

Roozbeh Razavi-Far
Ariel Ruiz-Garcia
Vasile Palade
Juergen Schmidhuber *Editors*

Generative Adversarial Learning: Architectures and Applications

Volume 217

Intelligent Systems Reference Library

Series Editors

Janusz Kacprzyk

Polish Academy of Sciences, Warsaw, Poland

Lakhmi C. Jain

KES International, Shoreham-by-Sea, UK

The aim of this series is to publish a Reference Library, including novel advances and developments in all aspects of Intelligent Systems in an easily accessible and well structured form. The series includes reference works, handbooks, compendia, textbooks, well-structured monographs, dictionaries, and encyclopedias. It contains well integrated knowledge and current information in the field of Intelligent Systems. The series covers the theory, applications, and design methods of Intelligent Systems. Virtually all disciplines such as engineering, computer science, avionics, business, e-commerce, environment, healthcare, physics and life science are included. The list of topics spans all the areas of modern intelligent systems such as: Ambient intelligence, Computational intelligence, Social intelligence, Computational neuroscience, Artificial life, Virtual society, Cognitive systems, DNA and immunity-based systems, e-Learning and teaching, Human-centred computing and Machine ethics, Intelligent control, Intelligent data analysis, Knowledge-based paradigms, Knowledge management, Intelligent agents, Intelligent decision making, Intelligent network security, Interactive entertainment, Learning paradigms, Recommender systems, Robotics and Mechatronics including human-machine teaming, Self-organizing and adaptive systems, Soft computing including Neural systems, Fuzzy systems, Evolutionary computing and the

Fusion of these paradigms, Perception and Vision, Web intelligence and Multimedia.

Indexed by SCOPUS, DBLP, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

More information about this series at [https://link.springer.com/
bookseries/8578](https://link.springer.com/bookseries/8578)

Editors

Roozbeh Razavi-Far, Ariel Ruiz-Garcia, Vasile Palade and
Juergen Schmidhuber

Generative Adversarial Learning: Architectures and Applications



Editors

Roozbeh Razavi-Far

Department of Electrical and Computer Engineering and School of
Computer Science, University of Windsor, Windsor, ON, Canada

Ariel Ruiz-Garcia

SeeChange.ai, Manchester, UK

Vasile Palade

Centre for Computational Science and Mathematical Modelling, Coventry
University, Coventry, UK

Juergen Schmidhuber

The Swiss AI Lab, IDSIA, University of Lugano, USI & SUPSI, Lugano,
Switzerland

ISSN 1868-4394

e-ISSN 1868-4408

Intelligent Systems Reference Library

ISBN 978-3-030-91389-2

e-ISBN 978-3-030-91390-8

<https://doi.org/10.1007/978-3-030-91390-8>

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are solely and exclusively
licensed by the Publisher, whether the whole or part of the material is
concerned, specifically the rights of translation, reprinting, reuse of
illustrations, recitation, broadcasting, reproduction on microfilms or in any
other physical way, and transmission or information storage and retrieval,
electronic adaptation, computer software, or by similar or dissimilar
methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG

The registered company address is: Gewerbestrasse 11, 6330 Cham,
Switzerland

Preface

The recent big advancements in Generative Adversarial Networks (GANs) have opened countless opportunities for synthetic data generation in many areas, including science, engineering, medicine, security, among others. Generative Adversarial Networks are built based on two dueling neural networks, in which one network aims to generate new samples and convince the competing network that the generated samples are real and not synthetic.

This book contains recent theoretical advancements on Generative Adversarial Networks and their interdisciplinary applications, such as for fair data generation, image generation, face aging, intrusion detection, industrial defect detection, action recognition, drug discovery, medical imaging, among other applications.

The book is intended for scientists, researchers, educators, and anyone interested in the latest methodological advancements on Generative Adversarial Networks and their applications. In addition, the book can be considered as a reference for various state-of-the-art architectures of GANs, major training challenges of GANs including convergence, vanishing gradients, and mode collapse, as well as the technical remedies to overcome these challenges.

The editors would like to convey many thanks to those who contributed to this book by sharing their insights and achievements in this emerging area in machine learning.

We are also very grateful for the assistance provided by Springer-Verlag.

Roozbeh Razavi-Far
Ariel Ruiz-Garcia
Vasile Palade
Juergen Schmidhuber
Ontario, Canada
Manchester, UK
Coventry, UK
Lugano, Switzerland

Contents

1 An Introduction to Generative Adversarial Learning: Architectures and Applications

Roozbeh Razavi-Far, Ariel Ruiz-Garcia and Vasile Palade

1.1 Book Outline

References

2 Generative Adversarial Networks: A Survey on Training, Variants, and Applications

Maryam Farajzadeh-Zanjani, Roozbeh Razavi-Far, Mehrdad Saif and Vasile Palade

2.1 Introduction

2.2 GAN Variants

2.2.1 Original Generative Adversarial Network (GAN)

2.2.2 Deep Convolutional Generative Adversarial Network (DCGAN)

2.2.3 Conditional Generative Adversarial Network (CGAN)

2.2.4 Information Maximizing Generative Adversarial Networks (InfoGAN)

2.2.5 Auxiliary Classifier Generative Adversarial Network (ACGAN)

2.2.6 Stacked Generative Adversarial Networks (StackGAN)

2.2.7 Cycle-Consistent Generative Adversarial Network (CycleGAN)

2.2.8 Wasserstein Generative Adversarial Network (WGAN)

2.2.9 Semi-Supervised Generative Adversarial Network (SSGAN)

2.2.10 Progressive Growing Generative Adversarial Network (Progressive GAN)

2.2.11 Style-Based Generative Adversarial Network (StyleGAN)

2.2.12 Bidirectional Generative Adversarial Network (BiGAN)

2.2.13 Bayesian Generative Adversarial Network (BGAN)

2.3 GAN Applications

2.4 Conclusions

References

3 Fair Data Generation and Machine Learning Through Generative Adversarial Networks

Xintao Wu, Depeng Xu, Shuhan Yuan and Lu Zhang

3.1 Introduction

3.2 Overview of FairGAN Framework

3.2.1 Definitions and Metrics of Fairness

3.2.2 FairGAN Framework

3.3 Simplified Fairness Aware Generative Adversarial Networks

3.3.1 Model Framework Design

3.3.2 Empirical Evaluation

3.4 Achieving Causal Fairness in Data Generation

3.4.1 Model Framework Design

3.4.2 Achieving Direct/Indirect/Counterfactual Fairness

3.4.3 Empirical Evaluation

3.5 Achieving Fairness in Classification

3.5.1 Model Framework Design

3.5.2 Achieving Fair Classification

3.5.3 Empirical Evaluation

3.6 Related Work

3.6.1 Dealing with Different Types of Structural Data

3.6.2 Dealing with Privacy

3.7 Future Directions

3.7.1 Variants Comparison and Architecture Design

3.7.2 Achieving Long-Term Fairness in Dynamic Decision Making

3.7.3 Achieving Fairness in Regression

3.7.4 Achieving Fairness in Recommendation

3.7.5 Open Source Software

3.8 Conclusions

References

4 Quaternion Generative Adversarial Networks

Eleonora Grassucci, Edoardo Cicero and Danilo Comminiello

4.1 Introduction

4.2 Quaternion Algebra

4.3 Generative Learning in the Quaternion Domain

4.3.1 The Quaternion Adversarial Framework

4.3.2 Quaternion Fully Connected Layers

4.3.3 Quaternion Convolutional Layers

4.3.4 Quaternion Pooling Layers

4.3.5 Quaternion Batch Normalization

4.3.6 Quaternion Spectral Normalization

4.3.7 Quaternion Weight Initialization

4.3.8 Training

4.4 GAN Architectures in the Quaternion Domain

4.4.1 Vanilla QGAN

4.4.2 Advanced QGAN

4.4.3 Evaluation Metrics

4.5 Experimental Evaluation

4.5.1 Evaluation of Spectral Normalization Methods

4.6 Conclusions

References

5 Image Generation Using Continuous Conditional Generative Adversarial Networks

Xin Ding, Yongwei Wang, Zuheng Xu, William J. Welch and Z. Jane Wang

5.1 Introduction and Motivation

5.2 Continuous Conditional Generative Adversarial Networks

5.2.1 Derivation of HVDL and SVDL Losses

5.2.2 A Rule of Thumb for Hyper-parameter Selection

5.2.3 Algorithms for Training CcGANs

5.3 Theoretical Analysis

5.4 Experiments

5.4.1 Case Study 1: Circular 2-D Gaussians

5.4.2 Case Study 2: UTKFace

5.5 Conclusion

References

6 Generative Adversarial Networks for Data Augmentation in Hyperspectral Image Classification

Dimitra Koumoutsou, Georgios Siolas, Eleni Charou and Georgios Stamou

6.1 Introduction

6.1.1 Dataset

6.1.2 Data Availability

6.1.3 Class Imbalance

6.1.4 Data Augmentation

6.2 Previous Work

6.2.1 Data Augmentation

6.2.2 Class Imbalance

6.2.3 Applications in Hyperspectral Imaging

6.3 Wasserstein GAN

6.4 Conditional Wasserstein Generative Adversarial Network with Gradient Penalty for Hyperspectral Image Generation

6.4.1 Wasserstein Generative Adversarial Network with Gradient Penalty

6.4.2 Hyperspectral Data Patch

6.4.3 Dimensionality Reduction

6.4.4 Discriminator and Generator Models

6.4.5 Training Process

6.5 Experimental Results

6.5.1 Evaluation Metrics

6.5.2 Experimental Setting

6.5.3 Spectral Signature

6.5.4 Data Augmentation

6.5.5 Visualizations

6.6 Conclusion and Future Work

References

7 Face Aging Using Generative Adversarial Networks

Bruno Kemmer, Rodolfo Simões and Clodoaldo Lima

7.1 Introduction

7.2 Generative Adversarial Networks

7.2.1 Mode Collapse

7.2.2 GANs Types

7.2.3 Reference Architectures for Facial Imaging

7.3 Databases Used for Facial Aging

7.3.1 FG-NET

7.3.2 UTKFace

7.3.3 CACD

7.3.4 MORPH

7.3.5 IMDB-Wiki

7.3.6 Cross-Age LFW (CALFW)

7.3.7 Other Databases

7.4 Experiments and Results

7.4.1 CAAE Experiments

7.4.2 IPCGAN Experiments

7.4.3 Recursive Chaining of Reversible Image-to-Image Translators Experiments, RCRIIT

7.4.4 Method Comparison

7.5 Summary

References

8 Embedding Time-Series Features into Generative Adversarial Networks for Intrusion Detection in Internet of Things Networks

Ehsan Hallaji, Roozbeh Razavi-Far and Mehrdad Saif

8.1 Introduction

8.2 Related Works

8.2.1 Adversarially Learned Anomaly Detection

8.2.2 GAN Ensemble for Anomaly Detection

8.2.3 Anomaly Detection with GAN

8.2.4 Unsupervised Change Detection with GAN

8.2.5 Unsupervised Change Detection with GAN

8.3 Internet of Things Network

8.3.1 Dataset

8.3.2 IoT Testbed

8.4 Feature Embedding

8.4.1 Extracting Sequential Changes

8.4.2 Generative Adversarial Cluster Analysis

8.4.3 Training Computational Complexity

8.5 Experimental Results

8.5.1 Experimental Setting

8.5.2 Results Analysis

8.6 Conclusion

References

9 Inspection of Lead Frame Defects Using Deep CNN and Cycle-Consistent GAN-Based Defect Augmentation

Chia-Feng Juang and Wei-Shane Chen

9.1 Introduction

9.2 Defect Inspection Using the Faster R-CNN

9.2.1 Materials

9.2.2 Defect Inspection Using the Faster R-CNN

9.3 Defect Augmentation Using the CycleGAN

9.3.1 Basic GAN Structure

9.3.2 CycleGAN Structure

9.3.3 CycleGAN Training

9.3.4 CycleGAN for Image Augmentation

9.4 Experiments

9.4.1 Experimental Result of CyleGAN

9.4.2 Experimental Result of Defect Inspection Using the Faster R-CNN

9.5 Conclusions

References

10 Adversarial Learning in Accelerometer Based Transportation and Locomotion Mode Recognition

Lukas Günthermann, Lin Wang, Ivor Simpson, Andrew Philippides and Daniel Roggen

10.1 Introduction

10.2 SHL Dataset

10.3 State of the Art in Classifying the SHL Dataset

10.4 Background on Generative Adversarial Networks

10.5 Methods

10.5.1 Feature Extraction

10.5.2 Feature Scaling

10.5.3 Dealing with Class Imbalance

10.5.4 Feature Selection

10.5.5 Proposed Generative Adversarial Network

10.5.6 Hyperparameter Tuning

10.6 Implementation

10.7 Results

10.7.1 User Specific Evaluation

10.7.2 User Semi-independent Evaluation

10.7.3 User Independent Evaluation

10.8 Discussion

10.8.1 Performance Evaluation

10.8.2 Hyperparameter Tuning

10.9 Conclusion

References

11 GANs for Molecule Generation in Drug Design and Discovery

Ziqiao Zhang, Fei Li, Jihong Guan, Zhenzhou Kong, Liming Shi and Shuigeng Zhou

11.1 Introduction

11.2 Preliminary Concepts

11.2.1 Molecular Representation

11.2.2 Evaluation Metrics

11.3 GAN-based Molecule Generation Models

11.3.1 Goal-Directed Models

11.3.2 Distribution-Learning Models

11.3.3 Applications of GAN-based Molecule Generative Models

11.4 Comparison with Other Generative Models for Molecule Generation

11.4.1 Molecule Generation Based on Other Generative Models

11.4.2 Advantages of GAN-based Models

11.4.3 Disadvantages of GAN Based Models

11.5 Challenges and Future Directions

11.5.1 New Molecular Representations

11.5.2 New Molecule Generation Models

11.5.3 Benchmarks and Metrics

11.5.4 New Pharmaceutical Objective Functions

11.5.5 Influence of Property Prediction Models

11.6 Conclusion

References

12 Improved Diagnostic Performance of Arrhythmia Classification Using Conditional GAN Augmented Heartbeats

Deepankar Nankani and Rashmi Dutta Baruah

12.1 Introduction

12.1.1 ECG Synthesis Using Generative Adversarial Network

12.1.2 Related Work on Heartbeat Classification

12.1.3 Contributions

12.2 Data

12.2.1 Dataset Description

12.2.2 Data Preprocessing

12.3 Methodology

12.3.1 ECG Generation Methodology

12.3.2 Augmentation Using Deep Convolutional Conditional GANs

12.3.3 ECG Classification Methodology

12.4 Results and Discussion

12.4.1 Computing Platform

12.4.2 Evaluation Metrics

12.4.3 DCCGAN Performance Evaluation

12.4.4 Beat Classification Performance

12.5 Conclusion and Future Work

References

13 Generative Adversarial Network Powered Fast Magnetic Resonance Imaging—Comparative Study and New Perspectives

Guang Yang, Jun Lv, Yutong Chen, Jiahao Huang and Jin Zhu

13.1 Introduction

13.1.1 Magnetic Resonance Imaging

13.1.2 Limitations of Magnetic Resonance Imaging

13.1.3 Conventional Acceleration Using Compressive Sensing

13.1.4 Deep Learning Based Fast MRI

13.1.5 GAN Powered Fast MRI

13.2 Methods

13.2.1 Fundamentals of MRI Reconstruction

13.2.2 CNN Based MRI Reconstruction

13.2.3 GAN Based MRI Reconstruction

13.2.4 Evaluation Methods

13.3 Benchmarking

13.4 Discussion

13.5 Conclusion

References

14 Generative Adversarial Networks for Data Augmentation in X-Ray Medical Imaging

Talib Iqbali and M. Arif Wani

14.1 Introduction

14.2 Previous Work on Using GANs and Transfer Learning in X-Ray Imaging

14.3 Deep Convolutional GAN (DCGAN) and Its Limitations for X-Ray Imaging

14.3.1 Architecture of DCGAN

14.3.2 DCGAN Training

14.4 Progressively Growing GAN (PGGAN)

14.4.1 Training of PGGAN

14.4.2 Phasing in a New Layer for Smooth Model Building

14.4.3 Transfer Learning for Model Building

14.5 Results and Discussion

14.5.1 Dataset

14.5.2 Augmentation Results

14.6 Conclusion

References

1. An Introduction to Generative Adversarial Learning: Architectures and Applications

Roozbeh Razavi-Far¹✉, Ariel Ruiz-Garcia² and Vasile Palade³✉

- (1) Department of Electrical and Computer Engineering and School of Computer Science, University of Windsor, 401 Sunset Avenue, Windsor, ON, N9B 3P4, Canada
(2) SeeChange.ai, Manchester, UK
(3) Centre for Data Science, Coventry University, Coventry, UK

✉ **Roozbeh Razavi-Far (Corresponding author)**
Email: roozbeh@uwindsor.ca

✉ **Vasile Palade**
Email: vasile.palade@coventry.ac.uk

Generative Adversarial Networks (GANs) continue to be one of the most popular deep learning approaches for synthetic data generation. Moreover, GANs are finding their way in other application domains, such as speech and audio synthesis, object detection and segmentation, text-to-image translation, and policy learning in deep reinforcement learning, among many others. This book presents a collection of chapters on the latest advancements on GANs and state-of-the-art applications.

GANs are made up of two neural networks: One network learns to generate new data samples, and the other tries to tell whether a data sample was produced by the generator network or it belongs to the training set. The main objective of the generator is to convince the discriminator that the sample it generated is part of the training set and not synthesized. This

approach is a special case of Artificial Curiosity, a learning paradigm introduced by Schmidhuber in 1990, in which two neural networks compete with each other in a minimax game [23, 25, 29]. Adversarial networks were also used for unsupervised learning in slightly later work by Schmidhuber, called Predictability Minimization [24, 26–28]. Nonetheless, simultaneous training of a generator and a discriminator model makes the learning process unstable and difficult to optimize. This added complexity often results in problems such as mode collapse, vanishing gradients, and non-convergence. The chapters included in this book propose novel solutions to address such problems and prove these solutions in the context of challenging applications.

Upon implementation of generative adversarial networks by training two neural networks in [5] and conditional adversarial networks in [19], a large number of GAN variations have been proposed, which are mainly reviewed in the next chapter [4]. Recently, there has been an increasing interest in applications of generative adversarial networks in a broad range of domains, such as image processing and vision, medical imaging, natural language processing, signal processing, monitoring and diagnostic, and security analysis.

Several GAN-based techniques have been developed to address different types of machine learning problems. Among all, generating synthetic samples to form a balanced dataset is one of the major applications of GAN [9, 20], which is also addressed in Chaps. 9 and 14. In addition, changing the architecture of the GAN by adding an auxiliary classifier to improve the quality of generated samples [18], or for the sake of semi-supervised learning [3], is useful and popular in recent literature. Besides modifications in the architecture of the GAN, there has been growing interest to develop GAN’s objective functions with respect to specific applications for generating knockoffs [11, 13], dimensionality reduction [2], and missing data imputation [16]. Moreover, transfer learning and reinforcement learning are two hot topics that combined with the concept of two duelling neural networks of GANs found several applications in both academia and industry [7, 22].

This book is intended for researchers, engineers, and practitioners looking to make use of and stay up to date with the latest advancements on GAN theoretical developments as well as its real-world applications.

1.1 Book Outline

The book consists of 13 chapters in addition to the first introductory chapter. These high quality contributed research chapters introducing theoretical advancements and state-of-the-art applications of GANs. Every chapter has undergone a rigorous review process by independent reviewers before being accepted. We provide a brief description of each chapter hereafter.

The next chapter, entitled “Generative Adversarial Networks: A Survey on Training, Variants, and Applications”, surveys different state-of-the-art GAN-based methods and their applications. The chapter firstly reviews state-of-the-art generative adversarial models, and, then, focuses on major training problems, including vanishing gradients, mode collapse, and convergence, and various solutions to address these problems. It then concentrates on recent advancements to adjust loss functions, modify the training process, and add auxiliary neural networks. This chapter finally summarizes various applications of generative adversarial networks [4].

The book then focuses on fair data generation. The chapter “Fair Data Generation and Machine Learning through Generative Adversarial Networks” presents a generative adversarial framework for so-called fair data generation and fair predictive learning. The proposed framework, called FairGAN, accommodates various fairness notions by changing the network architecture and objective functions [30].

The book then considers applications of GANs in image generation. This begins with the chapter “Quaternion Generative Adversarial Networks” that proposes a group of quaternion-valued generative adversarial networks (QGANs), which allows to process channels as a single entity and capture internal latent relations, at the same time, reducing the computational resources [6]. Chapter “Image Generation Using Continuous Conditional Generative Adversarial Networks” introduces continuous conditional generative adversarial networks (CcGAN), the first generative model for image generation with continuous and scalar conditions. Two novel discriminator losses and a label input method are proposed for CcGAN. Error bounds of both losses are also provided in this chapter [1]. Chapter “Generative Adversarial Networks for Data Augmentation in Hyperspectral Image Classification” aims to modify the

Gradient Penalty in Wasserstein Generative Adversarial Network for conditional generation of realistic hyperspectral data cubes [17]. Chapter “Face Aging using Generative Adversarial Networks” enumerates the most frequent GAN-based architectures for facial imaging, their main challenges, and available benchmark databases. It then compares different face aging architectures using the most frequent databases and measures the effectiveness of the resulting faces [15].

The book then focuses on other applications of GANs including intrusion detection in internet of things, defect detection in industrial data, and activity recognition. Chapter “Embedding Time-Series Features into Generative Adversarial Networks for Intrusion Detection in Internet of Things Networks” firstly reviews applications of generative adversarial networks for anomaly detection. It then proposes a novel method for embedding the time-series characteristics of the data stream into the GAN-based detectors. The proposed approach and other GAN-based approaches are evaluated for detecting intrusions in an Internet of Things (IoT) network [10]. Chapter “Inspection of Lead Frame Defects Using Deep CNN and Cycle-consistent GAN-based Defect Augmentation” proposes a defect augmentation technique by means of the Cycle-consistent Generative Adversarial Network (CycleGAN) in order to automatically generate different types of defects. CycleGAN automatically translates normal patches on lead frame images to defect patches, and, then, a linear weighting technique is used to blend the augmented defect patches into the lead frame images in order to obtain augmented defects that are consequently used to train the faster region-based convolutional neural network (R-CNN). The trained R-CNN is then used to detect and classify defects on lead frames [14]. Chapter “Adversarial Learning in Accelerometer based Transportation and Locomotion Mode Recognition” introduces a fully connected neural network as a classifier into the conditional GAN framework. The proposed framework aims to perform semi-supervised learning on labeled and unlabeled data samples by means of adversarial learning between the discriminator and generator and between the discriminator and classifier. The proposed method has been used to address the problem of improving the recognition of human activities from smartphone sensors when limited training data is available [8].

The last chapters of this book focuses on more specific applications of GANs in drug discovery, biomedical engineering, and medical imaging. Chapter “GANs for Molecule Generation in Drug Design and Discovery” aims to survey the applications of generative adversarial networks for molecule generation in drug design and discovery and provides the readers with an overview of the preliminary concepts of molecule generation. This chapter also compares GAN-based approaches and other competitors for molecule generation and discusses their limitations [32]. Chapter “Improved Diagnostic Performance of Arrhythmia Classification Using Conditional GAN Augmented Heartbeats” focuses on the generation and classification of normal beats, supraventricular ectopic beats, and ventricular ectopic beats by encoding class-related dependencies in the deep convolution conditional generative adversarial network (DCCGAN) model. The proposed DCCGAN model augments specific class beats. These augmented beats along with original beats are used to train a Parallel Convolutional Neural Network (PCNN) classifier. The trained PCNN improves life-threatening morphological arrhythmia detection [21]. Chapter “Generative Adversarial Network Powered Fast Magnetic Resonance Imaging—Comparative Study and New Perspectives” reviews generative adversarial network-based methods to solve fast MRI and conducts a comparative study on various anatomical datasets [31]. Chapter “Generative Adversarial Networks for Data Augmentation in X-Ray Medical Imaging” focuses on the use of GANs for data augmentation in X-ray medical imaging and proposes a Progressively Growing Generative Adversarial Network (PGGAN), which improves the accuracy of classifying pathologies using chest X-rays [12].

The chapters presented in this book bring us a step closer to solving some of the most challenging obstacles when training GANs, such as the mode collapse and training stability. Moreover, they also demonstrate the promising performance of GANs to address challenging topics such as image-to-image translation and realistic data generation. The latter has the potential for a broad impact and can facilitate training of deep neural networks where lack of data is a common problem, as demonstrated in the contributed chapter by Koumoutsou et al. [17].

The editors would like to express their gratitude to the authors of the contributed chapters, as well as the reviewers who helped ensure the high

quality of these chapters. Lastly, the editors hope the readers of this book will enjoy it and make good use of it.

References

1. Ding, X., Wang, Y., Xu, Z., Welch, W.J., Wang, Z.J.: Image generation using continuous conditional generative adversarial networks (Chap. 5). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 87–114. Springer, Berlin (2021)
2. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M.: Generative adversarial dimensionality reduction for diagnosing faults and attacks in cyber-physical systems. *Neurocomputing* **440**, 101–110 (2021) [\[Crossref\]](#)
3. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M., Parvania, M.: Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. *IEEE Trans. Smart Grid* **12**(4), 3468–3478, (2021)
4. Farajzadeh-Zanjani, M., Razavi-Far, R., Saif, M., Palade, V.: Generative adversarial networks: a survey on training, variants, and applications (Chap. 2). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 7–30. Springer, Berlin (2021)
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., Red Hook (2014)
6. Grassucci, E., Cicero, E., Comminiello, D.: Quaternion generative adversarial networks (Chap. 4). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 57–86. Springer, Berlin (2021)
7. Guimaraes, G.L., Sanchez-Lengeling, B., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models (2020). [arXiv:1705.10843](#)
8. Günthermann, L., Wang, L., Simpson, I., Philippides, A., Roggen, D.: Adversarial learning in accelerometer based transportation and locomotion mode recognition

(Chap. 10). R. Razavi-Far et al. (eds.), Generative Adversarial Learning: Architectures and Applications, Intelligent Systems Reference Library, pp. 205–222. Springer, Berlin (2021)

9. Hallaji, E., Razavi-Far, R., Palade, V., Saif, M.: Adversarial learning on incomplete and imbalanced medical data for robust survival prediction of liver transplant patients. *IEEE Access* **9**, 73641–73650 (2021) [\[Crossref\]](#)
10. Hallaji, E., Razavi-Far, R., Saif, M.: Embedding time-series features into generative adversarial networks for intrusion detection in internet of things networks (Chap. 8). R. Razavi-Far et al. (eds.), Generative Adversarial Learning: Architectures and Applications, Intelligent Systems Reference Library, pp. 169–184. Springer, Berlin (2021)
11. Hassani, H., Razavi-Far, R., Saif, M., Palade, V.: Generative adversarial network-based scheme for diagnosing faults in cyber-physical power systems. *Sensors* **21**(15) (2021)
12. Iqbali, T., Wani, M.A.: Generative adversarial networks for data augmentation in X-Ray medical imaging (Chap. 14). R. Razavi-Far et al. (eds.), Generative Adversarial Learning: Architectures and Applications, Intelligent Systems Reference Library, pp. 341–XXX. Springer, Berlin (2021)
13. Jordon, J., Yoon, J., van der Schaar, M.: KnockoffGAN: generating knockoffs for feature selection using generative adversarial networks. In: International Conference on Learning Representations (2019)
14. Juang, C.F., Chen, W.S.: Inspection of lead frame defects using deep CNN and cycle-consistent GAN-based defect augmentation (Chap. 9). R. Razavi-Far et al. (eds.), Generative Adversarial Learning: Architectures and Applications, Intelligent Systems Reference Library, pp. 185–204. Springer, Berlin (2021)
15. Kemmer, B., Simões, R., Lima, C.: Face aging using generative adversarial networks (Chap. 7). R. Razavi-Far et al. (eds.), Generative Adversarial Learning: Architectures and Applications, Intelligent Systems Reference Library, pp. 145–168. Springer, Berlin (2021)
16. Kim, J., Tae, D., Seok, J.: A survey of missing data imputation using generative adversarial networks. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 454–456 (2020)
- 17.

- Koumoutsou, D., Siolas, G., Charou, E., Stamou, G.: Generative adversarial networks for data augmentation in hyperspectral image classification (Chap. 6). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 115–144. Springer, Berlin (2021)
18. Li, C., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates Inc, Red Hook (2017)
 19. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014)
 20. Mullick, S.S., Datta, S., Das, S.: Generative adversarial minority oversampling. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1695–1704 (2019)
 21. Nankani, D., Baruah, R.D.: Improved diagnostic performance of arrhythmia classification using conditional GAN augmented heartbeats (Chap. 12). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 275–304. Springer, Berlin (2021)
 22. Sarmad, M., Lee, H.J., Kim, Y.: RI-GAn-net: a reinforcement learning agent controlled gan network for real-time point cloud shape completion. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5891–5900 (2019)
 23. Schmidhuber, J.: Making the world differentiable: on using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report, Technische University Munich (1990). Technical Report FKI-126-90
 24. Schmidhuber, J.: Learning factorial codes by predictability minimization. Technical Report, Department of Computer Science, University of Colorado at Boulder (1991). Technical Report CU-CS-565-91
 25. Schmidhuber, J.: A possibility for implementing curiosity and boredom model-building neural controllers. In: Proceeding of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 222–227. MIT Press/Bradford Books (1991)
 - 26.

Schmidhuber, J.: Learning factorial codes by predictability minimization. *Neural Comput.* **4**(6), 863–879 (1992)
[[Crossref](#)]

27. Schmidhuber, J.: Netzwerkarchitekturen, zielfunktionen und kettenregel. (network architectures, objective functions, and chain rule). Ph.D. thesis, Inst. f. Inf., Technische University Munich (1993). Habilitation Thesis
28. Schmidhuber, J.: Neural predictors for detecting and removing redundant information. In: *Adaptive Behavior and Learning*. Kluwer, Alphen aan den Rijn (1999)
29. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Netw.* **127**, 58–66 (2020)
[[Crossref](#)]
30. Wu, X., Xu, D., Yuan, S., Zhang, L.: Fair data generation and machine learning through generative adversarial networks (Chap. 3). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 31–56. Springer, Berlin (2021)
31. Yang, G., Lv, J., Chen, Y., Huang, J., Zhu, J.: Generative adversarial network powered fast magnetic resonance imaging - comparative study and new perspectives (Chap. 13). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 305–340. Springer, Berlin (2021)
32. Zhang, Z., Li, F., Guan, J., Kong, Z., Shi, L., Zhou, S.: GANs for molecule generation in drug design and discovery (Chap. 11). R. Razavi-Far et al. (eds.), *Generative Adversarial Learning: Architectures and Applications*, Intelligent Systems Reference Library, pp. 233–274. Springer, Berlin (2021)

2. Generative Adversarial Networks: A Survey on Training, Variants, and Applications

Maryam Farajzadeh-Zanjani¹✉, Roozbeh Razavi-Far¹✉, Mehrdad Saif¹✉
and Vasile Palade²✉

- (1) Department of Electrical and Computer Engineering, University of Windsor, 401 Sunset Avenue, Windsor, ON, N9B 3P4, Canada
- (2) Centre for Data Science, Coventry University, Coventry, CV1 5FB, UK

✉ **Maryam Farajzadeh-Zanjani (Corresponding author)**
Email: farajza@uwindsor.ca

✉ **Roozbeh Razavi-Far**
Email: roozbeh@uwindsor.ca

✉ **Mehrdad Saif**
Email: msaif@uwindsor.ca

✉ **Vasile Palade**
Email: vasile.palade@coventry.ac.uk

Abstract

In recent years, Generative Adversarial Network (GAN) and its variants have gained great popularity in both academia and industry. In this chapter, we survey different state-of-the-art GAN-based methods and their applications. These techniques vary in architecture and objective functions. The chapter firstly introduces generative models followed by the GAN's usual training problems, such as vanishing gradients, mode collapse, and

convergence. Then, the proposed solutions and strategies for improving GAN’s training and convergence are provided, including related tasks such as obtaining higher image quality when GANs are used in image processing applications. The chapter reviews state-of-the-art GANs and focuses on the main advancements that involve adjusting the loss function, modifying the training process, and adding auxiliary neural network(s). A summary of different applications of GANs is also provided.

2.1 Introduction

Generative algorithms are of paramount importance in many machine learning applications. The generative algorithms have been mainly used either for feature selection or realistic data generation [24]. They can be categorized depending on whether they relate to the explicit density models or implicit density models. Explicit density estimation explicitly defines and solves for the models containing the distribution. They usually have the problem of computational tractability and difficulty of learning from high-dimensional data. The generative models under this category include the Markov chain method (e.g., Boltzmann Machines), Variational Autoencoder (VAE), and fully visible belief nets. On the other hand, implicit density estimation relates to building a model that can sample from the distribution without explicitly defining it. The Generative Stochastic Network (GSN) [2] and generative adversarial networks (GANs) [26] are two well-known examples of the category of generative models with implicit density. GANs, specifically, as a novel class of deep generative models have recently drawn significant attention since they can be trained through backpropagation and do not require Markov chains for sampling.

While GANs have become a hot topic in deep learning and gained the attention of both industry and academia, they have become very popular indeed in the domain of image processing and computer vision. This is because of their interesting design, their nature of learning (i.e., unsupervised learning do not need labels), and their great potential to generate new high-quality images. A general road map of generative adversarial networks is provided in Fig. 2.1. This figure shows how the idea of a generative adversarial network is expanded and developed during the

decades. Moreover, some state-of-the-art GAN-based methods are labelled in Fig. 2.1 and explained in next sections of this chapter.

Although GANs could potentially capture rich underlying distributions of the given data, there are some challenges associated with the training of duelling networks of GANs. These generally include vanishing gradients, mode collapse, and convergence of the training algorithm.

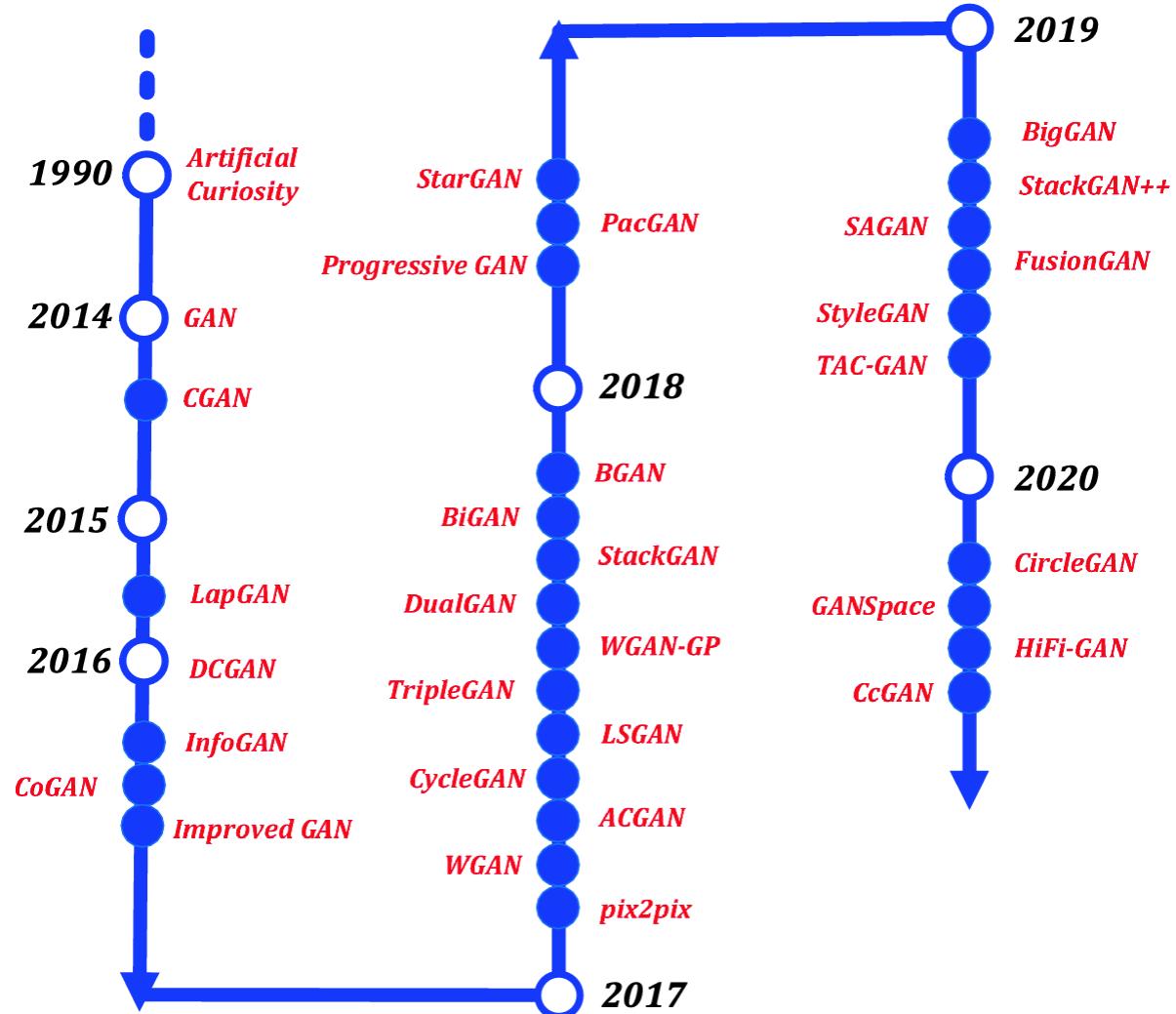


Fig. 2.1 A general road map of generative adversarial networks. It shows the expansion of idea of artificial curiosity [75] to different variants of generative adversarial networks (GAN) [26], including conditional GAN (CGAN) [53], Laplacian GAN (LapGAN) [15], Deep convolutional GAN (DCGAN) [62], Information Maximizing GAN (InfoGAN) [8], Coupled GAN (CoGAN) [48], Improved GAN represents Improved techniques for training GANs [71], Image-to-Image Translation with Conditional Adversarial Networks (pix2pix) [35], Wasserstein GAN (WGAN)

[4], Auxiliary Classifier GAN (ACGAN) [58], Cycle-Consistent GAN (CycleGAN) [98], Least Squares GAN (LSGAN) [51], Triple GAN (TripleGAN) [44], Unsupervised Dual Learning for Image-to-Image Translation (DualGAN) [92], Stacked GAN (StackGAN) [96], Bidirectional GAN (BiGAN) [17], Bayesian GAN (BGAN) [70], Progressive Growing GAN (Progressive GAN) [38], The Power of Two Samples in Generative Adversarial Networks (PacGAN) [45], Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation (StarGAN) [10], BigGAN [5], StackGAN++ [97], Self-Attention GAN (SAGAN) [95], FusionGAN [50], StyleGAN [39], Twin Auxiliary Classifiers GAN (TAC-GAN) [25], CircleGAN [76], GANspace [30], HiFiGAN [42], Continuous Conditional GAN (CcGAN) [16]

The difficulty of training GANs is mainly due to the adversarial process between the generator and discriminator. Two duelling neural networks are trained simultaneously in a zero-sum game, so any improvements to either the generator or discriminator involve the modification of parameters for the other model. For instance, considering a strong discriminator in adversarial learning need modifications of parameters for the generator. When the discriminator is strong, it is too confident in the predictions of original samples. This affects the trainability of the generator. Such a discriminator may not let the generator learn the distribution of data. In other words, in early training, while the generated samples and the original samples are too different, an optimal discriminator does not provide enough information to the generator for moving forward. In this case, the training of the generator performs so slowly and it can fail since the gradient for the Jensen–Shannon divergence vanishes. Although the improvement of min–max loss has been already proposed to overcome the vanishing gradient problem [26], many researchers tried different alternative objective functions and proposed extended variants of GANs. One of the most fundamental generative adversarial networks in this respect is Wasserstein GAN [4], which makes use of the Wasserstein distance and is explained later.

Another case of GAN’s failure is known as the mode collapse problem [19]. For instance, considering a generator in adversarial training, when it can generate such a plausible sample to the discriminator and learn to repeat it in every iteration. Then, the generator produces only a limited set of samples while the discriminator is trapped in the local minimum and can not keep progress. In other words, the generator rotates through generating

the same small set of samples plausible to the discriminator, while the discriminator has no strategy to escape; as a result, the diversity among generated samples is unsatisfactory, and this fails to be acknowledged. To this aim, various extensions of GANs have proposed some remedies including the Wasserstein loss, gradient penalty (as in WGAN-GP [4]), unrolled optimization of the discriminator [52], using implicit variational learning [79], boosting generative models [82], etc., to balance the generator’s and discriminator’s training and alleviate the over-optimization of the generator.

In addition to the above-mentioned training problems, GANs suffer from convergence failure when the models’ parameters destabilize. It is not easy to train two competing neural networks of GAN to achieve a stable model. Then, it is essential to find an equilibrium between the discriminator and generator. For instance, considering a generator produces samples far from the original distribution and too easy to classify for the discriminator, while the discriminator’s loss rapidly goes very close to zero. A promising solution for the GAN convergence failure is to consider regularization, adding noise to the discriminator [3], penalizing its weight, proper selection of the optimization function (e.g., Adam optimizer), and hyperparameters selection and tuning (e.g., batch size and learning rate) [68].

Moreover, feature matching [71] can be applied to prevent overtraining of a generator on the current discriminator in an iteration of adversarial training. Then, the generator needs to be trained to match the features that are potentially on a middle layer of the discriminator. For instance, mean and covariance feature matching has been employed in McGAN [54] for the sake of stable training of GANs.

On the other hand, minibatch discrimination [71] lets the discriminator look at multiple samples in combination instead of treating each sample independently. The original and generated samples are fed into the discriminator in distinctive batches, and the similarity of each sample with others in the same batch is calculated to avoid the generator collapsing at the same point and producing similar images. Moreover, to avoid overconfidence in the prediction of labels, one-sided label smoothing can be used [80]. While dropout regularization also can be employed for the sake of avoiding the overfitting problem [8]. In addition, historical averaging and

variational batch normalization are among other techniques to ease the GAN training process and improve its optimization [71].

Overall, various strategies have been proposed to improve GANs' training and ensure the stability. These methods are summarized and illustrated in Fig. 2.2. For example, probability ratio clipping on the generator side and sample reweighting on the discriminator side was proposed in [88]. Moreover, Salimans et al. discussed about feature matching, mini-batch discrimination, historical averaging and variational batch normalization as some solution for GAN training in [71]. In addition, choosing proper optimization functions [56] and balancing learning rate of the generator and the discriminator by two time-scale update rule [33] can enhance training stability. The other widely used method is one-sided label smoothing [80] to avoid an over confident discriminator and make the training more robust. Furthermore, several normalization approaches are shown in Fig. 2.2 including batch normalization [34], layer normalization [90], weight normalization [72], instance normalization [57, 83], group normalization [87], spectral normalization [89], and switchable normalization [49].

GANs have been the center of conversations and are widely used by many researchers in several domains for different purposes, such as fault diagnosis [22], signal reconstructions [59], dimensionality reduction [21], class imbalance learning [69], speech enhancement [61], text generation [14], etc. In this survey, we have organized the researches on GANs into two categories: (I) The major researches on GANs focused on the improvement of the architecture, or objective function, or both. These advancements mainly involve adjusting loss function, modifying the training process, and/or adding auxiliary neural network(s). The extended variants of GANs mostly aim to enhance the quality of generated samples, provide some remedies for the sake of GANs' training, overcome mode collapse and convergence problems, support inverse mapping, etc. The state-of-the-art GAN variants are presented in Sect. 2.2 of this chapter. (II) The researches related to the recent applications of generative adversarial networks and their variants in data science, engineering and medical fields [91], e.g., for missing data imputation [41], anomaly detection, cyber-attack classification [22], survival prediction of liver transplant patients [29], etc. The applications of GAN are provided in Sect. 2.3.

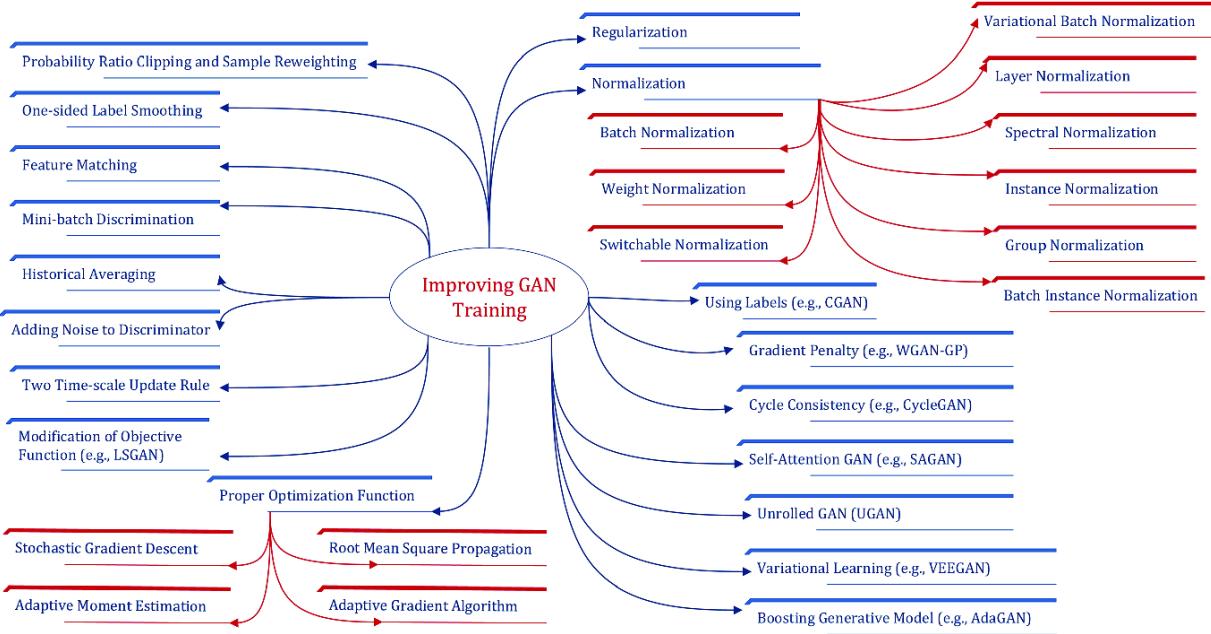


Fig. 2.2 A general overview of strategies for improving generative adversarial networks training

2.2 GAN Variants

This section presents the original GAN and its main variants.

2.2.1 Original Generative Adversarial Network (GAN)

GANs [26] consist of two neural networks that are arranged in competition with each other. These two duelling networks are named the generator \mathcal{G} and discriminator \mathcal{D} . The architecture of GAN is presented in Fig. 2.3. The idea behind adversarial learning of GANs is that \mathcal{G} tries to “fool” \mathcal{D} by generating samples that follow the distribution of real samples p_{data} , while \mathcal{D} tries to distinguish between these fake samples generated by \mathcal{G} and the real samples. To this aim, \mathcal{G} model a mapping function from a prior noise distribution $p_z(z)$ into $\mathcal{G}(z)$. The adversarial process performs through a min-max game between \mathcal{G} and \mathcal{D} . As a result, the generator develops samples so that \mathcal{D} can not discriminate and is deceived. GANs employ multi-layer perceptron (MLP) as the generator and the discriminator, which is more suitable for small datasets. The objective function of GAN [26] is formulated as follows:

(2.1)

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \in p_{data}} [\log \mathcal{D}(x)] + \mathbb{E}_{z \in p_z(z)} [\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

where \mathbb{E} stands for the mathematical expectancy. The discriminator (i.e., a defender) predicts the probability that a given sample came from the original training data rather than the generator (i.e., an attacker). The discriminative neural network-based model \mathcal{D} is trained to maximize the probability of assigning the real label to original training samples (i.e., and the fake label to the generated samples from \mathcal{G}). Meanwhile, the generative neural network-based model \mathcal{G} is trained to minimize $\log(1 - \mathcal{D}(\mathcal{G}(z)))$. These two multi-layer perceptrons of \mathcal{D} and \mathcal{G} play a two-player minimax game by updating their weights through backpropagations. Moreover, for the sake of having much stronger gradients early in adversarial learning, it is recommended to train \mathcal{G} for maximizing $\log(\mathcal{D}(\mathcal{G}(z)))$ instead of minimizing $\log(1 - \mathcal{D}(\mathcal{G}(z)))$.

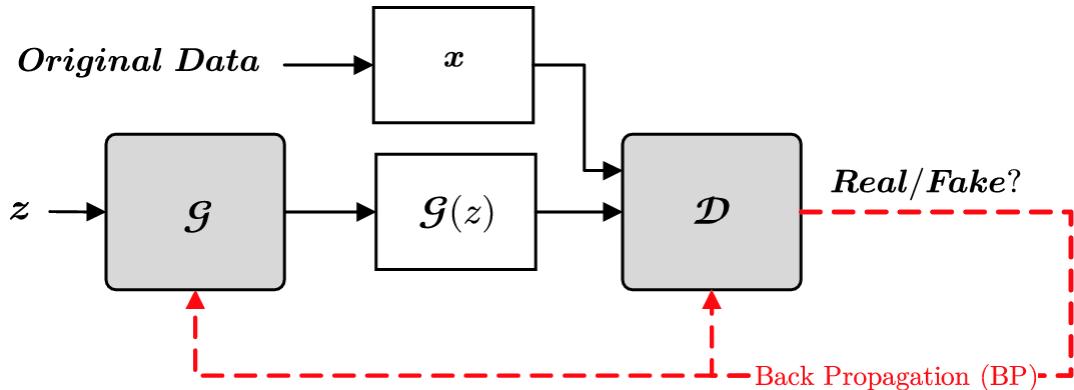


Fig. 2.3 Generative Adversarial Network [26] sets two neural networks of the generator \mathcal{G} and the discriminator \mathcal{D} in competition with each other, where noise $z \in p_z(z)$, $\mathcal{G}(z)$ shows the generated samples by \mathcal{G} , and $x \in p_{data}$ is a batch of samples from original data

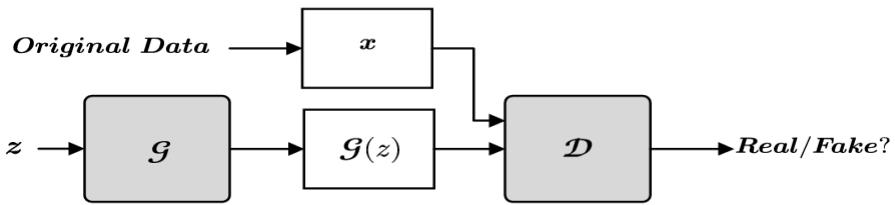
Despite the mesmerizing idea of adversarial training process of GANs to generate high-quality samples, there are some challenges to train these two duelling networks which grabbed the attention of several researchers to propose various new variants. Different alternatives were developed mainly by improvement in the GAN's structure and/or modifications on the objective function. These GANs' variants are reviewed in the following subsections.

2.2.2 Deep Convolutional Generative Adversarial Network (DCGAN)

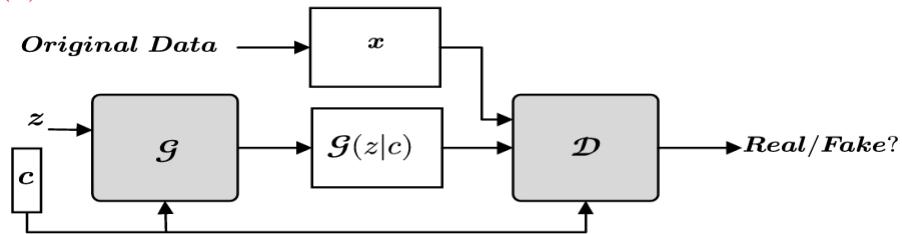
Deep Convolutional Generative Adversarial Network (DCGAN) [62] can be considered as a foundational support for GAN research. Similar to GAN, DCGAN has both the discriminator \mathcal{D} and generator \mathcal{G} with the same objective function, where the loss of each network is calculated during an adversarial process to update the generator's and the discriminator's weights. A schematic view of DCGAN is shown in Fig. 2.4.

However, for the sake of stable training, DCGAN made some improvements in the architecture of the GAN. The major difference between these two is that no convolution layers are used in GAN while deep convolutional networks are used (i.e., in place of GAN's fully connected networks) in DCGAN. The pooling layers are replaced with strided convolutions on \mathcal{D} and fractional strided-convolutions on \mathcal{G} . Moreover, the network in DCGAN consists of four convolution layers. On the discriminator side, these layers are joined with batch normalization (i.e., except for the first layer) and leaky ReLU activation function. On the other hand, the generator is in some fashion the reverse of the discriminator, with transpose convolution layer rather than convolution layer along with batch normalization (i.e., except for the last layer) and ReLU activation function at all layers except for the output, which uses Tanh activation.

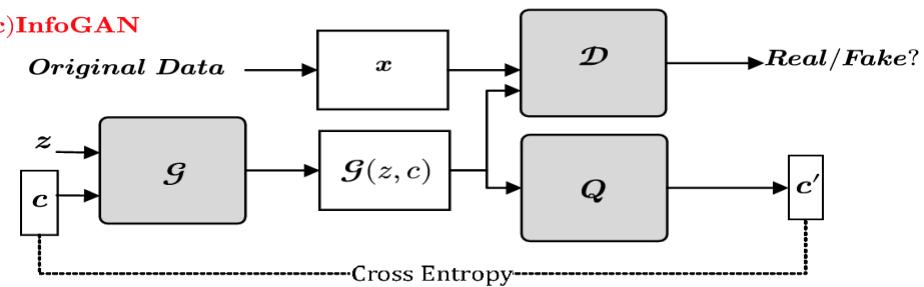
(a) GAN, DCGAN, LSGAN, WGAN



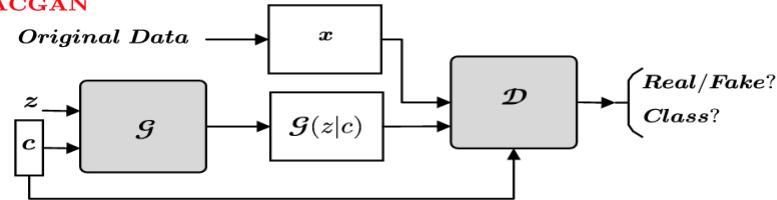
(b) CGAN



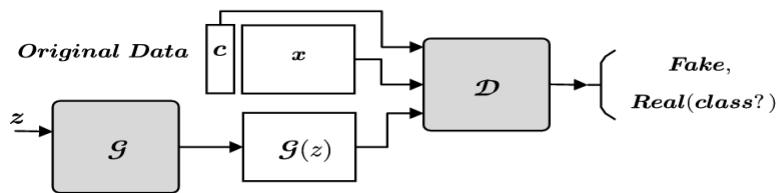
(c) InfoGAN



(d) ACGAN



(e) SSGAN



(f) BiGAN

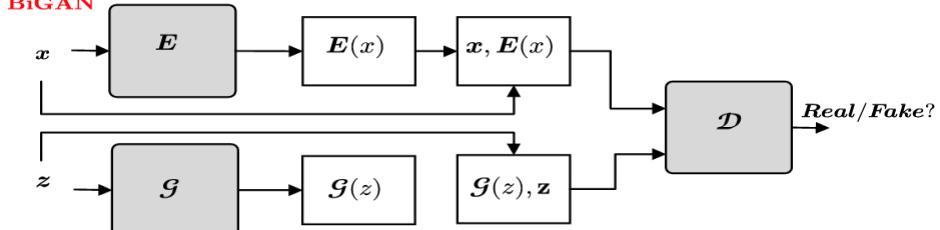


Fig. 2.4 A schematic view of variants of GAN including **a** Generative Adversarial Network (GAN) [26], Deep convolutional Generative Adversarial Network (DCGAN) [62], Least Squares Generative Adversarial Networks (LSGAN) [51], Wasserstein Generative Adversarial Network (WGAN) [4], **b** Conditional Generative Adversarial Network (CGAN) [53], **c** Information Maximizing Generative Adversarial Networks (InfoGAN [8]), **d** Auxiliary Classifier Generative Adversarial Network (ACGAN) [58], **e** Semi-supervised Generative Adversarial Network (SSGAN) [43], and **f** Bidirectional Generative Adversarial Network (BiGAN) [17]

2.2.3 Conditional Generative Adversarial Network (CGAN)

Conditional GAN [53] makes the generator capable of generating a certain class of data by simply adding a vector of features (i.e., that we wish to condition on), to both \mathcal{G} and \mathcal{D} . A schematic view of CGAN is shown in Fig. 2.4. This provides the generator with an ability to control the output. Then, CGAN training involves the training of the discriminator and generator that are conditioned on some extra information c . The objective function of CGAN is adopted for a conditional model as follows:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V_c(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \in p_{data}} [\log \mathcal{D}(x|c)] + \mathbb{E}_{z \in p_z(z)} [\log(1 - \mathcal{D}(\mathcal{G}(z|c)))] \quad (2.2)$$

in which c shows the condition, such as the class labels of the training samples. The conditional training of duelling networks can be also applied to DCGAN that takes into account convolutional neural network (CNN). Moreover, in [16], CGAN conditional on continuous, scalar conditions (i.e., regression labels) is proposed.

2.2.4 Information Maximizing Generative Adversarial Networks (InfoGAN)

Conditional GANs were developed to provide the generator with the ability to control the generated outputs. CGANs are supervised learning methods which take labels to generate samples in a structured way. However, to control more properties and learn more semantic features of data in an unsupervised fashion, Information Maximizing Generative Adversarial Networks (infoGAN) [8] was developed. InfoGAN, as its name suggests, makes use of a mutual information loss function for training. It also employs control variables along with the noise as input to the generator to capture the important features of the data in a structured way. Similar to

GANs, InfoGAN takes into consideration the minimax game, but regularization of information is also included in its objective as follows:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V_I(\mathcal{G}, \mathcal{D}) = V(\mathcal{G}, \mathcal{D}) - \lambda I(c; \mathcal{G}(z, c)) \quad (2.3)$$

where the first term $V(\mathcal{G}, \mathcal{D})$ is calculated as the objective function of GAN, λ is a regularization parameter, and I is a mutual information to measure the information loss in the generator. Maximizing I means to make the latent code c have as much informative features of the original samples as possible. Since the input noise vector is decomposed into two parts of incompressible noise z and the latent code c in InfoGAN, the generator network becomes $\mathcal{G}(z, c)$.

A modular view of infoGAN is presented in Fig. 2.4. In Fig. 2.4, c shows the conditional vector and can be a continuous code that encodes features in InfoGAN. Considering variational mutual information maximization to maximize $I(c; \mathcal{G}(z, c))$, then the objective of InfoGAN is modified as follows:

$$\min_{\mathcal{G}, \mathcal{Q}} \max_{\mathcal{D}} V_I(\mathcal{G}, \mathcal{D}, \mathcal{Q}) = V(\mathcal{G}, \mathcal{D}) - \lambda L_I(\mathcal{G}, \mathcal{Q}) \quad (2.4)$$

where $\mathcal{Q}(c|x)$ is an auxiliary distribution defined to update the generator model to get better samples, and $L_I(\mathcal{G}, \mathcal{Q})$ is the lower bound of $I(c; \mathcal{G}(z, c))$.

2.2.5 Auxiliary Classifier Generative Adversarial Network (ACGAN)

ACGAN [58], similar to GAN, is a generative adversarial network, but it employs supervised learning. In this regard, ACGAN might be similar to CGAN, which is a conditional generative model that makes use of supervised learning. Figure 2.4 shows a modular view of ACGAN. In Fig. 2.4, c denotes the conditional vector and can be applied as a one-hot vector to encode the class labels in ACGAN. However, the discriminator \mathcal{D} of ACGAN not only distinguishes fake and real samples but also needs to predict the class label of these generated samples. Moreover, in ACGAN the generated samples along with real data are fed to \mathcal{D} , while in CGAN, the class labels are also supplied to the discriminator. Then, ACGAN takes

into account two losses of the adversarial play and classification loss. This is how ACGAN differs from CGAN. In addition, motivated by ACGAN, Twin Auxiliary Classifiers GAN (TAC-GAN) is proposed in [25] to solve the problem of low intra-class diversity of ACGAN. In TAC-GAN, an auxiliary classifier is added to the adversarial network of ACGAN, which supports the generator to learn an unbiased distribution and be able to generate diverse and realistic images.

2.2.6 Stacked Generative Adversarial Networks (StackGAN)

StackGAN [96] utilizes a hierarchical stack of conditional GANs to generate high-quality images from text descriptions. In StackGAN, the concept of Conditioning Augmentation has been incorporated along with \mathcal{G} and \mathcal{D} to provide more diversity along with the generated samples. It is a two-staged generative adversarial network, where the first level generator (Stage-I GAN) is conditioned on the text, yielding a low-resolution image, then, the second level generator (Stage-II GAN) is conditioned both on low-resolution images (i.e., produced in Stage-I GAN) and text embedding over to complete the needs to produce a high-resolution realistic image.

Moreover, StackGAN++ is another variant of StackGAN proposed in [97], where several generators and discriminators are devised in a tree-like structure to achieve a higher-resolution image. The StackGAN architecture is illustrated in a simplified modular view in Fig. 2.5.

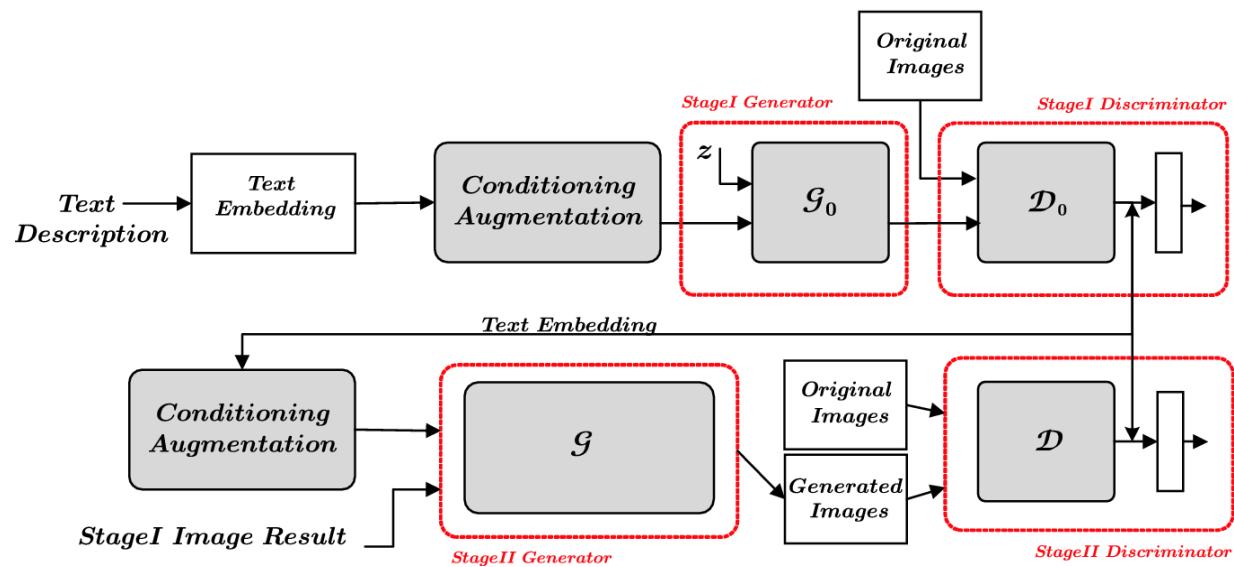


Fig. 2.5 Simplified view of StackGAN [96] including Conditioning Augmentation, Stage I and Stage II GAN. The first stage employs generative adversarial network and accepts the text description input to generate low-resolution images w.r.t. the given text. Stage II is constrained by Stage-I results. The low-resolution images are upgraded to a higher resolution image

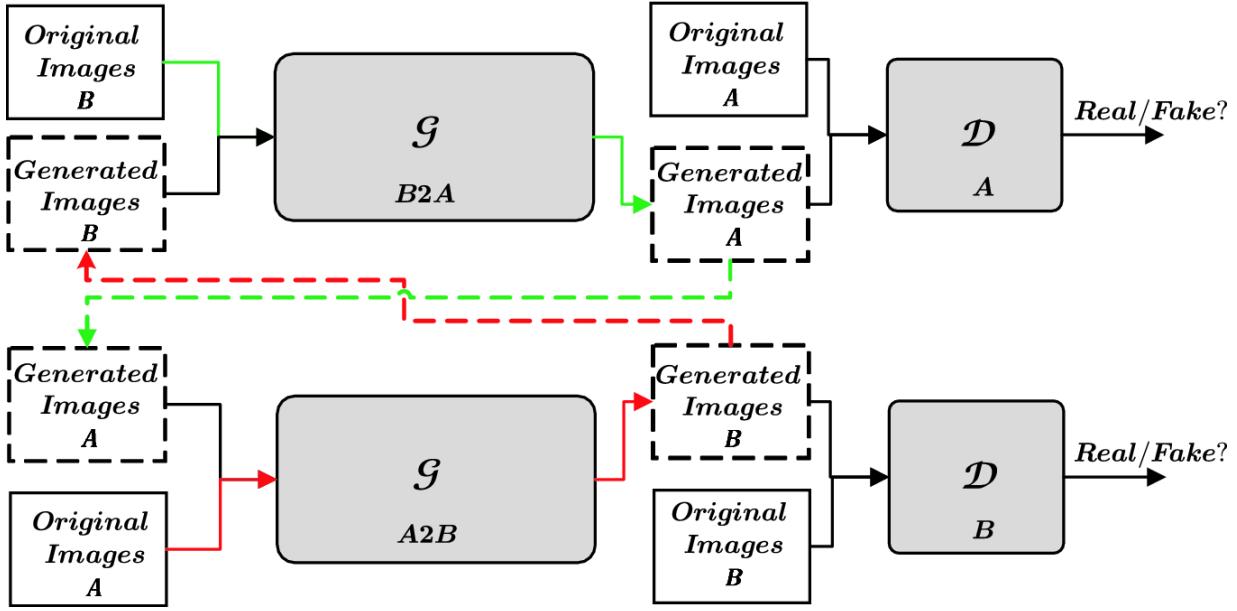


Fig. 2.6 Simplified view of CycleGAN [98] including two mapping functions of \mathcal{G}_{A2B} and \mathcal{G}_{B2A} , two discriminators of \mathcal{D}_A and \mathcal{D}_B . For the sake of better visualization, the cycle paths are colored in red and green w.r.t. translation from domain A to domain B, and vice versa

2.2.7 Cycle-Consistent Generative Adversarial Network (CycleGAN)

CycleGAN [98] is a generative network to convert an input image to a different version of that, where these two images are unpaired. The main goal of this translation is to learn a mapping from one domain to another domain in the absence of paired images. For the sake of “cycle consistency,” this translation is also reversible. In other words, given an input image A is converted to a target domain B, then image B can be used by CycleGAN to go back to the source domain, resulting in A. To this aim, two pairs of generators and discriminators are involved in CycleGAN; one for translation from the source domain to output, and another for conversion

back from the output to the source domain. The schematic view of CycleGAN is presented in Fig. 2.6.

Motivated by CycleGAN, a new generative model called Dual-GAN [92] was developed, which enables the image to image translations from two different domains in an unsupervised manner. Although CycleGAN and DualGAN are both trained for general-purpose image-to-image translations, their training procedures are different from each other. In addition, another generative model, named coupled GAN (CoGAN), that employs two GANs together to perform the unpaired image conversion is proposed in [48]. However, the adversarial networks of CoGAN learn a joint distribution over the unpaired images of source and target domains. In addition, they do not implement the cycle consistency as implemented in CycleGAN and DualGAN. Then, an improved version of CoGAN, which is involved with encoders, was also proposed in [47].

2.2.8 Wasserstein Generative Adversarial Network (WGAN)

Wasserstein Generative Adversarial Network (WGAN) [4] is an extension to GAN with an almost similar structure, where a generator and discriminator (i.e., referred as critic in WGAN) are combined in an adversarial network. Figure 2.4a shows a symbolic view of WGAN.

However, the training procedure of WGAN is different from GAN, since it makes use of Wasserstein distance to overcome the problem of mode collapse in GAN. The new Wasserstein loss function was defined, which has a smoother gradient. In addition, the critic may be updated several times compared to generator \mathcal{G} for each iteration. WGAN learns regardless of the performance of \mathcal{G} . However, there still exists some training instability in WGAN, which are discussed in [28]. Similar to other generative adversarial networks, WGAN has also gained the attention of several researchers to propose new WGAN-based methods. For instance, Wasserstein Generative Well-intentioned Network (GWIN) [11], GS-WGAN that is a gradient-sanitized approach for learning differentially private generators [7], and the Branch GAN [1] are some of these research works motivated by WGAN.

2.2.9 Semi-Supervised Generative Adversarial Network (SSGAN)

Semi-supervised learning makes use of a large amount of unlabelled data along with few labelled samples to train a predictive model. Motivated by the idea of semi-supervised learning and the high potential of generative adversarial networks to generate unlabeled images, different semi-supervised GANs were proposed. The schematic view of SSGAN is presented in Fig. 2.4. For instance, in [43] the discriminator turns into a multi-class semi-supervised classifier, which is trained on partially labelled data to distinguish fake samples and labels of real samples. To this aim, \mathcal{D} receives the unlabeled generated samples by \mathcal{G} to learn about fake samples, the unlabeled real samples to learn the samples originated from the real distribution of data, and the labelled real samples to know the decision boundaries w.r.t. the class labels. The proposed SSGAN in [43] uses feature matching GAN with the semi-supervised loss of \mathcal{D} , and, then, the objective function of the discriminator consists of two main terms of supervised loss over the labelled data and unsupervised loss of the samples with no labels.

In addition, the shortcomings of feature matching in semi-supervised GAN and why good semi-supervised classification performance and a good generator cannot be achieved together have been discussed in [12]. The idea of a complement generator to address the above issues is also examined, and, then, a new preferred generator for GAN-based semi-supervised learning is proposed [12].

2.2.10 Progressive Growing Generative Adversarial Network (Progressive GAN)

Progressive GAN [38] has modified the GAN structure and consequently the training methodology of GANs so that the training time is reduced, the variation of generated images is increased, and the stability of training to generate large high-quality images is sustained. The modification involves progressively expanding \mathcal{G} and \mathcal{D} depth in synchrony by adding layers step-wise during the training process. The progressive nature of training in the Progressive GAN enables the networks to first learn the large-scale structure of the image, and, then, the accurate scale details of the image are precisely learned rather than learning all scales of images at the same time together.

2.2.11 Style-Based Generative Adversarial Network (StyleGAN)

Style-Based Generative Adversarial Network (StyleGAN) [39] utilizes the idea of progressive training in Progressive GAN and having a major contribution on the generator structure to control the style of generated images. To achieve the scale-specific control, \mathcal{G} puts the input latent code into an intermediate latent space that is disentangled, and, then, the style over the generated images is regulated by mapping network and noise layers (i.e., two sources of randomness). It is also noticeable that the automated methods proposed in [39] are applicable to any generator architecture. For the sake of image quality, an improved version of StyleGAN is proposed in [40], where the model architecture and training methods are developed. In addition, the idea of StyleGAN grabbed the attention of the researchers to propose StyleRig [81], Rigging StyleGAN for 3D control over portrait images. StyleRig is a generative adversarial network that provides a face rig-like control, and also employs self-supervised training. Another work motivated by StyleGAN is PcapGAN [18], where the style vector is defined for the encoder in an adversarial network.

2.2.12 Bidirectional Generative Adversarial Network (BiGAN)

Bidirectional Generative Adversarial Network (BiGAN) [17] is an unsupervised feature learning framework able to learn the inverse mapping of input data. Figure 2.4f shows a simplified view of BiGAN. BiGAN makes no assumptions regarding the structure or type of data it is dealing with. It is a robust and generic method. BiGAN framework includes the generative adversarial networks of \mathcal{G} and \mathcal{D} , along with an encoder which is added to map the original data to latent space. In other words, the encoder learns to invert \mathcal{G} , while both encoder and generator aim to fool \mathcal{D} . Then, the discriminator should distinguish not only the generated samples by \mathcal{G} and the original samples, but also jointly in the data space (i.e., either original samples or generated samples) and the latent space (i.e., either an encoder output or z). A new variant of BiGAN referred as bidirectional conditional generative adversarial networks (BiCoGAN) was proposed in [36].

2.2.13 Bayesian Generative Adversarial Network (BGAN)

Bayesian GAN [70] is an advanced version of generative adversarial networks concerned with the Bayesian idea for unsupervised and semi-supervised learning. Bayesian GAN employs stochastic gradient Hamiltonian Monte Carlo to marginalize the weights of \mathcal{G} and \mathcal{D} . It can also handle the problem of mode-collapse in GAN.

In addition, the Bayesian learning technique is formulated along with the variational generative adversarial network to propose the bayesian VGAN in [9]. Also, the Bayesian CycleGAN is another advanced version of GAN motivated by the Bayesian learning technique [94].

Missing Data Imputation

Transfer learning

Feature Selection

Reinforcement learning

Dimensionality Reduction

Semi-supervised Learning

Classification

Class Imbalance learning

Task driven

Fig. 2.7 A general overview of task-driven applications of GANs

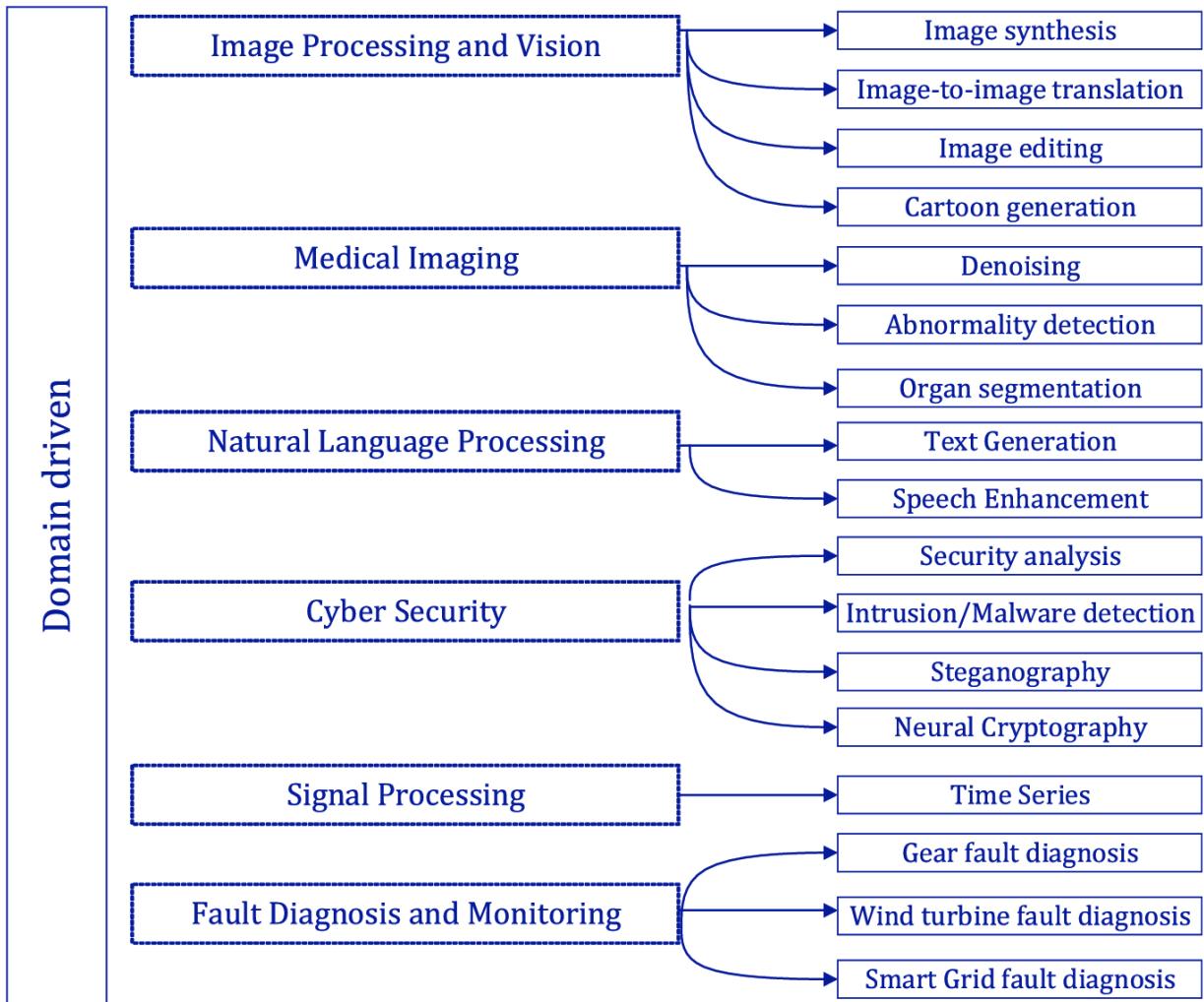


Fig. 2.8 A general overview of domain-driven applications of GANs

2.3 GAN Applications

GAN and its variants have a wide range of applications in numerous domains. Figures 2.7 and 2.8 show task-driven and domain-driven applications of GANs, respectively.

From a task-driven perspective illustrated in Fig. 2.7, GANs have been extensively used either for synthetic sample generation or for discriminative purposes. The challenge of class imbalance raises when the number of samples in a class of interest is less than the number of samples in other classes [66]. In the last decades, several class imbalance learning techniques have been proposed to tackle this inevitable practical problem [23, 65]. Recently, GANs have been employed by many researchers to produce

synthetic samples of the minority classes to solve a class imbalance problem [55, 60, 69]. GANs can be applied not only for classification (i.e., supervised learning), but also they have been employed in several semi-supervised cases. For instance, SSGAN [22, 43], CATGAN [78], SaliencyGAN [84] and TripleGAN [44] can be considered under semi-supervised GAN category. Recent applications of generative models for transfer learning [10, 92, 98] and reinforcement learning [27, 74] have also grabbed the attention of many researchers. In addition to applications of GANs for classification and semi-supervised learning, they have been employed for the sake of data processing. From the data processing perspective, accurate estimation of missing values of incomplete data is of paramount importance. Several researchers have been proposed missing data imputation techniques to complete categorical and numerical features [63]. Contemporary approaches for missing data imputation make use of GANs [29]. In [41], various missing data imputation techniques that make use of generative adversarial networks have been surveyed.

Table 2.1 List of recent GAN surveys on some specific domains

Title	Year	Reference
A Survey on Text Generation Using Generative Adversarial Networks	2021	[14]
Generative Adversarial Networks in Time Series: A Survey and Taxonomy	2021	[6]
Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy	2021	[86]
Creating Artificial Images for Radiology Applications Using Generative Adversarial Networks (GANs)—A Systematic Review	2020	[77]
A Survey of Missing Data Imputation Using Generative Adversarial Networks	2020	[41]
A State-of-the-Art Review on Image Synthesis With Generative Adversarial Networks	2020	[85]
Generative Adversarial Networks in Security: A Survey	2020	[20]
A Review of Generative Adversarial Networks and Its Application in Cybersecurity	2019	[93]
Generative Adversarial Network in Medical Imaging	2019	[91]

On the other hand, although learning from high-dimensional data has been challenging and numerous research works have been done in this domain [66], GANs provide efficient remedies for dimensionality reduction

[21] and feature selection [37]. For instance, in [32] a GAN-based diagnostic framework is proposed, which makes use of the most informative features and knockoffs for the sake of feature selection.

On the other hand, GANs have been applied on a wide range of datasets in several domains including but not limited to image processing and vision, medical imaging, natural language processing, security analysis, and fault diagnosis. A general overview of domain-driven applications of GANs is provided in Fig. 2.8. In vision and image processing applications, several GAN-based approaches have been proposed for image synthesis, image-to-image translation, image colorization, art generation, face age and cartoon generation [85]. As for natural language processing, [14] has surveyed the text generation using GAN. Moreover, speech enhancement through the generative adversarial network is also proposed in [61]. The application of GAN for signal processing includes signal reconstructions [59], electroencephalogram EEG biometrics learning [99], and spectrum sensing [13]. More specifically, generative adversarial networks in time series was reviewed in [6]. In the domain of cyber-security, GANs are mainly employed for intrusion detection, cyber attack identification, steganography, and neural cryptography [20, 73]. Although many research works have been performed for the sake of diagnosing faults and attacks and monitoring of cyber-physical systems using neural network-based techniques [31, 64, 67], more advanced methods for fault diagnosis and intrusion detection have been proposed in [22, 46], which make use of deep generative adversarial networks. Table 2.1 shows some of the most recent surveys on applications of generative adversarial networks in different domains including computer vision, radiology, cyber security, missing data imputation, and medical imaging.

2.4 Conclusions

This chapter presented a diverse survey of the state-of-the-art GAN-based techniques and their applications in different domains. Generative adversarial methods have been a hot topic and several GANs variants have been proposed by many researchers, mainly to improve the learning process of these methods. GANs and their variants not only have been employed to generate synthetic samples (e.g., class imbalance learning, dimensionality

reduction, and feature generation) but also have been applied for discriminative purposes (e.g., for semi-supervised learning and anomaly detection). Despite a wide range of GAN-based methods in image processing and vision, limited research works have been performed in other domains such as speech enhancement, text generation, signal processing, fault diagnosis and monitoring, and security analysis.

References

1. Adler, J., Lunz, S.: Banach wasserstein gan. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc., Red Hook (2018)
2. Alain, G., Bengio, Y., Yao, L., Yosinski, J., Thibodeau-Laufer, E., Zhang, S., Vincent, P.: Gsns: generative stochastic networks. Inf. Inference: J. IMA 5(2), 210–249 (2016). <https://doi.org/10.1093/imaiai/iaw003> [MathSciNet][Crossref][zbMATH]
3. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net (2017)
4. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 214–223. PMLR (2017)
5. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9, 2019. OpenReview.net (2019)
6. Brophy, E., Wang, Z., She, Q., Ward, T.: Generative adversarial networks in time series: a survey and taxonomy (2021). [arXiv:2107.11098](https://arxiv.org/abs/2107.11098)
7. Chen, D., Orekondy, T., Fritz, M.: Gs-wgan: a gradient-sanitized approach for learning differentially private generators. In: Larochelle, H., Ranzato, M., Hadsell,

- R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 12673–12684. Curran Associates, Inc., Red Hook (2020)
- 8. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: interpretable representation learning by information maximizing generative adversarial nets. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc., Red Hook (2016)
 - 9. Chien, J.T., Kuo, C.L.: Variational bayesian gan. In: 2019 27th European Signal Processing Conference (EUSIPCO), pp. 1–5 (2019). <https://doi.org/10.23919/EUSIPCO.2019.8903084>
 - 10. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: StarGAN: unified generative adversarial networks for multi-domain image-to-image translation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8789–8797 (2018). <https://doi.org/10.1109/CVPR.2018.00916>
 - 11. Cosentino, J., Zhu, J.: Generative well-intentioned networks. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc., Red Hook (2019)
 - 12. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R.: Good semi-supervised learning that requires a bad gan. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
 - 13. Davaslioglu, K., Sagduyu, Y.E.: Generative adversarial learning for spectrum sensing. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6 (2018). <https://doi.org/10.1109/ICC.2018.8422223>
 - 14. de Rosa, G.H., Papa, J.P.: A survey on text generation using generative adversarial networks. *Pattern Recogn.* **119**, 108098 (2021). <https://doi.org/10.1016/j.patcog.2021.108098>. <https://www.sciencedirect.com/science/article/pii/S0031320321002855>
 - 15. Denton, E., Chintala, S., Szlam, A., Fergus, R.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS’15, p. 1486–1494. MIT Press, Cambridge, MA, USA (2015)

16. Ding, X., Wang, Y., Xu, Z., Welch, W.J., Wang, Z.J.: CcGAN: continuous conditional generative adversarial networks for image generation. In: International Conference on Learning Representations (2021)
17. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. In: 5th International Conference on Learning Representations, ICLR (2017)
18. Dowoo, B., Jung, Y., Choi, C.: Pcapgan: packet capture file generator by style-based generative adversarial networks. In: 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1149–1154 (2019). <https://doi.org/10.1109/ICMLA.2019.00191>
19. Durall, R., Chatzimichailidis, A., Labus, P., Keuper, J.: Combating mode collapse in GAN training: an empirical analysis using hessian eigenvalues. In: VISIGRAPP (2021)
20. Dutta, I.K., Ghosh, B., Carlson, A., Totaro, M., Bayoumi, M.: Generative adversarial networks in security: a survey. In: 2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), pp. 0399–0405 (2020). <https://doi.org/10.1109/UEMCON51285.2020.9298135>
21. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M.: Generative adversarial dimensionality reduction for diagnosing faults and attacks in cyber-physical systems. Neurocomputing **440**, 101–110 (2021). <https://doi.org/10.1016/j.neucom.2021.01.076>
[Crossref]
22. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M., Parvania, M.: Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. IEEE Trans. Smart Grid, **12**(4), 3468–3478 (2021). <https://doi.org/10.1109/TSG.2021.3061395>
23. Farajzadeh-Zanjani, M., Razavi-Far, R., Saif, M.: Efficient sampling techniques for ensemble learning and diagnosing bearing defects under class imbalanced condition. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7 (2016). <https://doi.org/10.1109/SSCI.2016.7849879>
24. GM, H., Gourisaria, M.K., Pandey, M., Rautaray, S.S.: A comprehensive survey and analysis of generative models in machine learning. Computer Sci. Rev. **38**, 100285 (2020). <https://doi.org/10.1016/j.cosrev.2020.100285>. <https://www.sciencedirect.com/science/article/pii/S1574013720303853>
- 25.

- Gong, M., Xu, Y., Li, C., Zhang, K., Batmanghelich, K.: Twin auxilary classifiers gan. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 1328–1337. Curran Associates, Inc., Red Hook (2019)
26. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc., Red Hook (2014)
27. Guimaraes, G.L., Sanchez-Lengeling, B., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models (2017). [arXiv:1705.10843](https://arxiv.org/abs/1705.10843)
28. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein GANs. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
29. Hallaji, E., Razavi-Far, R., Palade, V., Saif, M.: Adversarial learning on incomplete and imbalanced medical data for robust survival prediction of liver transplant patients. IEEE Access **9**, 73641–73650 (2021). <https://doi.org/10.1109/ACCESS.2021.3081040>
[Crossref]
30. Hätkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: GANspace: discovering interpretable GAN controls. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020)
31. Hassani, H., Farajzadeh-Zanjani, M., Razavi-Far, R., Saif, M., Palade, V.: Design of a cost-effective deep convolutional neural network-based scheme for diagnosing faults in smart grids. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 1420–1425 (2019). <https://doi.org/10.1109/ICMLA.2019.00232>
32. Hassani, H., Razavi-Far, R., Saif, M., Palade, V.: Generative adversarial network-based scheme for diagnosing faults in cyber-physical power systems. Sensors **21**(15) (2021). <https://doi.org/10.3390/s21155173>
- 33.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)

34. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, pp. 448–456. JMLR.org (2015)
35. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5967–5976 (2017). <https://doi.org/10.1109/CVPR.2017.632>
36. Jaiswal, A., AbdAlmageed, W., Wu, Y., Natarajan, P.: Bidirectional conditional generative adversarial networks. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision - ACCV 2018, pp. 216–232. Springer International Publishing, Cham (2019)
[Crossref]
37. Jordon, J., Yoon, J., Schaar, M.: KnockoffGAN: generating knockoffs for feature selection using generative adversarial networks. In: ICLR (2019)
38. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
39. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4396–4405 (2019). <https://doi.org/10.1109/CVPR.2019.00453>
40. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8107–8116 (2020). <https://doi.org/10.1109/CVPR42600.2020.00813>
41. Kim, J., Tae, D., Seok, J.: A survey of missing data imputation using generative adversarial networks. In: 2020 International Conference on Artificial Intelligence

in Information and Communication (ICAIIC), pp. 454–456 (2020). <https://doi.org/10.1109/ICAIIC48513.2020.9065044>

42. Kong, J., Kim, J., Bae, J.: Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020)
43. Kumar, A., Sattigeri, P., Fletcher, T.: Semi-supervised learning with GANs: Manifold invariance with improved inference. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
44. LI, C., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
45. Lin, Z., Khetan, A., Fanti, G., Oh, S.: PacGAN: the power of two samples in generative adversarial networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18, pp. 1505–1514. Curran Associates Inc., Red Hook (2018)
46. Liu, J., Qu, F., Hong, X., Zhang, H.: A small-sample wind turbine fault detection method with synthetic fault data using generative adversarial nets. IEEE Trans. Ind. Inf. **15**(7), 3877–3888 (2019). <https://doi.org/10.1109/TII.2018.2885365> [Crossref]
47. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
48. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc., Red Hook (2016)
49. Luo, P., Zhang, R., Ren, J., Peng, Z., Li, J.: Switchable normalization for learning-to-normalize deep representation. IEEE Trans. Pattern Anal. Mach. Intell. **43**(2),

712–728 (2021). <https://doi.org/10.1109/TPAMI.2019.2932062>
[Crossref]

50. Ma, J., Yu, W., Liang, P., Li, C., Jiang, J.: FusionGAN: a generative adversarial network for infrared and visible image fusion. *Inf. Fusion* **48**, 11–26 (2019).
<https://doi.org/10.1016/j.inffus.2018.09.004>
[Crossref]
51. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2813–2821 (2017). <https://doi.org/10.1109/ICCV.2017.304>
52. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks (2017). <https://openreview.net/pdf?id=BydrOICle>
53. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). [arXiv: 1411.1784](https://arxiv.org/abs/1411.1784)
54. Mroueh, Y., Sercu, T., Goel, V.: McGan: mean and covariance feature matching GAN. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 2527–2535. PMLR (2017)
55. Mullick, S.S., Datta, S., Das, S.: Generative adversarial minority oversampling. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1695–1704 (2019)
56. Nagarajan, V., Kolter, J.Z.: Gradient descent gan optimization is locally stable. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
57. Nam, H., Kim, H.E.: Batch-instance normalization for adaptively style-invariant neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18, pp. 2563–2572. Curran Associates, Inc., Red Hook, NY, USA (2018)
58. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 2642–2651. PMLR (2017)

59. Oyamada, K., Kameoka, H., Kaneko, T., Tanaka, K., Hojo, N., Ando, H.: Generative adversarial network-based approach to signal reconstruction from magnitude spectrograms (2018). arXiv:Signal Processing
60. Pan, T., Chen, J., Xie, J., Zhou, Z., He, S.: Deep feature generating network: a new method for intelligent fault detection of mechanical systems under class imbalance. IEEE Trans. Ind. Inf. **17**(9), 6282–6293 (2021). <https://doi.org/10.1109/TII.2020.3030967>
[Crossref]
61. Pascual, S., Bonafonte, A., Serrà, J.: SEGAN: speech enhancement generative adversarial network (2017). [arXiv:1703.09452](https://arxiv.org/abs/1703.09452)
62. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings (2016)
63. Razavi-Far, R., Cheng, B., Saif, M., Ahmadi, M.: Similarity-learning information-fusion schemes for missing data imputation. Knowl.-Based Syst. **187**, 104805 (2020). <https://doi.org/10.1016/j.knosys.2019.06.013>
64. Razavi-Far, R., Davilu, H., Palade, V., Lucas, C.: Model-based fault detection and isolation of a steam generator using neuro-fuzzy networks. Neurocomputing **72**(13), 2939–2951 (2009). <https://doi.org/10.1016/j.neucom.2009.04.004>. Hybrid Learning Machines (HAIS 2007)/Recent Developments in Natural Computation (ICNC 2007)
65. Razavi-Far, R., Farajzadeh-Zanjani, M., Wang, B., Saif, M., Chakrabarti, S.: Imputation-based ensemble techniques for class imbalance learning. IEEE Trans. Knowl. Data Eng. **33**(5), 1988–2001 (2021). <https://doi.org/10.1109/TKDE.2019.2951556>
[Crossref]
66. Razavi-Far, R., Farajzadeh-Zanjani, M., Saif, M.: An integrated class-imbalanced learning scheme for diagnosing bearing defects in induction motors. IEEE Trans. Ind. Inf. **13**(6), 2758–2769 (2017). <https://doi.org/10.1109/TII.2017.2755064>
[Crossref]
67. Razavi-Far, R., Hallaji, E., Farajzadeh-Zanjani, M., Saif, M., Kia, S.H., Henao, H., Capolino, G.A.: Information fusion and semi-supervised deep learning scheme

- for diagnosing gear faults in induction machine systems. *IEEE Trans. Ind. Electr.* **66**(8), 6331–6342 (2019). <https://doi.org/10.1109/TIE.2018.2873546> [Crossref]
68. Roth, K., Lucchi, A., Nowozin, S., Hofmann, T.: Stabilizing training of generative adversarial networks through regularization. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., Red Hook (2017)
 69. Roy, S.K., Haut, J.M., Paoletti, M.E., Dubey, S.R., Plaza, A.: Generative adversarial minority oversampling for spectral-spatial hyperspectral image classification. *IEEE Trans. Geosci. Remote Sensing* 1–15 (2021). <https://doi.org/10.1109/TGRS.2021.3052048>
 70. Saatci, Y., Wilson, A.G.: Bayesian gan. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., Red Hook (2017)
 71. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training GANs. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., Red Hook (2016)
 72. Salimans, T., Kingma, D.P.: Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In: *NIPS’16*, p. 901–909. Curran Associates Inc., Red Hook, NY, USA (2016)
 73. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: protecting classifiers against adversarial attacks using generative models (2018). [arXiv:1805.06605](https://arxiv.org/abs/1805.06605)
 74. Sarmad, M., Lee, H.J., Kim, Y.: RL-GAn-net: a reinforcement learning agent controlled gan network for real-time point cloud shape completion. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5891–5900 (2019)
 75. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Netw.* **127**, 58–66 (2020). <https://doi.org/10.1016/j.neunet.2020.04.008> [Crossref][zbMATH]
 - 76.

- Shim, W., Cho, M.: Circlegan: generative adversarial learning across spherical circles. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 21081–21091. Curran Associates, Inc., Red Hook (2020)
77. Sorin, V., Barash, Y., Konen, E., Klang, E.: Creating artificial images for radiology applications using generative adversarial networks (gans) - a systematic review. *Acad. Radiol.* **27**(8), 1175–1185 (2020). <https://doi.org/10.1016/j.acra.2019.12.024>. <https://www.sciencedirect.com/science/article/pii/S1076633220300210>
78. Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016). [arXiv:1511.06390](https://arxiv.org/abs/1511.06390)
79. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: Veegan: Reducing mode collapse in GANs using implicit variational learning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017)
80. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016). <https://doi.org/10.1109/CVPR.2016.308>
81. Tewari, A., Elgharib, M., Bharaj, G., Bernard, F., Seidel, H.P., Pérez, P., Zollhöfer, M., Theobalt, C.: Stylerig: rigging styleGAN for 3d control over portrait images. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6141–6150 (2020). <https://doi.org/10.1109/CVPR42600.2020.00618>
82. Tolstikhin, I.O., Gelly, S., Bousquet, O., SIMON-GABRIEL, C.J., Schölkopf, B.: Adagan: boosting generative models. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Red Hook (2017). <https://proceedings.neurips.cc/paper/2017/file/d0010a6f34908640a4a6da2389772a78-Paper.pdf>
83. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: the missing ingredient for fast stylization (2016). [arXiv:1607.08022](https://arxiv.org/abs/1607.08022)
- 84.

- Wang, C., Dong, S., Zhao, X., Papanastasiou, G., Zhang, H., Yang, G.: SaliencyGAN: deep learning semisupervised salient object detection in the fog of iot. *IEEE Trans. Ind. Inf.* **16**(4), 2667–2676 (2020). <https://doi.org/10.1109/TII.2019.2945362>
[Crossref]
85. Wang, L., Chen, W., Yang, W., Bi, F., Yu, F.R.: A state-of-the-art review on image synthesis with generative adversarial networks. *IEEE Access* **8**, 63514–63537 (2020). <https://doi.org/10.1109/ACCESS.2020.2982224>
[Crossref]
86. Wang, Z., She, Q., Ward, T.E.: Generative adversarial networks in computer vision: a survey and taxonomy. *ACM Comput. Surv.* **54**(2) (2021). <https://doi.org/10.1145/3439723>
87. Wu, Y., He, K.: Group normalization. *Int. J. Computer Vis.* **128**, 742–755 (2020). <https://doi.org/10.1007/s11263-019-01198-w>
[Crossref]
88. Wu, Y., Zhou, P., Wilson, A.G., Xing, E., Hu, Z.: Improving gan training with probability ratio clipping and sample reweighting. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 5729–5740. Curran Associates, Inc., Red Hook (2020)
89. Xiaopeng, C., Jiangzhong, C., Yuqin, L., Qingyun, D.: Improved training of spectral normalization generative adversarial networks. In: 2020 2nd World Symposium on Artificial Intelligence (WSAI), pp. 24–28 (2020). <https://doi.org/10.1109/WSAI49636.2020.9143310>
90. Xu, J., Sun, X., Zhang, Z., Zhao, G., Lin, J.: Understanding and improving layer normalization. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., Red Hook (2019)
91. Yi, X., Walia, E., Babyn, P.: Generative adversarial network in medical imaging: a review. *Med. Image Anal.* **58**, 101552 (2019). <https://doi.org/10.1016/j.media.2019.101552>
92. Yi, Z., Zhang, H., Tan, P., Gong, M.: DualGAN: Unsupervised dual learning for image-to-image translation. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2868–2876 (2017). <https://doi.org/10.1109/ICCV.2017.310>

93. Yinka-Banjo, C., Ugot, O.A.: A review of generative adversarial networks and its application in cybersecurity. *Artif. Intell. Rev.* **53**, 1721–1736 (2019). [\[Crossref\]](#)
94. You, H., Cheng, Y., Cheng, T., Li, C.L., Zhou, P.: Bayesian cycleGAN via marginalizing latent sampling (2018). [arXiv:1811.07465](#)
95. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 97, pp. 7354–7363. PMLR (2019)
96. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.: Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5908–5916 (2017). <https://doi.org/10.1109/ICCV.2017.629>
97. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan++: realistic image synthesis with stacked generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **41**(8), 1947–1962 (2019). <https://doi.org/10.1109/TPAMI.2018.2856256>
[\[Crossref\]](#)
98. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2242–2251 (2017). <https://doi.org/10.1109/ICCV.2017.244>
99. Ödenizci, O., Wang, Y., Koike-Akino, T., Erdoğmuş, D.: Adversarial deep learning in eeg biometrics. *IEEE Signal Proc. Lett.* **26**(5), 710–714 (2019). <https://doi.org/10.1109/LSP.2019.2906826>
[\[Crossref\]](#)

3. Fair Data Generation and Machine Learning Through Generative Adversarial Networks

Xintao Wu¹✉, Depeng Xu¹✉, Shuhan Yuan²✉ and Lu Zhang¹✉
(1) University of Arkansas, Fayetteville, AR 72701, USA
(2) Utah State University, Logan, UT 84332, USA

✉ Xintao Wu (Corresponding author)
Email: xintaowu@uark.edu

✉ Depeng Xu
Email: depengxu@uark.edu

✉ Shuhan Yuan
Email: shuhan.yuan@usu.edu

✉ Lu Zhang
Email: lz006@uark.edu

Abstract

In this chapter, we present a fair generative adversarial networks framework (named FairGAN) for fair data generation and fair predictive learning. The FairGAN framework can accommodate various fairness notions by changing the network architecture and objective functions of generators and discriminators. Under the FairGAN framework, we present three previously published model designs, Simplified-FairGAN [1], Causal-FairGAN [2], and FairGAN+ [3], discuss their designs, fairness, utility, convergence, and implementation. We then present a short literature review of closely related works of using GANs for structural data generation and achieving differential privacy in GAN. At the end, we introduce several future research

directions including architecture design, achieving long-term fairness in dynamic decision making, achieving fairness in regression and recommendation, and open source development. This paper expects to advance understanding and applicability of GAN from image data generation to fair data generation and fair predictive learning.

3.1 Introduction

Fair machine learning is receiving an increasing attention in machine learning fields to fight against discrimination. Discrimination refers to unjustified distinctions in decisions against individuals based on their membership in a certain demographic group. Decision making like college admission, employment, loan application, and insurance could incur discrimination. Research has been conducted on discrimination discovery to analyze the historical training data for unveiling historical discriminatory practices, and on discrimination removal to remove discrimination from data and predictive model to achieve fairness. In the literature, many methods have been proposed to modify the training data for mitigating biases and achieving fairness. These methods include: massaging [4], reweighting [5], sampling [6], disparate impact removal [7], causal-based removal [8–11] and fair representation learning [12–15]. Some research work proposed to mitigate discriminatory bias in training data by modifying labels of training data [4, 16] or even whole records [6, 7]. Some other work proposed to mitigate discriminatory bias in predictions by adjusting the learning process [17–19] or changing predicted labels [20, 21]. Refer to some surveys, e.g., [22], for details.

Nevertheless, methods of modifying the training data are limited by the amount of available data at hand. On the other hand, instead of modifying existing datasets, one can generate high quality synthetic data. In this paper, based on generative adversarial networks, we present a unified framework, named FairGAN, for generating fair data that have good utility and also producing fair predictive models.

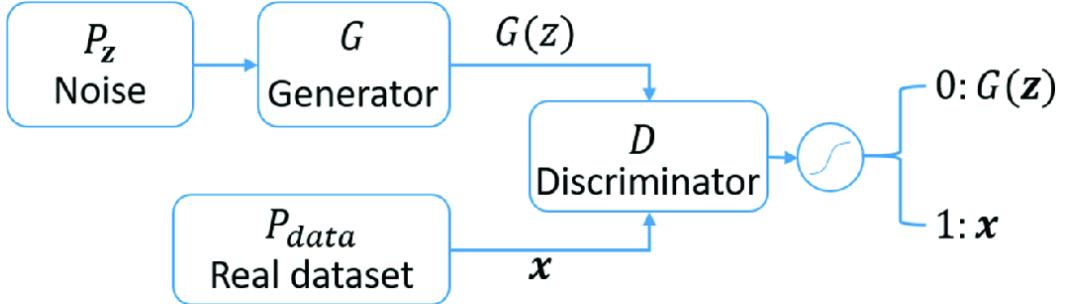


Fig. 3.1 The structure of GAN

Generative adversarial networks (GAN) [23] uses an implicit model that samples directly from the distribution represented by the model, and estimates the real data distribution through an adversarial game. Due to its huge success in image generation, GAN was also extended to structural data generation (e.g., medGAN [24] and tableGAN [25]), text generation (e.g., MaskGAN [26] and SeqGAN [27]), and graph generation based on embedding [28]. GAN consists of two components: a generator G and a discriminator D , both of which are multilayer neural networks. Figure 3.1 shows the structure of the GAN model. Given random noise variable $\mathbf{z} \sim P(\mathbf{Z})$ as input, the generator $G(\mathbf{z})$ attempts to learn a generative distribution P_G to match the real data distribution P_{data} . Meanwhile, the discriminator D is a binary classifier to predict whether a data sample is from real data distribution or the fake data generated by the generator. By playing the adversarial game, GAN is formalized as

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim P_{data}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{Z}}} [\log(1 - D(G(\mathbf{z})))].$$

Compared with other generative models, GAN has several advantages such as parallel execution, few restrictions of generator function design, asymptotically consistent, and free from the computationally expensive Markov chains [29], thus achieving huge success in image/video/text generation tasks.

The FairGAN framework also consists two components: generators for generating synthetic data, and discriminators for posing fairness and utility constraints. Within the framework, we can develop specific model designs by modifying the network architecture and objective functions to meet different fairness requirements, including statistical parity, causal fairness, fairness in classification. Under the FairGAN framework, we present three previously published FairGAN models, Simplified-FairGAN [1] that can

achieve a simple correlation-based fairness criterion in generated data, Causal-FairGAN [2] for generating data that achieve various causality-based fairness criteria, and FairGAN+ [3] that can simultaneously achieve fair data generation and fair classification. We then present a short literature review of closely related works of using GANs for structural data generation and achieving differential privacy in GAN. At the end, we present future research directions, e.g., how to achieve long-term fairness, fairness in regression, and fairness in recommendation under the FairGAN framework, and conclude this chapter.

3.2 Overview of FairGAN Framework

In this section, we first introduce definitions and metrics of fairness and then present the general framework for fairness aware generative adversarial networks.

3.2.1 Definitions and Metrics of Fairness

Assume that we have a dataset $\mathcal{D} = \{\mathbf{X}, Y, S\} \sim P_{\text{data}}$, where S denotes the protected attribute (e.g., gender, race, etc.), Y denotes the decision (e.g., hiring, loan approval, etc.), and \mathbb{E} denotes a set of non-protected attributes. A predictive model $\eta : \mathbf{X} \rightarrow Y$ is trained on the dataset to output prediction $p_z(z)$ of Y from \mathbb{E} . Roughly speaking, fairness is achieved if decision outcomes are independent of given variables, especially those sensitive ones such as the traits of individuals.

Various notions have been proposed for quantifying fairness for the dataset and the predictive model. Correlation or association-based notions are among the first endeavor to define fairness in the dataset, including statistical parity, conditional statistical parity, and ϵ -fairness. **Statistical parity** is defined as: $P(Y = 1|S = 1) = P(Y = 1|S = 0)$ which requires that the label is independent of the protected attribute. **Conditional statistical parity** is similarly defined as

$P(Y = 1|S = 1, X) = P(Y = 1|S = 0, X)$ given a third attribute X . The ϵ -fairness quantifies the fairness of data through the error rate of predicting the protected attribute λ given the unprotected attributes \mathcal{X} . If the error rate is low, it means λ is predictable by \mathcal{X} . Formally, a dataset $\mathcal{D} = (\mathbf{X}, Y, S)$ is said to be ϵ -fair if for any classification algorithm $f : \mathbf{X} \rightarrow S$,

$BER(f(\mathbf{X}), S) > \epsilon$, with empirical probabilities estimated from \mathcal{D} ,

where BER (balanced error rate) is defined as

$BER(f(\mathbf{X}), S) = \frac{1}{2}(P[f(\mathbf{X}) = 0|S = 1] + P[f(\mathbf{X}) = 1|S = 0])$. BER indicates the average class-conditioned error of f on distribution \mathcal{D} over the pair (\mathbf{X}, S) .

The statistical parity based notions can be easily extended to the predictive model by replacing Y with the predicted label $p_z(z)$. For example, the classifier achieves statistical parity if

$P(\eta(\mathbf{X}) = 1|S = 1) = P(\eta(\mathbf{X}) = 1|S = 0)$. In addition, fairness notions that take classification performance into consideration have also been proposed including equality of odds and equality of opportunity. **Equality of odds** is defined as

$P(\eta(\mathbf{X}) = 1|Y = y, S = 1) = P(\eta(\mathbf{X}) = 1|Y = y, S = 0)$, where $y \in \{0, 1\}$. It means that, for $Y = 1$, the classifier η has equal true positive rates (TPR) between two subgroups $S = 1$ and $S = 0$, and for $Y = 0$, the classifier η has equal false positive rates (FPR) between two subgroups. In many binary classification cases, $Y = 1$ is a more important outcome. With only requiring non-discrimination on the specific outcome group, the equality of odds can be relaxed to the **equality of opportunity**, $P(\eta(\mathbf{X}) = 1|Y = 1, S = 1) = P(\eta(\mathbf{X}) = 1|Y = 1, S = 0)$. It indicates that individuals who qualify for a desirable outcome should have an equal chance of being correctly classified for this outcome.

When treating discrimination or fairness as a causal relation between the decision and the protected attribute, causality-based fairness notions based on Pearl's structural causal model [30] have been proposed, including total effect, direct discrimination, indirect discrimination, and counterfactual fairness, defined based on various types of causal effects. The causal effect of one variable X on another variable Y is inferred by comparing the difference in interventional distributions under different interventions with the help of causal modeling and the *do*-operator. Based on how the intervention is transferred in the causal model (graph), there are mainly three types of causal effects: total causal effect, path-specific effect, counterfactual effect [30]. The total causal effect measures the causal effect of S on Y where the intervention is transferred along all causal paths from S to Y . As a causal fairness notion based on that, there is no **total effect** if $BER(f(\mathbf{X}), S) > \epsilon$

where $V(\mathcal{G}, \mathcal{D})$ represents the interventional distribution of Y when forcing variable S to take certain value s . The path-specific effect measures the causal effect of S on Y where the intervention is transferred only along a subset of causal paths from S to Y , which is also referred to as the π -specific effect denoting the subset of causal paths as π . Given a path set π , the π -specific effect of the value change of S from 0 to 1 on Y (with reference $S = 1$) is given by $P(Y_{S=1|\pi}) - P(Y_{S=0|\pi})$, where $P(Y_{S=s|\pi})$ represents the interventional distribution where the intervention is transferred only along π .

There is no **direct discrimination** in the data if $P(Y_{S=1|\pi_d}) = P(Y_{S=0|\pi_d})$, where π_d is the path set that only contains the direct edge from S to Y , i.e., $S \rightarrow Y$. Given a subset of attributes $\mathbf{R} \subseteq \mathbf{X}$ that cannot be objectively justified in decision making, there is no **indirect discrimination** in the data if $P(Y_{S=1|\pi_i}) = P(Y_{S=0|\pi_i})$, where π_i is the set of causal paths from S to Y that pass through \mathbb{E} . In the total effect and path-specific effect, the intervention is performed on the whole population. The counterfactual effect measures the causal effect while the intervention is performed conditioning on only certain individuals or groups specified by a subset of observed variables $\mathbf{O} = \mathbf{o}$. Given a context $\mathbf{O} = \mathbf{o}$, the counterfactual effect of the value change of S from 0 to 1 on Y is given by $P(Y_{S=1|\mathbf{o}}) - P(Y_{S=0|\mathbf{o}})$. **Counterfactual fairness** is achieved in the data if $P(Y_{S=1|\mathbf{o}}) = P(Y_{S=0|\mathbf{o}})$ under any context $\mathbf{O} = \mathbf{o}$.

3.2.2 FairGAN Framework

We present the framework of fairness aware generative adversarial networks (FairGAN) for generating continuous and discrete data with fair constraints. The FairGAN framework consists of generators and discriminators. To generate continuous and discrete data, the generator G is composed by two components, a regular generator that is trained to generate the salient representations, and a decoder that seeks to construct the synthetic data from the salient representations. The decoder, pretrained from an autoencoder, is able to generate both types of data from the output of the regular generator. A basic autoencoder contains an encoder and a decoder. The objective function of the autoencoder is to make the reconstructed input \mathbf{x}' close to the original input \mathbf{x} .

To ensure the generated data to meet fair requirements, we introduce constraint terms to the objective functions of generator and discriminator and/or add new generators or discriminators. Symbolically, FairGAN consists of two components: generator(s) G for generating synthetic data, and discriminator(s) D for posing fairness and utility constraints. The objective is to generate a dataset $\hat{\mathcal{D}} = \{\hat{\mathbf{X}}, \hat{Y}, \hat{S}\} \sim P_G$ such that $\mathcal{D} \approx \hat{\mathcal{D}}$ and $\hat{\mathcal{D}}$ achieves a variety of fairness notions. Meanwhile, we also want to make predicted decisions $p_z(z)$ be fair by co-training η with generators and discriminators.

3.3 Simplified Fairness Aware Generative Adversarial Networks

In this section, we first introduce the work of using GAN to achieve a simple statistical-based fairness criterion in generated data but no guarantee on classifier.

3.3.1 Model Framework Design

In [1], Xu et al. developed the Simplified-FairGAN model that achieves fair data generation in terms of disparate treatment and disparate impact [7]. Disparate treatment indicates the discriminatory outcomes are due to explicitly using the protected attribute to make decisions. Disparate impact indicates the discriminatory outcomes are not explicitly from the protected attribute but from the proxy unprotected attributes. To handle both disparate treatment and disparate impact, the notion of statistical parity is used to measure the potential discrimination caused by the correlation between class label Y and protected attribute S and requires

$P(Y = 1|S = 1) = P(Y = 1|S = 0)$. Simplified-FairGAN consists of one generator G and two discriminators D_1 and D_2 as shown in Fig. 3.2. The generator generates fake samples $\{\hat{\mathbf{X}}, \hat{Y}\}$ conditioned on the protected attribute S . One discriminator aims to ensure the generated data $\{\hat{\mathbf{X}}, \hat{Y}, \hat{S}\}$ close to the real data $\{\mathbf{X}, Y, S\}$ while the other discriminator aims to ensure there are no correlation between $\hat{\mathbf{X}}$ and \hat{S} and no correlation between \hat{Y} and \hat{S} .

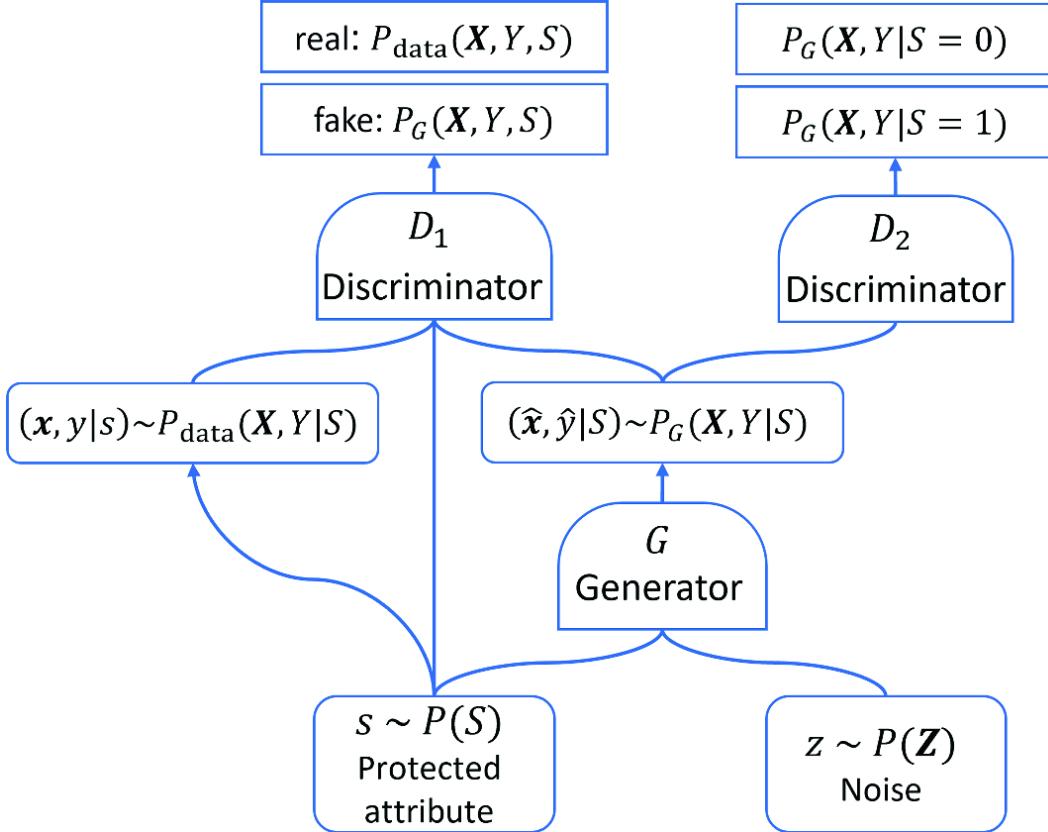


Fig. 3.2 The structure of Simplified-FairGAN [1]

More specifically, in Simplified-FairGAN, every generated sample has a corresponding value of the protected attribute $s \sim P_{\text{data}}(s)$. Generator G generates a fake pair (\hat{x}, \hat{y}) following the conditional distribution $P_G(x, y|s)$. The fake pair (\hat{x}, \hat{y}) is generated by a noise variable ϵ and the protected attribute s , namely, $(\hat{x}, \hat{y}) = G(z, s)$, $z \sim P_z(z)$, where $P_z(z)$ is a prior distribution. Hence, the generated fake sample $(\hat{x}, \hat{y}, \hat{s})$ is from the joint distribution $P_G(x, y, s) = P_G(x, y|s)P_G(s)$, where $P_G(s) = P_{\text{data}}(s)$. Discriminator D_1 is trained to distinguish between the real data from $P_{\text{data}}(x, y, s)$ and the generated fake data from $P_G(x, y, s)$. Its minimax function is

$$V_1(G, D_1) = \mathbb{E}_{s \sim P_{\text{data}}(s), (x, y) \sim P_{\text{data}}(x, y|s)} [\log D_1(x, y, s)] + \mathbb{E}_{\hat{s} \sim P_G(s), (\hat{x}, \hat{y}) \sim P_G(x, y|s)} [\log(1 - D_1(\hat{x}, \hat{y}, \hat{s}))]$$

To achieve fairness, another discriminator D_1 with the minimax function $V_2(G, D_2) = \mathbb{E}_{(\hat{x}, \hat{y}) \sim P_G(x, y|s=1)} [\log D_2(\hat{x}, \hat{y})] + \mathbb{E}_{(\hat{x}, \hat{y}) \sim P_G(x, y|s=0)} [\log(1 - D_2(\hat{x}, \hat{y}))]$ is added to distinguish the two categories of generated samples,

$P_G(\mathbf{x}, y|s = 1)$ and $P_G(\mathbf{x}, y|s = 0)$. The value function of the minimax game is formulated as

$\min_G \max_{D_1, D_2} V(G, D_1, D_2) = V_1(G, D_1) + \lambda V_2(G, D_2)$, where λ is a hyperparameter that specifies a trade off between utility and fairness of data generation. Discriminator D_1 aims to ensure

$P_G(\mathbf{x}, y|s = 1) = P_G(\mathbf{x}, y|s = 0)$, making the generated samples not encode any information supporting to predict the value of protected attribute s . Once the generated sample $\mathcal{G}(z, c)$ cannot be used to predict the protected attribute \hat{s} (s), the correlation between $\mathcal{G}(z, c)$ and s is removed, i.e., $\{\hat{\mathbf{x}}, \hat{y}\} \perp\!\!\!\perp s$. Hence, the generated data is free from disparate treatment and disparate impact.

3.3.2 Empirical Evaluation

Xu et al. [1] evaluated the effectiveness of Simplified-FairGAN and compared its performance with the regular GAN model from two perspectives, fairness and utility. Note that the regular GAN aims to generate the synthetic samples that have the same distribution as the real data, i.e., $P_G(\mathbf{x}, y, s) = P_{\text{data}}(\mathbf{x}, y, s)$. The regular GAN model cannot achieve fair data generation when the input real data contains discrimination. In evaluation, the autoencoder was pretrained for 200 epochs. Both the encoder and the decoder have one hidden layer with the dimension size as 128. The generator is a feedforward neural network with two hidden layers, each having 128 dimensions. The discriminator is also a feedforward network with two hidden layers where the first layer has 256 dimensions and the second layer has 128 dimensions. Simplified-FairGAN was first trained without the fair constraint for 2,000 epochs (D_1 and G) and then trained with the fair constraint for another 2,000 epochs (D_1 , D_1 and G).

The UCI Adult income dataset [31] was used in the evaluation. The Adult dataset contains 48,842 instances. The decision indicates whether the income is higher than \$50 k per year, and the protected attribute is gender. Each instance in the dataset consists of 14 attributes and each attribute is converted to a one-hot vector. The combined feature vector has 57 dimensions. The risk difference (

$RD(\mathcal{D}) = P(y = 1|s = 1) - P(y = 1|s = 0)$) in the Adult dataset is 0.1989, which indicates discrimination against female. The dataset generated

from GAN has the similar risk difference to the real dataset. On the contrary, the risk difference of data generated by Simplified-FairGAN is 0.0411, which is below the fairness threshold 0.05. In addition, the ϵ -fairness (disparate impact) is evaluated by calculating the balanced error rate (BER). The BER in the original real dataset is 0.1538, which means a classifier can predict s given x with high accuracy. Hence, there is a high disparate impact in the real dataset. On the contrary, the BER in the data generated by Simplified-FairGAN is 0.3862, which indicates the small disparate impact. Moreover, they evaluated the data utility of generated datasets by calculating the Euclidean distance of joint and conditional probabilities ($P(x, y)$, $P(x, y, s)$, and $P(x, y|s)$) and showed Simplified-FairGAN can preserve good utility.

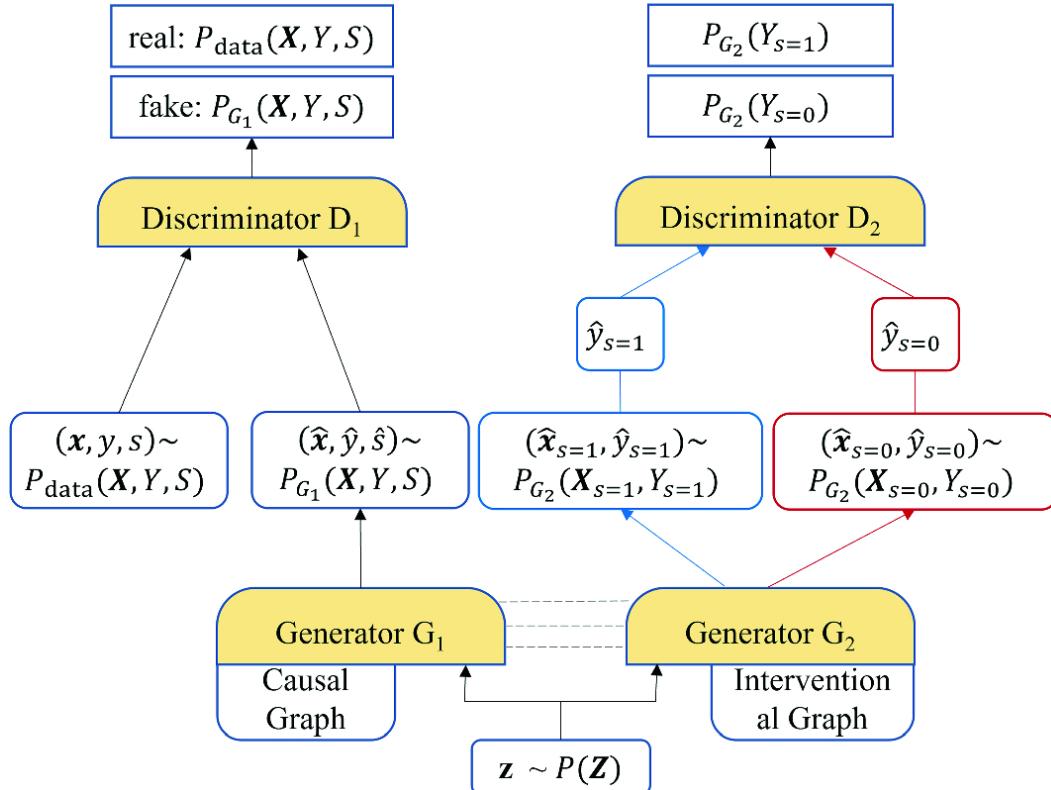


Fig. 3.3 The structure of Causal-FairGAN

3.4 Achieving Causal Fairness in Data Generation

Simplified-FairGAN can only achieve fairness in terms of some simple correlation-based fairness criterion. However, the causal blindness of

Simplified-FairGAN makes it unable to achieve causality-based fairness notions which capture and quantify fairness as causal effects, differentiate them from spurious effects, measure fairness along specific causal paths, and also estimate the counterfactual effect. In [2], Xu et al. developed Causal-FairGAN for generating data that can achieve various causal-based fairness criteria based on total effect [32], direct/indirect discrimination [8], and counterfactual fairness [11], given a causal graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V} = \{\mathbf{X}, Y, S\}$.

3.4.1 Model Framework Design

Figure 3.3 shows the framework of Causal-FairGAN. In designing Causal-FairGAN, the first challenge is how to preserve the causal relationship among all attributes in the generator. Causal-FairGAN [2] leverages CausalGAN [33] to reflect the causal structure by arranging the network structure of the generator following the given causal graph. The generator plays the role of the causal model that agrees with this causal graph in terms of both the graph structure and conditional distributions. The idea is to partition the generator into $|\mathbf{V}|$ sub-neural networks such that each of them generates the value for one attribute in \mathbb{E} . Then, if a node is a parent of another node in the causal graph, the output of the sub-neural network corresponding to the first node is designed as the input of the sub-neural network corresponding to the second node. As a result, the generator can be used to simulate the real causal model.

Another challenge is how to generate and compare the original distribution and interventional distribution simultaneously. This is because, on one hand we need to generate close-to-real data, and on the other hand we also need to satisfy various causal-based fairness criteria that generally compare the interventional distributions of Y under two different interventions $do(S = 1)$ and $do(S = 0)$. To address this challenge, two generators are adopted in Causal-FairGAN. One generator G_1 plays the role of causal model similar to CausalGAN and aims to generate observational data whose distribution is close to the real observational distribution, while the other generator G_2 explicitly plays the roles of different interventional models based on the type of causal effects and aims to generate interventional data that satisfy the criterion. The two generators share the

input noises and parameters to reflect the connections between the two causal models, and differ in connections of sub-neural networks to reflect the intervention. As a result, G_1 generates samples $(\hat{\mathbf{x}}, \hat{y}, \hat{s}) \sim P_{G_1}(\mathbf{X}, Y, S)$, and G_1 generates samples $(\hat{\mathbf{x}}_{S=1}, \hat{y}_{S=1}) \sim P_{G_2}(\mathbf{X}_{S=1}, Y_{S=1})$, if $S = 1$, $(\hat{\mathbf{x}}_{S=1}, \hat{y}_{S=1}) \sim P_{G_2}(\mathbf{X}_{S=1}, Y_{S=1})$, if $S = 1$.

Causal-FairGAN also adopts two discriminators. One discriminator D_1 is designed to distinguish between the real observational data

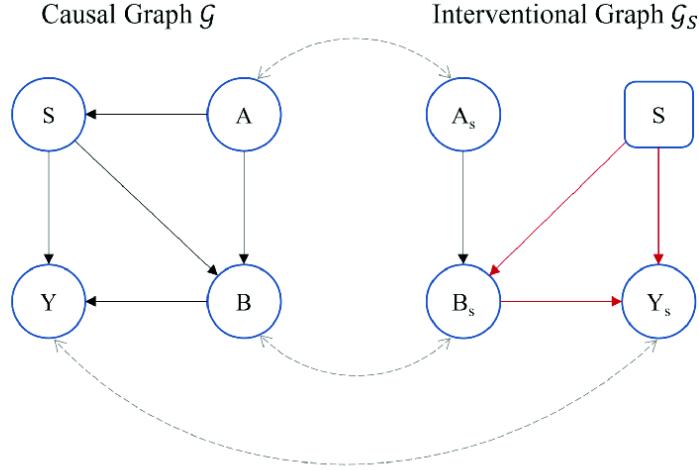
$(\mathbf{x}, y, s) \sim P_{\text{data}}(\mathbf{X}, Y, S)$ and the generated fake observational data $(\hat{\mathbf{x}}, \hat{y}, \hat{s}) \sim P_{G_1}(\mathbf{X}, Y, S)$. The other discriminator D_1 tries to distinguish the two intervention distributions under $do(S = 1)$ and $do(S = 1)$, i.e., $\hat{y}_{S=1} \sim P_{G_2}(Y_{S=1})$ and $\hat{y}_{S=0} \sim P_{G_2}(Y_{S=0})$. Putting the generators and discriminators together, generator G_1 plays the adversarial game with the discriminator D_1 with the minimax function

$$J_1(G_1, D_1) = \mathbb{E}_{(\mathbf{x}, y, s) \sim P_{\text{data}}(\mathbf{X}, Y, S)}[\log D_1(\mathbf{x}, y, s)] + \mathbb{E}_{(\hat{\mathbf{x}}, \hat{y}, \hat{s}) \sim P_{G_1}(\mathbf{X}, Y, S)}[1 - \log D_1(\hat{\mathbf{x}}, \hat{y}, \hat{s})]$$

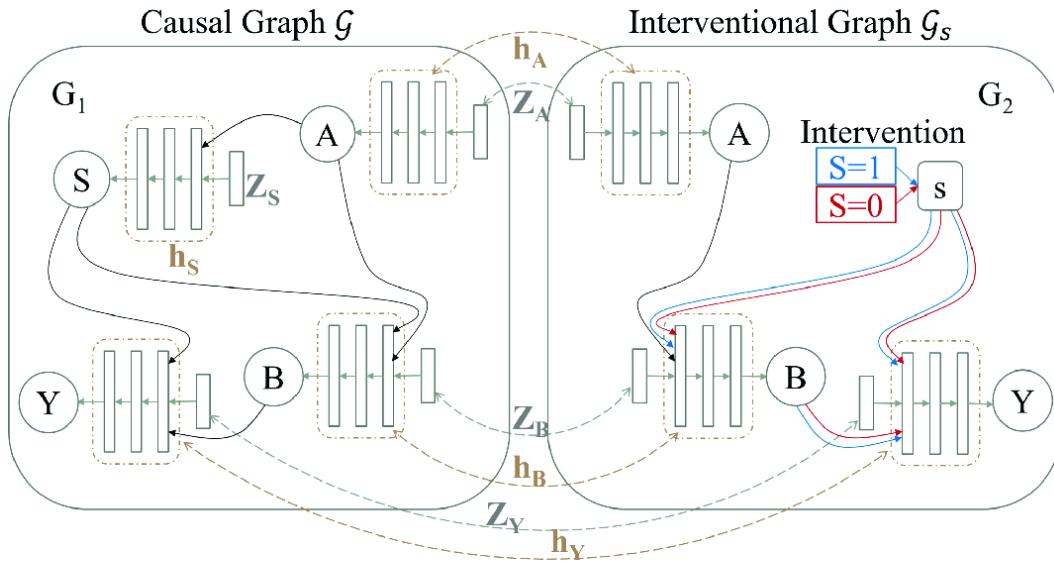
, and generator G_1 plays the adversarial game with the discriminator D_1 with the minimax function

$$J_2(G_2, D_2) = \mathbb{E}_{\hat{y}_{S=1} \sim P_{G_2}(Y_{S=1})}[\log D_2(\hat{y}_{S=1})] + \mathbb{E}_{\hat{y}_{S=0} \sim P_{G_2}(Y_{S=0})}[1 - \log D_2(\hat{y}_{S=0})]$$

. Here J_1 aims to achieve $P_{G_1}(\mathbf{X}, Y, S) = P_{\text{data}}(\mathbf{X}, Y, S)$, i.e., to make the generated observational data indistinguishable from the real data whereas J_1 aims to make the generated interventional data satisfy the fairness criterion, e.g., $P_{G_2}(Y_{S=1}) = P_{G_2}(Y_{S=0})$ in terms of total effect. As G_1 and G_1 share the same sets of parameters, the data generated by G_1 will be close to the data and also will satisfy the fairness criterion.



(a) Causal graph \mathcal{G} and interventional graph \mathcal{G}_s



(b) Generators G_1 and G_2

Fig. 3.4 An example of the generators G_1 and G_2 for Causal-FairGAN based on total effect

Figure 3.4 shows an illustrative example which involves 4 variables $\{\mathbf{X} = \{A, B\}, Y, S\}$ and focuses causal fairness in terms of total effect. In the causal graph and the interventional graph under $do(S = 1)$, the double headed arrows indicate the pair of nodes that share the same hidden variables and the function in the causal model. Figure 3.4b shows the structures of the generators where G_1 agrees with causal graph \mathcal{G} and G_2 agrees with

interventional graph \mathcal{G}_s . The double headed arrows indicate the sharing of noises and parameters of sub-neural networks. As can be seen, the edge from A to S is deleted in interventional graph \mathcal{G}_s , which is also reflected in generator G_1 . In addition, for each pair of nodes in the graphs, e.g., B in \mathcal{G} and B in \mathcal{G}_s , the corresponding sub-neural networks are also synchronized.

3.4.2 Achieving Direct/Indirect/Counterfactual Fairness

Direct/Indirect Fairness. Both direct and indirect causal fairness are based on path-specific effects which concern causal effects transmitted along a certain set of paths only. To deal with direct and indirect causal fairness, Causal-FairGAN simulates in generator G_1 the situation where the intervention is transferred along the given path set only, which is then compared with the situation where the intervention is transferred along all causal paths. To address this challenge, Causal-FairGAN develops two settings of connections for generator G_1 : the interventional setting which is to simulate the first situation, and the reference setting which is to simulate the second situation. Then, discriminator D_1 is designed to distinguish two path-specific interventional distributions to achieve direct and indirect causal fairness.

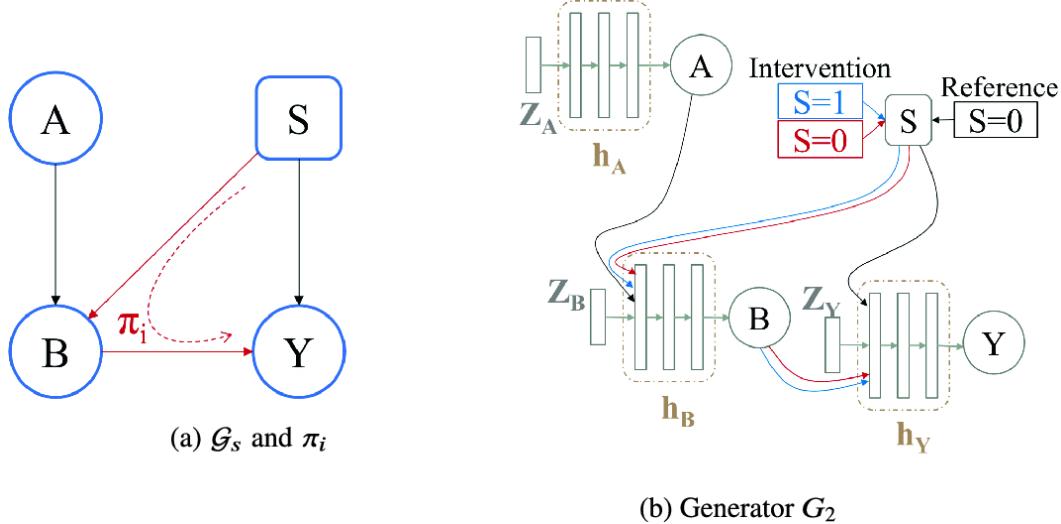


Fig. 3.5 An example of the generator G_1 for Causal-FairGAN based on indirect discrimination

For example, given a path set π_i that contains the paths pass through unjustified attributes, indirect fairness requires that

$P(Y_{S=1|\pi_i}) = P(Y_{S=0|\pi_i})$ with reference $S = 1$. Figure 3.5b shows the interventional graph \mathcal{G}_s of the causal graph in Fig. 3.4, where $\pi_i = \{S \rightarrow B \rightarrow Y\}$. Generator G_1 is designed to follow the interventional graph \mathcal{G}_s , as shown in Fig. 3.5b. The reference setting is denoted by links in black, and the interventional setting is denoted by links in red and green. In the reference setting, S is always set as 0, and the effect of its value is transmitted along both paths from S to Y . In the interventional setting, on the other hand, S can be both 1 and 0. The effect of $S = 1$ is transmitted along the red links as they represent the path in π_i , and the effect of $S = 1$ is transmitted along the green links as they represent the path not in π_i . As a result, the interventional distribution generated by G_1 , $P_{G_2}(\mathbf{X}_{S=s|\pi}, Y_{S=s|\pi})$, simulates the path-specific effect. To achieve indirect causal fairness, discriminator D_1 is designed to distinguish between two interventional distributions $\hat{y}_{S=1|\pi_i} \sim P_{G_2}(Y_{S=1|\pi_i})$ and $\hat{y}_{S=0|\pi_i} \sim P_{G_2}(Y_{S=0|\pi_i})$. By playing the adversarial game with G_1 , the corresponding value function J_1 aims to achieve $P_{G_2}(Y_{S=1|\pi_i}) = P_{G_2}(Y_{S=0|\pi_i})$.

Counterfactual fairness. The intervention is performed conditioning on a subset of variables $\mathbf{O} = \mathbf{o}$. Thus, different from previous fairness criteria that concern the interventional model only, counterfactual fairness concerns the connection between the original causal model and the interventional model. Causal-FairGAN reflects this connection by building a direct dependency between the samples generated by G_1 and the samples generated by G_1 . For each noise vector ϵ , Causal-FairGAN first generates the observational sample by using G_1 , and observes whether in the sample $\mathbf{O} = \mathbf{o}$. Only for those noise vectors with $\mathbf{O} = \mathbf{o}$ in the generated samples, Causal-FairGAN uses them for generating interventional samples by using G_1 . Thus, the interventional distribution generated by G_1 is conditioned on $\mathbf{O} = \mathbf{o}$, denoted by $P_{G_2}(\mathbf{X}_{S=s}, Y_{S=s}|\mathbf{o})$. Finally, discriminator D_1 is designed to distinguish between $\hat{y}_{S=1|\mathbf{o}} \sim P_{G_2}(Y_{S=1}|\mathbf{o})$ and $\hat{y}_{S=0|\mathbf{o}} \sim P_{G_2}(Y_{S=0}|\mathbf{o})$, producing the value function that aims to achieve $P_{G_2}(Y_{S=1}|\mathbf{o}) = P_{G_2}(Y_{S=0}|\mathbf{o})$.

3.4.3 Empirical Evaluation

Xu et al. [2] evaluated the performance of Causal-FairGAN in generating fair data for different types of causal fairness. They used the UCI Adult income dataset [31] with the same setting of data preprocessing and causal graph construction in [8]. They compared with CausalGAN [33] that preserves the causal structure of the original data but is unaware of the fairness constraint. They also compared Causal-FairGAN with other fair data generating approaches for different fairness respectively as other approaches may only be able to achieve one or two types of fairness.

For total effect, they compared with Simplified-FairGAN [1], which removes all information correlated to S in other attributes. The original data has a total effect of 0.1936, and CausalGAN preserves similar total effect. Simplified-FairGAN produces no total effect, but with the worst utility in terms of χ^2 . This is because Simplified-FairGAN removes too much information, including both causal and spurious, due to its causal blindness. The generated data by Causal-FairGAN based on total effect ($\lambda = 1$) produces no total effect, and also preserves good data utility.

For indirect discrimination, they compared with PSE-DR [8], which is a direct/indirect discrimination removing algorithm by modifying the causal graph and generating new fair data based on the modified causal graph. For indirect discrimination, they considered all the paths passing through *marital_status* as π_i . Similar to total effect, Causal-FairGAN preserves indirect discrimination close to the original data, and Simplified-FairGAN removes indirect discrimination but causes the largest utility loss. On the other hand, PSE-DR and Causal-FairGAN with specific causal effect ($\lambda = 1$) can remove indirect discrimination and also have good data utility.

For counterfactual fairness, they compared with A1 and A3 [11]. A1 generates fair decisions using a classifier that is built on non-descendants of S . A3 is similar to A1 but presupposes an additive noise model for estimating noise terms, which are then used for building the classifier. For counterfactual fairness, they considered the observation of two attributes, i.e., $\mathbf{O} = \{\text{race}, \text{native_country}\}$, which has 4 value combinations. The original data and CausalGAN contain biases in terms of counterfactual fairness in all subgroups. A1 is counterfactual fair as expected since it is proved to be so in [11]. However, the data utility is bad especially in terms of classifier accuracy, since it only uses non-descendants of *sex* in labeling decisions. A3 cannot achieve counterfactual fairness, probably because its

linear assumption does not fit the original data well. Finally, Causal-FairGAN with counterfactual fairness ($\lambda = 1$) achieves both counterfactual fairness and good data utility.

3.5 Achieving Fairness in Classification

Simplified-FairGAN and Causal-FairGAN cannot guarantee fair classification for models trained from generated data as classification performance-based fairness notions such as equality of odds or equality of opportunity [21] cannot be achieved via generating fair data alone. In [3], Xu et al. developed a new architecture, FairGAN+, to simultaneously achieve fair data generation and fair classification under the FairGAN framework. The main idea is to incorporate the classifier into the FairGAN architecture and learn the generative model and classifier simultaneously and also adopt another discriminator to train the classifier through adversarial learning. As a result, in addition to releasing fair data, FairGAN+ can produce a classifier that makes guaranteed discrimination-free predictions.

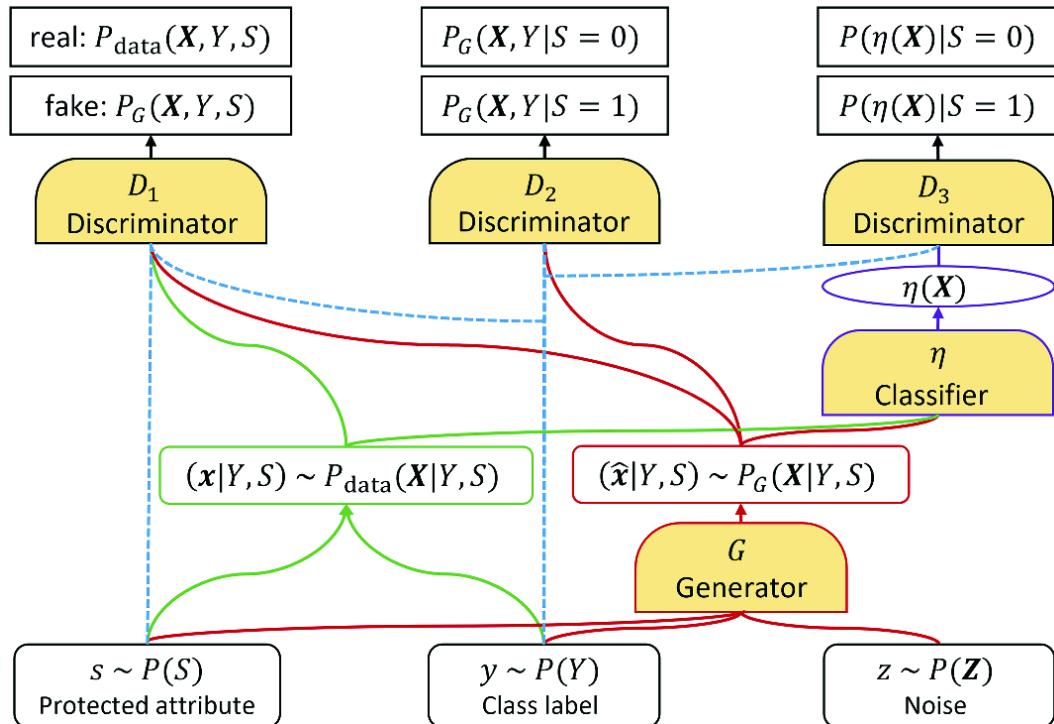


Fig. 3.6 The structure of Cla-FairGAN

3.5.1 Model Framework Design

For ease of presentation, we use the correlation-based fairness to describe our architecture because FairGAN+ mainly introduces modifications on discriminators. The causality-based fairness can be enforced by using generators for observational data and interventional data.

Figure 3.6 shows the structure of FairGAN+ that consists of one generator G , one classifier η and three discriminators D_1 , D_2 and D_3 . Each generated sample $G(\mathbf{z})$ has a corresponding value pair of the protected attribute $P_{\text{data}}(\mathbf{x}, y, s)$ and the class label $y \sim P_{\text{data}}(Y)$. The generator $G(\mathbf{z})$ generates fake $\hat{\mathbf{x}}$ following the conditional distribution $P_G(\mathbf{X}|Y, S)$ given a noise variable ϵ . The classifier $\eta : \mathbf{X} \rightarrow Y$ outputs the prediction $p_z(z)$ from \mathbb{E} . The prediction $p_z(z)$ is trained to both accurately predict the label Y and be free from discrimination. The objective function of FairGAN+ is $J = V + L$, where V is the value function of the overall minimax games between generator, classifier and discriminators; L is the objective function of the classifier. The value function V of the overall minimax game is defined as:

$$\min_{G, \eta} \max_{D_1, D_2, D_3} V(G, \eta, D_1, D_2, D_3) = V_1(G, D_1) + \lambda V_2(G, D_2) + \mu V_3(\eta, D_3),$$

where hyperparameter λ is used to specify a trade-off between data utility and fairness of data generation, μ to specify a trade-off between classification accuracy and classification fairness of η .

The first value function J_1 aims to make the generated data samples match the real data distribution, where the generator G seeks to learn the joint distribution $P_G(\mathbf{X}, Y, S)$ over real data $P_{\text{data}}(\mathbf{X}, Y, S)$, and the discriminator D_1 is trained to distinguish between the real data from $P_{\text{data}}(\mathbf{X}, Y, S)$ and the generated fake data from $P_G(\mathbf{X}, Y, S)$. The generator plays the adversarial game with D_1 to generate close-to-real fake data. The second value function J_1 aims to make the generated samples not encode any information supporting to predict the value of the protected attribute S . Therefore, D_1 is trained to distinguish values of the protected attribute of generated samples ($P_G(\mathbf{X}, Y|S=1)$ and $P_G(\mathbf{X}, Y|S=0)$) while the generator G aims to fool the discriminator D_1 . Once the generated samples $\{\hat{\mathbf{X}}, \hat{Y}\}$ cannot be used to predict the protected attribute S , the correlation between $\{\hat{\mathbf{X}}, \hat{Y}\}$ and S is removed, i.e., $\{\hat{\mathbf{X}}, \hat{Y}\} \perp\!\!\!\perp S$. Hence, we

can achieve fairness in data generation. The third value function J_1 aims to make the prediction $p_z(z)$ of samples not encode any information supporting to predict the value of the protected attribute S . Therefore, D_1 is trained to distinguish values of the protected attribute from the predictions made by η , i.e., $P(\eta(\mathbf{X}) = 1|S = 1)$ and $P(\eta(\mathbf{X}) = 1|S = 0)$ while the classifier η aims to fool the discriminator D_1 . Once the prediction of η cannot be used to predict the protected attribute S , the correlation between $p_z(z)$ and S is removed, i.e., $\eta(\mathbf{X}) \perp S$. The released η expects to achieve the desired fairness notion in classification.

FairGAN+ uses the feedback loop of various components to improve utility and fairness of both generated data and classifier. For example, the classifier η maximizes the log-likelihood of the correct class labels as it makes more accurate predictions during training, and the generator G also maximizes log-likelihood of the correct class labels as it generates samples that match each class accordingly. Hence in the feedback loop of D_1, G, η , improving the utility of G can improve the utility of η on making correct predictions and improving η can improve G on generating more realistic samples conditioned on each class. Similarly, in the feedback loop of D_2, G, D_3, η , improving the fairness of G can improve η on making fair predictions, and improving the fairness of η can improve G on generating fair data. As a result, by simultaneously learning a generative model and a classifier, FairGAN+ can perform better than a standalone generative model or a standalone classifier.

3.5.2 Achieving Fair Classification

To achieve correlation-based classification fairness such as statistical parity, equality of odds, and equality of opportunity, FairGAN+ enforces the independence or conditional independence between the predicted class label $p_z(z)$ and the protected attribute S . Demographic parity simply requires that the predictions of η is independent of the protected attribute S . The value function J_1 is set as

$$V_3(\eta, D_3) = \mathbb{E}_{\mathbf{x} \sim P(\mathbf{X}|Y, S=1)}[\log D_3(\eta(\mathbf{x}))] + \mathbb{E}_{\mathbf{x} \sim P(\mathbf{X}|Y, S=0)}[\log(1 - D_3(\eta(\mathbf{x})))]$$

. Once the prediction of η cannot be used to predict the protected attribute S , the correlation between $p_z(z)$ and S is removed, i.e., $\eta(\mathbf{X}) \perp S$. Then the classifier η achieves demographic parity. Equality of odds requires

$P(\eta(\mathbf{X}) = 1 | Y = y, S = 1) = P(\eta(\mathbf{X}) = 1 | Y = y, S = 0)$, where $y \in \{0, 1\}$. Hence the discriminator D_1 requires the corresponding real label y of the prediction $p_z(z)$. This can be achieved by defining the third value function

$V_3 = \mathbb{E}_{y \sim P(Y), \mathbf{x} \sim P(\mathbf{X}|Y=y, S=1)}[\log D_3(\eta(\mathbf{x})|Y=y)] + \mathbb{E}_{y \sim P(Y), \mathbf{x} \sim P(\mathbf{X}|Y=y, S=0)}[\log(1 - D_3(\eta(\mathbf{x})|Y=y))]$. J_1 aims to make the prediction $p_z(z)$ of samples not encode any information supporting to predict the value of the protected attribute S given the real label $Y = y$. Once the prediction of η cannot be used to predict the protected attribute S given the real label $Y = y$, the conditional correlation between $p_z(z)$ and S is removed, i.e., $\eta(\mathbf{X}) \perp S | Y = y$. For equality of opportunity, we only need to consider the case $Y = 1$, so the discriminator D_1 only aims to constrain a subgroup instead of the whole population.

To achieve classification-based fairness from the causal perspective, a predictive model $\eta : \mathbf{X} \rightarrow Y$, which outputs the prediction based on the interventional distribution generated by generators, and another discriminator D_1 , which is trained to distinguish values of the protected attribute from the predictions on the interventional data made by η , need to be introduced. The predictive model is expected to satisfy causal fairness by playing adversarial games.

3.5.3 Empirical Evaluation

In [3], Xu et al. conducted evaluation using the Adult dataset. Experimental results showed both FairGAN+ and Simplified-FairGAN can generate fair data that satisfy statistical fairness and ϵ -fairness while preserving the distribution of real data precisely. Compared with Simplified-FairGAN, FairGAN+ has a slightly better trade-off between utility and fairness of data generation as the co-training of the generative model and the classifier improves data generation.

To evaluate the performance of fair classification, they compared the logistic regression (LR) classifier in FairGAN+ with the LR model trained (not built-in) on the generated fair data from Simplified-FairGAN. Note that Simplified-FairGAN does not include the classifier component and simply assumes that fair data generation could automatically achieves fair classification. They also compared with ACGAN [34] that aims to generate the synthetic samples that have the same distribution as the real data given

the values of labels, i.e., $P_G(\mathbf{X}|Y, S) = P_{data}(\mathbf{X}|Y, S)$. ACGAN also contains a classifier, but does not enforce fairness in either data generation or classification.

The regular LR directly trained on the real data has a risk difference of 0.1834, which indicates unfair predictions. The LR model from ACGAN has a high risk difference 0.1119 and that from Simplified-FairGAN has a lower risk difference 0.0901 (but still above the threshold 0.05). This demonstrates that Simplified-FairGAN’s assumption on fair classification from fair data generation does not always hold. On the contrary, the LR model from FairGAN₊ has a low risk difference 0.0141. Evaluation results based on equality of odds showed similar observations. They also evaluated the accuracy of different classifiers on the real Adult dataset. The accuracy of the classifier from FairGAN₊ is slightly lower than other classifiers. In FairGAN₊, small utility loss is caused by playing the adversarial game with the third discriminator D_1 to achieve respective notions of classification fairness.

3.6 Related Work

In this section, we give a brief review on research works of using GANs for structural data generation. We refer the interested readers to [35] for a comprehensive review on image data generation via GANs. We also cover recent works on achieving privacy in GAN based models.

3.6.1 Dealing with Different Types of Structural Data

There have been quite some research works on using GANs to generate structural data in the past few years. Choi et al. [24] first introduced medGAN that combines an auto-encoder and a GAN to generate heterogeneous non-time-series continuous data. Srivastava et al. [36] developed VeeGAN that trains the generator and reconstructor networks jointly when modeling the multimodal distribution of continuous columns of tabular data. The joint training encourages the reconstructor network not only to map the data distribution to a Gaussian, but also to approximately reverse the action of the generator, thus avoiding mode collapse. Camino et al. [37] studied the generation of multi-categorical discrete tabular data using

GANs. Park et al. [25] developed tableGAN for generating synthetic data via a convolutional neural network which optimizes the label column’s quality.

To address data imbalance in tabular data generation, Xu et al. [38] proposed conditional tabular GAN (CTGAN) that uses a conditional generator and augments the training procedure with mode-specific normalization. Other works of addressing imbalanced data generation include [39–41]. In particular, Engelmann and Lessmann [40] proposed an oversampling method based on a conditional Wasserstein GAN. It can effectively model tabular datasets with numerical and categorical variables and considers the down-stream classification task through an auxiliary classifier loss. Fakoor et al. [41] proposed a data augmentation strategy for tabular data based on Gibbs sampling and introduced a self-attention pseudo likelihood estimator. Camino et al. [39] showed through experimental comparisons that undersampling majority class or oversampling minority class improves generative models on runtime and utility. Zheng et al. [42] developed one-class adversarial nets (OCAN) that uses a complementary GAN model to generate fake abnormal records using training data with only normal records.

Several GAN based models have been proposed to generate various medical data. Yahi et al. [43] used GANs to generate continuous time-series medical records. For the Electronic Health Record (EHR) data, Che et al. [44] proposed ehrGAN that augments the training dataset in a semi-supervised learning manner to generate plausible labeled data. Rashidian et al. [45] proposed SMOOTH-GAN that uses the Wasserstein GANs with gradient penalty loss. Similarly, Walia et al. [46] developed the Wasserstein conditional GANS with gradient penalty (WCGAN-GP) to generate tabular mixed data. It would be important to study how to achieve fairness in those models as medical data and their built decision models often incur discrimination.

3.6.2 Dealing with Privacy

How to achieve privacy in GAN based data generation models has also received extensive attention. Xie et al. [47] proposed a differentially private GAN (DPGAN) model for synthetic data generation. DPGAN adds Gaussian noise to gradients of the generator and discriminator during the learning procedure modifies gradient clipping by only clipping on the weights instead

of gradients. Zhang et al. [48] developed Privacy Preserving Generative Adversarial Network (PPGAN) that perturbs the objective function of discriminator by injecting Laplace noises based on functional mechanism to guarantee the differential privacy of training data.

To further effectively handle discrete data and capture the information from a mixture of discrete-continuous data, Torfi et al. [49] built on convolutional autoencoders and convolutional generative adversarial networks. Jordon et al. [50] proposed PATE-GAN that replaces the GAN discriminator with a Private Aggregation of Teacher Ensembles (PATE) [51, 52] mechanism. Chin-Cheong et al. [53] applied differentially private SGD (DPSGD) on WGAN-GP to generate private heterogeneous EHR data. Xu et al. [54] proposed GANobfuscator that also uses DPSGD in private training. Torkzadehmahani et al. [55] introduced a differentially private conditional GAN (DP-CGAN) training framework based on a new clipping and perturbation strategy, which computes the per-example gradients of the discriminator loss on both real and fake data and clips the per-example gradients on real data and fake data separately. Takahashi et al. [56] proposed PRIVAE that uses term-wise DPSGD in the learning process of variational autoencoder.

Chen et al. [57] proposed gradient-sanitized Wasserstein generative adversarial networks (GS-WGAN) that applies selective gradient sanitization scheme to ensure DP training of the generator. It allows for training GANs in both centralized and federated data scenarios. Yoon et al. [58] proposed anonymization through data synthesis using generative adversarial networks (ADS-GAN) which uses the conditional GAN framework to generate synthetic data while minimize patient identifiability. Under the decentralized setting, Augenstein et al. [59] proposed Fed-Avg GAN to adapt GAN training by using the DP-Fed-Avg [60] algorithm, providing user-level DP guarantee under trusted server.

3.7 Future Directions

In this section, we point out a few important research directions.

3.7.1 Variants Comparison and Architecture Design

Leveraging generative adversarial networks for fair machine learning is an important new research area. In addition to the FairGAN framework and three models presented in the previous section, there have been several variants developed in the past few years. Sattigeri et al. [61] developed a fairness aware GAN geared towards high dimensional image data. Its fairness discriminator is only fed the outcome, thus retaining information about the protected attribute in the generated image and preserving the natural image structure. Abusitta et al. [62] proposed a fair GAN framework that supports designers to decide on the amount of data that needs to be synthetically sampled and labeled. To avoid the difficulties which arise from double-network adversarial learning, Adel et al. [63] proposed an adversarial fairness framework that does not need the separate network architecture representing the adversary and instead designing a neural network that is uninformative about sensitive attributes of data as well as predictive of class labels. Choi et al. [64] proposed a weakly supervised algorithm for overcoming dataset bias for deep generative models. The algorithm adjusts importance weights by using an additional small, unlabeled reference dataset as the supervision signal, thus sidestepping the need for explicit labels on the underlying bias factors. Yu et al. [65] developed InclusiveGAN that considers the data coverage problem of minority inclusion and harmonizes adversarial training (GAN) with reconstructive generation, implicit maximum likelihood estimation. For graph data, Dai and Wang [66] proposed FairGNN to reduce the bias of GNNs. It is imperative to compare FairGAN variants in terms of fairness, utility, accuracy, and convergence and design new architectures to improve performance.

It is also important to investigate how to meet other requirements such as privacy preservation and explainability in GAN based data generation and predictive learning. Huang et al. [67] developed Generative Adversarial Privacy and Fairness (GAPF) for learning private and fair representations from data. GAPF provides privacy guarantees against strong information-theoretic adversaries and enforces demographic parity. A latest work [68] adapted CTGAN [38] to generate sparse, in-distribution counterfactual model explanations which match a desired target prediction with a conditional generative model, allowing batches of counterfactual instances to be generated with a single forward pass on mixed-type tabular data.

3.7.2 Achieving Long-Term Fairness in Dynamic Decision Making

Previous works consider static settings where each individual has only one decision record made from the individual profile, and fairness notions are proposed for one-shot decisions only. However, in many situations each individual is involved in a sequence of decisions. At each time step the decision is made from the current individual profile. Then, the decision will in turn affect the individual profile at the next time step through a feedback loop, and subsequently affect following decisions. For example, consider a lending system where each user is characterized by the credit score used by the bank in making loan decisions. After the bank grants the loan to the user, if the user can repay the loan in time, his/her credit score will increase, otherwise it will decline. As a result, for any user, the decision made by the bank in granting the loan or not will affect the decision made for the next time when the user applies for the loan again. Much recent research [69–71] has shown that imposing static fairness notions that are intended to protect disadvantage groups may actually harm fairness in the long run.

The idea of long-term fairness has been proposed and received increasing attention in recent research [71]. To define long-term fairness, it has been proposed to treat profile attributes \mathbb{E} as a representation of the qualification or reputation of an individual and measure long-term fairness by the difference in \mathbb{E} 's distributions between different groups over time. For example, in [72], long-term fairness is defined as being achieved if $\mathbb{E}[\mathbf{X}|S = 1, T = t^*] - \mathbb{E}[\mathbf{X}|S = 0, T = t^*] = 0$ where λ denotes the time that represents the long-term stage. As another example, in [69] long-term fairness is measured by $\mathbb{E}[\mathbf{X}|S = 0, T = 0] - \mathbb{E}[\mathbf{X}|S = 0, T = t^*]$. Based on these notions, we can leverage the FairGAN framework to generate fair sequence data as well as predictive models whose delayed impacts satisfy long-term fairness. To generate sequence data, one can use recurrent neural networks (RNN) or other deep neural nets for sequence modeling instead of feed-forward neural networks as generators. In each time step of the generator, we can simulate both mechanisms of how decisions are made from individual profiles as well as mechanisms of how individual profiles are reshaped by decisions. Then, we can include long-term fairness

constraints as discriminators, and simultaneously train a predictive model for making long-term fair decisions.

3.7.3 Achieving Fairness in Regression

Fair regression has received much attention recently. Fair regression can be naturally defined as the task of minimizing the expected loss of real-valued predictions subject to some fairness constraints. Recent research on fair regression focused on developing new fairness notions and approximation methods as target variables and even the sensitive attribute are continuous in the regression setting.

Fairness notions under the regression setting are in principle based on some form of independence, e.g., the independence of model prediction \hat{Y} and sensitive attribute A , the independence of prediction error $\hat{Y} - Y$ and sensitive attribute A , and the conditional independence of \hat{Y} and A given Y . Different from the classification setting, variables of Y and \hat{Y} (even A) become continuous in the regression setting. Researchers have developed quantitative metrics based on moment constraints, such as mean difference [73], mean squared error difference [74], and Pearson correlation [75]. Based on the FairGAN framework, we can modify and/or introduce new generators and discriminators to capturing and enforcing the above regression fairness notions. As a result, we can achieve fair data generation and fair regression. However, these simplified metrics fail to capture subtle effects. For example, the predicted values may have different variances across groups. Recently, researchers started to propose fairness metrics based on distributions/densities instead of simple point estimate [76], and develop approximation methods [77] for achieving fairness in regression. How to adapt these advanced fairness notions is a challenging and open direction.

3.7.4 Achieving Fairness in Recommendation

Recommender systems, which study user behavior and recommend personalized items, such as products, jobs, and courses, are ubiquitous on the web. Since recommender systems make recommendations based on historical data, they can easily incur discrimination that may already exist in the data. For example, in some job searching website, the recommender system could recommend high-pay jobs to white people while

recommending low-pay jobs to African Americans. The goal of fair recommendation is to learn a fair recommendation algorithm that can recommend suitable items to both groups but also be fair with respect to protected attribute. The challenge is that modern recommender systems usually need user feedback to enhance the personalization. For example, contextual multi-armed bandits [78, 79] dynamically select the suitable items based on personal and item information and user feedback. As a result, it cannot be directly solved using ideas similar to that of FairGAN+.

Achieving fairness in recommendation has been received increasing attention recently [80–82]. Based on the FairGAN framework, we can also achieve fair recommendations. Specifically, we can leverage GAN to generate discrimination-free context information as well as discrimination-free recommended items as training samples. Meanwhile, to further ensure fairness in recommendation algorithms, an adversarial game as a constraint is incorporated by detecting whether recommended items are from protected or unprotected group. If the discriminator is unable to distinguish which group the recommended items are from, we consider the algorithms achieve the fair recommendation.

3.7.5 Open Source Software

In order to help the transition of fairness research algorithms to use in real systems and to share a framework for research community, it is important to implement an open-source library for the FairGAN models. The library should provide a comprehensive metrics and algorithms for discrimination discovery, fair generation of both static and dynamic data, fair classification, and fair recommendation. Domain users and developers can incorporate them into their real systems and/or design their own GAN-based models via the provided APIs to achieve fairness in their applications.

3.8 Conclusions

Fair machine learning is receiving an increasing attention in machine learning fields. There are two major challenges in fair machine learning: (1) releasing fair training data and (2) building models that make fair predictions. In this paper, based on generative adversarial networks (GAN), we presented a unified framework, named FairGAN, for generating data that

have good utility and also satisfy various fairness requirements. The proposed framework consists two components: generators for generating synthetic data, and discriminators for posing fairness and utility constraints. Within the framework, we can develop different model designs by modifying the network architecture and objective functions to meet different fairness requirements. We presented the Simplified-FairGAN for achieving simple correlation-based fairness criteria in generated data, Causal-FairGAN for achieving various causality-based fairness criteria, and FairGAN+ for achieving fairness for both generated data and learned classifiers. We also introduced other fairness aware GAN models and introduced several future research directions including how to achieve long-term fairness, fairness in regression, and fairness in recommendation under the FairGAN framework.

Acknowledgements

This work was supported in part by NSF 1841119, 1910284, 1920920, and 2137335.

References

1. Xu, D., Yuan, S., Zhang, L., Wu, X.: Fairgan: fairness-aware generative adversarial networks. In: IEEE International Conference on Big Data, pp. 570–575 (2018).
<https://doi.org/10.1109/BigData.2018.8622525>
2. Xu, D., Wu, Y., Yuan, S., Zhang, L., Wu, X.: Achieving causal fairness through generative adversarial networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16 (2019)
3. Xu, D., Yuan, S., Zhang, L., Wu, X.: Fairgan+: achieving fair data generation and classification through generative adversarial nets. In: 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9–12, 2019, pp. 1401–1406. IEEE (2019)
4. Kamiran, F., Calders, T.: Classifying without discriminating. In: 2009 2nd International Conference on Computer, Control and Communication, pp. 1–6. IEEE (2009)
5. Calders, T., Kamiran, F., Pechenizkiy, M.: Building classifiers with independency constraints. In: 2009 IEEE International Conference on Data Mining Workshops,

pp. 13–18. IEEE (2009)

6. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* **33**(1), 1–33 (2012) [[Crossref](#)]
7. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15, pp. 259–268. ACM Press (2015)
8. Zhang, L., Wu, Y., Wu, X.: A causal framework for discovering and removing direct and indirect discrimination. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 3929–3935 (2017)
9. Zhang, L., Wu, Y., Wu, X.: Causal modeling-based discrimination discovery and removal: criteria, bounds, and algorithms. *IEEE Trans. Knowl. Data Eng.* (2018)
10. Wu, Y., Zhang, L., Wu, X., Tong, H.: P_c-fairness: a unified framework for measuring causality-based fairness. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8–14 December 2019*, pp. 3399–3409. Canada, Vancouver, BC (2019)
11. Kusner, M.J., Loftus, J., Russell, C., Silva, R.: Counterfactual fairness. In: *Advances in Neural Information Processing Systems*, pp. 4066–4076 (2017)
12. Edwards, H., Storkey, A.: Censoring representations with an adversary. *Phys. Rev. D* **93**(2), 023519 (2015)
13. Xie, Q., Dai, Z., Du, Y., Hovy, E., Neubig, G.: Controllable invariance through adversarial feature learning. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, p. 00004 (2017)
14. Madras, D., Creager, E., Pitassi, T., Zemel, R.S.: Learning adversarially fair and transferable representations (2018). [arXiv:1802.06309](#)
15. Zhang, B.H., Lemoine, B., Mitchell, M.: Mitigating unwanted biases with adversarial learning. In: AAAI Conference on AI, Ethics and Society (2018)
16. Žliobaite, I., Kamiran, F., Calders, T.: Handling conditional discrimination. In:

2011 IEEE 11th International Conference on Data Mining (ICDM), pp. 992–1001. IEEE (2011)

17. Kamishima, T., Akaho, S., Sakuma, J.: Fairness-aware learning through regularization approach. In: 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 643–650. IEEE (2011)
18. Zafar, M.B., Valera, I., Rodriguez, M.G., Gummadi, K.P.: Fairness constraints: mechanisms for fair classification. In: AISTATS (2017)
19. Wu, Y., Zhang, L., Wu, X.: On convexity and bounds of fairness-aware classification. In: The World Wide Web Conference, WWW, San Francisco, CA, USA, May 13–17, 2019, pp. 3356–3362 (2019). <https://doi.org/10.1145/3308558.3313723>
20. Kamiran, F., Calders, T., Pechenizkiy, M.: Discrimination aware decision tree learning. In: 2010 IEEE International Conference on Data Mining, pp. 869–874 (2010)
21. Hardt, M., Price, E., Srebro, N., et al.: Equality of opportunity in supervised learning. In: Advances in Neural Information Processing Systems, pp. 3315–3323 (2016)
22. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning (2019). [arXiv:1908.09635](https://arxiv.org/abs/1908.09635)
23. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. In: NIPS (2014). [arXiv:1406.2661](https://arxiv.org/abs/1406.2661)
24. Choi, E., Biswal, S., Malin, B.A., Duke, J., Stewart, W.F., Sun, J.: Generating multi-label discrete patient records using generative adversarial networks. In: Doshi-Velez, F., Fackler, J., Kale, D.C., Ranganath, R., Wallace, B.C., Wiens, J. (eds.) Proceedings of the Machine Learning for Health Care Conference, MLHC 2017, Boston, Massachusetts, USA, 18–19 August 2017. Proceedings of Machine Learning Research, vol. 68, pp. 286–305. PMLR (2017)
25. Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., Kim, Y.: Data synthesis based on generative adversarial networks. Proc. VLDB Endow. **11**(10), 1071–1083 (2018)
[Crossref]
26. Fedus, W., Goodfellow, I., Dai, A.M.: Maskgan: better text generation via filling in

- the _____. In: ICLR (2018). [arXiv:1801.07736](https://arxiv.org/abs/1801.07736)
27. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: sequence generative adversarial nets with policy gradient. In: AAAI (2017). [arxiv:1609.05473](https://arxiv.org/abs/1609.05473)
 28. Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., Guo, M.: Graphgan: graph representation learning with generative adversarial nets (2017). [arXiv:1711.08267](https://arxiv.org/abs/1711.08267) [cs, stat]
 29. Goodfellow, I.J.: NIPS 2016 tutorial: generative adversarial networks (2017). [arXiv:1701.00160](https://arxiv.org/abs/1701.00160)
 30. Pearl, J.: Causality: Models, 2nd edn. Reasoning and Inference. Cambridge University Press, New York (2009)
 31. Dheeru, D., Karra Taniskidou, E.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2017)
 32. Zhang, J., Bareinboim, E.: Fairness in decision-making—the causal explanation formula. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
 33. Kocaoglu, M., Snyder, C., Dimakis, A.G., Vishwanath, S.: Causalgan: Learning causal implicit generative models with adversarial training (2017). [arXiv:1709.02023](https://arxiv.org/abs/1709.02023)
 34. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: Proceedings of the 34th International Conference on Machine Learning, pp. 2642–2651. ICML’17 (2017)
 35. Gui, J., Sun, Z., Wen, Y., Tao, D., Ye, J.: A review on generative adversarial networks: algorithms, theory, and applications (2020). [arXiv:2001.06937](https://arxiv.org/abs/2001.06937)
 36. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: VEEGAN: reducing mode collapse in gans using implicit variational learning. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, pp. 3308–3318 (2017)
 37. Camino, R., Hammerschmidt, C.A., State, R.: Generating multi-categorical samples with generative adversarial networks (2018). [arXiv:1807.01202](https://arxiv.org/abs/1807.01202)
 38. Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular

- data using conditional GAN. In: NeurIPS, pp. 7333–7343 (2019)
- 39. Camino, R., Hammerschmidt, C.A., State, R.: Minority class oversampling for tabular data with deep generative models (2020). [arXiv:2005.03773](https://arxiv.org/abs/2005.03773)
 - 40. Engelmann, J., Lessmann, S.: Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning (2020). [arXiv:2008.09202](https://arxiv.org/abs/2008.09202)
 - 41. Fakoor, R., Mueller, J., Erickson, N., Chaudhari, P., Smola, A.J.: Fast, accurate, and simple models for tabular data via augmented distillation. In: NeurIPS (2020)
 - 42. Zheng, P., Yuan, S., Wu, X., Li, J., Lu, A.: One-class adversarial nets for fraud detection. In: AAAI (2019)
 - 43. Yahi, A., Vanguri, R., Elhadad, N., Tatonetti, N.P.: Generative adversarial networks for electronic health records: A framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories (2017). [arXiv:1712.00164](https://arxiv.org/abs/1712.00164)
 - 44. Che, Z., Cheng, Y., Zhai, S., Sun, Z., Liu, Y.: Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In: ICDM, pp. 787–792. IEEE Computer Society (2017)
 - 45. Rashidian, S., Wang, F., Moffitt, R.A., Garcia, V., Dutt, A., Chang, W., Pandya, V., Hajagos, J.G., Saltz, M.M., Saltz, J.H.: SMOOTH-GAN: towards sharp and smooth synthetic EHR data generation. In: AIME. Lecture Notes in Computer Science, vol. 12299, pp. 37–48. Springer (2020)
 - 46. Walia, M., Tierney, B., McKeever, S.: Synthesising tabular data using wasserstein conditional gans with gradient penalty (WCGAN-GP). In: AICS. CEUR Workshop Proceedings, vol. 2771, pp. 325–336. CEUR-WS.org (2020)
 - 47. Xie, L., Lin, K., Wang, S., Wang, F., Zhou, J.: Differentially private generative adversarial network (2018). [arXiv:1802.06739](https://arxiv.org/abs/1802.06739)
 - 48. Zhang, X., Ding, J., Errapotu, S.M., Huang, X., Li, P., Pan, M.: Differentially private functional mechanism for generative adversarial networks. In: GLOBECOM, pp. 1–6. IEEE (2019)
 - 49. Torfi, A., Fox, E.A., Reddy, C.K.: Differentially private synthetic medical data generation using convolutional gans (2020). [arXiv:2012.11774](https://arxiv.org/abs/2012.11774)
 - 50. Jordon, J., Yoon, J., van der Schaar, M.: PATE-GAN: generating synthetic data with differential privacy guarantees. In: ICLR (Poster). OpenReview.net (2019)

51. Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I.J., Talwar, K.: Semi-supervised knowledge transfer for deep learning from private training data. In: ICLR. OpenReview.net (2017)
52. Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Erlingsson, Ú.: Scalable private learning with PATE. In: ICLR. OpenReview.net (2018)
53. Chin-Cheong, K., Sutter, T.M., Vogt, J.E.: Generation of differentially private heterogeneous electronic health records (2020). [arXiv:2006.03423](https://arxiv.org/abs/2006.03423)
54. Xu, C., Ren, J., Zhang, D., Zhang, Y., Qin, Z., Ren, K.: Ganobfuscator: mitigating information leakage under GAN via differential privacy. IEEE Trans. Inf. Forensics Secur. **14**(9), 2358–2371 (2019)
[\[Crossref\]](#)
55. Torkzadehmahani, R., Kairouz, P., Paten, B.: DP-CGAN: differentially private synthetic data and label generation. In: CVPR Workshops, pp. 98–104. Computer Vision Foundation/IEEE (2019)
56. Takahashi, T., Takagi, S., Ono, H., Komatsu, T.: Differentially private variational autoencoders with term-wise gradient aggregation (2020). [arXiv:2006.11204](https://arxiv.org/abs/2006.11204)
57. Chen, D., Orekondy, T., Fritz, M.: GS-WGAN: A gradient-sanitized approach for learning differentially private generators. In: NeurIPS (2020)
58. Yoon, J., Drumright, L.N., van der Schaar, M.: Anonymization through data synthesis using generative adversarial networks (ADS-GAN). IEEE J. Biomed. Health Inf. **24**(8), 2378–2388 (2020)
[\[Crossref\]](#)
59. Augenstein, S., McMahan, H.B., Ramage, D., Ramaswamy, S., Kairouz, P., Chen, M., Mathews, R., y Arcas, B.A.: Generative models for effective ML on private, decentralized datasets. In: ICLR. OpenReview.net (2020)
60. McMahan, H.B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. In: ICLR (Poster). OpenReview.net (2018)
61. Sattigeri, P., Hoffman, S.C., Chenthamarakshan, V., Varshney, K.R.: Fairness gan: Generating datasets with fairness properties using a generative adversarial network. IBM J. Res. Devel. **63**(4/5), 3–1 (2019)
62. Abusitta, A., Aïmeur, E., Wahab, O.A.: Generative adversarial networks for mitigating biases in machine learning systems. In: Giacomo, G.D., Catalá, A.,

Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) ECAI (2020)

63. Adel, T., Valera, I., Ghahramani, Z., Weller, A.: One-network adversarial fairness. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27– February 1, 2019. pp. 2412–2420. AAAI Press (2019)
64. Choi, K., Grover, A., Singh, T., Shu, R., Ermon, S.: Fair generative modeling via weak supervision. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 1887–1898. PMLR (2020)
65. Yu, N., Li, K., Zhou, P., Malik, J., Davis, L., Fritz, M.: Inclusive GAN: improving data and minority coverage in generative models. In: ECCV (22). Lecture Notes in Computer Science, vol. 12367, pp. 377–393. Springer (2020)
66. Dai, E., Wang, S.: Fairgnn: Eliminating the discrimination in graph neural networks with limited sensitive attribute information (2020). [arXiv:2009.01454](https://arxiv.org/abs/2009.01454)
67. Huang, C., Chen, X., Kairouz, P., Sankar, L., Rajagopal, R.: Generative adversarial models for learning private and fair representations (2018)
68. Looveren, A.V., Klaise, J., Vacanti, G., Cobb, O.: Conditional generative models for counterfactual explanations (2021). [arXiv:2101.10123](https://arxiv.org/abs/2101.10123)
69. Liu, L., Dean, S., Rolf, E., Simchowitz, M., Hardt, M.: Delayed impact of fair machine learning. In: International Conference on Machine Learning, pp. 3156–3164 (2018)
70. Tu, R., Zhang, X., Liu, Y., Kjellström, H., Liu, M., Zhang, K., Zhang, C.: How do fair decisions fare in long-term qualification? In: Thirty-fourth Conference on Neural Information Processing Systems (2020)
71. Zhang, X., Liu, M.: Fairness in learning-based sequential decision algorithms: a survey (2020). [arXiv:2001.04861](https://arxiv.org/abs/2001.04861)
72. Hu, L., Chen, Y.: A short-term intervention for long-term fairness in the labor market. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 1389–1398. International World Wide Web Conferences Steering Committee (2018)
73. Calders, T., Karim, A., Kamiran, F., Ali, W., Zhang, X.: Controlling attribute effect in linear regression. In: 2013 IEEE 13th International Conference on Data Mining, pp. 71–80. IEEE (2013)

74. Johnson, K.D., Foster, D.P., Stine, R.A.: Impartial predictive modeling: ensuring group fairness in arbitrary models, pp. arXiv–1608 (2016)
75. Komiyama, J., Takeda, A., Honda, J., Shima, H.: Nonconvex optimization for regression with fairness constraints. In: International Conference on Machine Learning, pp. 2737–2746. PMLR (2018)
76. Agarwal, A., Dudík, M., Wu, Z.S.: Fair regression: quantitative definitions and reduction-based algorithms. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 120–129. PMLR (2019)
77. Steinberg, D., Reid, A., O’Callaghan, S., Lattimore, F., McCalman, L., Caetano, T.S.: Fast fair regression via efficient approximations of mutual information (2020). [arXiv:2002.06200](https://arxiv.org/abs/2002.06200)
78. Hill, D.N., Nassif, H., Liu, Y., Iyer, A., Vishwanathan, S.: An efficient bandit algorithm for realtime multivariate optimization. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1813–1821. ACM (2017)
79. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web, pp. 661–670. ACM (2010)
80. Burke, R., Sonboli, N., Ordonez-Gauger, A.: Balanced neighborhoods for multi-sided fairness in recommendation. In: Conference on Fairness, Accountability and Transparency, pp. 202–214 (2018)
81. Mehrotra, R., McInerney, J., Bouchard, H., Lalmas, M., Diaz, F.: Towards a fair marketplace: counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2243–2251 (2018)
82. Chen, Y., Cuellar, A., Luo, H., Modi, J., Nemlekar, H., Nikolaidis, S.: Fair contextual multi-armed bandits: theory and experiments. In: Conference on Uncertainty in Artificial Intelligence, pp. 181–190 (2020)

4. Quaternion Generative Adversarial Networks

Eleonora Grassucci¹✉, Edoardo Cicero¹ and Danilo Comminiello¹✉
(1) Department of Information Engineering, Electronics and
Telecommunications (DIET), Sapienza University of Rome, Via
Eudossiana 18, 00184 Rome, Italy

✉ Eleonora Grassucci (Corresponding author)
Email: eleonora.grassucci@uniroma1.it

✉ Danilo Comminiello
Email: danilo.comminiello@uniroma1.it

Abstract

Latest Generative Adversarial Networks (GANs) are gathering outstanding results through a large-scale training, thus employing models composed of millions of parameters requiring extensive computational capabilities.

Building such huge models undermines their replicability and increases the training instability. Moreover, multi-channel data, such as images or audio, are usually processed by real-valued convolutional networks that flatten and concatenate the input, often losing intra-channel spatial relations. To address these issues related to complexity and information loss, we propose a family of quaternion-valued generative adversarial networks (QGANs). QGANs exploit the properties of quaternion algebra, e.g., the Hamilton product, that allows to process channels as a single entity and capture internal latent relations, while reducing by a factor of 4 the overall number of parameters. We show how to design QGANs and to extend the proposed approach even to advanced models. We compare the proposed QGANs with

real-valued counterparts on several image generation benchmarks. Results show that QGANs are able to obtain better FID scores than real-valued GANs and to generate visually pleasing images. Furthermore, QGANs save up to 75% of the training parameters. We believe these results may pave the way to novel, more accessible, GANs capable of improving performance and saving computational resources.

This work has been supported by “Progetti di Ricerca Grandi” of Sapienza University of Rome under grant number RG11916B88E1942F.

4.1 Introduction

Generative models including generative adversarial networks (GANs) [10] and variational autoencoders (VAEs) [24] have been recently spectators of an increasing widespread development due to the massive availability of large datasets covering a large range of applications. The demand to learn such complex data distributions leads to define models far from the original approach of a simple GAN, which was characterized by fully connected layers and evaluated on benchmark datasets such as the MNIST [10].

Multiple pathways have been covered to improve GANs generation ability. A first branch aims at stabilizing the training process which is notoriously unstable, often leading to a lack of convergence. This includes constraining the discriminator network to be 1-Lipschitz by introducing a gradient penalty in the loss function, normalizing the spectral norm of the network weights or adding a consistency regularization [1, 16, 26, 41]. Other significant improvements are gained by architectural innovations such as self-attention modules, flexible activation functions or style-based generator [14, 21, 22, 40]. A crucial improvement in the quality of image generation has been brought by broadly scaling up the networks and involving wider batch sizes [2, 23, 34]. Indeed, BigGAN closed the visual quality gap between GANs generated images and real-world samples in ImageNet [2]. Most of the latest GANs are somehow inspired to it.

However, these impressive results come at the cost of huge models with hundred of millions of free parameters that require large computational resources. This drastically reduces the accessibility and the diffusion of this kind of models. Moreover, GANs are notorious fragile models, thus the

training with this amount of parameters may result in unstable or less handy process. Furthermore, when dealing with multidimensional inputs, such as images, 3D audio, multi-sensor signals or human-pose estimation, among others, real-valued networks break the original structure of the inputs. Channels are processed as independent entities and just concatenated in a tensor without exploiting any intra-channel correlation.

In order to address these limitations, neural networks in hypercomplex domains have been proposed. Among them, quaternion neural networks (QNNs) leverage the properties of the non-commutative quaternion algebra to define lower-complexity models and preserve relations among channels. Indeed, QNNs process channels together as a single entity, thus maintaining the original input design and correlation. Due to this feature, QNNs are able to capture internal relations while saving up to the 75% of free parameters thanks to hypercomplex-valued operations, including the Hamilton product.

Encouraged by the promising results of other generative models in the quaternion domain [12, 13] and the need to make deep GANs more accessible, we introduce the family of quaternion generative adversarial networks (QGANs). QGANs are completely defined in the quaternion domain and, among other properties, they exploit the quaternion convolutions derived from the hypercomplex algebra [6, 8, 28, 29] to improve the generation ability of the model while reducing the overall number of parameters. A first attempt to introduce quaternion convolutions in GANs has been recently made in [35]. Here, we define the core-blocks of the quaternion generative adversarial framework that we use to formulate a vanilla QGAN. Then, we explain how to derive more advanced QGANs to prove the superior generation ability of the proposed approach in multiple image generation benchmarks. We show that the quaternion spectral normalized GAN (QSNGAN) is able to earn a better FID and a more pleasant visual quality of the generated images with respect to its real-valued counterpart thanks to the quaternion inner operations. Moreover, the proposed QSNGAN has just 75% the number of free parameters with respect to the real-valued SNGAN.

We believe that these theoretical statements and empirical results lay the foundations for novel deep GANs in hypercomplex domains capable of grasping internal input relations while scaling down computational

requirements, thus saving memory and being more accessible. To the best of our knowledge, this is the first time that a generative adversarial framework has been completely defined in a hypercomplex domain.

The contribution of this chapter is threefold:

- (i) we introduce the family of quaternion generative adversarial networks (QGANs), proving their enhanced generation ability and lower-complexity with respect to its real-valued counterpart on different benchmark datasets¹;
- (ii) we define the theoretically correct approach to apply the quaternion batch normalization (QBN) and redefine existing approaches as its approximations;
- (iii) we propose and define the spectral normalization in the quaternion domain (QSN), proving its efficacy on two image generation benchmarks.

The chapter is organized as follows. Section 4.2 presents the fundamental properties of quaternion algebra, while Sect. 4.3 describes the quaternion adversarial framework and the quaternion-valued core blocks employed in QGANs. Section 4.4 lays the foundations for the quaternion generative adversarial networks and presents a simple quaternion vanilla GAN and a more advanced and complex QGAN model. Section 4.5 proves the effectiveness of the presented QGANs on a thorough empirical evaluation, and, finally, conclusions are drawn in Sect. 4.6.

4.2 Quaternion Algebra

Quaternions are hypercomplex numbers of rank 4, being a direct non-commutative extension of complex-valued numbers. The quaternion domain \mathcal{D} lies in a four-dimensional associative normed division algebra over real numbers, belonging to the class of Clifford algebras [38]. A quaternion is defined as the composition of one scalar element and three imaginary ones:

$$q = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k} = q_0 + \mathbf{q} \quad (4.1)$$

with $q_0, q_1, q_2, q_3 \in \mathbb{R}$ and being

$\hat{i} = (1, 0, 0)$, $\hat{j} = (0, 1, 0)$, $\hat{\kappa} = (0, 0, 1)$ unit axis vectors representing the orthonormal basis in \mathcal{G}_s . A *pure quaternion* is a quaternion without its scalar part q_0 , resulting in the vector $\mathbf{q} = q_1\hat{i} + q_2\hat{j} + q_3\hat{\kappa}$. As for complex numbers, also the quaternion algebra relies upon the relations among the imaginary components:

$$\hat{i}^2 = \hat{j}^2 = \hat{\kappa}^2 = -1 \quad (4.2)$$

$$\hat{i}\hat{j} = \hat{i} \times \hat{j} = \hat{\kappa}; \quad \hat{j}\hat{\kappa} = \hat{j} \times \hat{\kappa} = \hat{i}; \quad \hat{\kappa}\hat{i} = \hat{\kappa} \times \hat{i} = \hat{j} \quad (4.3)$$

While the scalar product of two quaternions q and p is simply defined as the element-wise product $q \cdot p = q_0p_0 + q_1p_1 + q_2p_2 + q_3p_3$, quaternion vector multiplication, denoted with \times , is not commutative, i.e., $\hat{i}\hat{j} \neq \hat{j}\hat{i}$. In fact:

$$\hat{i}\hat{j} = -\hat{j}\hat{i}; \quad \hat{j}\hat{\kappa} = -\hat{\kappa}\hat{j}; \quad \hat{\kappa}\hat{i} = -\hat{i}\hat{\kappa}.$$

Due to the non-commutative property, the quaternion product is introduced, commonly known as Hamilton product. We will see that Hamilton product plays a crucial role in neural networks. It is defined as:

$$\begin{aligned} qp &= (q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{\kappa})(p_0 + p_1\hat{i} + p_2\hat{j} + p_3\hat{\kappa}) \\ &= (q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3) \\ &\quad + (q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3)\hat{i} \\ &\quad + (q_2p_0 + q_3p_1 + q_0p_2 - q_1p_3)\hat{j} \\ &\quad + (q_3p_0 - q_2p_1 + q_1p_2 + q_0p_3)\hat{\kappa}. \end{aligned} \quad (4.4)$$

The above product can be rewritten in a more concise form as:

$$qp = q_0p_0 - \mathbf{q} \cdot \mathbf{p} + q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p}, \quad (4.5)$$

where $q_0p_0 - \mathbf{q} \cdot \mathbf{p}$ is the scalar element of the new quaternion in output and $q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p}$ is instead the vector part of the quaternion. From (4.5) it is easy to define a concise form of product for pure quaternions too:

$$\mathbf{q}\mathbf{p} = -\mathbf{q} \cdot \mathbf{p} + \mathbf{q} \times \mathbf{p}. \quad (4.6)$$

where the scalar product is the same as before for full quaternions and the vector product is

$$P(\eta(\mathbf{X}) = 1 | Y = y, S = 1) = P(\eta(\mathbf{X}) = 1 | Y = y, S = 0).$$

Similarly to complex numbers, the complex conjugate of a quaternion can be defined as:

$$q^* = q_0 - q_1\hat{i} - q_2\hat{j} - q_3\hat{\kappa} = q_0 - \mathbf{q} \quad (4.7)$$

Also the norm is defined and it is equal to

$$|q| = \sqrt{qq^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

that is the euclidean norm in \mathcal{G}_s .

Indeed, q is said to be a *unit quaternion* if $\mathcal{G}(z, c)$, as well as a *pure unit quaternion* if $V(\mathcal{G}, \mathcal{D})$. Moreover, a quaternion q is endowed with an inverse determined by:

$$q^{-1} = \frac{q^*}{|q|^2}.$$

Note that for unit quaternions, the relation $q^* = q^{-1}$ holds.

A quaternion has also a polar form:

$$q = |q| (\cos(\theta) + \mathbf{v} \sin(\theta)) = |q| e^{\mathbf{v}\theta} \quad (4.8)$$

where $\theta \in \mathbb{R}$ is the argument of the quaternion, $\cos(\theta) = q_0 / \|q\|$, $\sin(\theta) = \|\bar{q}\| / \|q\|$ and $\mathbf{v} = \bar{q} / \|\bar{q}\|$ is a pure unit quaternion.

Following, quaternions show interestingly properties when they can be interpreted as points and hyperplanes in \mathcal{G}_s . Among them, we find involutions, which are generally defined as self-inverse mappings or mappings that are their own inverse. Quaternions have an infinite number of involutions [7] that can be generalized by the formula:

$$q^{\mathbf{v}} = -\mathbf{v}q\mathbf{v} \quad (4.9)$$

where q is an arbitrary quaternion to be involved and \mathbf{x} is any unit vector and the axis of the involution. Among the infinite involutions, the most relevant ones are the three perpendicular involutions defined as:

$$\begin{aligned} q^{\hat{i}} &= -\hat{i}q\hat{i} = q_0 + q_1\hat{i} - q_2\hat{j} - q_3\hat{\kappa} \\ q^{\hat{j}} &= -\hat{j}q\hat{j} = q_0 - q_1\hat{i} + q_2\hat{j} - q_3\hat{\kappa} \\ q^{\hat{\kappa}} &= -\hat{\kappa}q\hat{\kappa} = q_0 - q_1\hat{i} - q_2\hat{j} + q_3\hat{\kappa} \end{aligned} \quad (4.10)$$

which are the first identified involutions [5] and they are crucial for the study of the second-order statistics of a quaternion signal, as we will see in the next section.

4.3 Generative Learning in the Quaternion Domain

In this section, we introduce the quaternion adversarial approach as well as the fundamental quaternion-valued operations employed to define the family of QGANs in next sections. It is worth noting that in a quaternion neural network each element is a quaternion, including inputs, weights, biases and outputs.

4.3.1 The Quaternion Adversarial Framework

Generative adversarial networks are built upon a minimax game between the generator network (G) and the discriminator one (D), as a special case of the concept initially proposed to implement artificial curiosity [32, 33]. They are trained in an adversarial fashion through the following objective function introduced in [10]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \{\log D(x)\} + \mathbb{E}_{z \sim p_z(z)} \{\log(1 - D(G(z)))\} \quad (4.11)$$

where p_{data} is the real data distribution and p_z is the noise distribution. The two terms in the objective are two cross-entropies [15]. Indeed, the first term is the cross-entropy between $[1 \ 0]^T$ and $[D(x) \ 1 - D(x)]^T$, whereas the second term is the cross-entropy between $[1 \ 0]^T$ and $[D(G(z)) \ 1 - D(G(z))]^T$. In order to introduce the family of QGANs, first we need to delineate this adversarial approach in the hypercomplex domain. Thus, we define the cross-entropy function for quaternions which has to take the four components into account, as suggested in [28] for the quaternion mean squared error, by replacing real numbers with hypercomplex numbers and computing the operations element-wise. Thus, the quaternion cross-entropy (QCE) between the target quaternion q and the estimated one \tilde{q} can be defined as follows:

(4.12)

$$\begin{aligned} \text{QCE}(q, \tilde{q}) = \frac{1}{N} \sum_{n=1}^N & [q_0 \log(\tilde{q}_0) + (1 - q_0) \log(1 - \tilde{q}_0) \\ & + q_1 \log(\tilde{q}_1) + (1 - q_1) \log(1 - \tilde{q}_1) \\ & + q_2 \log(\tilde{q}_2) + (1 - q_2) \log(1 - \tilde{q}_2) \\ & + q_3 \log(\tilde{q}_3) + (1 - q_3) \log(1 - \tilde{q}_3)]. \end{aligned}$$

More in general, several objective functions proposed to train GANs can be redefined in the quaternion domain. Among the most common ones, we find the Wasserstein distance with a gradient penalty that enforces the Lipschitz continuity of the discriminator, which is defined as follows [1, 16]:

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} \{D(x)\} - \mathbb{E}_{z \sim p(z)} \{D(G(z))\} - \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} \{(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\} \quad (4.13)$$

where the last term is the gradient penalty that is a regularization technique for the discriminator, and \hat{x} is an interpolation as in [16].

Other works [3, 26] consider instead the hinge loss, which is given, respectively for the discriminator and the generator, by:

$$V(D, \hat{G}) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \{\min(0, -1 + D(x))\} + \mathbb{E}_{z \sim p_z(z)} \{\min(0, -1 - D(G(z)))\}, \quad (4.14)$$

$$V(\hat{D}, G) = -\mathbb{E}_{z \sim p_z(z)} \{\hat{D}(G(z))\}. \quad (4.15)$$

Being (4.13) and (4.14) the composition of expected values and cross-entropies, both the definitions of the Wasserstein loss and of the hinge loss in the quaternion domain are straightforwardly derived by following the procedure shown for the adversarial loss in (4.11).

4.3.2 Quaternion Fully Connected Layers

In real-valued neural networks, fully connected layers are generally defined as:

$$\mathbf{y}_r = \phi(\mathbf{W}_r \mathbf{x}_r + \mathbf{b}_r) \quad (4.16)$$

where $\mathbf{W}_r \mathbf{x}_r$ performs the multiplication between the weight matrix \mathcal{D}_A and the input \mathbf{x}_r , J_1 is the bias and $|\mathbf{V}|$ is any activation function. In order to define the same operation in the quaternion domain, we represent the

quaternion weight matrix as $\mathbf{W} = \mathbf{W}_0 + \mathbf{W}_1\hat{i} + \mathbf{W}_2\hat{j} + \mathbf{W}_3\hat{k}$, the quaternion input as $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1\hat{i} + \mathbf{x}_2\hat{j} + \mathbf{x}_3\hat{k}$ and the quaternion bias as $\mathbf{b} = \mathbf{b}_0 + \mathbf{b}_1\hat{i} + \mathbf{b}_2\hat{j} + \mathbf{b}_3\hat{k}$. Therefore, \mathbf{Wx} in (4.16), is performed by a vector multiplication between two quaternions, as defined in (4.4), i.e., by the Hamilton product $\mathbf{W} \otimes \mathbf{x}$:

$$\begin{aligned}\mathbf{W} \otimes \mathbf{x} &= (\mathbf{W}_0\mathbf{x}_0 - \mathbf{W}_1\mathbf{x}_1 - \mathbf{W}_2\mathbf{x}_2 - \mathbf{W}_3\mathbf{x}_3) \\ &\quad + (\mathbf{W}_1\mathbf{x}_0 + \mathbf{W}_0\mathbf{x}_1 - \mathbf{W}_3\mathbf{x}_2 + \mathbf{W}_2\mathbf{x}_3)\hat{i} \\ &\quad + (\mathbf{W}_2\mathbf{x}_0 + \mathbf{W}_3\mathbf{x}_1 + \mathbf{W}_0\mathbf{x}_2 - \mathbf{W}_1\mathbf{x}_3)\hat{j} \\ &\quad + (\mathbf{W}_3\mathbf{x}_0 - \mathbf{W}_2\mathbf{x}_1 + \mathbf{W}_1\mathbf{x}_2 + \mathbf{W}_0\mathbf{x}_3)\hat{k}.\end{aligned}\tag{4.17}$$

Note that \mathbf{W} has dimensionality $\frac{1}{4}|\mathbf{W}_r|$ since it is composed of four submatrices $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2$ and \mathbf{W}_3 each one with 1/16 the dimension of \mathcal{D}_A . This is a key feature of QNNs since the results of the quaternion layer with product $\mathbf{W} \otimes \mathbf{x}$ has the same output dimension of the real-valued layer built upon $\mathbf{W}_r\mathbf{x}_r$ but with 1/4 the number of parameters to train. Note also that the submatrices are shared over each component of the quaternion input. The sharing allows the weights to capture internal relations among quaternion elements since each characteristics in a component will have an influence in the other components through the common weights. In this way the relations among components are preserved and captured by the weights of the network which is able to process inputs without losing intra-channel information. The bias \hat{s} is then added with a sum component by component. Finally, in QNNs the activation functions are applied to the input element-wise resulting in the so called *split activation functions*. That is, suppose to consider a common Rectified Linear Unit (ReLU) activation function and the quaternion $\mathbf{z} = \mathbf{W} \otimes \mathbf{x} + \mathbf{b}$, the final result \mathbf{y} of the layer will be:

$$\mathbf{y} = \text{ReLU}(\mathbf{z}_0) + \text{ReLU}(\mathbf{z}_1)\hat{i} + \text{ReLU}(\mathbf{z}_2)\hat{j} + \text{ReLU}(\mathbf{z}_3)\hat{k}.\tag{4.18}$$

4.3.3 Quaternion Convolutional Layers

Convolutional layers are generally applied to multichannel inputs, such as images. Supposing to deal with color images, real-valued neural networks break the structure of the input and concatenates the red, green and blue

(RGB) channels in a tensor. Quaternion-valued convolutions, instead, preserve the correlations among the channels and encapsulate the image in a quaternion as [27, 28, 39]:

$$\mathbf{x} = 0 + R\hat{i} + G\hat{j} + B\hat{k} \quad (4.19)$$

The image channels are the real coefficients of the imaginary units while the scalar part is set to 0. Encapsulating channels in a quaternion allows to treat them as a single entity and thus to preserve intra-channels relations. A visual explanation of the quaternion representation of color images is depicted in Fig. 4.1.

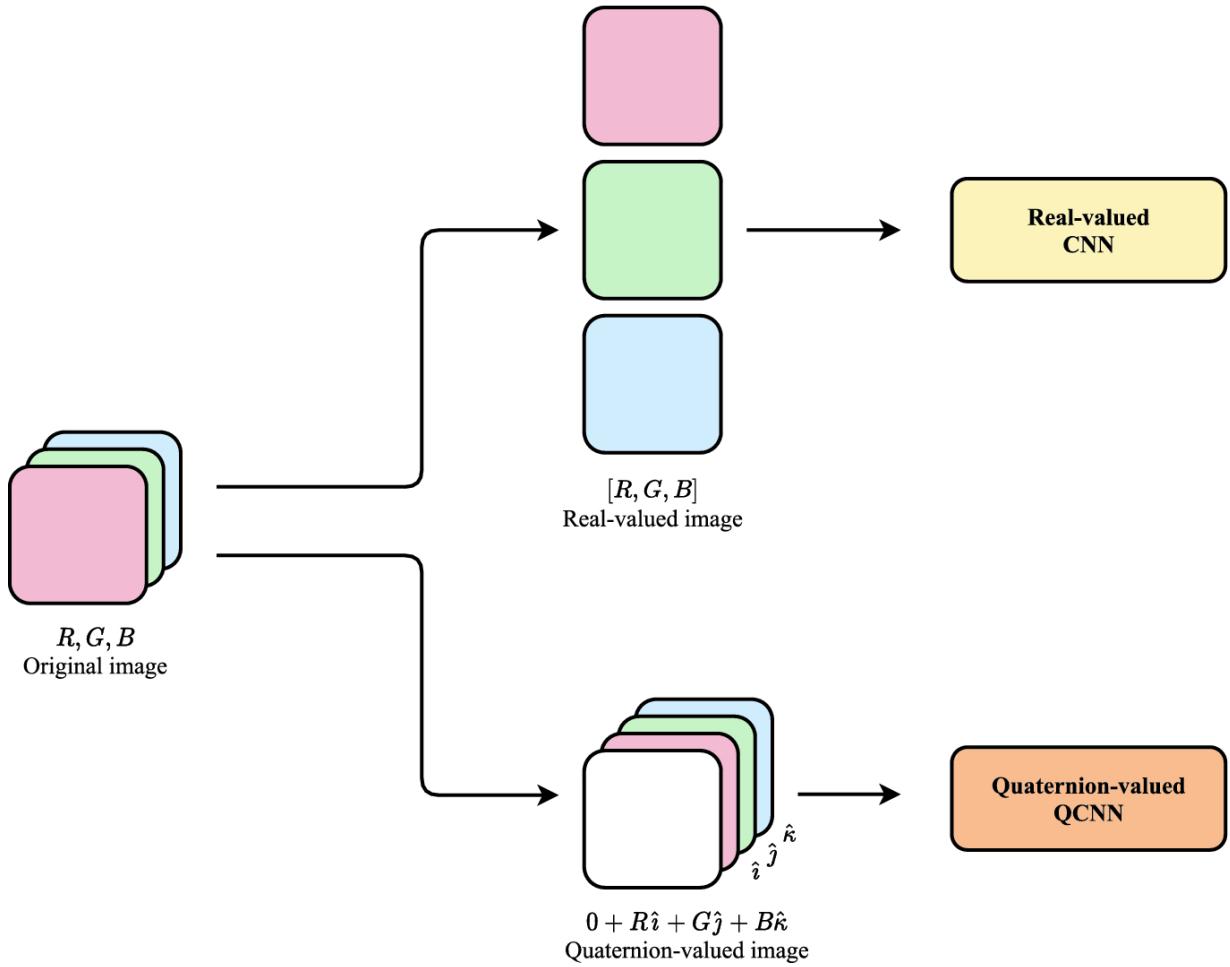


Fig. 4.1 Visual explanation of an R, G, B image processed by real and quaternion-valued networks. On the left, the original three-channels image. The image can be processed in two ways: (i) As a tensor of independent channels by a standard real-valued convolutional network as on the top of the figure. (ii) As a single entity, encapsulating it in a quaternion, and considering internal relations among channels as

quaternion-valued convolutional network does in the bottom of the figure. It is worth noting how the real-valued network does not consider any correlation among channels while quaternion ones preserve the relations among channels

Similarly to the definition of fully connected layers in the previous section, let us consider now a real-valued convolutional layer delineated by:

$$\mathbf{y} = \phi(\mathbf{W}_r * \mathbf{x}_r + \mathbf{b}_r) \quad (4.20)$$

where $*$ is the convolution operator. Quaternion convolutional layers are built with the same procedure depicted for fully connected layers thus considering the Hamilton product instead of the standard vector multiplication. That is, the convolution operator $\mathbf{W}_r * \mathbf{x}_r$ is replaced for quaternion weights and inputs with

$$\begin{aligned} \mathbf{W} * \mathbf{x} = & (\mathbf{W}_0 * \mathbf{x}_0 - \mathbf{W}_1 * \mathbf{x}_1 - \mathbf{W}_2 * \mathbf{x}_2 - \mathbf{W}_3 * \mathbf{x}_3) \\ & + (\mathbf{W}_1 * \mathbf{x}_0 + \mathbf{W}_0 * \mathbf{x}_1 - \mathbf{W}_3 * \mathbf{x}_2 + \mathbf{W}_2 * \mathbf{x}_3) \hat{i} \\ & + (\mathbf{W}_2 * \mathbf{x}_0 + \mathbf{W}_3 * \mathbf{x}_1 + \mathbf{W}_0 * \mathbf{x}_2 - \mathbf{W}_1 * \mathbf{x}_3) \hat{j} \\ & + (\mathbf{W}_3 * \mathbf{x}_0 - \mathbf{W}_2 * \mathbf{x}_1 + \mathbf{W}_1 * \mathbf{x}_2 + \mathbf{W}_0 * \mathbf{x}_3) \hat{k}. \end{aligned} \quad (4.21)$$

A visual explanation of the operation is shown in Fig. 4.2. While real-valued convolutional layer has to learn each filter independently, quaternion convolution allow the sharing of filters, thus reducing the number of free parameters to train.

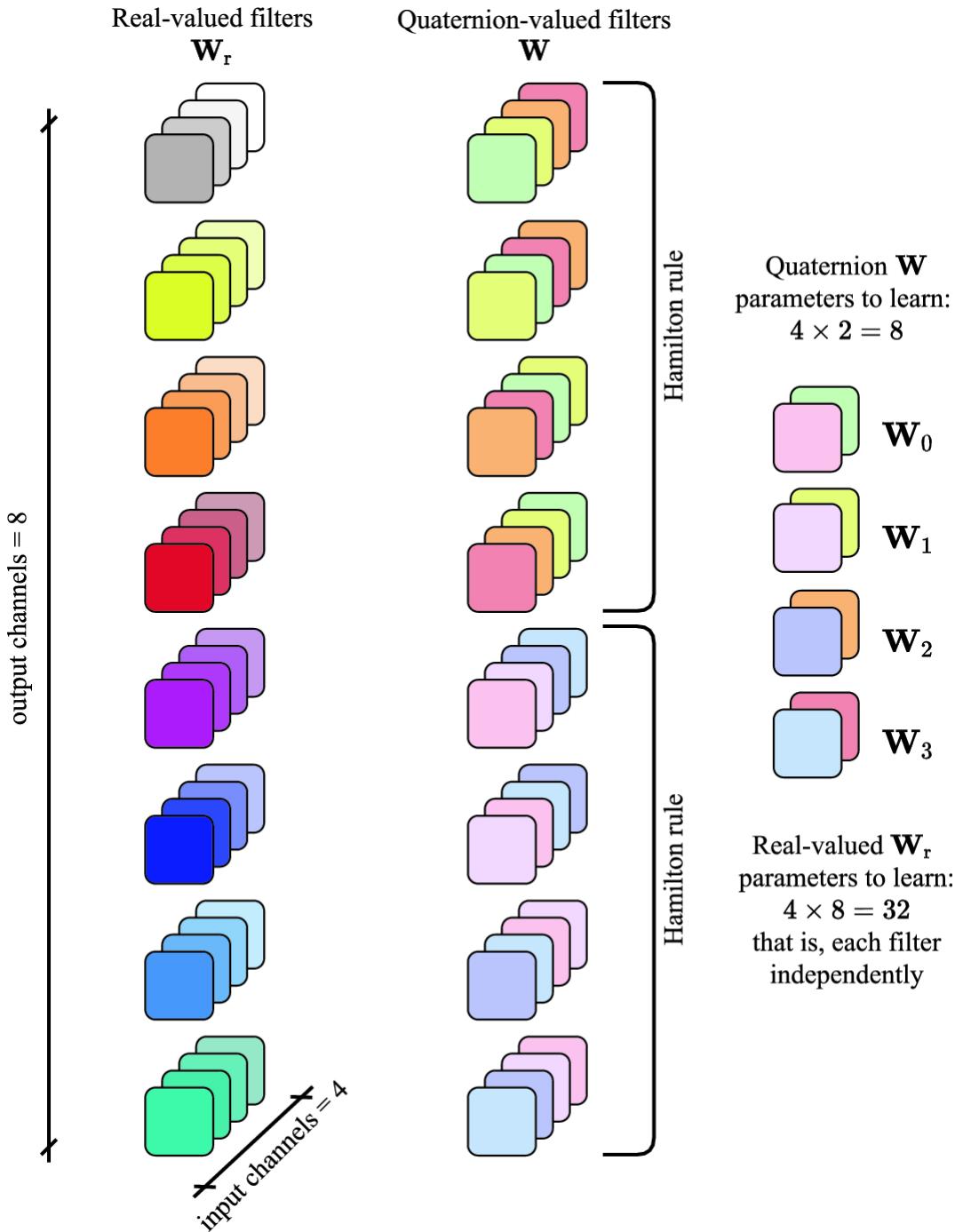


Fig. 4.2 Real and quaternion convolution filters for a layer l with 4 channels in input and 8 channels in output. On the left, the real-valued network has to learn each filter independently, thus resulting in $4 \times 8 = 32$ filters to learn. On the right, the quaternion filters are adjusted following the Hamilton product rule in (4.21) and shared among the 4 input components. As a result, the dimension of the weight matrix \mathbf{W} is the same as \mathcal{D}_A , but the quaternion filters to learn are just $4 \times 2 = 8$, that is,

the 75% of the real-valued filters. The so-composed filters are then employed to perform convolution

Note that in convolutional networks the sharing weights are crucial to properly process channels. Indeed, the RGB channels of an image interact with each other by resulting in combined colors, such as yellow or violet, through a representation of pixels in the color space. Nonetheless, real-valued networks are not able to catch these interactions since they process input channels separately, while QCNNs not only preserve the input design but also capture these relations through the sharing of weights. Actually, QCNNs perform a double learning: the convolution operator has the task of learning external relations among the pixels of the image, while the Hamilton product accomplishes the learning among the channels. Furthermore, as for linear layers, QCNNs are built with 1/4 the number of parameters with respect to their real-valued counterpart.

4.3.4 Quaternion Pooling Layers

Many neural networks make use of pooling layers, such as max pooling or average pooling, to extract high-level information and reduce input dimensions. As done before for previous layers in the quaternion domain, also this set of operations can be redefined in the quaternion domain.

The simplest examples of pooling in the hypercomplex domain are average and sum poolings. Indeed, applying these operations to each quaternion component, as done for split activation function, will not affect the final result [39]. A different approach must be defined, instead, for max pooling. Indeed, the maximum of a single component is not guaranteed by the maximum of all the other components. In order to address this issue, a guidance matrix has to be introduced. As in [39], the matrix is built through the quaternion amplitude and keeps trace of the maximum position, which is then mapped back to the original quaternion matrix in order to proceed with the pooling computation. However, max pooling operations are rarely employed in GANs, thus we only make use of average and sum pooling in our experiments.

4.3.5 Quaternion Batch Normalization

Introduced in [20], batch normalization (BN) has immediately became an ever-present module in neural networks. The idea behind BN is to normalize inputs to have zero mean and unit variance. This normalization helps the generalization ability of the network among different batches of training data and between train and test data distribution. Moreover, reducing the internal covariate shift remarkably improves the training speed, thus leading to a faster convergence of the model. For these reasons, also QNNs are endowed with batch normalization. However, different versions of this method were proposed in literature. An elegant whitening procedure based on the standard covariance matrix is introduced in [8]. In that paper, the Cholesky decomposition is used to compute the square root of the inverse of the covariance matrix, which is often intractable. The authors assert that approach ensures zero mean, unit variance and decorrelation among components. However, the covariance matrix is not able to recover the complete second-order statistics in the quaternion domain [4] and the decomposition requires heavy matrix calculations and computational time [19]. Another remarkable approach is introduced in [36], where the input is standardized computing the average of the variance of each component. Nevertheless, describing the second-order statistics of a signal in the quaternion domain needs meticulous computations and the approach in [36] is an approximation of the complete variance. Notwithstanding the approximation, this method allows to notably reduce computational time.

The proper theoretically procedure to reach a centered, decorrelated and unit-variance quaternion signal would be represented by performing a whitening procedure. Ideally, we should consider the covariance matrix and then decompose it to whiten the input in order to avoid computing the square root of the inverse which is often unfeasible. However, due to the interactions among components, second-order statistics for quaternion random variables are not completely described by the standard covariance matrix [4]. For this reason, the augmented covariance matrix should be considered instead. Such matrix is augmented with the complementary covariance matrices $\mathbf{C}_{\mathbf{qq}^i}$, $\mathbf{C}_{\mathbf{qq}^j}$, $\mathbf{C}_{\mathbf{qq}^k}$ that are the covariance matrices of the quaternion with its three perpendicular involutions $\mathbf{q}^i, \mathbf{q}^j, \mathbf{q}^k$. Thus, the

augmented covariance matrix, which completely characterizes the second-order information of the augmented quaternion vector \tilde{q} , is defined as:

$$\tilde{\mathbf{C}}_{\mathbf{qq}} = \mathbb{E} \{ \tilde{\mathbf{q}} \tilde{\mathbf{q}}^H \} = \begin{bmatrix} \mathbf{C}_{\mathbf{qq}} & \mathbf{C}_{\mathbf{qq}^i} & \mathbf{C}_{\mathbf{qq}^j} & \mathbf{C}_{\mathbf{qq}^k} \\ \mathbf{C}_{\mathbf{qq}^i}^H & \mathbf{C}_{\mathbf{q}^i \mathbf{q}^i} & \mathbf{C}_{\mathbf{q}^i \mathbf{q}^j} & \mathbf{C}_{\mathbf{q}^i \mathbf{q}^k} \\ \mathbf{C}_{\mathbf{qq}^j}^H & \mathbf{C}_{\mathbf{q}^j \mathbf{q}^i} & \mathbf{C}_{\mathbf{q}^j \mathbf{q}^j} & \mathbf{C}_{\mathbf{q}^j \mathbf{q}^k} \\ \mathbf{C}_{\mathbf{qq}^k}^H & \mathbf{C}_{\mathbf{q}^k \mathbf{q}^i} & \mathbf{C}_{\mathbf{q}^k \mathbf{q}^j} & \mathbf{C}_{\mathbf{q}^k \mathbf{q}^k} \end{bmatrix} \quad (4.22)$$

where $(\cdot)^H$ is the conjugate transpose operator. The formulation in (4.22) recovers the complete statistical information of a general quaternion signal. Thus, the theoretically procedure should be delineated as:

$$\bar{\mathbf{x}} = \tilde{\mathbf{C}}_{\mathbf{qq}}^{-1/2} (\mathbf{x} - \mathbb{E} \{ \mathbf{x} \}) \quad (4.23)$$

or substituting the inverse square root $\tilde{\mathbf{C}}_{\mathbf{qq}}^{-1/2}$ with a decomposition of it.

However, the construction of the augmented covariance matrix may be quite difficult and computational expensive due to the computation of each sub-covariance matrix. Moreover, $\tilde{\mathbf{C}}_{\mathbf{qq}}^{-1/2}$ includes skew-symmetric sub-matrices [4], that make the decomposition more difficult.

In order to simplify the calculation of (4.22) and make it more feasible for practical applications, a particular case can be considered by leveraging the \mathbb{Q} -properness property [4, 13, 37]. The \mathbb{Q} -properness entails that the quaternion signal is not correlated with its involutions, implying vanishing complementary covariance matrices, i.e., $\mathbf{C}_{\mathbf{qq}^i} = \mathbf{C}_{\mathbf{qq}^j} = \mathbf{C}_{\mathbf{qq}^k} = 0$. Also, for \mathbb{Q} -proper random variables the following relation holds:

$$\text{var} \{ \mathbf{q}_c \} = \mathbb{E} \{ \mathbf{q}_c^2 \} = \sigma^2, \quad c = \{0, 1, 2, 3\} \quad (4.24)$$

Thus, considering a \mathbb{Q} -proper quaternion, the covariance in (4.22) becomes:

$$\tilde{\mathbf{C}}_{\mathbf{qq}} = \mathbb{E} \{ \tilde{\mathbf{q}} \tilde{\mathbf{q}}^H \} = \begin{bmatrix} \mathbf{C}_{\mathbf{qq}} & 0 & 0 & 0 \\ 0 & \mathbf{C}_{\mathbf{q}^i \mathbf{q}^i} & 0 & 0 \\ 0 & 0 & \mathbf{C}_{\mathbf{q}^j \mathbf{q}^j} & 0 \\ 0 & 0 & 0 & \mathbf{C}_{\mathbf{q}^k \mathbf{q}^k} \end{bmatrix} = 4\sigma^2 \mathbf{I} \quad (4.25)$$

Assuming \mathbb{Q} -properness for a random variable saves a lot of calculations and computational costs. Notwithstanding the theoretical correctness of the above defined approach, quaternion batch normalization (QBN) techniques adopted so far in the literature relies in some approximations.

We assume the input signal is \mathbb{Q} -proper, thus we consider the covariance in (4.25) and build the normalization as follows:

$$\bar{\mathbf{x}} = \frac{\mathbf{x} - \mu_q}{\sqrt{\text{var}\{\mathbf{x}\} + \epsilon}} = \frac{\mathbf{x} - \mu_q}{\sqrt{4\sigma^2 + \epsilon}} \quad (4.26)$$

where μ_q is the quaternion input mean value, which is a quaternion itself, and it is defined as:

$$\mu_q = \frac{1}{N} \sum_{n=1}^N q_{0,n} + q_{1,n}\hat{i} + q_{2,n}\hat{j} + q_{3,n}\hat{k} = \bar{q}_0 + \bar{q}_1\hat{i} + \bar{q}_2\hat{j} + \bar{q}_3\hat{k}. \quad (4.27)$$

The final output is computed as follows:

$$\text{QBN}(\mathbf{x}) = \gamma \bar{\mathbf{x}} + \beta \quad (4.28)$$

where β is a shifting quaternion parameter and η is a scalar parameter.

In conclusion, the QBN proposed by [8] is an elegant approximation, nevertheless it is not able to catch the complete second-order statistics information, while requiring heavy computations [19]. Thus, we believe that considering \mathbb{Q} -proper signals, which are indeed very frequent, is a good approximation which also extremely reduces the computational requirements. For our experiments, we adopt the method represented by (4.28).

4.3.6 Quaternion Spectral Normalization

Among the wide variety of proposed techniques to stabilize GANs training, the spectral normalization (SN) [26] is one of the most widespread method. Previously, the crucial importance of having a Lipschitz-bounded discriminator function was introduced in [1, 16]. Lately, it was proved that no restriction on the discriminator space leads to the *gradient uninformativeness* problem [42]. This means that the gradient of the optimal discriminative function has no information about the real distribution, thus providing useless feedbacks to the generator. Forcing a function to be

Lipschitz continuous means controlling how fast it increases and bound the gradients, thus mitigating gradient explosions [11, 42]. In [1], a method based on weight clipping was proposed to force the discriminator to be 1-Lipschitz. Later, such approach has been improved by adding a gradient penalty (GP) that constraints the gradient norm to be at most 1 [16]. The latter method is reproposed in several state-of-the-art GANs and combined with other regularization techniques to improve performance, as suggest [25]. However, being built on the gradients with respect to the inputs, the gradient penalty cannot impose a regularization outside the support of the fake and real data distribution. Moreover, it requires consistent computations. The spectral normalization, instead, directly operates on the weights of the network being free of the support limit and its computations is faster than other methods [26]. It aims at controlling the Lipschitz constant of the discriminator by constraining the spectral norm of each layer.

A generic function f is K -Lipschitz continuous if, for any two points π_i , π_i , the following property holds:

$$\frac{\|f(x_1) - f(x_2)\|}{|x_1 - x_2|} \leq K \quad (4.29)$$

being $\|\cdot\|$ the l_2 norm. The Lipschitz norm $\frac{1}{4}|\mathbf{W}_r|$ of a function f is equal to $P_G(\mathbf{x}, y | s = 1)$, where $\sigma(\cdot)$ is the spectral norm of the matrix in input, that is, the largest singular value of the matrix.

For a generic linear layer $f(h) = \mathbf{W}\mathbf{x} + \mathbf{b}$, the Lipschitz norm is:

$$\|f\|_{\text{Lip}} = \sup_h \sigma(\nabla f(h)) = \sup_h \sigma(\mathbf{W}) = \sigma(\mathbf{W}) \quad (4.30)$$

Assuming the Lipschitz norm of each layer activation being equal to 1, constraint that is satisfied for many popular activation functions including ReLU and Leaky ReLU [26], we can apply the Lipschitz bound to the whole network by following $\|f_1 \circ f_2\|_{\text{Lip}} \leq \|f_1\|_{\text{Lip}} \cdot \|f_2\|_{\text{Lip}}$.

Finally, the SN is defined as

$$\bar{W}_{SN}(\mathbf{W}) = \frac{\mathbf{W}}{\sigma(\mathbf{W})} \quad (4.31)$$

and it ensures that the weight matrix \mathbf{W} always satisfies the constraint $P_G(\mathbf{x}, y|s)$. In [26] the authors underline that applying the original singular value decomposition algorithm to compute $\sigma(\mathbf{W})$ may result in an extremely heavy algorithm. To address the computational complexity, they suggest to estimate the largest singular value via the power iteration method.

In order to control the Lipschitz constraint in a QGAN, in this section we explore two methods to define the spectral normalization in the quaternion domain. A first approach aims at normalizing the weights \mathbf{W} by operating on each submatrix $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ independently, by computing the spectral norm separately. That is, through the power iteration method as above, we compute $\sigma_0(\mathbf{W}_0), \sigma_1(\mathbf{W}_1), \sigma_2(\mathbf{W}_2), \sigma_3(\mathbf{W}_3)$ and then normalize each submatrix with the corresponding norm. This method forces each submatrix to have spectral norm equal to 1. However, it never takes the whole weight matrix \mathbf{W} into account. Moreover, the relations among the components of the quaternion matrix is not considered, losing the characteristic property of QNNs.

The second method, similarly to the real-valued SN, normalizes the whole matrix \mathbf{W} together, by imposing the constraint to the complete matrix and not to the singular submatrices. Therefore, the spectral norm is computed by taking the complete weight matrix into account and considering the relations among the quaternion components. However, while the spectral norm is computed as in (4.30), the normalization step is applied differently from the SN in (4.31). Instead of normalizing the whole matrix as in (4.31), being the weight matrix \mathbf{W} designed by a composition of the submatrices $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$, we can leverage this quaternion setup to save computational costs and normalize each submatrix $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$. The normalized submatrices $\bar{\mathbf{W}}_{0,QSN}, \bar{\mathbf{W}}_{1,QSN}, \bar{\mathbf{W}}_{2,QSN}, \bar{\mathbf{W}}_{3,QSN}$ will result in a normalized weight matrix $\bar{\mathbf{W}}_{QSN}(\mathbf{W})$ with a more efficient computation than normalizing the full matrix \mathbf{W} .

An empirical comparison between the two methods is reported in Sect. 4.5. We investigate the two techniques in a plain QGAN and prove that the latter approach is stabler and gains better performance in both the considered benchmarks. We deem it more appropriate both theoretically and

empirically and we use it in our further experiments. From now on, we refer to such approach as the quaternion spectral normalization (QSN).

4.3.7 Quaternion Weight Initialization

Weight initialization has often a crucial role in networks convergence and in the reduction of the risk of vanishing or exploding gradients [9]. This procedure becomes even more important when dealing with quaternion weights. Indeed, due to the interactions among the elements of the quaternion, the initialization cannot be random nor component-aware. For these reasons, an appropriate initialization has to be introduced.

First, consider a weight matrix \mathbf{W} with $E\{|\mathbf{W}|\} = 0$. The initialization is based on a normalized pure quaternion u generated for each weight submatrix from a uniform distribution in $[0, 1]$. By using the polar form of a quaternion, we can define the initialization of the weight matrix as

$$\mathbf{W} = |\mathbf{W}|e^{u\theta} = |\mathbf{W}|(\cos(\theta) + u \sin(\theta)) \quad (4.32)$$

where each matrix component is initialized as

$$\begin{aligned} \mathbf{W}_0 &= \phi \cos(\theta) \\ \mathbf{W}_1 &= \phi u_1 \sin(\theta) \\ \mathbf{W}_2 &= \phi u_2 \sin(\theta) \\ \mathbf{W}_3 &= \phi u_3 \sin(\theta) \end{aligned} \quad (4.33)$$

where the angle \hat{x} is randomly generated in the interval $[-\pi, \pi]$ and ϕ is randomly sampled in the interval of the standard deviation around zero $\mathcal{Q}(c|x)$. The standard deviation is set according to the initialization method chosen, either [9] or [17]. In the first case, we set $\sigma = 1/\sqrt{2(n_{in} + n_{out})}$ whereas in the latter we set $\sigma = 1/\sqrt{2n_{in}}$. In both the equations, n_{in} is the number of neurons in the input layer and n_{out} the number of neurons in the output layer. The variance of \mathbf{W} can be written as:

$$\text{var } \{\mathbf{W}\} = E\{|\mathbf{W}|^2\} - E\{|\mathbf{W}|\}^2. \quad (4.34)$$

However, similarly to the QBN in the previous section, in order to reduce the computations, the component $E\{|\mathbf{W}|\}^2$ can be considered equal to 0

[28, 29]. This is equivalent to considering a \mathbb{Q} -proper quaternion signal whose augmented covariance matrix has off-diagonal elements equal to 0 and trace equal to \mathcal{D}_A . Consequently, the variance is computed by considering only the first term of (4.34) as:

$$\text{var } \{\mathbf{W}\} = E \{|\mathbf{W}|^2\} = 4\sigma^2 \quad (4.35)$$

4.3.8 Training

The forward phase of a QNN is the same as its real-valued counterpart. Therefore, the input flows from the first to the last layer of the network. It may be interesting to note that in eq. (4.17) the order of the weight and the input can be inverted, thus changing the output of the product, resulting in an inverted QNN [28, 29]. For what concerns the backward phase, it worth mentioning that the gradient of a general quaternion loss function \mathcal{X} is computed for each component of the quaternion weight matrix \mathbf{W} as in the ensuing equation:

$$\frac{\delta \mathcal{L}}{\delta \mathbf{W}} = \frac{\delta \mathcal{L}}{\delta \mathbf{W}_0} + \frac{\delta \mathcal{L}}{\delta \mathbf{W}_1} \hat{i} + \frac{\delta \mathcal{L}}{\delta \mathbf{W}_2} \hat{j} + \frac{\delta \mathcal{L}}{\delta \mathbf{W}_3} \hat{k}. \quad (4.36)$$

Then, the gradient is propagated back following the chain rule. Indeed, as defined in [28], the backpropagation of quaternion neural networks is just an extension of the method for their real-valued counterpart. Consequently, QNNs can be easily trained as real-valued networks via backpropagation.

4.4 GAN Architectures in the Quaternion Domain

The previous section describes the main blocks and the framework to build and train a GAN in the quaternion domain. In this section we go further, presenting the complete definition of a plain QGAN in Sect. 4.4.1 and of an advanced state-of-the-art QGAN composed of complex blocks in Sect. 4.4.2. First, in order to setting up a QGAN, each input, weight, bias and output has to be manipulated to become a quaternion. Therefore, weight matrices are initialized as composed by the four submatrices, similarly to (4.17) and (4.21). Real-valued operations such as multiplications or convolutions in the networks are replaced with their quaternion counterparts, completing the redefinition of the layers in the quaternion

domain. The input is handled as a quaternion and processed as a single entity. For images, a pure quaternion is considered as in (4.19), while for other kind of multidimensional signals, the scalar part is considered too. The initialization of the weights is then applied following the description in Sect. 4.3.7. This accurate definition of QGANs grants to design a model with a fewer number of free parameters with respect to the same real-valued model and consequently to save memory and computational requirements.

4.4.1 Vanilla QGAN

In the original GAN [10], both the generator (G) and the discriminator (D) are defined by fully connected layers. Due to the limited expressivity of this design with complex data such as images, in [30] the authors propose to replace dense layers with more suitable operations for this kind of data and to build G and D by stacking several convolutional layers. State-of-the-art GANs are based on the deep convolutional GAN (DCGAN) [30]. In particular, the DCGAN increases the spatial dimensionality by means of transposed convolutions in the generator and decreases it in the discriminator with convolutions. Furthermore, this architecture defines batch normalization in every layer except for the last layer of G and for the first layer of D , in order to let the networks learn the correct statistics of the data distribution.

By redefining the DCGAN in the quaternion domain (QDCGAN) it is possible to explore the potential of the quaternion algebra in a simple GAN framework. The QDCGAN generator is defined by an initial quaternion fully connected layer and then by interleaving quaternion transposed convolutions with quaternion batch normalization and split ReLU activation functions except for the last layer which ends up with a split Tanh function. The discriminator has the same structure of the generator but with quaternion transposed convolutions replaced by quaternion convolutions to decrease the dimensionality and with a final fully connected quaternion layer that returns as output the real/fake decision by means of a sigmoid split activation. The QDCGAN, as its real-valued counterpart, optimizes the original loss in (4.11) (Fig. 4.3).

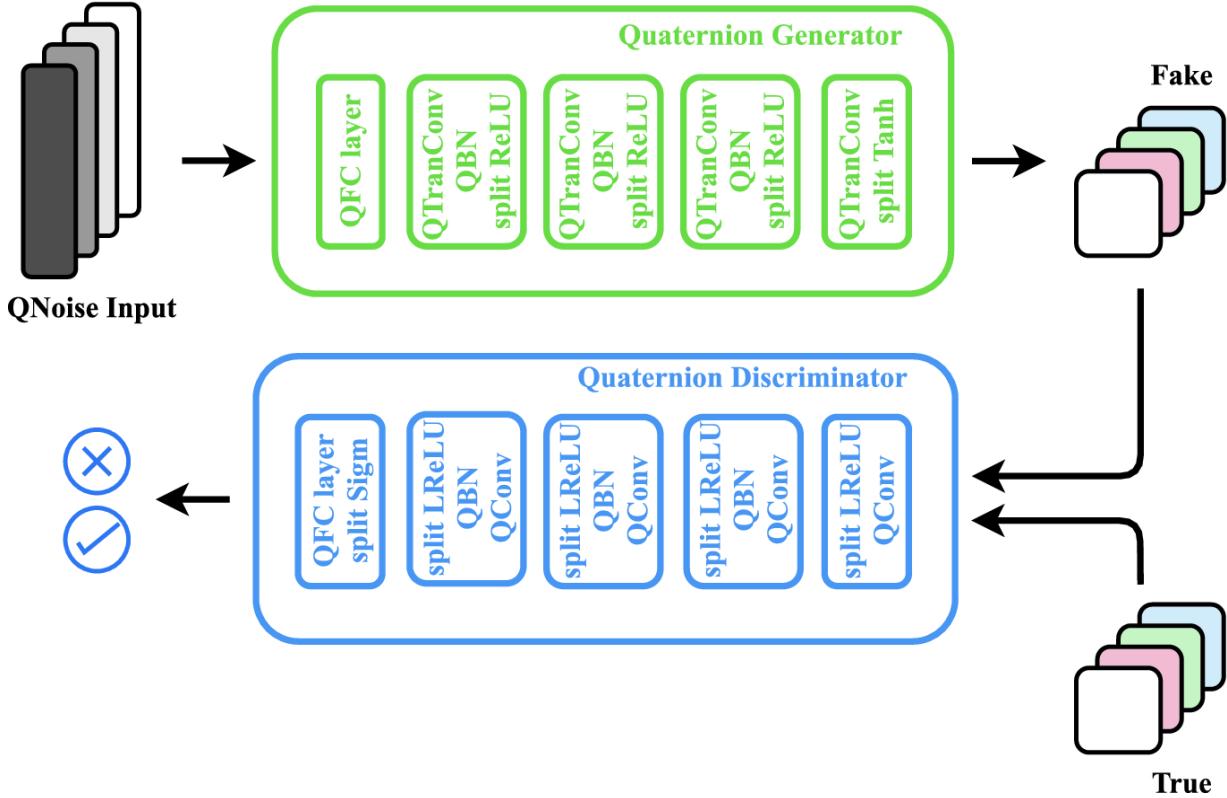


Fig. 4.3 Quaternion Vanilla GAN architecture. Each parameter including inputs, weights and outputs is a quaternion. The generator (green network) takes a quaternion noise signal and generates a batch of quaternion images with four channels. The discriminator tries to distinguish between fake and real quaternion samples exploiting the properties of quaternion algebra

4.4.2 Advanced QGAN

The above presented Vanilla QGAN is just a plain example to give a general idea on how to build GANs in the quaternion domain. In this section, we consider a more advanced model, the spectral normalized GAN (SNGAN) [26] and we present the steps to define its quaternion counterpart.

The quaternion spectral normalized GAN (QSNGAN) is trained in an adversarial fashion through the hinge loss defined in (4.14) and (4.15) for the discriminator and generator respectively, as suggested in [3, 26]. The overall architecture of the model is inspired by [3]. Both the generator and the discriminator networks are characterized by quaternion convolutional layers in order to leverage the properties of the Hamilton product. To mitigate the vanishing gradient problem and obtain better performance, a

series of residual blocks with upsampling in the generator and downsampling in the discriminator can be adopted [26]. A scheme of the residual block of the proposed QSNGAN is depicted in Fig. 4.4. The discriminative network plays a crucial role in GANs training, thus it is more complex with respect to the generator network. It takes in input the two sets of quaternion images with four channels in a first residual block, as illustrated in Fig. 4.5. The output of the block is the decision on whether they come from the fake or real distribution.

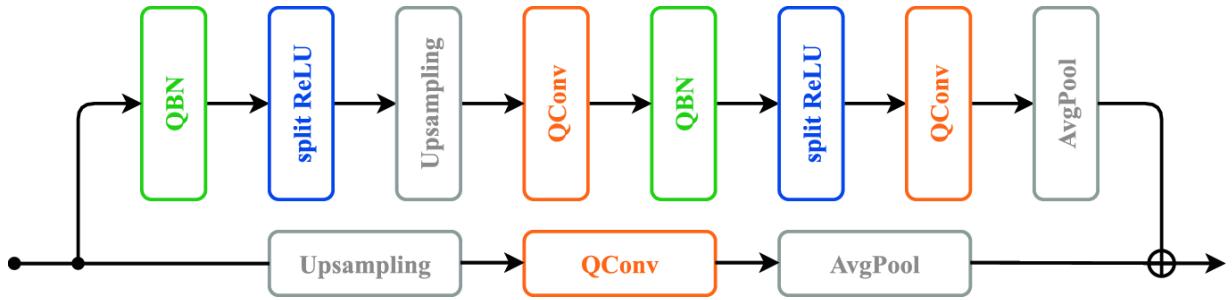


Fig. 4.4 Quaternion residual block (QResBlock) architecture inspired by [26] and redefined in the quaternion domain. QBN is omitted in the discriminator network and replaced by QSN. Grey blocks means they are used exclusively in the generator or in the discriminator. The generator considers the upsampling steps in the residual and in the shortcut pass while the discriminator the average pooling ones, except for the last residual block of the discriminator which keep the dimension invariant

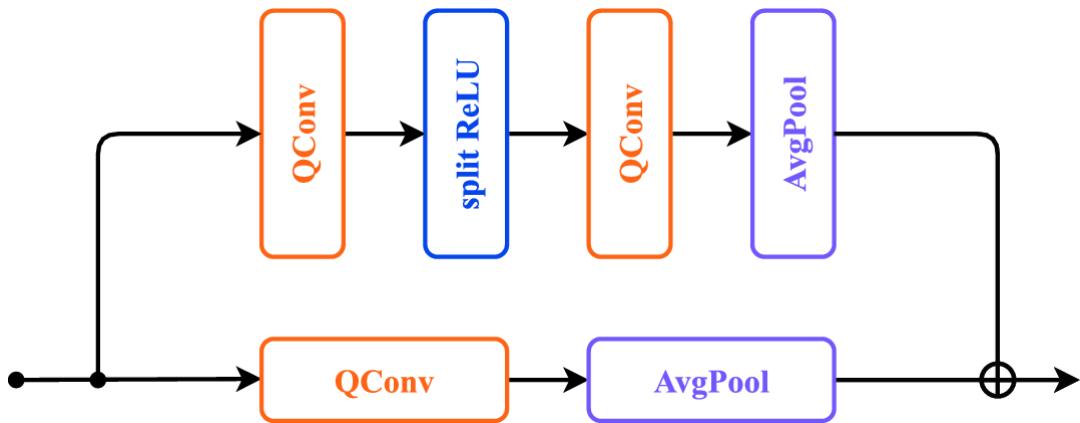


Fig. 4.5 First discriminator quaternion residual block (First QResBlock) with quaternion convolutions and average pooling layers to downsample the input

In order to guarantee a fair comparison with the SNGAN, we consider a real-valued noise signal in input to the generator and handle it with an

initial real-valued fully connected layer. The output of the first layer is then encapsulated in a quaternion signal with a procedure similar to the one considered in Sect. 4.3.3 to handle colored images. The signal is then processed by the quaternion generator up to the last layer, which generates the four-channel fake image. The original SNGAN considers batch normalization in the generator and spectral normalization in the discriminator. We keep the same structure and consider the proposed QBN in (4.28) for the first network and the QSN introduced in Sect. 4.3.6 for the discriminator. In particular, we exploit the QSN with spectral norm computed over the whole weight matrix, which is theoretically proper and ensures stabler results.

The definition of the SNGAN in the quaternion domain allows to save parameters, as we will explore in the next section. Moreover, the QSNGAN, processing the channels as a single entity through the quaternion convolutions based on the Hamilton product, is able to capture the relations among them and to capture any intra-channel information, which the SNGAN, conversely, loses. The latter property turns into an improved generation ability by the QSNGAN that properly grasps the real data distribution. The architecture of the proposed QSNGAN is reported in Fig. 4.6. In the scheme, the forward phase flows from left to right for the top network (quaternion generator) and from right to left for the second network (quaternion discriminator).

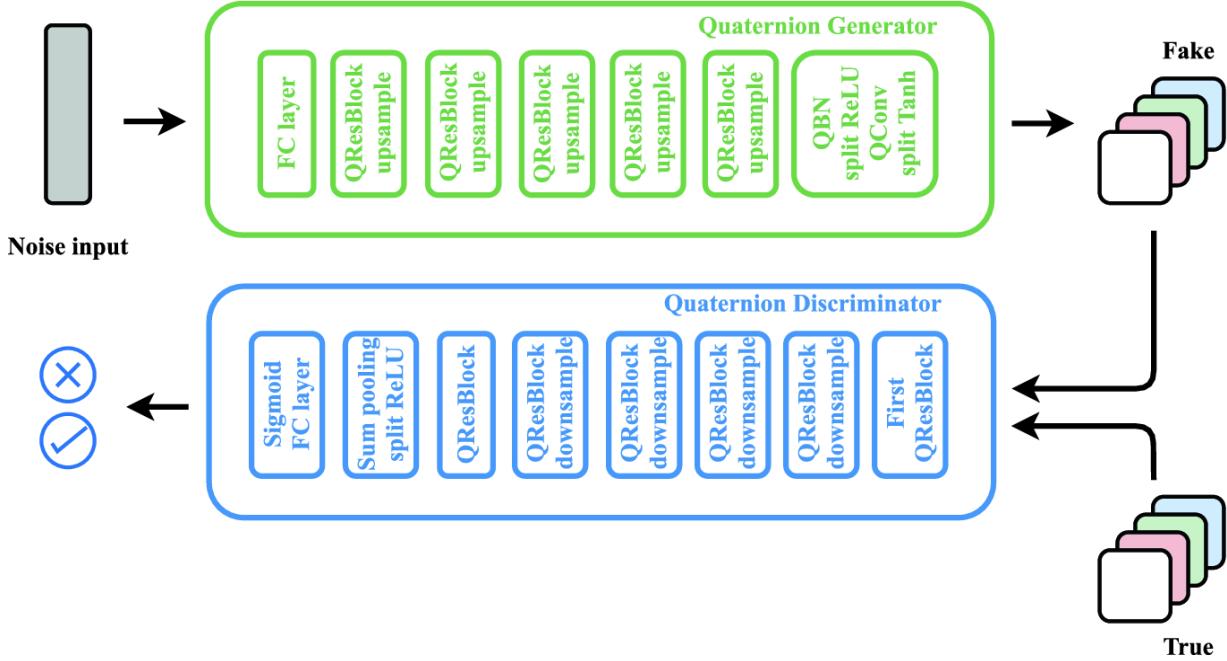


Fig. 4.6 QSNGAN architecture schema. The generator network (top) takes in input a real-valued signal, processes it with a a fully connected layer and then encapsulates it in a quaternion signal. The residual blocks are depicted in Fig. 4.4. The generator outputs a quaternion-valued sample of images that, together with a sample from the real distribution, goes to the input of the discriminator network (bottom). It handles the samples through a series of residual blocks (the first one is illustrated in Fig. 4.5, the other ones in Fig. 4.4) up to the last layer which outputs the real or fake decision

4.4.3 Evaluation Metrics

In order to evaluate the performance of the generative networks, we consider two objective metrics, the Fréchet Inception Distance (FID) [18] as main metric, as it is more consistent with human evaluations, and the Inception Score (IS) [31]. The Fréchet inception distance embeds the generated and the real samples into the Inception convolutional features and models the two distributions as Gaussian signals evaluating the empirical means μ_g, μ_{data} and covariances $z \in p_z(z)$ and then computes the Fréchet distance as:

$$\text{FID}(p_g, p_{\text{data}}) = \|\mu_g - \mu_{\text{data}}\| + \text{Tr}(\mathbf{C}_g + \mathbf{C}_{\text{data}} - 2(\mathbf{C}_g \mathbf{C}_{\text{data}})^{1/2}) \quad (4.37)$$

where (\hat{x}, \hat{y}) refers to the trace operator. Being the FID a distance between real and fake distributions, the lower the FID value, the better the generated samples.

Instead, the IS considers the inception model to get the conditional distribution $p(y|x)$ of the generated samples. IS expects the conditional distribution to have low entropy since the images represent meaningful objects, while the marginal distribution $p(y)$ should have high entropy due to the diversity among the samples. It is defined as:

$$\text{IS}(p_g) = \exp \left(\mathbb{E}_{x \sim p_g} \{ \text{KL}[p(y|x) || p(y)] \} \right) \quad (4.38)$$

where KL is the Kullback–Leibler divergence. Conversely to the FID, higher IS values stands for better generated samples. However, IS has some drawbacks since it does not consider the true data distribution and, moreover, it is not able to detect mode collapse, thus we consider the FID score as main metric and the IS in support to it.

4.5 Experimental Evaluation

In order to evaluate the effectiveness of our proposed approach, we conduct a collection of experiments on the unsupervised image generation task. We take two datasets into account: the CelebA-HQ [21], which contains 27 k images for training and 3 k images for testing, and the 102 Oxford Flowers, which contains approximately 7 k images for training and a few less than 1 k images for testing. We reshape the samples of both the dataset to 128×128 and then test the real-valued SNGAN and the proposed QSNGAN. We use the Adam optimizer and keep the same hyperparameters fixed as in [26], i.e., learning rate equal to 0.0002, and the optimizer parameters equal to D_1, G, η , D_1, G, η . We just vary the number of critic iterations, considering two experiments with critic iterations equal to 1 and then equal to 5 in order to better investigate the behavior of our QSNGAN, which may have a different balance between generator and discriminator networks with respect to the SNGAN. In every experiment, we fix the batch size to 64 and we perform 100 k training iterations for the CelebA-HQ and 50 k for the 102 Oxford Flowers. We have also considered to endow the SNGAN and the QSNGAN with a gradient penalty, as in (4.13), but we did not notice any improvement in the experiments, thus meaning that both the SN and the QSN adequately control the discriminator to be Lipschitz continuous.

The QSNGAN generator is a quaternion convolutional network as in Fig. 4.6. The initial fully connected layer, which takes the noise of size 128 in input, is composed of $4 \times 4 \times 1024$ neurons. The following quaternion residual blocks illustrated in Fig. 4.4 stack 1024, 512, 256, 128 and 64 filters. This means that, as an example, the first residual block is built by interleaving QBNs, split ReLUs and quaternion convolutions with 1024 kernels and an upsampling module with scale factor equal to 2. Further, at the end of the last residual connection, we stack a QBN, a split ReLU activation function and a final quaternion convolutional layer of dimension 64 to refine the output image, which is then passed to a split Tanh function to bound it in the interval (X, S) . For each quaternion convolution, we fix the kernel size to 3 and the stride and the padding to 1. Conversely, the shortcut in the residual block is composed of an upsampling module and a quaternion convolution with kernel size equal to 1 and null padding. The network built through this procedure has less than 10 M of free parameters with respect to the 32 M parameters of its real-valued counterpart. This means that the checkpoint for inference saves more than the 75% of disk memory, as shown in Table 4.1.

Table 4.1 Summary of number of networks parameters and memory requirements for real-valued SNGAN and its quaternion-valued counterpart QSNGAN models for CelebA-HQ. The proposed method saves more than the 75% of total free parameters and memory disk for model checkpoints

Model	#Params G	#Params D	#Params Tot	Disk Memory ^a
SNGAN	32,150,787	29,022,213	61,173,000	~ 115 GB
QSNGAN	9,631,204	7,264,901	16,896,105	~ 35 GB

^aGenerator checkpoint for inference

The QSNGAN discriminator is still a quaternion convolutional network as in Fig. 4.6, but it is slightly more complex. At the beginning, the real images are encapsulated in a quaternion as depicted in Sect. 4.3.3, resulting in a batch of four-channel images. Obviously, the images generated by the generator network are already comprised of four channels and defined as quaternions.

Table 4.2 Results summary for the 128×128 CelebA-HQ and 102 Oxford Flowers datasets. The proposed QSNGAN obtains a lower FID in each dataset considered. The values of the IS support the FID results. According to the objective metrics, the proposed QSNGAN generates more visually pleasant and diverse samples with respect to the real-valued baseline counterpart. The QSNGAN seems to be more robust to the choice of the hyper-parameter regarding the discriminator iterations (Critic iter) while the real-valued model fails when changing the original setting which fixes the parameter equal to 5

		FID ↓		IS ↓	
Model	Critic iter	CelebA-HQ	102 Oxford flowers	CelebA-HQ	102 Oxford flowers
SNGAN	1	>200 ^a	>200 ^a	<2.000 ^a	2.797 ± 0.196
	5	34.483	<u>165.058</u>	<u>2.032 ± 0.062</u>	<u>2.977 ± 0.146</u>
QSNGAN	1	29.417	175.484	2.249 ± 0.164	2.754 ± 0.256
	5	<u>33.068</u>	115.838	2.026 ± 0.082	3.000 ± 0.141

^a Discriminator collapses and training fails, thus metrics results are not comparable

The first residual block of the discriminator in Fig. 4.5 is a spectrally-normalized quaternion convolution block with 64 3×3 filters and split ReLU activation functions. The shortcut, instead, as for the generator network, is a 3×3 quaternion convolution with padding equal to 0. In this case, however, both the residual and the shortcut part ends with a 3×3 split average pooling. The images flow then to a stack of five residual blocks built as in Fig. 4.4 with, respectively, 128, 256, 512, 1024 and 1024 filters. Nevertheless, the residual section of each block has a split average pooling to operate downsampling and the shortcut is comprised of a quaternion convolution and another average pooling. The downsampling procedure is applied in each residual block except for the last one, which is a refiner and leaves the dimensionality unchanged. Every weight is normalized through the QSN introduced in Sect. 4.3.6. The configurations for kernel size, stride and padding are the same of the generator. At the end of the residual block stack, we apply a split ReLU and a global sum pooling before passing the batch to the final spectrally-normalized fully connected layer which, by means of a sigmoid, returns the real/fake decision. As for the generator, also the quaternion discriminator allows to save parameters

while learning the internal relations among channels. This saving is underlined in Table 4.1, which reports the exact number of parameters for the quaternion model and the real-valued one. The quaternion GAN can obtain equal or better results when trained with less parameters since it leverages the properties of quaternion algebra, including the Hamilton product, that allow to capture also the relations among channels and catch more information on the input. Consequently, the training procedure needs less parameters to learn the real distribution and to generate images from it.

The objective evaluation is reported in Table 4.2. We perform the computations of FID and IS on the test images (3 k for the CelebA-HQ and slightly less than 1 k for the 102 Oxford Flowers). As shown in Table 4.2, the proposed method stands out in the generation of samples from both the dataset according to the metrics considered. Moreover, the two QSNGANs with critic iterations 1 and 5 score a lower FID with respect to the best configuration of the SNGAN model. The proposed method performs better with one critic per generator iterations, while the real-valued model fails with this configuration. Overall, the QSNGAN seems to be more robust to the choice of the critic iterations with respect to the SNGAN, which is more fragile. The IS strengthen the results obtained with the FID, as it reports higher scores for the proposed method in every dataset.

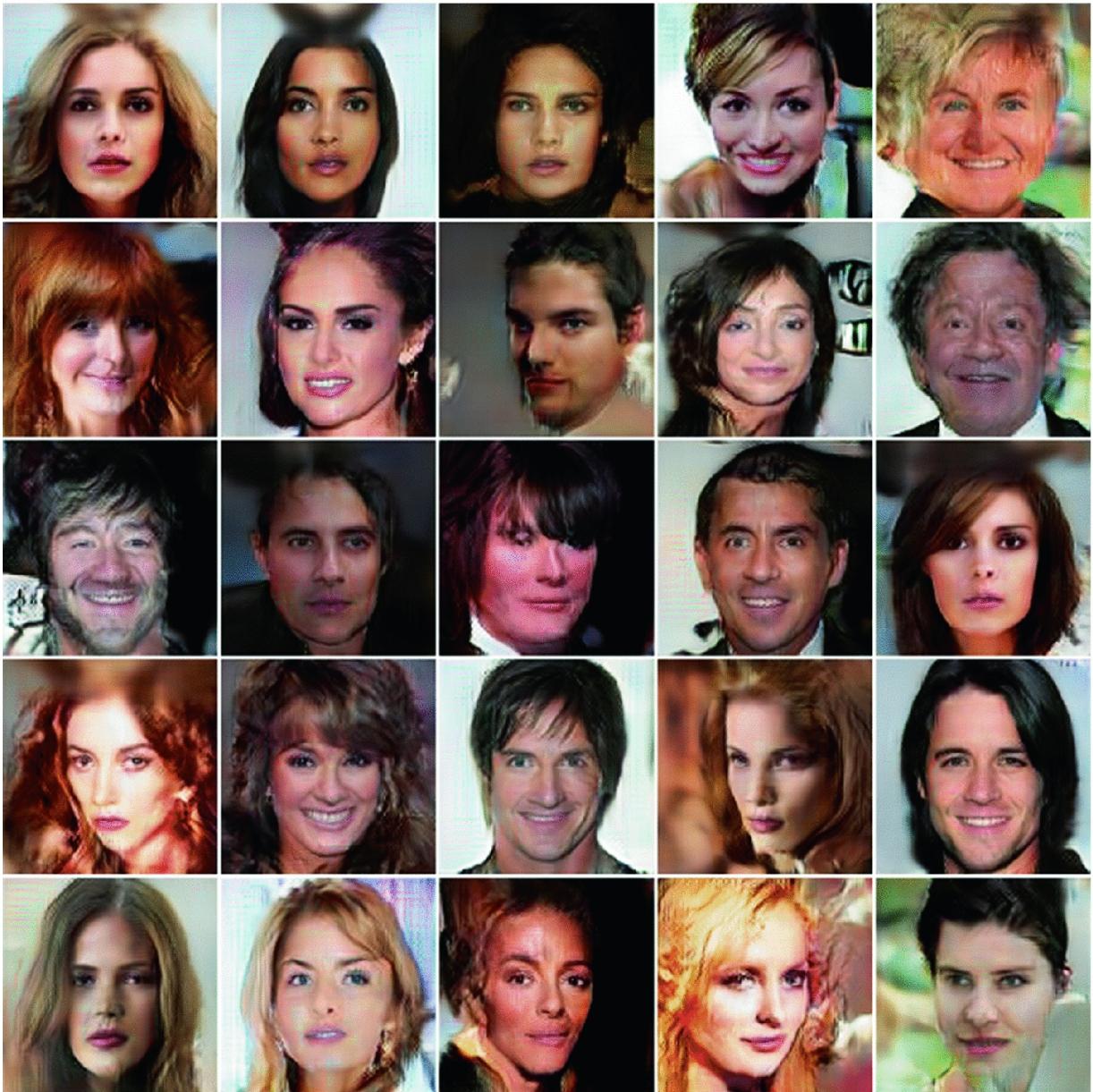


Fig. 4.7 Randomly generated samples from the real-valued SNGAN on the CelebA-HQ dataset after 100 k training iterations. Sometimes this model fails to detect border attributes such as hair and neck which may fade on the background. Indeed, only few samples seem to be visually pleasant while in some other cases the network fails to generate likable images

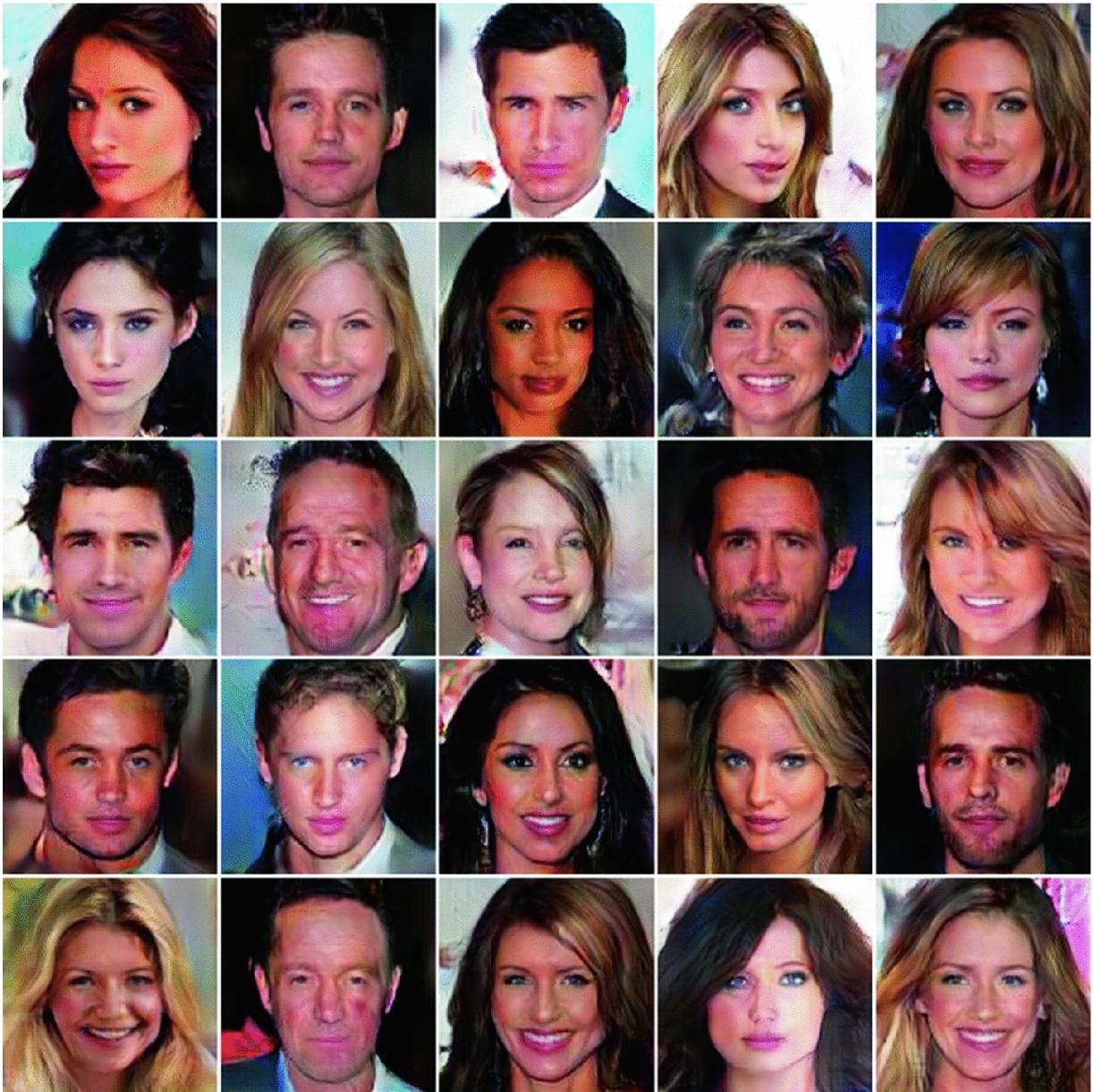


Fig. 4.8 Randomly generated samples from our QSNGAN on the CelebA-HQ dataset after 100 k training iterations. These images are part of the test samples which gained a FID of 29.417 and IS 2.249 ± 0.164 . The proposed method is able to generate visually pleasant images, well distinguishing the background from the face. Moreover, we do not observe mode collapse as samples have different attributes such as genre, hair color, pose and smile, among others



Fig. 4.9 Randomly generated samples from the SNGAN model on the 102 Oxford Flowers dataset after 50 k training iterations. SNGAN misleads some pixels in the images and depicted objects are not always distinguishable



Fig. 4.10 Randomly generated samples from the proposed QSNGAN on the 102 Oxford Flowers dataset after 50 k training iterations. Flowers contain many different colors shades and most of the objects are clearly defined. This set of figures sow the improved generation ability of our proposed method with respect to its real-valued counterpart

The visual inspection of the generated samples underlines the improved ability of our QSNGAN. Figures 4.7 and 4.8 show a randomly selected 128×128 batch of generated images for the real-valued SNGAN and the proposed QSNGAN, respectively. On one hand, SNGAN seems to be quite

unstable and prone to the input noise, thus alternating some good quality images with bad generated ones. Overall, the SNGAN is not always able to distinguish the background from some parts of the character, sometimes confusing attributes such as the neck or the hair as part of the environment, and letting them vanishing. On the other hand, the QSNGAN sample in Fig. 4.8 shows visually pleasant images, with a clear distinction between subject and background. It also shows a higher definition of faces attributes, including the most difficult ones, such as eyebrows, beard or skin shades. In addition, colors seem to be more vivid and samples are diverse in terms of pose, genre, expression, and hair color, among others. Concerning the second dataset, the generated samples for the SNGAN are shown in Fig. 4.9, while the batch from the QSNGAN is reported in Fig. 4.10. As it is clear from Table 4.2, the results for this dataset are preliminary but encouraging. Even in this case the proposed approach gains a lower FID and a higher IS than the real-valued model. Additionally, in SNGAN samples pixels are evident and often misleading, thus confusing the flower object with the colored background. On the other hand, the images generated from our QSNGAN contain more distinct subjects. Furthermore, the proposed method better catches every color shade thanks to the quaternion algebra properties, which allow the network learning internal relations among channels without losing intra-channel information.

In conclusion, the proposed quaternion-valued QSNGAN shows an improved ability in capturing the real data distribution by leveraging the quaternion algebra properties in each experiment we conduct. It can generate better and more vivid samples according to visual inspections and to objective metrics with respect to its real-valued counterpart. Furthermore, the proposed method has less than the 75% of free parameters with respect to the SNGAN which also has worse generation performance.

4.5.1 Evaluation of Spectral Normalization Methods

This section reports the tests we conduct to evaluate the two quaternion spectral normalization methods described in Sect. 4.3.6. In order to investigate the performance of the normalizing approaches, we validate two smaller models with respect to the ones introduced in the previous subsection on the CIFAR10 and STL10 datasets. CIFAR10 contains 50 k

32×32 images for training and 10 k for testing while STL10 has 105 k 32×32 images in the train split and 8 k in the test one.

Table 4.3 Summary results for comparison of the two quaternion spectral normalization methods depicted in Sect. 4.3. We consider the SNGAN proposed in [26] as baseline to define two simple models in the quaternion domain and then test the different QSN approaches. QSN Split refers to the first method that normalizes the submatrices independently while QSN Full stands for the normalization of the whole weight matrix together. No QSN is a model without any spectral normalization method. While the latter fails, the QSN Full generates better images according to the FID in both datasets

	FID ↓		IS ↓	
Config	CIFAR10	STL10	CIFAR10	STL10
No QSN	70.312	91.567	4.031 ± 1.327	<u>4.744 ± 0.643</u>
QSN Split	<u>35.417</u>	<u>75.112</u>	4.7128 ± 1.270	4.455 ± 0.092
QSN Full	31.966	59.611	<u>4.317 ± 0.951</u>	4.987 ± 0.485

We examine three different configurations: the first one does not involve any QSN method, thus the discriminator network is not constrained to be 1-Lipschitz. We run this experiment in order to check the effectiveness of the spectral normalization methods that we propose. The second configuration applies a split computation of the spectral norm for each quaternion component and normalize each weight submatrix $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ independently. The last approach computes the spectral norm of the whole weight matrix and uses it to normalize each component. Respectively, we refer to these methods as No QSN, QSN Split and QSN Full.

To assess the performances, we build the same SNGANs presented in [26] by redefining them in the quaternion domain. We adopt the quaternion core residual blocks we define in the previous section and in Fig. 4.4, while reducing the model dimension. For CIFAR10, we set up a generator with the initial linear layer $4 \times 4 \times 256$ and then pile up three quaternion residual blocks, each one with 256 filters. As before, we end up with a stack of QBN, split ReLU and a quaternion convolution with a final split Tanh to generate the 32×32 images in the range (\mathbf{X}, S) . The discriminator, in which the QSN methods act in each layer, begins with a first residual block

(Fig. 4.5) with 128 filters and then proceeds with three blocks composed of 128 kernels. As in Fig. 4.6, the network ends with a global sum pooling and a fully connected layer with sigmoid to output the decision probability. The so-defined QSNGAN for CIFAR10 is comprised of less than 2 M parameters. It is worth noting that the real-valued counterpart presented in [26] has more than 5 M of free parameters.

The model to generate the 32×32 STL10 images is deeper than the previous one and is composed of 5,545,188 parameters. The structure is the same but it contains an initial layer of $4 \times 4 \times 256$ and then the residual blocks with 256, 128 and 64 filters. The final refiner quaternion convolutional layers has 64 kernels. The discriminator, instead, has one residual blocks more than the model for CIFAR10 and the filters are, respectively from the first to the last block, 64, 128, 256, 512, 1024 with a final 512 fully connected layer with sigmoid.

As we can see in Table 4.3, the unbounded model with no QSN fails in generating images from both CIFAR10 and STL10. Indeed, the FID is much higher than the other approaches. This proves the effectiveness of the proposed QSN full method which computes the spectral norm of each layer taking all the components into account. As a matter of fact, the proposed approach is capable to generate improved quality images in every experiment we conduct.

4.6 Conclusions

In this paper we introduce the family of quaternion-valued GANs (QGANs) that leverages the properties of quaternion algebra. We have rigorously defined each core block employed to build the proposed QGANs, including the quaternion adversarial framework. Moreover, we have provided a meticulous experimental evaluation on different image generation benchmarks to prove the effectiveness of our method. We have shown that the proposed QGAN has an improved generation ability with respect to the real-valued counterpart, according to the FID and IS metrics and to a visual inspection. Moreover, our method saves up to the 75% of free parameters. We believe that these results lay the foundations for novel deep GANs, capturing higher levels of input information and better grasping the real

data distribution, while significantly reducing the overall number of parameters.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN (2017). [arXiv:1701.07875v3](https://arxiv.org/abs/1701.07875v3)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (ICLR) (2019)
3. Chen, T., Zhai, X., Ritter, M., Lucic, M., Houlsby, N.: Self-supervised GANs via auxiliary rotation loss. In: IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12146–12155 (2019)
4. Cheong Took, C., Mandic, D.P.: Augmented second-order statistics of quaternion random signals. *Signal Process.* **91**(2), 214–224 (2011)
[Crossref]
5. Chernov, V.: Discrete orthogonal transforms with data representation in composition algebras. In: Proceedings of the Scandinavian Conference on Image Analysis, pp. 357–364 (1995)
6. Comminiello, D., Lella, M., Scardapane, S., Uncini, A.: Quaternion convolutional neural networks for detection and localization of 3D sound events. In: IEEE International Conference on Acoustics, Speech and Signal Process. (ICASSP), pp. 8533–8537. Brighton, UK (2019)
7. Ell, T.A., Sangwine, S.J.: Quaternion involutions and anti-involutions. *Comput. Math. Appl.* **53**(1), 137–143 (2007)
[MathSciNet][Crossref]
8. Gaudet, C., Maida, A.: Deep quaternion networks. In: IEEE International Joint Conference on Neural Network (IJCNN). Rio de Janeiro, Brazil (2018)
9. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: 27th International

Conference on Neural Information Processing Systems (NIPS), vol. 2, pp. 2672–2680. MIT Press, Cambridge, MA, USA (2014)

11. Gouk, H., Frank, E., Pfahringer, B., Cree, M.J.: Regularisation of neural networks by enforcing Lipschitz continuity. *Mach. Learn.* **110**(2), 393–416 (2021) [[MathSciNet](#)][[Crossref](#)]
12. Grassucci, E., Comminiello, D., Uncini, A.: An information-theoretic perspective on proper quaternion variational autoencoders. *Entropy* **23**(7) (2021)
13. Grassucci, E., Comminiello, D., Uncini, A.: A quaternion-valued variational autoencoder. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Toronto, Canada (2021)
14. Grassucci, E., Scardapane, S., Comminiello, D., Uncini, A.: Flexible generative adversarial networks with non-parametric activation functions. In: Progress in Artificial Intelligence and Neural Systems, vol. 184. Smart Innovation, Systems and Technologies, Springer (2021)
15. Gui, J., Sun, Z., Wen, Y., Tao, D., Ye, J.P.: A review on generative adversarial networks: algorithms, theory, and applications (2020). [arXiv:2001.06937v1](#)
16. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems (NIPS) (2017)
17. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1026–1034 (2015)
18. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Neural Information Processing Systems (NIPS), pp. 6626–6637 (2017)
19. Hoffmann, J., Schmitt, S., Osindero, S., Simonyan, K., Elsen, E.: AlgebraNets (2020). [arXiv:2006.07360v2](#)
20. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML), pp. 448–456. JMLR.org (2015)
21. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for

- improved quality, stability, and variation. In: International Conference on Learning Representations (ICLR) (2018)
22. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 4401–4410. Computer Vision Foundation/IEEE (2019)
 23. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8107–8116. IEEE (2020)
 24. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes, pp. 1–14 (2014). [arXiv:1312.6114v10](https://arxiv.org/abs/1312.6114v10)
 25. Kurach, K., Lucic, M., Zhai, X., Michalski, M., Gelly, S.: A large-scale study on regularization and normalization in GANs. In: International Conference on Machine Learning (ICML) (2019)
 26. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks (2018). [arXiv:1802.05957v1](https://arxiv.org/abs/1802.05957v1)
 27. Parcollet, T., Morchid, M., Linarès, G.: Quaternion convolutional neural networks for heterogeneous image processing. In: IEEE International Conference on Acoustics, Speech and Signal Process. (ICASSP), pp. 8514–8518. Brighton, UK (2019)
 28. Parcollet, T., Morchid, M., Linarès, G.: A survey of quaternion neural networks. Art. Intell. Rev. (2019)
 29. Parcollet, T., Ravanelli, M., Morchid, M., Linarès, G., Trabelsi, C., De Mori, R., Bengio, Y.: Quaternion recurrent neural networks. In: International Conference on Learning Representations (ICLR), pp. 1–19. New Orleans, LA (2019)
 30. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016). [arXiv:1511.06434v2](https://arxiv.org/abs/1511.06434v2)
 31. Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Neural Information Processing Systems (NIPS), pp. 2234–2242 (2016)
 32. Schmidhuber, J.: A possibility for implementing curiosity and boredom in model-building neural controllers. In: Proceedings of the First International Conference

on Simulation of Adaptive Behavior on From Animals to Animats, pp. 222–227. MIT Press, Cambridge, MA, USA (1991)

33. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Netw.* **127**, 58–66 (2020)
[[Crossref](#)]
34. Schönfeld, E., Schiele, B., Khoreva, A.: A U-Net based discriminator for generative adversarial networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8207–8216 (2020)
35. Sfikas, G., Giotis, A.P., Retsinas, G., Nikou, C.: Quaternion generative adversarial networks for inscription detection in byzantine monuments. In: Pattern Recognition. ICPR International Workshops and Challenges, pp. 171–184. Springer International Publishing (2021)
36. Vecchi, R., Scardapane, S., Comminiello, D., Uncini, A.: Compressing deep-quaternion neural networks with targeted regularisation. *CAAI Trans. Intell. Technol.* **5**(3), 172–176 (2020)
[[Crossref](#)]
37. Vía, J., Ramírez, D., Santamaría, I.: Proper and widely linear processing of quaternion random vectors. *IEEE Trans. Inf. Theory* **56**(7), 3502–3515 (2010)
[[Crossref](#)]
38. Ward, J.P.: Quaternions and Cayley Numbers. Algebra and Applications. Mathematics and Its Applications, vol. 403. Kluwer Academic Publishers, Dordrecht (1997)
39. Yin, Q., Wang, J., Luo, X., Zhai, J., Jha, S.K., Shi, Y.: Quaternion convolutional neural network for color image classification and forensics. *IEEE Access* **7**, 20293–20301 (2019)
[[Crossref](#)]
40. Zhang, H., Goodfellow, I.J., Metaxas, D.N., Odena, A.: Self-attention generative adversarial networks. In: International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, vol. 97, pp. 7354–7363. PMLR (2019)
41. Zhang, H., Zhang, Z., Odena, A., Lee, H.: Consistency regularization for generative adversarial networks. In: International Conference on Machine

Learning (ICML) (2020)

42. Zhou, Z., Liang, J., Song, Y., Yu, L., Wang, H., Zhang, W., Yu, Y., Zhang, Z.: Lipschitz generative adversarial nets. In: International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, vol. 97, pp. 7584–7593. PMLR (2019)
-

Footnotes

- 1 The implementation of the QGANs is available online at <https://github.com/eleGAN23/QGAN>.

5. Image Generation Using Continuous Conditional Generative Adversarial Networks

Xin Ding¹✉, Yongwei Wang²✉, Zuheng Xu¹✉, William J. Welch¹✉ and Z. Jane Wang²✉

- (1) Department of Statistics, University of British Columbia, Vancouver, V6T1Z4, Canada
(2) Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, V6T1Z4, Canada

✉ **Xin Ding (Corresponding author)**
Email: xin.ding@stat.ubc.ca

✉ **Yongwei Wang**
Email: yongweiw@ece.ubc.ca

✉ **Zuheng Xu**
Email: zuheng.xu@stat.ubc.ca

✉ **William J. Welch**
Email: will@stat.ubc.ca

✉ **Z. Jane Wang**
Email: zjanew@ece.ubc.ca

Abstract

This chapter is a tutorial for the *continuous conditional generative adversarial network* (CcGAN) [8], the first generative model for image generation with continuous, scalar conditions (termed *regression labels*). Existing *conditional GANs* (cGANs) are mainly designed for categorical

conditions (e.g., class labels); conditioning on regression labels is mathematically distinct and raises two fundamental problems: (P1) Since there may be very few (even zero) real images for some regression labels, minimizing existing empirical versions of cGAN losses (a.k.a. empirical cGAN losses) often fails in practice; (P2) Since regression labels are scalar and infinitely many, conventional label input methods are not applicable. CcGAN solves the above problems, respectively, by (S1) reformulating existing empirical cGAN losses to be appropriate for the continuous scenario; and (S2) proposing a novel method to incorporate regression labels into the generator and the discriminator. The reformulation in (S1) leads to two novel empirical discriminator losses, termed the *hard vicinal discriminator loss* (HVDL) and the *soft vicinal discriminator loss* (SVDL) respectively, and a novel empirical generator loss. The error bounds of a discriminator trained with HVDL and SVDL are derived under mild assumptions. As case studies, experiments show that CcGAN can outperform cGAN in the continuous image generation scenario.

5.1 Introduction and Motivation

Conditional generative adversarial networks (cGANs) were first proposed in [14], aiming to estimate the distribution of images conditioning on some auxiliary information, especially class labels. Subsequent studies [3, 17, 18, 26] confirm the feasibility of generating diverse, high-quality (even photo-realistic), and class-label consistent fake images from class-conditional GANs. Unfortunately, these cGANs do not work well for image generation with continuous, scalar conditions, termed *regression labels*, due to two problems:

Problem 1 (P1): cGANs are often trained to minimize the empirical versions of their losses (a.k.a. the empirical cGAN losses) on some training data, a principle also known as the *empirical risk minimization* (ERM) [25]. The success of ERM relies on a large sample size for each distinct condition. Unfortunately, we usually have only a few real images for some regression labels. Moreover, since regression labels are continuous, some values may not even appear in the training set.

Consequently, a cGAN cannot accurately estimate the image distribution conditional on such missing labels.

Problem 2 (P2): In class-conditional image generation, class labels are often encoded by one-hot vectors or label embedding and then fed into the generator and discriminator by hidden concatenation [14], an auxiliary classifier [18] or label projection [17]. A precondition for such label encoding is that the number of distinct labels (e.g., the number of classes) is finite and known. Unfortunately, in the continuous scenario, we may have infinite distinct regression labels.

A naive approach to solve **(P1)–(P2)** is to “bin” the regression labels into a series of disjoint intervals and still train a cGAN in the class-conditional manner (these interval are treated as independent classes) [19]. However, this approach has four shortcomings: (1) our experiments in Sect. 5.4 show that this approach often makes cGANs collapse; (2) we can only estimate the image distribution conditional on membership in an interval and not on the target label; (3) a large interval width leads to high label inconsistency; (4) inter-class correlation is not considered (images in successive intervals have similar distributions).

In machine learning, *vicinal risk minimization* (VRM) [4, 25] is an alternative rule to ERM. VRM assumes that a sample point shares the same label with other samples in its vicinity. Motivated by VRM, in generative modeling conditional on regression labels where we estimate a conditional distribution $p(x|y)$ (x is an image and y is a regression label), it is natural to assume that a small perturbation to y results in a negligible change to $p(x|y)$. This assumption is consistent with our perception of the world. For example, the image distribution of facial features for a population of 15-year-old teenagers should be close to that of 16 year olds.

This chapter therefore introduces the *continuous conditional GAN* (CcGAN) to tackle **(P1)** and **(P2)** [6–8]. CcGAN is the first generative model for image generation conditional on regression labels. It is noted that [23] and [22] train GANs in an unsupervised manner and synthesize unlabeled fake images for a subsequent image regression task. [20]

proposes a semi-supervised GAN for dense crowd counting. CcGAN is fundamentally different from these works since they do not estimate the conditional image distribution. Recently, [9] integrates cGAN with semi-supervised learning for the power grid diagnosis, which is still within the category of conventional cGAN [14]. Reference [29] formulates the symmetric image registration problem in the form of cGAN; however, the condition is an image instead of a continuous scalar, which is also fundamentally different from this study. In the following sections, we will introduce the derivation of CcGAN, then provide theoretical analysis and empirically evaluate the effectiveness of CcGAN.

5.2 Continuous Conditional Generative Adversarial Networks

This section introduces continuous conditional generative adversarial networks (CcGAN), including the derivations of two discriminator losses (i.e., HVDL, SVDL) and a generator loss function. We also provide a rule-of-thumb formula for hyper-parameter selections and our training algorithms for CcGAN.

5.2.1 Derivation of HVDL and SVDL Losses

In this section, we provide the solutions **(S1)-(S2)** to **(P1)-(P2)** in a one-to-one manner by introducing the *continuous conditional GAN* (CcGAN). Please note that theoretical cGAN losses (e.g., the vanilla cGAN loss [14], the Wasserstein loss [2], and the hinge loss [16]) are suitable for both class labels and regression labels; however, their empirical versions fail in the continuous scenario (i.e., **(P1)**). Our first solution **(S1)** focuses on reformulating these empirical cGAN losses to fit into the continuous scenario. Without loss of generality, we only take the vanilla cGAN loss as an example to show such reformulation (the empirical versions of the Wasserstein loss and the hinge loss can be reformulated similarly).

The vanilla discriminator loss and generator loss [14] are defined as:

(5.1)

$$\begin{aligned}
\mathcal{L}(D) &= -\mathbb{E}_{y \sim p_r(y)} [\mathbb{E}_{x \sim p_r(x|y)} [\log(D(\mathbf{x}, y))] \\
&\quad - \mathbb{E}_{y \sim p_g(y)} [\mathbb{E}_{x \sim p_g(x|y)} [\log(1 - D(\mathbf{x}, y))]]] \\
&= -\int \log(D(\mathbf{x}, y)) p_r(\mathbf{x}, y) d\mathbf{x} dy - \int \log(1 - D(\mathbf{x}, y)) p_g(\mathbf{x}, y) d\mathbf{x} dy, \\
\mathcal{L}(G) &= -\mathbb{E}_{y \sim p_g(y)} [\mathbb{E}_{z \sim q(z)} [\log(D(G(\mathbf{z}, y), y))] \\
&= -\int \log(D(G(\mathbf{z}, y), y)) q(\mathbf{z}) p_g(y) d\mathbf{z} dy,
\end{aligned} \tag{5.2}$$

where $\mathbf{x} \in \mathcal{X}$ is an image, $Y = y$ is a label, $p_z(z)$ and $p_g(y)$ are respectively the true and fake label marginal distributions, $P(\mathbf{x}, y)$ and $p_g(\mathbf{x}|y)$ are respectively the true and fake image distributions conditional on y , $p_r(\mathbf{x}, y)$ and $p_g(\mathbf{x}, y)$ are respectively the true and fake joint distributions of \mathbf{x} and y , and $q(\mathbf{z})$ is the probability density function of $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Since the distributions in the losses of Eqs. (5.1) and (5.2) are unknown, for class-conditional image generation, [14] follows ERM and minimizes the empirical losses:

$$\widehat{\mathcal{L}}^\delta(D) = -\frac{1}{N^r} \sum_{c=1}^C \sum_{j=1}^{N_c^r} \log(D(\mathbf{x}_{c,j}^r, c)) - \frac{1}{N^g} \sum_{c=1}^C \sum_{j=1}^{N_c^g} \log(1 - D(\mathbf{x}_{c,j}^g, c)), \tag{5.3}$$

$$\widehat{\mathcal{L}}^\delta(G) = -\frac{1}{N^g} \sum_{c=1}^C \sum_{j=1}^{N_c^g} \log(D(G(\mathbf{z}_{c,j}), c)), \tag{5.4}$$

where C is the number of classes, N^r and N^g are respectively the number of real and fake images, N_c^r and N_c^g are respectively the number of real and fake images with label c , $\mathbf{x}_{c,j}^r$ and $\mathbf{x}_{c,j}^g$ are respectively the j -th real image and the j -th fake image with label c , and the $\mathbf{z}_{c,j}$ are independently and identically sampled from $q(\mathbf{z})$. Equation (5.3) implies we estimate $p_r(\mathbf{x}, y)$ and $p_g(\mathbf{x}, y)$ by their empirical probability density functions as follows:

(5.5)

$$\hat{p}_r^\delta(\mathbf{x}, y) = \frac{1}{N^r} \sum_{c=1}^C \sum_{j=1}^{N_c^r} \delta(\mathbf{x} - \mathbf{x}_{c,j}^r) \delta(y - c),$$

$$\hat{p}_g^\delta(\mathbf{x}, y) = \frac{1}{N^g} \sum_{c=1}^C \sum_{j=1}^{N_c^g} \delta(\mathbf{x} - \mathbf{x}_{c,j}^g) \delta(y - c),$$

where $\delta(\cdot)$ is a Dirac delta mass centered at 0. However, $\hat{p}_r^\delta(\mathbf{x}, y)$ and $\hat{p}_g^\delta(\mathbf{x}, y)$ in Eq. (5.5) are not good estimates in the continuous scenario because of **(P1)**.

To overcome **(P1)**, we introduce the *hard vicinal estimate* (HVE), a novel estimate for each of $p_r(\mathbf{x}, y)$ and $p_g(\mathbf{x}, y)$. The HVEs of $p_r(\mathbf{x}, y)$ and $p_g(\mathbf{x}, y)$ are:

$$\begin{aligned} \hat{p}_r^{\text{HVE}}(\mathbf{x}, y) &= C_1 \cdot \left[\frac{1}{N^r} \sum_{j=1}^{N^r} \exp\left(-\frac{(y - y_j^r)^2}{2\sigma^2}\right) \right] \cdot \left[\frac{1}{N_{y,\kappa}^r} \sum_{i=1}^{N^r} {}^1\{\lvert y - y_i^r \rvert \leq \kappa\} \delta(\mathbf{x} - \mathbf{x}_i^r) \right], \\ \hat{p}_g^{\text{HVE}}(\mathbf{x}, y) &= C_2 \cdot \left[\frac{1}{N^g} \sum_{j=1}^{N^g} \exp\left(-\frac{(y - y_j^g)^2}{2\sigma^2}\right) \right] \cdot \left[\frac{1}{N_{y,\kappa}^g} \sum_{i=1}^{N^g} {}^1\{\lvert y - y_i^g \rvert \leq \kappa\} \delta(\mathbf{x} - \mathbf{x}_i^g) \right], \end{aligned} \quad (5.6)$$

where \mathbf{x}_i^r and \mathbf{x}_i^g are respectively real image i and fake image i , y_i^r and y_i^g are respectively the labels of \mathbf{x}_i^r and \mathbf{x}_i^g , π and π are two positive hyper-parameters, \mathcal{G}_s and \mathcal{G}_s are two constants making these two estimates valid probability density functions, $N_{y,\kappa}^r$ is the number of the y_i^r satisfying $\log(\mathcal{D}(\mathcal{G}(z)))$, $N_{y,\kappa}^r$ is the number of the y_i^g satisfying $\lvert y - y_i^g \rvert \leq \kappa$, and 1 is an indicator function with support in the subscript. The terms in the first square brackets of \hat{p}_r^{HVE} and \hat{p}_g^{HVE} imply we estimate the marginal label distributions $p_z(z)$ and $p_g(y)$ by *kernel density estimates* (KDEs) [24]. The terms in the second square brackets are designed based on the assumption that a small perturbation to y results in negligible changes to $P(\mathbf{x}, y)$ and $p_g(\mathbf{x}|y)$. If this assumption holds, we can use images with labels in a small vicinity of y to estimate $P(\mathbf{x}, y)$ and $p_g(\mathbf{x}|y)$.

Similar to HVE, we can obtain the *soft vicinal estimate* (SVE), which is an intuitive alternative to HVE. The SVEs of $p_r(\mathbf{x}, y)$ and $p_g(\mathbf{x}, y)$ are:

(5.7)

$$\hat{p}_r^{\text{SVE}}(\mathbf{x}, y) = C_3 \cdot \left[\frac{1}{N^r} \sum_{j=1}^{N^r} \exp \left(-\frac{(y - y_j^r)^2}{2\sigma^2} \right) \right] \cdot \left[\frac{\sum_{i=1}^{N^r} w^r(y_i^r, y) \delta(\mathbf{x} - \mathbf{x}_i^r)}{\sum_{i=1}^{N^r} w^r(y_i^r, y)} \right],$$

$$\hat{p}_g^{\text{SVE}}(\mathbf{x}, y) = C_4 \cdot \left[\frac{1}{N^g} \sum_{j=1}^{N^g} \exp \left(-\frac{(y - y_j^g)^2}{2\sigma^2} \right) \right] \cdot \left[\frac{\sum_{i=1}^{N^g} w^g(y_i^g, y) \delta(\mathbf{x} - \mathbf{x}_i^g)}{\sum_{i=1}^{N^g} w^g(y_i^g, y)} \right],$$

where \mathcal{G}_s and \mathcal{G}_s are two constants making these two estimates valid probability density functions,

$$w^r(y_i^r, y) = e^{-\nu(y_i^r - y)^2} \quad \text{and} \quad w^g(y_i^g, y) = e^{-\nu(y_i^g - y)^2}, \quad (5.8)$$

and the hyper-parameter $\nu > 0$. In Eq. (5.7), similar to the HVEs, we estimate $p_z(z)$ and $p_g(y)$ by KDEs. Instead of using samples in a hard vicinity, the SVEs use all respective samples to estimate $P(\mathbf{x}, y)$ and $p_g(\mathbf{x}|y)$ but each sample is assigned with a weight based on the distance of its label from y .

Two diagrams in Fig. 5.1 visualize the process of using hard/soft vicinal samples to estimate $p(\mathbf{x}|y)$, i.e., a univariate Gaussian distribution conditional on its mean y . More specifically, as shown in Fig. 5.1, let us use the red curves denote the conditional univariate distribution $p(\mathbf{x}|y)$ that we are going to estimate. Two neighboring blue curves denote two conditional distributions $P(\mathbf{x}, y)$ and $P(\mathbf{x}, y)$, where conditions π_i and π_i are within the vicinity of condition y . To achieve a better estimate of $p(\mathbf{x}|y)$ given limited observations (with labels as y), HVE (Eq. (5.6)) estimates $p(\mathbf{x}|y)$ by drawing samples from neighboring distributions within a hard vicinity defined by $y \pm \kappa$ around the groundtruth label y ; while SVE (Eq. (5.7)) tries to estimate $p(\mathbf{x}|y)$ by drawing samples from the soft vicinity around y , and the soft vicinity can be constructed by a weight decay curve (Eq. (5.8)).

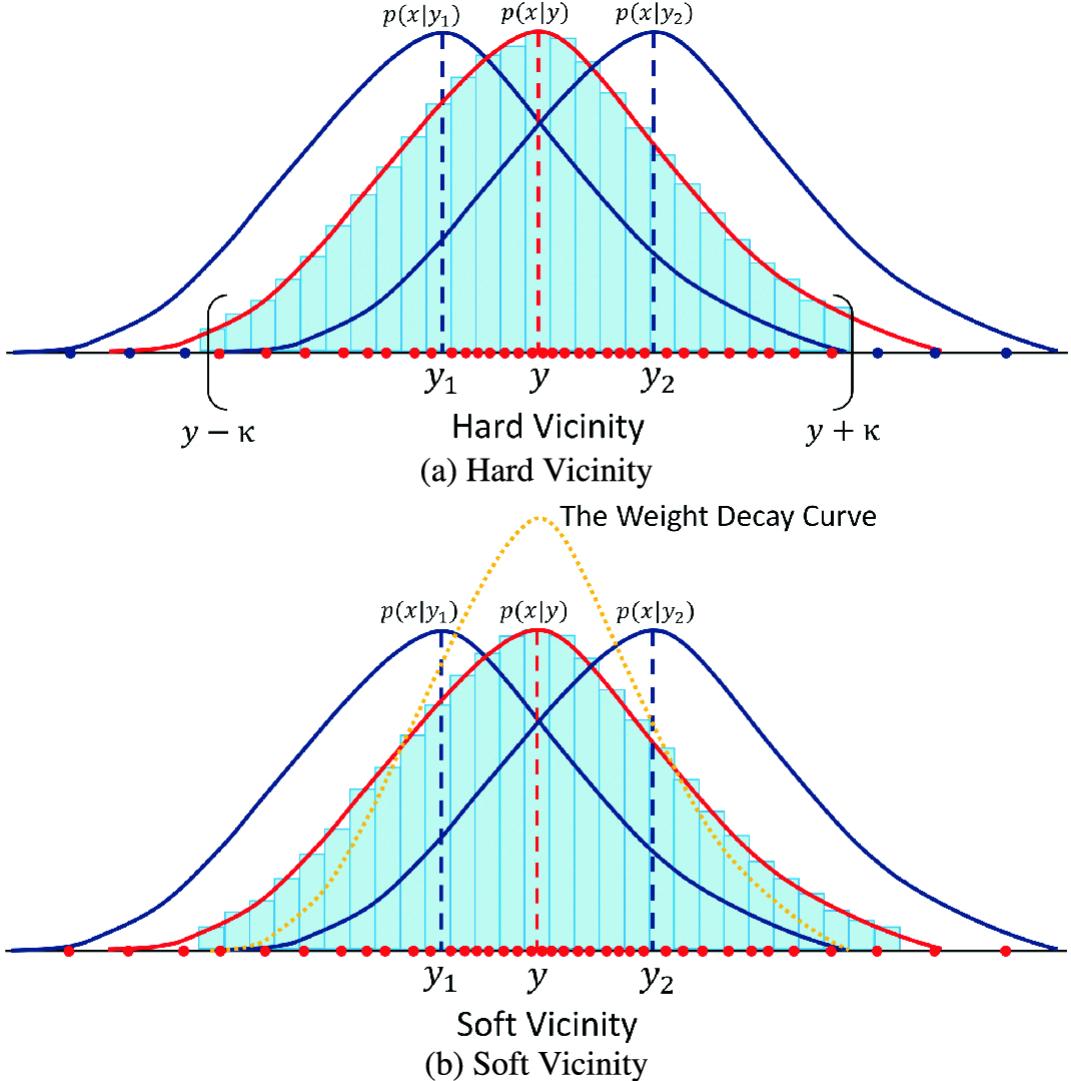


Fig. 5.1 HVE (Eq. (5.6)) and SVE (Eq. (5.7)) estimate $p(x|y)$ (a univariate Gaussian conditional on y) using two samples in hard and soft vicinities, respectively, of y . To estimate $p(x|y)$ (the red Gaussian curve) only from samples drawn from $P(x, y)$ and $P(x, y)$ (the blue Gaussian curves), estimation is based on the samples (red dots) in a hard vicinity (defined by $y \pm \kappa$) or a soft vicinity (defined by the weight decay curve) around y . The histograms in blue are samples in the hard or soft vicinity. The labels π_i , y , and π_i on the x -axis denote the means of x conditional on π_i , y , and π_i , respectively

By plugging Eqs. (5.6) and (5.7) into Eq. (5.1), we derive the *hard vicinal discriminator loss* (HVDL) and the *soft vicinal discriminator loss* (SVDL) as follows:

(5.9)

$$\begin{aligned}
\widehat{\mathcal{L}}^{\text{HVDL}}(D) &= - \frac{C_5}{N^r} \sum_{j=1}^{N^r} \sum_{i=1}^{N^r} \mathbb{E}_{\epsilon^r \sim \mathcal{N}(0, \sigma^2)} \left[\frac{1_{\{|y_j^r + \epsilon^r - y_i^r| \leq \kappa\}}}{N_{y_j^r + \epsilon^r, \kappa}^r} \log(D(\mathbf{x}_i^r, y_j^r + \epsilon^r)) \right] \\
&\quad - \frac{C_6}{N^g} \sum_{j=1}^{N^g} \sum_{i=1}^{N^g} \mathbb{E}_{\epsilon^g \sim \mathcal{N}(0, \sigma^2)} \left[\frac{1_{\{|y_j^g + \epsilon^g - y_i^g| \leq \kappa\}}}{N_{y_j^g + \epsilon^g, \kappa}^g} \log(1 - D(\mathbf{x}_i^g, y_j^g + \epsilon^g)) \right], \\
\widehat{\mathcal{L}}^{\text{SVDL}}(D) &= - \frac{C_7}{N^r} \sum_{j=1}^{N^r} \sum_{i=1}^{N^r} \mathbb{E}_{\epsilon^r \sim \mathcal{N}(0, \sigma^2)} \left[\frac{w^r(y_i^r, y_j^r + \epsilon^r)}{\sum_{i=1}^{N^r} w^r(y_i^r, y_j^r + \epsilon^r)} \log(D(\mathbf{x}_i^r, y_j^r + \epsilon^r)) \right] \\
&\quad - \frac{C_8}{N^g} \sum_{j=1}^{N^g} \sum_{i=1}^{N^g} \mathbb{E}_{\epsilon^g \sim \mathcal{N}(0, \sigma^2)} \left[\frac{w^g(y_i^g, y_j^g + \epsilon^g)}{\sum_{i=1}^{N^g} w^g(y_i^g, y_j^g + \epsilon^g)} \log(1 - D(\mathbf{x}_i^g, y_j^g + \epsilon^g)) \right], \tag{5.10}
\end{aligned}$$

where $\epsilon^r \triangleq y - y_j^r$, $\epsilon^g \triangleq y - y_j^g$, and \mathcal{G}_s , \mathcal{G}_s , \mathcal{G}_s , and \mathcal{G}_s are some constants.

Generator's loss function: In CcGAN, the generator's loss function is given by Eq. (5.11). In Eq. (5.11), we add Gaussian noise to seen labels to let the generator generate images conditional on both seen and unseen labels.

$$\widehat{\mathcal{L}}^\epsilon(G) = - \frac{1}{N^g} \sum_{i=1}^{N^g} \mathbb{E}_{\epsilon^g \sim \mathcal{N}(0, \sigma^2)} \log(D(G(\mathbf{z}_i, y_i^g + \epsilon^g), y_i^g + \epsilon^g)). \tag{5.11}$$

How do HVDL, SVDL, and Eq. (5.11) overcome (P1)?

The solution (**S1**) includes:

- Given a label y as the condition, we use images in a hard/soft vicinity of y to train the discriminator instead of just using images with label y . It enables us to estimate $P(\mathbf{x}, y)$ when there are not enough real images with label y .
- From Eqs. (5.9) and (5.10), we can see that the KDEs in Eqs. (5.6) and (5.7) are adjusted by adding Gaussian noise to the labels. Moreover, in Eq. (5.11), we add Gaussian noise to seen labels (assume y_i^g 's are seen) to train the generator to generate images at unseen labels. This enables estimation of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ when y' is not in the training set.

To solve **(P2)**, CcGAN introduces a novel label input mechanism to the conditional generative adversarial networks. As illustrated in Fig. 5.2, for the generator, the scalar regression label is replicated before being added element-wisely to the projected noise vector in the latent space.

Specifically, the Gaussian noise vector D_1, G, η is projected to a space in \mathbb{R}^{1024} using a fully-connected (FC) layer. The regression label y is repeated multiple times to form a label vector of dimension \mathbb{R}^{1024} . Scalar condition y is embedded to the generator by adding the resulted label vector to the projected noise vector in the \mathbb{R}^{1024} space. An empirical explanation of its effectiveness is that, the label information can be well preserved by replicating the scalar label and adding with the projected noise in the latent space.

For the discriminator D (in Fig. 5.2), we will firstly compute the convolutional output and reshape it as a vector in \mathbb{R}^{16384} . Then, we project the scalar to the space \mathbb{R}^{16384} , and calculate the inner product between two vectors we obtained. We also project the vectorized convolutional output to a scalar and add it with the inner product result. By activating the addition result with a sigmoid function, we obtain the discriminator's prediction score.

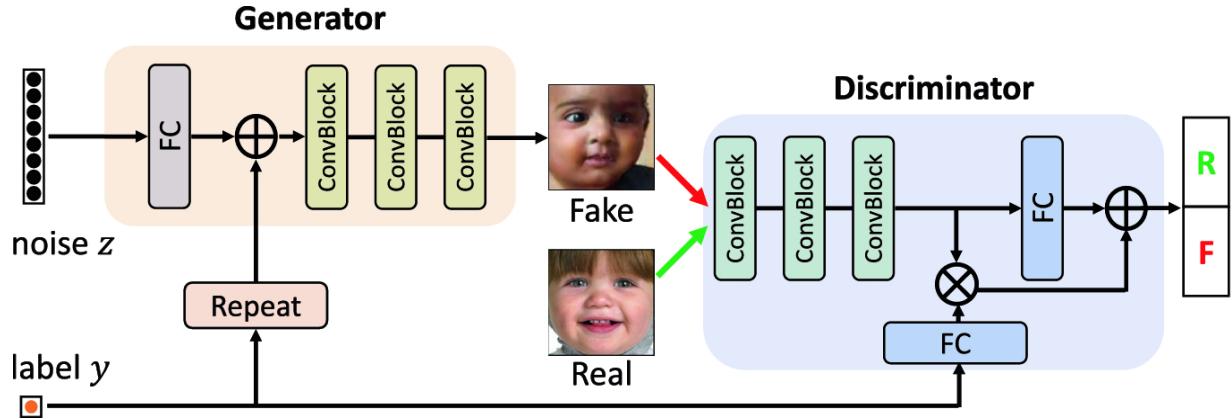


Fig. 5.2 An illustration of the label input mechanism for the generator G and the discriminator D in CcGAN. To input the scalar label y to G , firstly we project the noise vector D_1, G, η to a space \mathbb{R}^{1024} with a fully-connected (FC) layer; we then repeat the scalar regression label y to the same space \mathbb{R}^{1024} , and add the replicated label vector to the projected noise vector in \mathbb{R}^{1024} . In D , firstly we compute the output from three convolutional blocks with dimension \mathbb{R}^{16384} ; we then project y to the space \mathbb{R}^{16384} with a FC layer, and calculate their inner product to obtain a scalar. Next, we

map the convolutional output to another scalar using a FC layer. Finally we add these two scalars (followed by a sigmoid function) to produce real/fake predictions

5.2.2 A Rule of Thumb for Hyper-parameter Selection

The hyper-parameter π is computed based on a rule-of-thumb formula for the bandwidth selection of KDE [24], i.e., $\sigma = (4\hat{\sigma}_{y^r}^5/3N^r)^{1/5}$, where $\hat{\sigma}_{y^r}$ is the sample standard deviation of normalized labels in the training set. Let $\kappa_{\text{base}} = \max \left(y_{[2]}^r - y_{[1]}^r, y_{[3]}^r - y_{[2]}^r, \dots, y_{[N_{\text{uy}}^r]}^r - y_{[N_{\text{uy}}^r-1]}^r \right)$, where $y_{[l]}^r$ is the l -th smallest normalized distinct real label and $x_{c,j}^r$ is the number of normalized distinct labels in the training set. The π is set as a multiple of p_{data} (i.e., $\kappa = m_\kappa \kappa_{\text{base}}$) where the multiplier m_κ stands for 50% of the minimum number of neighboring labels used for estimating $P(\mathbf{x}, y)$ given a label y . For example, $\mathbf{R} \subseteq \mathbf{X}$ implies using 2 neighboring labels (one on the left while the other one on the right). In empirical experiments, m_κ is generally set as 1 or 2. In some extreme case when many distinct labels have too few real samples, we may consider increasing m_κ . It is also observed that $\nu = 1/\kappa^2$ works well in experiments [8].

5.2.3 Algorithms for Training CcGANs

The algorithms to train CcGANs are shown in Algorithm 1 for HVDL and Algorithm 2 for SVDL, respectively. It is worth mentioning that CcGAN does not require any specific network architecture, therefore it can use the state-of-art architectures in practice such as SNGAN [16] and BigGAN [3].

Algorithm 1: An algorithm for CcGAN training with HVDL.

Data: N^r real image-label pairs $\Omega^r = \{(\mathbf{x}_1^r, y_1^r), \dots, (\mathbf{x}_{N^r}^r, y_{N^r}^r)\}$, N_{uy}^r ordered distinct labels $\Upsilon = \{y_{[1]}^r, \dots, y_{[N_{uy}^r]}^r\}$ in the dataset, preset σ and κ , number of iterations K , the discriminator batch size m^d , and the generator batch size m^g .

Result: Trained generator G .

```
1 for  $k = 1$  to  $K$  do
2   Train D;
3   Draw  $m^d$  labels  $Y^d$  with replacement from  $\Upsilon$ ;
4   Create a set of target labels  $Y^{d,\epsilon} = \{y_i + \epsilon | y_i \in Y^d, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \dots, m^d\}$  ( $D$  training is conditional on these labels) ;
5   Initialize  $\Omega_d^r = \emptyset$ ,  $\Omega_d^f = \emptyset$ ;
6   for  $i = 1$  to  $m^d$  do
7     Randomly choose an image-label pair  $(\mathbf{x}, y) \in \Omega^r$  satisfying  $|y - y_i - \epsilon| \leq \kappa$  where  $y_i + \epsilon \in Y^{d,\epsilon}$  and let  $\Omega_d^r = \Omega_d^r \cup (\mathbf{x}, y_i + \epsilon)$ . ;
8     Randomly draw a label  $y'$  from  $U(y_i + \epsilon - \kappa, y_i + \epsilon + \kappa)$  and generate a fake image  $\mathbf{x}'$  by evaluating  $G(z, y')$ , where  $z \sim \mathcal{N}(\mathbf{0}, I)$ . Let  $\Omega_d^f = \Omega_d^f \cup (\mathbf{x}', y_i + \epsilon)$ . ;
9   end
10  Update  $D$  with samples in set  $\Omega_d^r$  and  $\Omega_d^f$  via gradient-based optimizers based on Eq.(5.9);
11  Train G;
12  Draw  $m^g$  labels  $Y^g$  with replacement from  $\Upsilon$ ;
13  Create another set of target labels  $Y^{g,\epsilon} = \{y_i + \epsilon | y_i \in Y^g, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \dots, m^g\}$  ( $G$  training is conditional on these labels) ;
14  Generate  $m^g$  fake images conditional on  $Y^{g,\epsilon}$  and put these image-label pairs in  $\Omega_g^f$  ;
15  Update  $G$  with samples in  $\Omega_g^f$  via gradient-based optimizers based on Eq.(5.11) ;
16 end
```

Algorithm 2: An algorithm for CcGAN training with SVDL.

Data: N^r real image-label pairs $\Omega^r = \{(\mathbf{x}_1^r, y_1^r), \dots, (\mathbf{x}_{N^r}^r, y_{N^r}^r)\}$, N_{uy}^r ordered distinct labels $\Upsilon = \{y_{[1]}^r, \dots, y_{[N_{\text{uy}}^r]}^r\}$ in the dataset, preset σ and ν , number of iterations K , the discriminator batch size m^d , and the generator batch size m^g .

Result: Trained generator G .

```

1 for  $k = 1$  to  $K$  do
2   Train D;
3   Draw  $m^d$  labels  $Y^d$  with replacement from  $\Upsilon$ ;
4   Create a set of target labels  $Y^{d,\epsilon} = \{y_i + \epsilon | y_i \in Y^d, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \dots, m^d\}$  ( $D$  training is conditional on these labels) ;
5   Initialize  $\Omega_d^r = \phi$ ,  $\Omega_d^f = \phi$ ;
6   for  $i = 1$  to  $m^d$  do
7     Randomly choose an image-label pair  $(\mathbf{x}, y) \in \Omega^r$  satisfying  $e^{-\nu(y-y_i-\epsilon)^2} > 10^{-3}$  where  $y_i + \epsilon \in Y^{d,\epsilon}$  and let  $\Omega_d^r = \Omega_d^r \cup (\mathbf{x}, y_i + \epsilon)$ . This step is used to exclude real images with too small weights. ;
8     Compute  $w_i^r(y, y_i + \epsilon) = e^{-\nu(y_i+\epsilon-y)^2}$ ;
9     Randomly draw a label  $y'$  from  $U(y_i + \epsilon - \sqrt{-\frac{\log 10^{-3}}{\nu}}, y_i + \epsilon + \sqrt{-\frac{\log 10^{-3}}{\nu}})$  and generate a fake image  $\mathbf{x}'$  by evaluating  $G(z, y')$ , where  $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Let  $\Omega_d^f = \Omega_d^f \cup (\mathbf{x}', y_i + \epsilon)$ . ;
10    Compute  $w_i^g(y', y_i + \epsilon) = e^{-\nu(y_i+\epsilon-y')^2}$ ;
11  end
12  Update  $D$  with samples in set  $\Omega_d^r$  and  $\Omega_d^f$  via gradient-based optimizers based on Eq.(5.10);
13  Train G;
14  Draw  $m^g$  labels  $Y^g$  with replacement from  $\Upsilon$ ;
15  Create another set of target labels  $Y^{g,\epsilon} = \{y_i + \epsilon | y_i \in Y^g, \epsilon \in \mathcal{N}(0, \sigma^2), i = 1, \dots, m^g\}$  ( $G$  training is conditional on these labels) ;
16  Generate  $m^g$  fake images conditional on  $Y^{g,\epsilon}$  and put these image-label pairs in  $\Omega_g^f$  ;
17  Update  $G$  with samples in  $\Omega_g^f$  via gradient-based optimizers based on Eq.(5.11) ;
18 end

```

5.3 Theoretical Analysis

In this section, we derive the error bounds of a discriminator trained with (\mathbf{X}, S) and $\widehat{\mathcal{L}}^{\text{SVDL}}$ under the theoretical loss \mathcal{X} . First, without loss of generality, we assume $L_I(\mathcal{G}, \mathcal{Q})$. We then introduce some notations as follows.

Let \mathcal{D} stand for the *Hypothesis Space* of D , where the hypothesis space \mathcal{D} consists of a set of functions that can be represented by D (a neural

network with determined architecture). Let $\hat{p}_r^{\text{KDE}}(y)$ and $\{\hat{\mathbf{X}}, \hat{Y}\}$ stand for the KDEs of $p_z(z)$ and $p_g(y)$ respectively. Let $p_w^r(y'|y) \triangleq \frac{w^r(y',y)p^r(y')}{W^r(y)}$, $p_w^g(y'|y) \triangleq \frac{w^g(y',y)p^g(y')}{W^g(y)}$, $W^r(y) \triangleq \int w^r(y',y)p_r(y')dy'$ and $W^g(y) \triangleq \int w^g(y',y)p_g(y')dy'$. Denote by KL the optimal discriminator [10] which minimizes \mathcal{X} but may not be in \mathcal{D} . Let $\tilde{D} \triangleq \arg \min_{D \in \mathcal{D}} \mathcal{L}(D)$. Let $\hat{D}^{\text{HVDL}} \triangleq \arg \min_{D \in \mathcal{D}} \hat{\mathcal{L}}^{\text{HVDL}}(D)$; similarly, we define $p(\mathbf{x}|y)$. Please note that KL may not be covered by the hypothesis space \mathcal{D} . The χ^2 is the minimizer of \mathcal{X} in the hypothesis space \mathcal{D} . Thus, $\mathcal{L}(\tilde{D}) - \mathcal{L}(D^*)$ should be a non-negative constant. In CcGAN, we minimize $\hat{\mathcal{L}}^{\text{HVDL}}(D)$ or $\hat{\mathcal{L}}^{\text{SVDL}}(D)$ with respect to $S \rightarrow Y$, so we are more interested in the distance of $\mathcal{G}(z, c)$ and $p(\mathbf{x}|y)$ from KL , i.e., $\mathcal{L}(\hat{D}^{\text{HVDL}}) - \mathcal{L}(D^*)$ and $\mathcal{L}(\hat{D}^{\text{SVDL}}) - \mathcal{L}(D^*)$.

Definition 1 (*Hölder Class*) Define the Hölder class of functions

$$\Sigma(L) \triangleq \{p : \forall t_1, t_2 \in \mathcal{Y}, \exists L > 0, \text{s.t.} |p'(t_1) - p'(t_2)| \leq L|t_1 - t_2|\}. \quad (5.12)$$

We will work with the following assumptions: **(A1)** All D 's in \mathcal{D} are measurable and uniformly bounded by U . Let

$$U \triangleq \max\{\sup_{D \in \mathcal{D}} [-\log D], \sup_{D \in \mathcal{D}} [-\log(1 - D)]\} \text{ and } U < \infty;$$

(A2) For $\forall \mathbf{x} \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, $\exists g^r(\mathbf{x}) > 0$ and $U < \infty$, s.t.

$\sigma_0(\mathbf{W}_0), \sigma_1(\mathbf{W}_1), \sigma_2(\mathbf{W}_2), \sigma_3(\mathbf{W}_3)$ with $\int g^r(\mathbf{x})d\mathbf{x} = M^r$; **(A3)** For

$\forall \mathbf{x} \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, $\exists g^g(\mathbf{x}) > 0$ and $U < \infty$, s.t.

$|p_g(\mathbf{x}|y') - p_g(\mathbf{x}|y)| \leq g^g(\mathbf{x})|y' - y|$ with $\int g^g(\mathbf{x})d\mathbf{x} = M^g$; **(A4)**

$\mathcal{D} = (\mathbf{X}, Y, S)$ and $p_g(y) \in \Sigma(L^g)$.

Theorem 1 Assume that (A1)-(A4) hold, then $y \in \{0, 1\}$, with probability at least $1 - \delta$,

$$(5.13)$$

$$\begin{aligned} \mathcal{L}(\hat{D}^{\text{HVDL}}) - \mathcal{L}(D^*) &\leq 2U \left(\sqrt{\frac{C_{1,\delta}^{\text{KDE}} \log N^r}{N^r \sigma}} + L^r \sigma^2 \right) + 2U \left(\sqrt{\frac{C_{2,\delta}^{\text{KDE}} \log N^g}{N^g \sigma}} + L^g \sigma^2 \right) \\ &+ 2\kappa U(M^r + M^g) + 2U \sqrt{\frac{1}{2} \log \left(\frac{8}{\delta} \right)} \left(\mathbb{E}_{y \sim \hat{p}_r^{\text{KDE}}(y)} \left[\sqrt{\frac{1}{N_{y,\kappa}^r}} \right] \right. \\ &\left. + \mathbb{E}_{y \sim \hat{p}_g^{\text{KDE}}(y)} \left[\sqrt{\frac{1}{N_{y,\kappa}^g}} \right] \right) + \mathcal{L}(\tilde{D}) - \mathcal{L}(D^*), \end{aligned}$$

for some constants $C_{1,\delta}^{\text{KDE}}, C_{2,\delta}^{\text{KDE}}$ depending on \hat{s} .

Theorem 2 Assume that (A1)-(A4) hold, then $y \in \{0, 1\}$, with probability at least $1 - \delta$,

$$\begin{aligned} \mathcal{L}(\hat{D}^{\text{SVDL}}) - \mathcal{L}(D^*) &\leq 2U \left(\sqrt{\frac{C_{1,\delta}^{\text{KDE}} \log N^r}{N^r \sigma}} + L^r \sigma^2 \right) + 2U \left(\sqrt{\frac{C_{2,\delta}^{\text{KDE}} \log N^g}{N^g \sigma}} + L^g \sigma^2 \right) \\ &+ 4U \sqrt{\frac{1}{2} \log \left(\frac{16}{\delta} \right)} \left(\frac{1}{\sqrt{N^r}} \mathbb{E}_{y \sim \hat{p}_r^{\text{KDE}}(y)} \left[\frac{1}{W^r(y)} \right] + \frac{1}{\sqrt{N^g}} \mathbb{E}_{y \sim \hat{p}_g^{\text{KDE}}(y)} \left[\frac{1}{W^g(y)} \right] \right) \\ &+ 2U \left(M^r \mathbb{E}_{y \sim \hat{p}_r^{\text{KDE}}(y)} \left[\mathbb{E}_{y' \sim \hat{p}_w^r(y'|y)} |y' - y| \right] + M^g \mathbb{E}_{y \sim \hat{p}_g^{\text{KDE}}(y)} \left[\mathbb{E}_{y' \sim \hat{p}_w^g(y'|y)} |y' - y| \right] \right) \\ &+ \mathcal{L}(\tilde{D}) - \mathcal{L}(D^*), \end{aligned} \tag{5.14}$$

for some constant $C_{1,\delta}^{\text{KDE}}, C_{2,\delta}^{\text{KDE}}$ depending on \hat{s} .

Please refer to [7] for proofs of Theorems 1 and 2.

How to interpret Theorems 1 and 2?

The error bounds in both theorems reflect the distance of $\mathcal{G}(z, c)$ and $p(x|y)$ from KL. Enlightened by the two upper bounds, when implementing CcGAN, we should:

- avoid letting D output extreme values (close to 0 or 1) so that U is kept at a moderate level;
- avoid using a too small or a too large π or x to keep the third and fourth terms moderate in Eqs. (5.13) and (5.14).

5.4 Experiments

As case studies, in this section, we evaluate the effectiveness of CcGAN on 2-D Gaussians (i.e., a simulation dataset) and UTKFace (i.e., a real image dataset), respectively. For a fair comparison, cGAN and CcGAN use the same network architecture (a customized architecture for Circular 2-D Gaussians, and the SNGAN [16] architecture for UTKFace) except for the label input modules. For stability, image labels are normalized to $[0, 1]$ in the UTKFace dataset during training.

5.4.1 Case Study 1: Circular 2-D Gaussians

We first test on a synthetic dataset generated from 120 2-D Gaussians with different means (i.e., the underlying groundtruth data distribution is known to us).

5.4.1.1 Experimental Setup

The means of the 120 Gaussians are evenly arranged on a unit circle centered at the origin O of a 2-D space. The Gaussians share a common covariance matrix $\tilde{\sigma}^2 \mathbf{I}_{2 \times 2}$, where $\tilde{\sigma} = 0.02$. We generate 10 samples from each Gaussian for training. Figure 5.3 shows 1,200 training samples (blue dots) from these Gaussians with their means (red dots) on a unit circle. The unit circle can be seen as a clock where we take the mean at 12 o'clock (point A) as the baseline point. Given another mean on the circle (point B), the label y for samples generated from the Gaussian with mean B is defined as the clockwise angle (in radians) between line segments OA and OB . E.g., the label for samples from the Gaussian at A is 0. Both cGAN and CcGAN are trained on this training set. When implementing cGAN, angles are treated as class labels (each Gaussian is treated as a class); while when implementing CcGAN, angles are treated as real numbers.

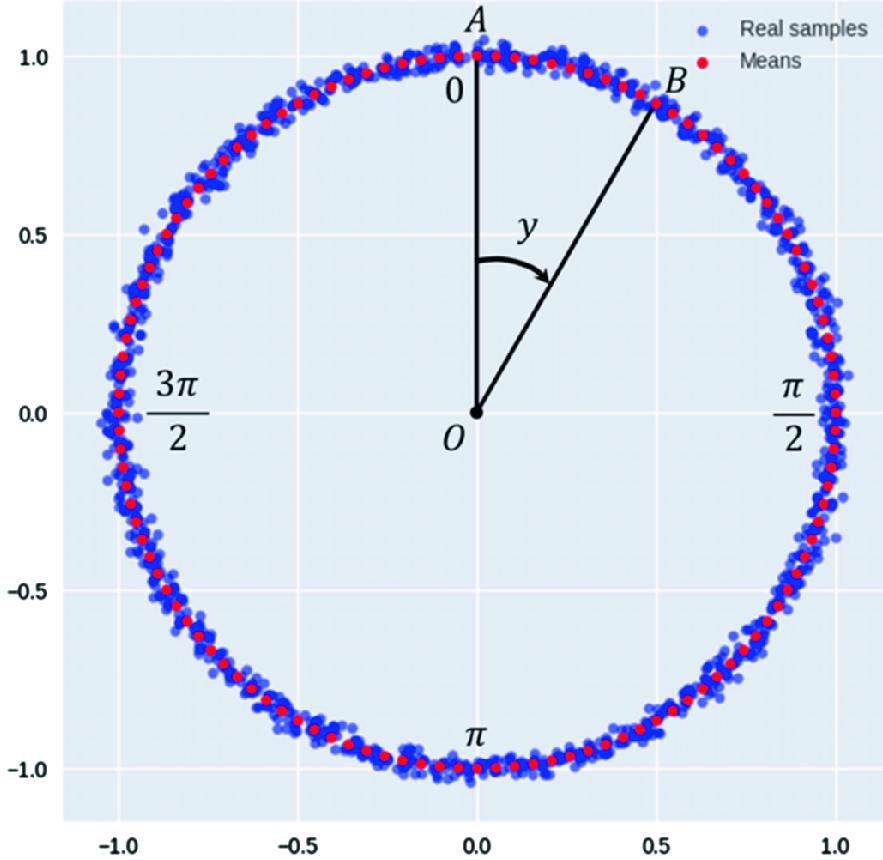


Fig. 5.3 Visual results for 1,200 training samples from 120 Gaussian (10 samples per Gaussian) on the Circular 2-D Gaussians dataset

The network architectures of cGAN and CcGAN are shown in Tables 5.1 and 5.2, respectively. Both cGAN and CcGAN are trained for 6,000 iterations on the training set with the Adam [13] optimizer (with D_1, G, η and $\beta_2 = 0.999$), a constant learning rate 5×10^{-5} and batch size 128. We use the rule of thumb formulae in Sect. 5.2.2 to select the hyper-parameters of HVDL and SVDL, where we let $\mathbf{R} \subseteq \mathbf{X}$. Thus, the three hyper-parameters in this experiments are set as follows: 128×128 , 128×128 , $x \in p_{data}$.

Table 5.1 Network architectures for the generator and discriminator of **cGAN** in the simulation. “fc” denotes a fully-connected layer. “BN” stands for batch normalization. The label y is treated as a class label and encoded by label-embedding [1] so its dimension equals to the number of distinct angles in the training set (i.e., $y \in \mathbb{R}^{120}$)

(a) Generator	(b) Discriminator
$z \in \mathbb{R}^2 \sim N(0, I); y \in \mathbb{R}^{120}$	A sample $x \in \mathbb{R}^2$
concat(z, y) $\in \mathbb{R}^{122}$	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	concat(output of previous layer, y) $\in \mathbb{R}^{220}$, where $y \in \mathbb{R}^{120}$ is the label of x .
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 2	fc \rightarrow 1; Sigmoid

Table 5.2 Network architectures for the generator and discriminator of CcGAN in the simulation. The label y is treated as a real scalar so its dimension is 1. We do not directly input y into the generator and discriminator. We first convert each y into the coordinate of the mean represented by this y , i.e., $P_G(\mathbf{x}, y|s = 1)$. Then we insert this coordinate into the networks

(a) Generator	(b) Discriminator
$z \in \mathbb{R}^2 \sim N(0, I); y \in \mathbb{R}$	A sample $x \in \mathbb{R}^2$ with label $y \in \mathbb{R}$
concat($z, \sin(y), \cos(y)$) $\in \mathbb{R}^4$	concat($x, \sin(y), \cos(y)$) $\in \mathbb{R}^4$
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 100; ReLU
fc \rightarrow 100; BN; ReLU	fc \rightarrow 1; Sigmoid
fc \rightarrow 2	

For testing, we choose 360 points evenly distributed on the unit circle as the means of 360 Gaussians. For each Gaussian, we generate 100 samples, yielding a test set with 36,000 samples. It should be noted that, among these 360 Gaussians, at least 240 are not used at the training. In other words, there are at least 240 labels in the testing set which do not appear in the training set. For each test angle, we generate 100 fake samples from each trained GAN, yielding 36,000 fake samples from each GAN in total. The

quality of these fake samples is evaluated. We repeat the whole experiment three times and report in Table 5.3 the average quality over three repetitions.

5.4.1.2 Evaluation Metrics

In the label-conditional scenario, each fake sample \mathbf{x} with label y is compared with the mean $P_G(\mathbf{x}, y|s = 1)$ of a Gaussian on the unit circle with label y . A fake sample is defined as “high-quality” if its Euclidean distance from \mathbf{x} to $P_G(\mathbf{x}, y|s = 1)$ is smaller than 128×128 . A mode (i.e., a Gaussian) is said to be recovered if at least one high-quality sample is assigned to it. We also measure the quality of fake samples with label y by computing the 2-Wasserstein Distance (\mathcal{W}_2) [21] between $p_r(\mathbf{x}|y) = \mathcal{N}([\sin(y), \cos(y)]^\top, \tilde{\sigma} \mathbf{I})$ and $p_g(\mathbf{x}|y) = \mathcal{N}(\mu_y^g, \Sigma_y^g)$, where we assume $p_g(\mathbf{x}|y)$ is Gaussian and its mean and covariance are estimated by the sample mean and sample covariance of 100 fake samples with label y .

5.4.1.3 Experimental Results

Quantitative results: In Table 5.3, we report the average percentage of high-quality fake samples and the average percentage of recovered modes over 3 repetitions. We also report the average \mathcal{W}_2 over 360 testing angles. We can see CcGAN substantially outperforms cGAN.

Table 5.3 Average quality of 36,000 fake samples from cGAN and CcGAN over three repetitions with standard deviations after the “ \pm ” symbol. “ \downarrow ” (“ \uparrow ”) indicates lower (higher) values are preferred

Method	% High quality \downarrow	% Recovered modes \downarrow	2-Wasserstein dist. \downarrow
cGAN (120 classes)	68.8 ± 4.8	68.8 ± 4.8	$\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1\hat{i} + \mathbf{x}_2\hat{j} + \mathbf{x}_3\hat{k}$
CcGAN (HVDL)	99.3 ± 0.4	100.0 ± 0.0	$3.03 \times 10^{-4} \pm 5.05 \times 10^{-5}$
CcGAN (SVDL)	99.3 ± 0.4	100.0 ± 0.0	$3.03 \times 10^{-4} \pm 5.05 \times 10^{-5}$

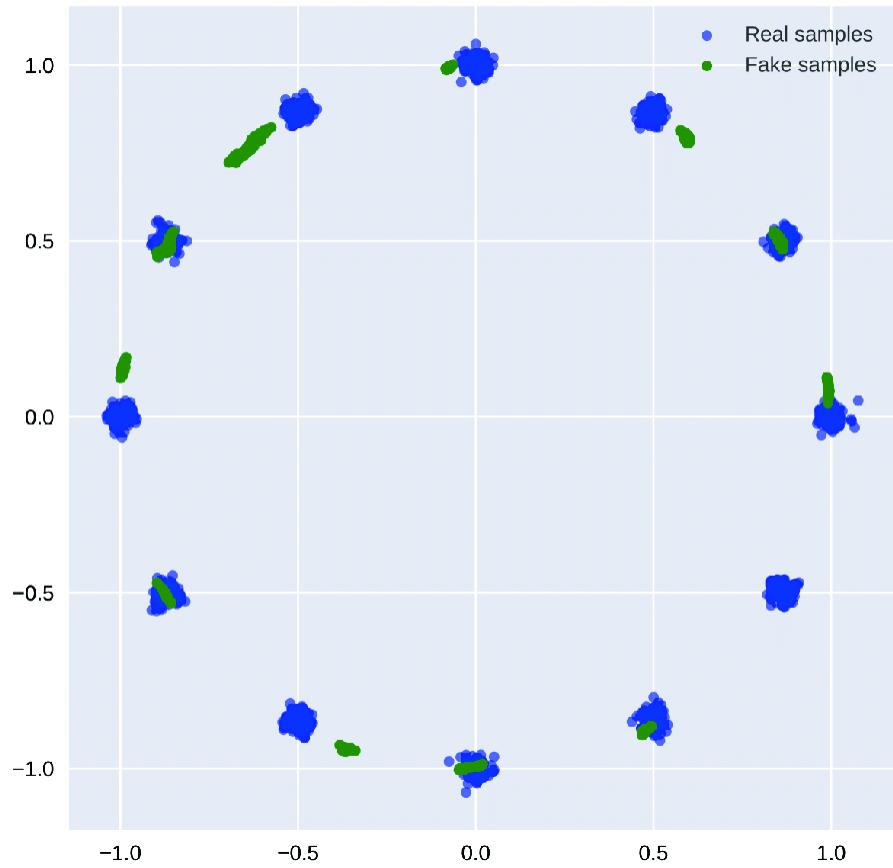


Fig. 5.4 Visual results for 100 fake samples generated by cGAN at each of 12 means not appearing on the Circular 2-D Gaussians training set. Green and blue dots stand for fake and real samples, respectively

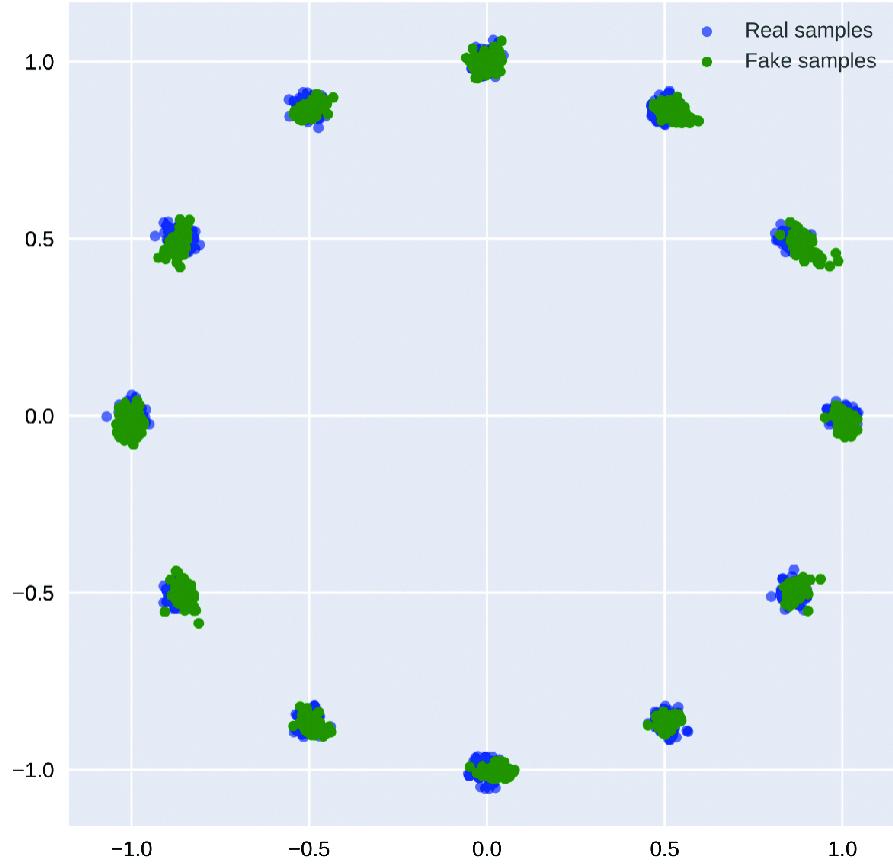


Fig. 5.5 Visual results for 100 fake samples generated by CcGAN (HVDL) at each of 12 means not appearing on the Circular 2-D Gaussians training set. Green and blue dots stand for fake and real samples, respectively

Visual results: Besides quantitative comparisons, we make visual comparisons between cGAN and CcGAN on the circular 2-D Gaussians dataset. We select 12 angles which do not appear in the training set. We then use cGAN and CcGAN to generate 100 samples for each unobserved angle. In Figs. 5.4, 5.5 and 5.6 we visualize the 100 fake samples generated by cGAN, CcGAN (HVDL) and CcGAN (SVDL), respectively. We observe that the fake samples from the two CcGAN methods are more realistic, which visually confirms the observation from the numerical metrics (Table 5.3).

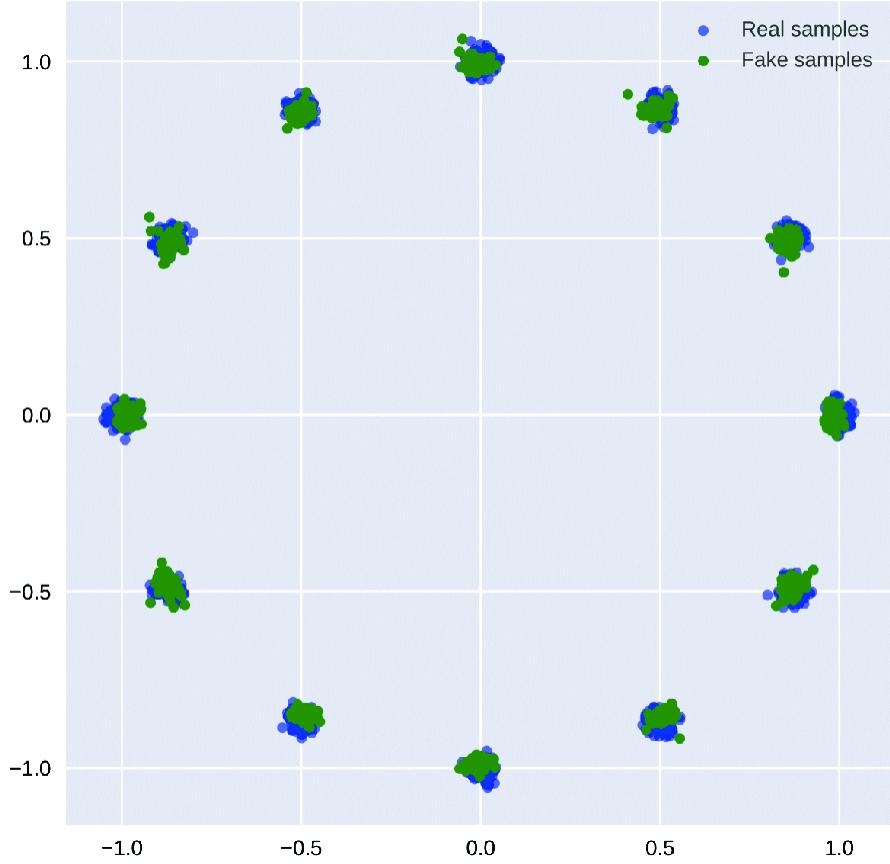


Fig. 5.6 Visual results for 100 fake samples generated by CcGAN (SVNL) at each of 12 means not appearing on the Circular 2-D Gaussians training set. Green and blue dots stand for fake and real samples, respectively

Varying number of Gaussians for training data generation: Finally, we study the influence of the number of Gaussians used for training data generation on the performance of cGAN and CcGAN. We vary the number of Gaussians from 120 to 10 with step size 10 but keep other settings in Sect. 5.4.1.1 unchanged and plot the line graphs of 2-Wasserstein Distance (log scale) versus the number of Gaussians in Fig. 5.7. Reducing the number of Gaussians for training implies a larger gap between any two consecutive distinct angles in the training set. As the number of Gaussians decreases, the continuous scenario gradually degenerates to the categorical scenario, therefore the assumption that a small perturbation to y results in a negligible change to $p(x|y)$ is no longer satisfied. Consequently, the 2-Wasserstein distances of the two CcGAN methods gradually increase and eventually surpass the 2-Wasserstein distance of cGAN when the number of

Gaussians is small (e.g., less than 40). Note that reducing the number of Gaussians in the training data generation will not improve the performance of cGAN in the testing because many angles seen in the testing stage (we evaluate each method on 360 angles) do not appear in the training set.

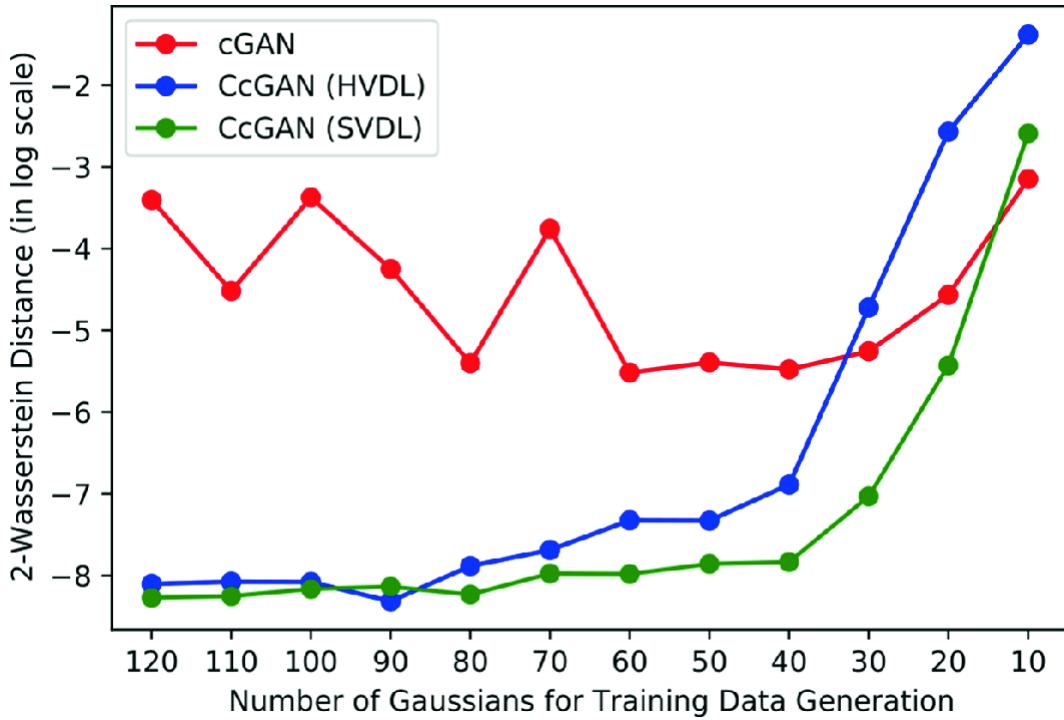


Fig. 5.7 Line graphs of 2-Wasserstein Distance (log scale) versus the number of Gaussians for training data generation. As the number of Gaussians decreases, the continuous scenario gradually degenerates to the categorical scenario, therefore the assumption that a small perturbation to y results in a negligible change to $p(x|y)$ is no longer satisfied. Consequently, the 2-Wasserstein distances of two CcGAN methods gradually increase and eventually surpass the 2-Wasserstein distance of cGAN when the number of Gaussians is small (e.g., less than 40)

5.4.2 Case Study 2: UTKFace

In the second case study, we evaluate CcGAN and cGAN on UTKFace [27], a dataset consisting of RGB images of human faces which are labeled by age.

5.4.2.1 Experimental Setup

The UTKFace dataset is an age regression dataset [27], with human face images collected in the wild. We use the preprocessed version (cropped and

aligned), with ages spanning from 1 to 60. After data cleaning (i.e., removing images of very low quality or with clearly wrong labels), the overall number of images is 14760. Images are resized to 32×32 . The histogram of UTKFace dataset w.r.t. ages 1–60 is shown in 5.8.

From Fig. 5.8, we can see UTKFace dataset is very imbalanced so the samples from the minority age groups are unlikely to be chosen at each iteration during the GAN training. Consequently, cGAN and CcGAN may not be well-trained at these minority age groups. To increase the chance of drawing these minority samples during training, we randomly replicate samples in the minority age groups to ensure that the sample size of each age is more than 200.

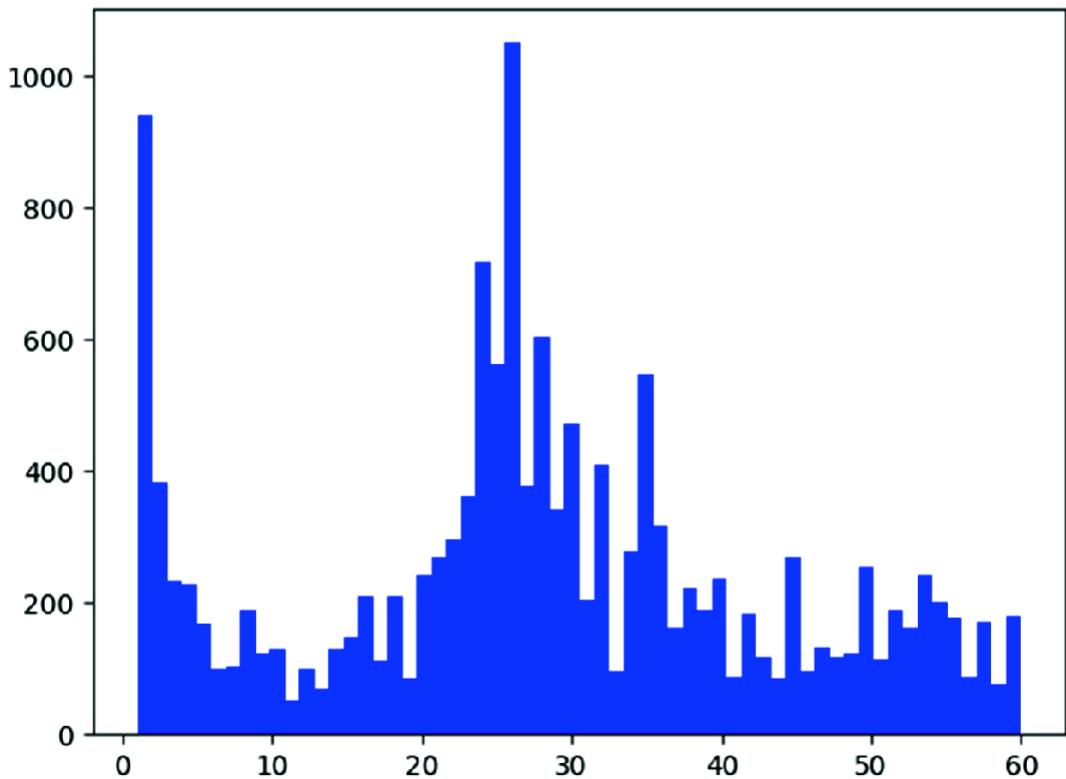


Fig. 5.8 The histogram of UTKFace dataset with ages ranging from 1 to 60

The UTKFace dataset is a more sophisticated dataset compared with the circular 2-D Gaussians, thus it requires networks with deeper layers. We employ the SNGAN architecture [16] in both cGAN and CcGAN consisting of residual blocks for the generator and the discriminator. Moreover, for the generator in cGAN, the regression labels are input into the network by the

label embedding [1] and the conditional batch normalization [5]. For the discriminator in cGAN, the regression labels are fed into the network by the label embedding and the label projection [17]. For CcGAN, the regression labels are fed into networks following the label input method in Sect. 5.2.

When implementing cGAN, each age is treated as a class. For CcGAN we use the rule of thumb formulae in Sect. 5.2.2 to select the three hyper-parameters of HVDL and SVDL, i.e., $\sigma \approx 0.041$, $\sigma \approx 0.041$ and $x \in p_{data}$. The cGAN and CcGAN are trained for 40,000 iterations on the training set with the Adam [13] optimizer (with D_1, G, η and $\beta_2 = 0.999$), a constant learning rate 10^{-4} and batch size 512. In testing, we generate 1,000 fake images from each trained GAN for each age.

5.4.2.2 Evaluation Metrics

On UTKFace, to comprehensively evaluate (1) the visual quality, (2) the intra-label diversity, and (3) the label consistency (whether assigned labels of fake images are consistent with their true labels) of fake images, we study an overall metric and three separate metrics here. (i) **Intra-FID** [17] is utilized as the overall metric. It computes the *Fréchet inception distance* (FID) [11] separately at each of the 60 evaluation ages and reports the average FID score. (ii) **Naturalness Image Quality Evaluator (NIQE)** [15] measures the visual quality only. (iii) **Diversity** is the average entropy of predicted race of fake images over evaluation ages. (iv) **Label Score** is the average absolute error between assigned labels and predicted labels. We describe more details of four metrics as follows.

- **Intra-FID** [17]: *We take Intra-FID as the overall score to evaluate the quality of fake images and we prefer the small Intra-FID score.* At each evaluation age, we compute the FID [11] between real images and fake images in terms of the bottleneck feature of the pre-trained AE. The Intra-FID score is the average FID over all 60 evaluation ages.
- **NIQE** [15]: *NIQE is used to evaluate the visual quality of fake images with the real images as the reference and we prefer the small NIQE score.* We train one NIQE model with real images at each of the 60 ages, so we have 60 NIQE models. During evaluation, a NIQE score is computed for each evaluation age based on the NIQE model at that age. Finally, we report the average and standard deviations of the 60 NIQE scores over

the 60 ages. Note that the NIQE is implemented by the NIQE module in MATLAB.

- **Diversity:** *Diversity is used to evaluate the intra-label diversity and the larger the better.* The UTKFace dataset consists of face images from 5 races based on which we train the classification-oriented ResNet-34. At each evaluation age, we ask the pre-trained ResNet-34 to predict the age of fake images and an entropy is computed based on these predicted ages. The diversity will be computed as the average of the 60 entropies over all evaluation ages.
- **Label Score:** *Label Score is used to evaluate the label consistency and the smaller the better.* We ask the pre-trained regression-oriented ResNet-34 to predict the ages of all fake images and the predicted ages are then compared with the assigned ages. The Label Score is defined as the average absolute distance between the predicted ages and assigned ages over all fake images, which is equivalent to the Mean Absolute Error (MAE).

Before we conduct the evaluation in terms of the four metrics, we first train an autoencoder (AE), a regression CNN and a classification CNN on all real images in UTKFace. The bottleneck dimension of the AE is 512 and the AE is trained to reconstruct the real images in UTKFace with MSE as the loss function. The regression CNN is trained to predict the age of a given image. The classification CNN is trained to predict the race of a given face image. The autoencoder and both two CNNs are trained for 200 epochs with a batch size 256.

5.4.2.3 Experimental Results

Quantitative results: We evaluate the quality of fake images by Intra-FID, NIQE, Diversity (entropy of predicted races), and Label Score. We report in Table 5.4 the average quality of 60,000 fake images.

Table 5.4 Average quality of 60,000 fake UTKFace images from cGAN and CcGAN with standard deviations after the “ \pm ” symbol. “ \downarrow ” (“ \downarrow ”) indicates lower (higher) values are preferred

Method	Intra-FID \downarrow	NIQE \downarrow	Diversity \downarrow	Label score \downarrow
cGAN (60 classes)	4.516 ± 0.965	4.516 ± 0.965	4.516 ± 0.965	11.087 ± 8.119

Method	Intra-FID ↓	NIQE ↓	Diversity ↓	Label score ↓
CcGAN (HVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167
CcGAN (SVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	10.739 ± 8.340

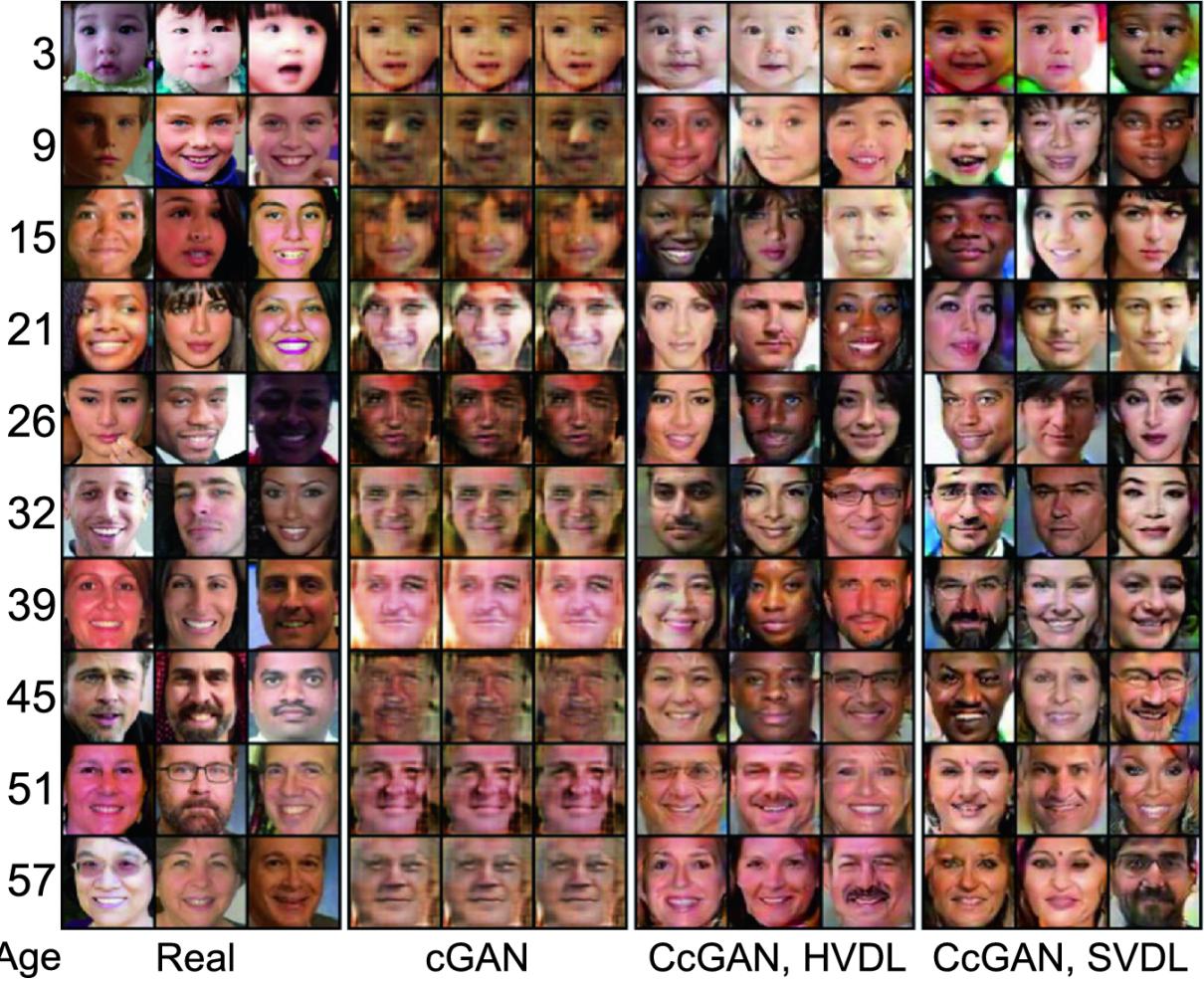


Fig. 5.9 Three UTKFace example images for each of 10 ages: real images and example fake images from cGAN and two CcGANs, respectively. CcGANs produce face images with **higher visual quality and more diversity**

Visual results: We also show in Fig. 5.9 some example fake images from cGAN and CcGAN. We observe that CcGANs generate fake face images with clearly higher visual quality and more diversity. Moreover, we evaluate and compare the FID and NIQE metrics at each age between cGAN and CcGANs. Figures 5.10 and 5.11 show the line graphs of FID/NIQE versus age label on UTKFace. We observe that, two CcGANs consistently outperform cGAN across all regression labels. The graphs of

CcGANs also appear smoother than those of cGAN because of HVDL and SVDL.

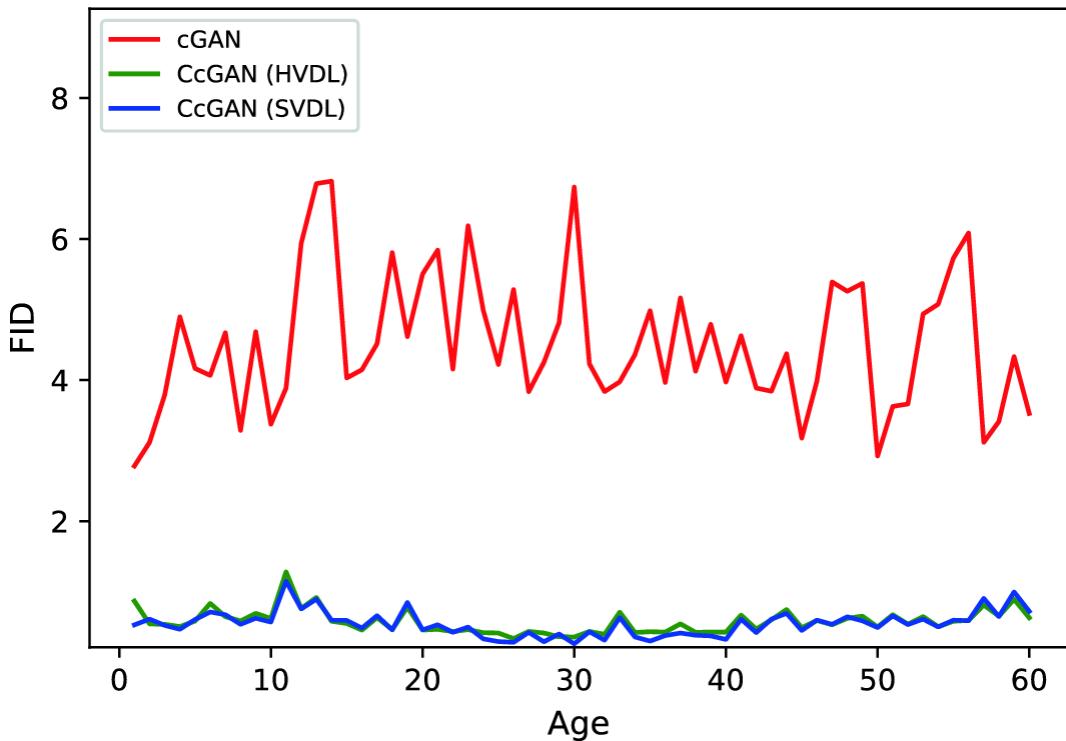


Fig. 5.10 Line graphs of FID versus regression labels on UTKFace

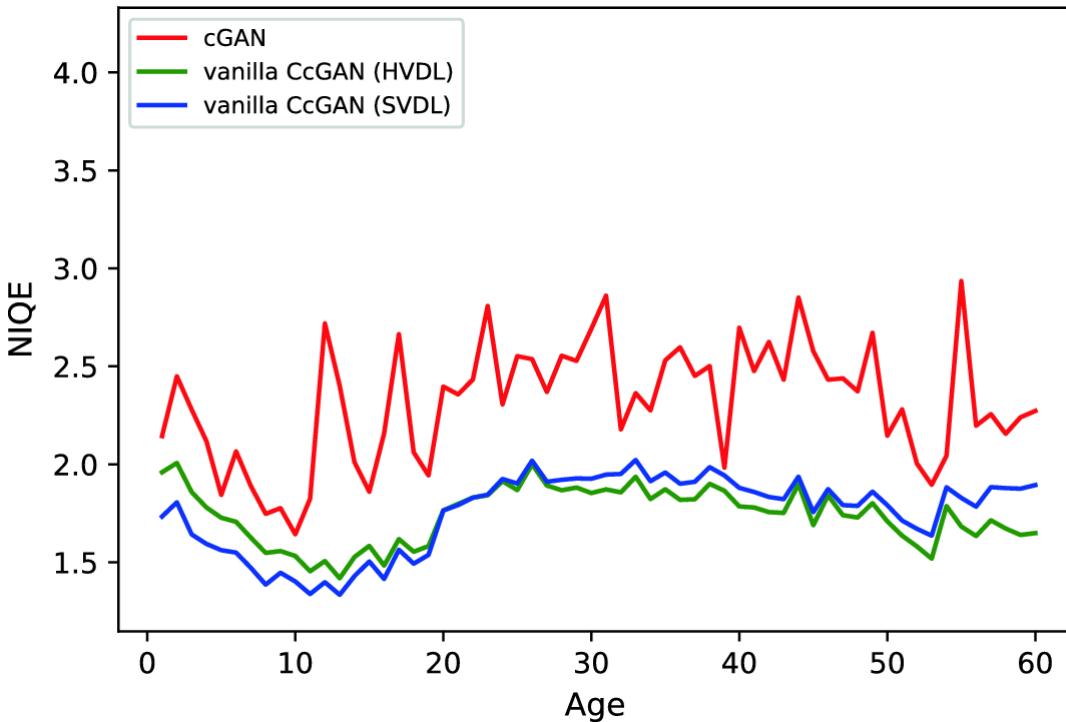


Fig. 5.11 Line graphs of NIQE versus regression labels on UTKFace

Interpolation: To perform label interpolation experiments, we keep the noise vector z fixed and vary label from age 3 to age 57 for CcGANs with HVDL and SVDL losses. The interpolation results are illustrated in 5.12. As age y increases, we observe the synthetic face gradually becomes older in appearance. This observation convincingly shows that both HVDL and SVDL based CcGANs do not simply memorize or overfit to the training set. Indeed, CcGANs demonstrate continuous control over synthetic images with respect to ages.

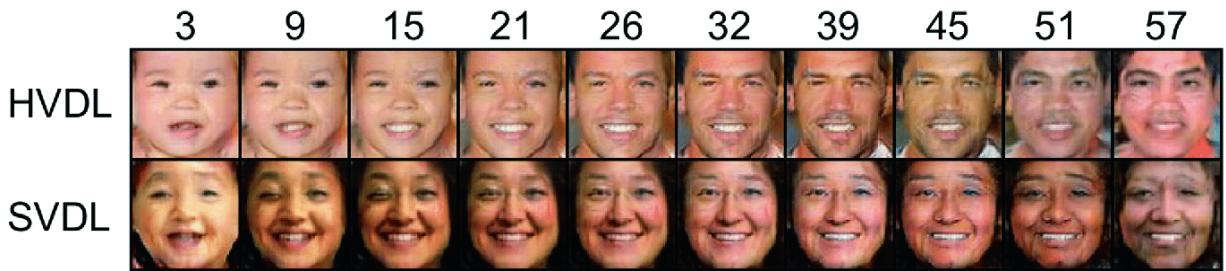


Fig. 5.12 Some examples of generated UTKFace images from CcGAN when the discriminator is trained with HVDL and SVDL. We fix the noise π but vary the label y from 3 to 57

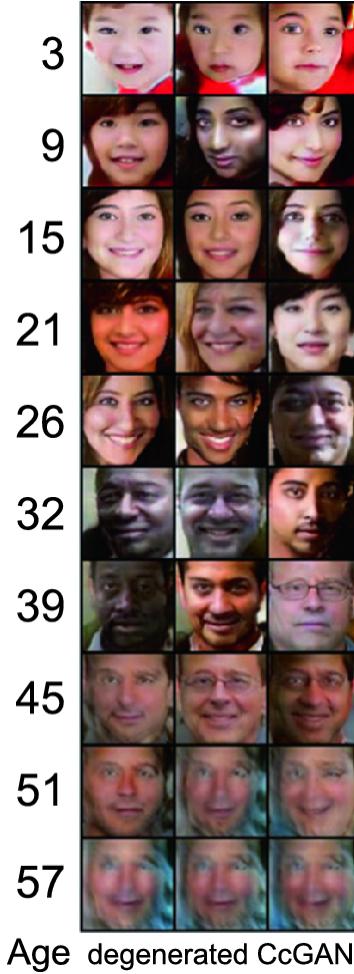


Fig. 5.13 Some example UTKFace fake images from a degenerated CcGAN

Degenerated CcGAN: We consider the extreme cases of the CcGANs on the UTKFace dataset. As shown in Fig. 5.13, the degenerated CcGAN fails to generate facial images at some ages (e.g., 51 and 57) because of too small sample sizes.

cGAN: different number of classes: In the last experiment, we bin samples into different number of classes based on ground-truth labels, in order to increase the number of training samples at each class. Then we train cGAN using samples from the binned classes. We experimented with two different bin setting, i.e., binning image samples into 60 classes and 40 classes, respectively. The results are reported in Fig. 5.14. The results demonstrate cGANs consistently fail to generate diverse synthetic images with labels aligned with their conditional information. Moreover, the image quality is much worse than those from the CcGANs. In conclusion,

compared with existing cGANs, CcGANs have substantially better performance in terms of the image quality and diversity.

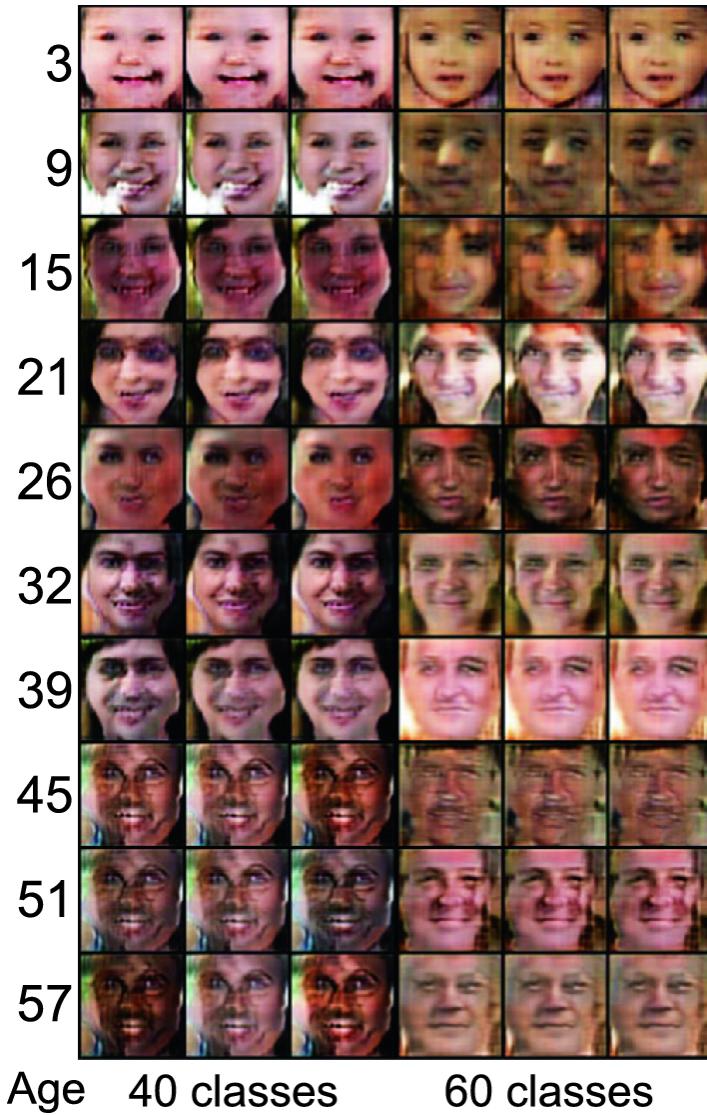


Fig. 5.14 Example UTKFace fake images from cGAN when we bin the age range into different number of classes

Training on images for odd ages only and testing on even-numbered ages: In this section, we train cGAN and CcGAN on images for odd ages only and test them on even-numbered ages. Since the training set in this experiment is half of the one in the main study in Sect. 5.4.2, we reduce the number of iterations for the GAN training from 40,000 to 20,000 while other settings are unchanged. The quantitative results for cGAN and two

CcGAN methods are summarized in Table 5.5. From Table 5.5, we can see two CcGAN methods are still much better than cGAN in terms of all metrics except Label Score, since CcGAN is designed to sacrifice some (not too much) label consistency for much better visual quality and diversity.

Training with smaller sample sizes: The histogram in Fig. 5.8 shows that the UTKFace dataset is highly imbalanced. To balance the training data and also test the performance of cGAN and CcGAN under smaller sample sizes, we vary the maximum sample size for each distinct age in the training from 200 to 50. Note that, in the main study in Sect. 5.4.2, we do not restrict the maximum sample size. Since we have a much smaller sample size, we reduce the number of iterations for the GAN training from 40,000 to 20,000 and slightly increase m_κ in Supp. 5.2.2 from 1 to 2 (we therefore use a wider hard/soft vicinity). We visualize the line graphs of Intra-FID versus the maximum sample size for each age of cGAN and CcGAN in Fig. 5.15. From the figure, we can clearly see that a smaller sample size worsens the performance of both cGAN and CcGAN. Moreover, the Intra-FID scores of cGAN always stay at a high level and are much larger than those of two CcGAN methods.

Table 5.5 Training cGAN and CcGAN on images for odd ages only and testing them on even-numbered ages

Method	Intra-FID ↓	NIQE ↓	Diversity ↓	Label score ↓
cGAN (30 classes)	4.516 ± 0.965	4.516 ± 0.965	4.516 ± 0.965	0.572 ± 0.167
CcGAN (HVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	11.087 ± 8.119
CcGAN (SVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	11.087 ± 8.119

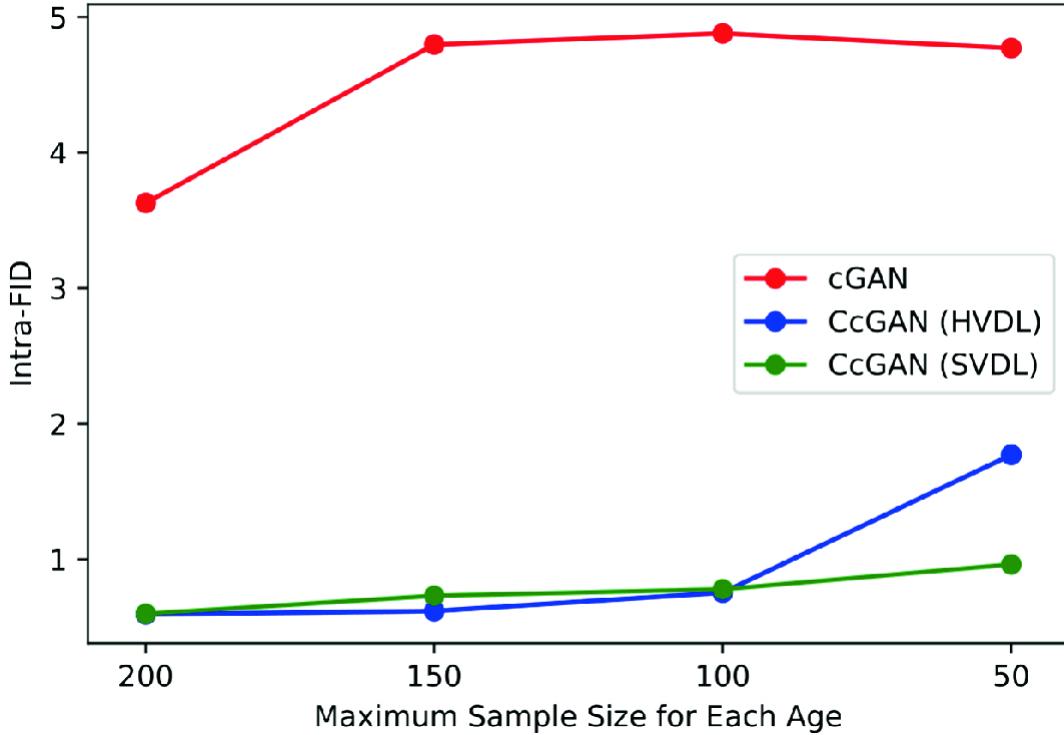


Fig. 5.15 Line graphs of Intra-FID versus the maximum sample size for each age in the training set

DiffAugment cannot save cGAN in the continuous scenario: DiffAugment [28] is a concurrent work which proposes differentiable transformations (i.e., translation, random brightness, contrast, saturation, and cutout) on both real and fake images during the GAN training. This method shows promising results in improving the performance of unconditional GANs (e.g., StyleGAN2 [12]) and class-conditional GANs (e.g., BigGAN [3]) when training samples are limited. However, DiffAugment is fundamentally different from CcGAN since it is designed for the unconditional and class-conditional scenarios rather than the continuous scenario. Even though incorporating DiffAugment into the cGAN training in the continuous scenario, the two problems **(P1)** and **(P2)** discussed in Sect. 5.1 are still unsolved. First, **(P1)** is still unsolved because DiffAugment does not provide a solution better than binning the regression labels into a series of disjoint intervals to tackle the problem that some regression labels do not exist in the training set. Second, since DiffAugment is designed for the unconditional and class-conditional scenarios where the number of distinct conditions is always finite and known, DiffAugment

doesn't provide a solution to **(P2)**. Besides these two unsolved problems, another concern of DiffAugment in the continuous scenario is that the ordinal information in the regression labels is not utilized while CcGAN implicitly uses this ordinal information to construct the soft/hard vicinity.

To support our arguments, we incorporate DiffAugment into the cGAN training in the UTKFace experiment while other settings are kept constant. When implementing DiffAugment, we use the official codes from the GitHub repository of DiffAugment.¹ The strongest transformation combination (Color + Translation + Cutout) is used in the cGAN training. Quantitative results from cGAN+DiffAugment are summarized in Table 5.6 and some example images from cGAN+DiffAugment are shown in Fig. 5.16. The quantitative results show that DiffAugment substantially improves the visual quality and diversity of the baseline cGAN; however, the performance of cGAN+DiffAugment is still much worse than that of the two CcGAN methods. The visual results also support the quantitative evaluations. Therefore, cGAN+DiffAugment still does not solve the two fundamental problems in the continuous scenario, since it is not designed for this purpose.

Table 5.6 Average quality of 60,000 fake UTKFace images from cGAN and CcGAN with standard deviations after the “ \pm ” symbol. “ \downarrow ” (“ \uparrow ”) indicates lower (higher) values are preferred

Method	Intra-FID \downarrow	NIQE \downarrow	Diversity \downarrow	Label score \downarrow
cGAN (60 classes)	4.516 ± 0.965	4.516 ± 0.965	4.516 ± 0.965	11.087 ± 8.119
cGAN (60 classes) + DiffAugment	4.516 ± 0.965	4.516 ± 0.965	4.516 ± 0.965	11.087 ± 8.119
CcGAN (HVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167
CcGAN (SVDL)	0.572 ± 0.167	0.572 ± 0.167	0.572 ± 0.167	10.739 ± 8.340

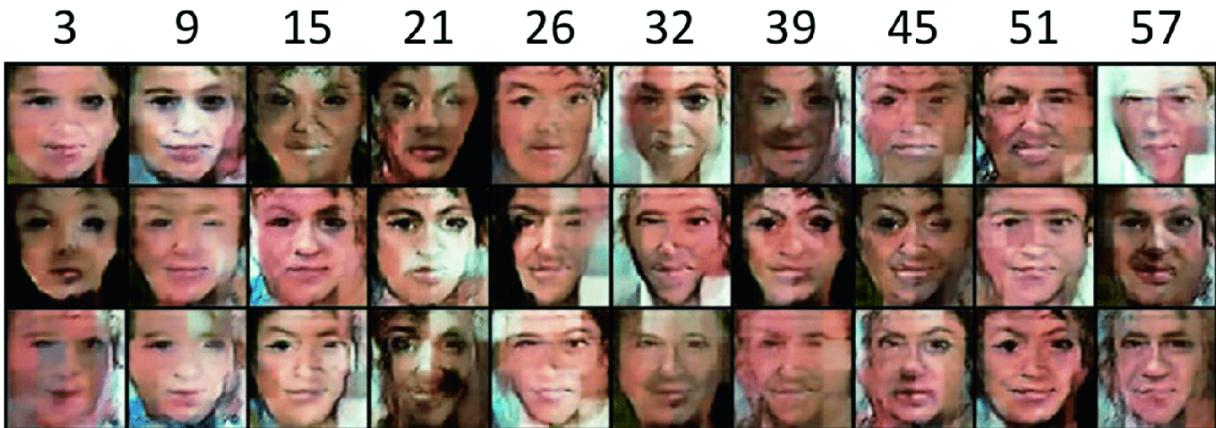


Fig. 5.16 Some example UTKFace fake images from cGAN+DiffAugment. Even with the help of DiffAugment, cGAN still has poor visual quality in the continuous scenario

5.5 Conclusion

This chapter studies CcGAN, the first generative model for image generation conditional on regression labels. In CcGAN, we study two novel empirical discriminator losses (HVDL and SVDL), a novel empirical generator loss and a novel label input method to overcome the two problems of existing cGANs. The error bounds of a discriminator trained under HVDL and SVDL are derived in this work. Finally, in our case studies, we demonstrate the superiority of CcGAN to cGAN on the Circular 2-D Gaussians and UTKFace datasets. In the future, we will investigate more advanced network architectures and novel label input mechanisms to further improve the performance of CcGAN. Moreover, we will evaluate the effectiveness of CcGAN on image datasets with higher image resolution (e.g., 128×128).

References

1. Akata, Zeynep, Perronnin, Florent, Harchaoui, Zaid, Schmid, Cordelia: Label-embedding for image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(7), 1425–1438 (2015)
[\[Crossref\]](#)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223.

International Convention Centre, Sydney, Australia (2017). PMLR

3. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019)
4. Chapelle, O., Weston, J., Bottou, L., Vapnik, V.: Vicinal risk minimization. In: Advances in Neural Information Processing Systems, pp. 416–422 (2001)
5. De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.: Modulating early visual processing by language. In: Advances in Neural Information Processing Systems, pp. 6594–6604 (2017)
6. Ding, X., Wang, Y., Wang, Z.J., Welch, W.J.: Efficient subsampling for generating high-quality images from conditional generative adversarial networks (2021). [arXiv:2103.11166](https://arxiv.org/abs/2103.11166)
7. Ding, X., Wang, Y., Xu, Z., Welch, W.J., Wang, Z.J.: Continuous conditional generative adversarial networks for image generation: novel losses and label input mechanisms (2020). [arXiv:2011.07466](https://arxiv.org/abs/2011.07466)
8. Ding, X., Wang, Y., Zuheng, X., Welch, W.J., Wang, Z.J.: CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation. In: International Conference on Learning Representations (2021)
9. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M., Parvania, M.: Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. IEEE Trans. Smart Grid (2021)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
11. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems, pp. 6626–6637 (2017)
12. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of style GAN. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8110–8119 (2020)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) Conference Track Proceedings 3rd International Conference on

Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015
(2015)

14. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). [arXiv: 1411.1784](#)
15. Mittal, A., Soundararajan, R., Bovik, A.C.: Making a “completely blind” image quality analyzer. *IEEE Signal Process. Lett.* **20**(3), 209–212 (2012)
16. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks (2018). [arXiv:1802.05957](#)
17. Miyato, T., Koyama, M.: cGANs with projection discriminator. In: International Conference on Learning Representations (2018)
18. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2642–2651 (2017)
19. Olmschenk, G.: Semi-supervised Regression with Generative Adversarial Networks Using Minimal Labeled Data. Ph.D. thesis (2019)
20. Olmschenk, G., Chen, J., Tang, H., Zhu, Z.: Dense crowd counting convolutional neural networks with minimal data using semi-supervised dual-goal generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition: Learning with Imperfect Data Workshop (2019)
21. Peyré, G., Cuturi, M., et al.: Computational optimal transport. *Found. Trends® Mach. Learn.* **11**(5–6), 355–607 (2019)
22. Rezagholiradeh, M., Haidar, M.A.: Reg-GAN: semi-supervised learning based on generative adversarial networks for regression. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2806–2810. IEEE (2018)
23. Rezagholizadeh, M., Haidar, M.A., Wu, D.: Semi-supervised regression with generative adversarial networks, 22 Nov 2018. US Patent App. 15/789,518
24. Silverman, B.W.: Density Estimation for Statistics and Data Analysis, vol. 26. CRC Press (1986)
25. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd edn. Springer (2000)
- 26.

Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Proceedings of the 36th International Conference on Machine Learning, vol. 97, pp. 7354–7363 (2019)

27. Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5810–5818 (2017)
 28. Zhao, S., Liu, Z., Lin, J., Zhu, J.Y., Han, S.: Differentiable augmentation for data-efficient GAN training. In: Advances in Neural Information Processing Systems (2020)
 29. Zheng, Y., Sui, X., Jiang, Y., Che, T., Zhang, S., Yang, J., Li, H.: Symreg-GAN: symmetric image registration with generative adversarial networks. IEEE Trans. Pattern Anal. Mach. Intell. (2021)
-

Footnotes

- 1 <https://github.com/mit-han-lab/data-efficient-gans>.

6. Generative Adversarial Networks for Data Augmentation in Hyperspectral Image Classification

Dimitra Koumoutsou¹✉, Georgios Siolas²✉, Eleni Charou¹✉ and
Georgios Stamou²✉

- (1) National Centre for Scientific Research “Demokritos”, Institute of Informatics & Telecommunications, Athens, Greece
- (2) Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece

✉ **Georgios Siolas**

Email: gsiolas@islab.ntua.gr

✉ **Eleni Charou**

Email: exarou@iit.demokritos.gr

✉ **Georgios Stamou**

Email: gstam@cs.ntua.gr

Abstract

Hyperspectral Imaging (HSI), or imaging spectroscopy, is an imaging method with numerous applications. Unlike conventional images, hyperspectral data contain information collected in narrow wavelength intervals at hundreds of bands across the electromagnetic spectrum. The resulting image has a high spatial and spectral resolution allowing for multiple features to be extracted from machine learning applications. However, HSI acquisition is expensive, resulting in low data availability. Especially in the field of remote sensing, hyperspectral datasets only have a

limited number of labeled samples, and significant class imbalances are common. Therefore, synthetic samples are useful in complementing existing datasets used in classification tasks. To this end, a modification of a Gradient Penalty Wasserstein Generative Adversarial Network (WGAN-GP) is proposed for conditional generation of realistic hyperspectral data cubes that refrains from commonly used computationally intense model architectures. Dimensionality reduction is introduced as a preprocessing step that further reduces complexity while retaining only important information. The efficacy of the model is proven by verifying the similarity of the synthetic to the real samples, evaluated by comparing their spectral information and their performance on classification outcomes using spatio-spectral models.

6.1 Introduction

Hyperspectral imaging, or imaging spectroscopy [16], is essentially recording a “scene” in multiple spectra, ranging from the visible to the Near Infra-Red (NIR) [44]. Unlike multispectral imaging, where there are ~ 10 bands recorded [26], hyperspectral imaging collects continuous and contiguous spectral bands at hundreds of narrow wavelength intervals. The resulting image is what is called a “hyperspectral scene”, which is the depiction of a geographic location that contains both spectral and spatial information. Especially in the spectral domain, the resolution of the image is very high. Due to this fact, hyperspectral data offers high potential for analysis. Hyperspectral imaging has been established as a unique kind of imaging, fusing spatial and spectral information. Consequently, robust processing strategies need to be employed to maximize information extraction.

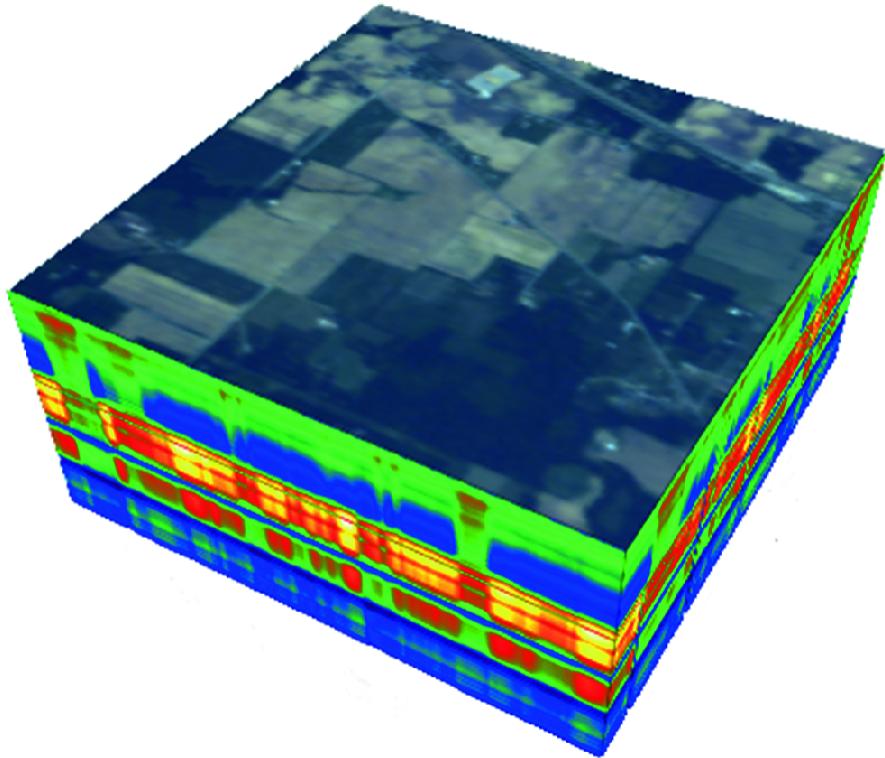


Fig. 6.1 The hyperspectral cube. The first 2 dimensions correspond to the spatial information, while the third dimension is the spectral dimension

The data recorded from a hyperspectral sensor have a 3-dimensional structure, which is called data cube, as shown in Fig. 6.1. The first 2 dimensions correspond to the spatial information, while the third dimension is the spectral dimension. Each spatial pixel is a vector (pixel vector) containing the spectral information, which contains the values of the reflected radiance or reflectance at the given position. If we select a pixel vector and plot it as a function of the corresponding wavelengths, the resulting depiction will be the average spectral signature of all materials and objects covered in the ground area of the pixel. The main advantages over conventional imaging methods are outlined next.

To begin with, a high-resolution image carries large amounts of information which are not possible to be extracted from lower-dimensional image data types. With proper processing and selection of the right algorithms and information extraction methods, it is possible to carry out novel tasks or significantly ameliorate existing ones [36]. Another asset of hyperspectral data is that they provide the “spectral signature” of the

depicted scene. Because each material/object reflects and emits radiation in varying proportions, the reflected radiation from the Earth's surface interacts with each material in a unique way. Differences can be quantified not only amongst different objects, but also for slight variations of the same material [5]. The result of this interaction between radiation and physical objects is called spectral signature because it is unique for each material. This property of the different materials/objects lies in the fact that way in which they reflect, emit, or absorb radiation is unique and characteristic of their physical properties, such as their geometry and chemical composition. Therefore, hyperspectral data can be an asset for Remote Sensing applications if processed appropriately.

Here, we propose a Generative Adversarial Network (GAN) for synthetic hyperspectral data generation. Previous approaches to hyperspectral data augmentation have focused on simple image transformations or computationally heavy deep learning models. In this study, we propose a light model that avoids some, albeit reliable, computationally intense architectures, like convolutional models. Another novelty presented is to reduce the dimensionality before training a GAN model to avoid costly computations that convey little to no useful information to the network. In fact, it is shown for the used dataset that over 95% of the total variance of the data can be retained while performing a 90% reduction on the spectral dimension. Generated samples undergo rigorous testing and evaluation to show that their spectral distribution is consistent with the distribution of the real data. Thus we prove that they are reliable and can be used for data augmentation purposes to compensate for limited data availability or class imbalances.

6.1.1 Dataset

To evaluate our approach the *Indian Pines* hyperspectral dataset is employed, which contains a hyperspectral scene and the ground truth labels for selected pixels of the scene. The dataset is openly available online.¹ The Indian Pines dataset consists of 145×145 pixels in the spatial dimension and 224 spectral bands, although the corrected version used here has 220 bands in the range of 400 to 2500 nm. Data was captured with the AVIRIS sensor in North-Western Indiana, USA, with a GSD of 20 m and are labeled

with 16 classes of vegetation not all of which are mutually exclusive; more specifically there are two-thirds agriculture, and one-third forest or other natural vegetation. The labels of the classes are ‘Alfalfa’, ‘Corn-notill’, ‘Corn-mintill’, ‘Corn’, ‘Grass-pasture’, ‘Grass-trees’, ‘Grass-pasture-mowed’, ‘Hay-windrowed’, ‘Oats’, ‘Soybean-notill’, ‘Soybean-mintill’, ‘Soybean-clean’, ‘Wheat’, ‘Woods’, ‘Buildings-Grass-Trees-Drives’ and ‘Stone-Steel-Towers’.

A false RGB composite of the Indian Pines scene is given In Fig. 6.2. Three bands are used to resemble the RGB color space. This figure aims to provide the reader with a visual perception of the hyperspectral scene which is as close as possible to conventional RGB imaging methods. Overlayed on the RGB composite is a depiction of the ground truth. This is done by drawing the labeled samples over the false RGB composite with different colors for each class. It is evident that labeled samples are few compared to the entire image area. It is important to remember that for a classification task the ground truth should be split in training and test sets, and training data is further split in training and validation sets. In a classification task where 30%, 20% or 10% of total samples is used for training, it is evident that the model is very likely to suffer from lack of training data. Also, classes with few samples are very prone to under-representation in the training set. This proves that generating samples to add during training is of essence in this task.



Fig. 6.2 Indian Pines Dataset with colored overlay of ground truth values. Each one of the 16 provided classes is depicted with a distinctive color

In the next paragraphs, we present two notable issues that often compromise hyperspectral image classification outcomes, and how they manifest in the Indian Pines Dataset. Both are addressed in this study with the use of a conditional GAN model for Data Augmentation.

6.1.2 Data Availability

Implementing a Machine Learning model effectively mostly comes down to providing it with the right training data. This is true even for shallow neural network architectures, but in the case of deep architectures, like deep

learning models, data play a pivotal role. In fact, we know that training a deep learning model with insufficient data results in worse performance than using the same data to train a less complex machine learning model. Hyperspectral data though are inherently abundant and complex so to fully exploit their potential deep learning models are preferred. Unfortunately, collecting data with hyperspectral sensors is a non-trivial and expensive process and is only used in sophisticated applications. Also, collecting and labeling those data is a time-consuming task for researchers. Hence, it is evident that despite the enormous potential of hyperspectral imaging, there is a significant lack of available data. For this reason, we explore the possibility of generating synthetic samples that resemble the real distribution as closely as possible. The generated data could be used to augment training sets used in deep learning architectures in cases where original data availability is limited.

6.1.3 Class Imbalance

The class imbalance problem in a dataset can be described formally as an unequal distribution of samples per class label. There are various issues that occur from a class imbalance problem; the classification accuracy declines, classes with few samples are under-represented in the model, the classifier may overfit on one or a few densely-populated classes, etc. This could be an issue both in a binary classification setting or a multi-class one. In the first case, one class overshadows the other, while in the latter there may be one or few majority classes that “monopolize” the model. One problem with class imbalance is that Overall Accuracy, which is one of the most common metrics used when evaluating a model, is no longer valid. This stems from the fact that Overall Accuracy considers the total number of correctly labeled samples, which in the case of an unbalanced dataset can be high due to overfitting on a majority class or classes. However, finding and processing “weak” classes may be equally or more important than majority classes, depending on the nature of the task. In this case, the overall accuracy metric is deceitful. In Fig. 6.3 we observe the significant class imbalance problem for the Indian Pines Dataset, which is a benchmark dataset for the majority of hyperspectral image classification models in the literature.

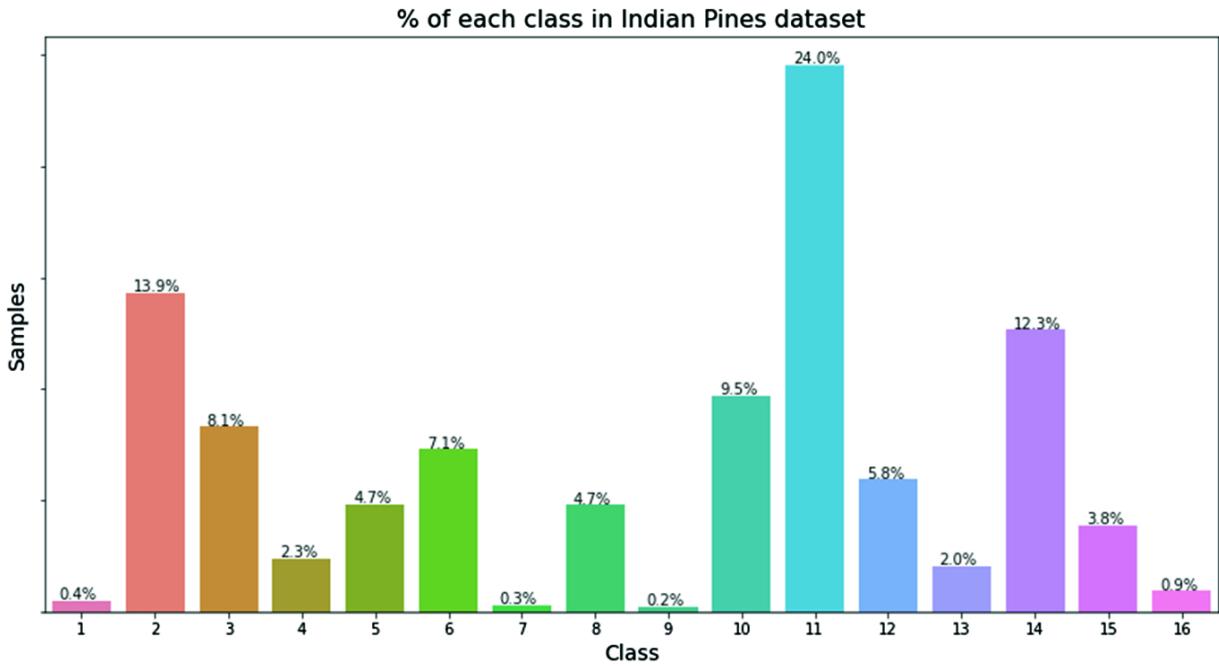


Fig. 6.3 Percentages of training samples per class for the Indian Pines Dataset. The class imbalance issue is evident

6.1.4 Data Augmentation

Data Augmentation is a method utilized under proof that better classification performance can be achieved in deep learning using a dataset augmented with artificial, generated data. Several studies have established the fact that bigger datasets result in better Deep Learning models [47]. However, collecting and annotating large datasets is not always realistic because of the time-consuming hand-crafted tasks involved in data collection and labelling. Data Augmentation is essentially an inflation of the training dataset with data warping or oversampling [46]. One or both methods can be used depending on the specifics of each application. The first transforms original samples while preserving the sample label. Data warping can be in the form of geometric and color transformations, random erasing, random occlusion, neural style transfer, etc. Oversampling augmentations generate synthetic samples which are used as an augmentation set of training samples. Examples of using data augmentation for oversampling include mixing images or using generative modeling, such as GANs.

For the purpose of this study, we focus on two main problems solved by data augmentation techniques. Solving these problems efficiently is crucial

for deep learning models especially and data augmentation techniques have proved efficient for both issues. The first is the problem of class imbalance, which is essentially an unequal distribution of labeled samples across each label set. The second is the lack of training samples in a classification task, which is common with limited-samples datasets, or in a situation where samples should not be used for training for various reasons (i.e. confidential medical information). Lately, using Generative Adversarial Networks to oversample data is gaining in popularity. For limited data problems, generative models can produce labeled samples that complement the dataset and promote good generalization. Oversampling with generative models also helps with class imbalances, either by training the generative model to only generate samples from “weak” classes, or by balancing out the entire dataset.

6.2 Previous Work

In this section we first give an outline of the most notable works on data augmentation and class imbalance, with a focus on GAN based methods. Then, the specific issue of data augmentation for Hyperspectral Imaging is explored. We see how employed methods have evolved in the last years from classic approaches based on transformations on the original data to recent studies employing deep neural networks and generative models that generate data that resemble the original data distribution.

6.2.1 Data Augmentation

The need for data augmentation has been dealt with in many works. When considering oversampling techniques to augment training sets with limited samples, methods range from naive to complex ones. Simple methods would be random oversampling with trivial geometric transformations, e.g. rotations. Other oversampling methods include color augmentations (though these are not suitable in applications where color is of the essence), filtering with kernels, random erasing, etc. These methods may be sufficient for trivial tasks, or to experiment on how class imbalance affects classification by tweaking class percentages. Krizhevsky et al. successfully used simple data augmentation methods, namely image transformations, to improve the performance of ImageNet and reduce overfitting [25].

However, these naïve methods have certain limitations and drawbacks that lead research to more sophisticated approaches, such as Deep Learning-based ones. The Data Augmentation Generative Adversarial Network (DAGAN) was proposed in [1] to assist vanilla classifiers and few-shot learning systems. Also, data augmentation is crucial in certain scientific areas, the most notable of which being medical applications. Due to the lack of available data for rare cases or the confidentiality of medical records, it is often the case that deep learning models targeting medical diagnosis suffer from limited data availability or class imbalances. Synthetic Magnetic Resonance Images (MRI) of slices of the human brain were generated using GANs by Calimeri et al. [7], whereas Computed Tomography (CT) images were generated with GANs to assist liver lesion classification in [15].

6.2.2 Class Imbalance

On the equally important problem of class imbalance, many approaches have been proposed [4]. Manifestations of this issue on Support Vector Machines were reviewed by [34] who showed that training on imbalanced datasets may produce suboptimal models. Class imbalance was also studied by [6], who investigated the impact of class imbalance on image data processing with Convolutional Neural Networks. To address this issue, several methods have been studied. Leevy et al. reviewed many proposed methods that address this issue in [27]. One notable approach to solving the class imbalance problem is to intervene in the data-level [24], which is what data augmentation methods do. Oversampling “weak” classes is based on counterbalancing class distributions so that the model trained on the data is not biased towards one or a few classes that have more samples. One approach to oversampling is Random Oversampling (ROS). ROS is essentially duplicating samples from the “weak” classes, from the original dataset, in a random fashion. Another advanced method is Synthetic Minority Over-sampling Technique (SMOTE) which was proposed by [8] and generates new samples by interpolating new points from existing ones, which has also been successfully used in medical datasets [38].

Several variations of the GAN model have also been proposed to compensate for class imbalances in various applications. Standard oversampling methods were outperformed by using GAN-generated samples from minority classes, even in the presence of high imbalance

ratios, as suggested by Douzas and Bacao [12]. An Adversarial Autoencoder Network [33] was used by Lim et al. to augment rare samples in an anomaly detection application [30]. Huang et al. studied how class imbalance can affect weather classification tasks and proposed a GAN Ensemble Learning approach [22]. Finally, studies on GANs have dealt with the problems of dimensionality reduction and class imbalance in [13, 39].

6.2.3 Applications in Hyperspectral Imaging

Although deep learning and generative modeling have established their potential in data augmentation and in solving the class imbalance issue, there are so far only a few works that employ those methods for synthetic hyperspectral data generation. For this reason, it would be beneficial for studies to concentrate their effort on this task. Our goal in this work is to further investigate how these problems can be addressed using modern GAN architectures. A comprehensive review of notable works on this subject is given in the following paragraphs.

Random Occlusion was proposed in [19], as a data augmentation method for training Convolutional Neural Networks (CNNs). In this paper, the pixels of different rectangular spatial regions from the original image are randomly occluded. The result is the generation of training images with various levels of occlusion. This approach not only augments the dataset but also reduces the risk of overfitting. The results show that this method achieves improved classification accuracies at a low computational cost.

Image transformations were proposed in [48] with the aim of enhancing original datasets by applying three image transformation techniques. More specifically, the three employed operations were flipping, translation, and rotation. They were used to generate data, which were then used to augment the training dataset for a classification model.

The issue of class imbalance was addressed in [28]. Their work was based into splitting the dataset in two categories; densely populated classes, and small-sized classes. The proposed algorithm is based on the orthogonal complement subspace projection (OCSP) and is used to select samples from large-size classes. Then, for small-size sets, the algorithm creates artificial samples to augment those classes. To outline the results of the proposed method, the impact on representation-based classifiers is studied.

Experimental results on a Support Vector Machine (SVM) show that the proposed OCSP algorithm outperforms several other proposed solutions.

Class imbalance was also addressed in [14]. Their method was based on the Rotation forest (RoF) ensemble classifier, which although efficient for HSI classification, suffers from the class imbalance issue. The conventional RoF method introduces biases for the densely populated classes and misclassifies the under-represented classes. In their study they proposed an adaptive ensemble method, which was based on Synthetic Minority Oversampling Technique (SMOTE) for data augmentation and RoF, using differentiated sampling rates. Results showed that the proposed method could generate diversity in the training set and the classifier could achieve better performance than the SVM, the Random Forest (RF), or the original RoF.

A CNN-based method for hyperspectral data augmentation was proposed in [29]. Because conventional data augmentation techniques have limitations, they chose to address the problem with a method called Pixel-Block Pair (PBP). This method uses a deep CNN to extract PBP features. The final label assignment is carried out using decision fusion.

The use of Generative models has so far been limited, although they have the potential to generate samples for data augmentation while avoiding certain drawbacks and restrictions of conventional methods. Two studies are reviewed here, both using Generative Adversarial Networks (GAN) to generate samples. The first one was proposed in [49]. This method addressed data augmentation by considering the complexity and non-linearity of hyperspectral data. They proposed two models, both of which are GAN models that augment the dataset and then use the discriminator model to perform the classification task. The first method was a 1D-GAN and the second a 3D-GAN, which considers spatio-spectral information. Finally, a Generative Adversarial Network with an Auxiliary Classifier (AC-GAN) was proposed in [3] for conditional data generation.

6.3 Wasserstein GAN

Generative models express a representation of a probability distribution function over multiple variables. Some Generative models allow the probability distribution function to be evaluated explicitly, while others may

only implicitly use the probability distribution, for example by drawing samples from that distribution. Generative networks belong in the second category; they allow sampling from the posterior of the desired space given a latent variable. The commonly used Generative Adversarial Network that was introduced in 2014 by Ian Goodfellow et al. is a type of Generative Model [17]. Generative Adversarial Networks are special cases of Artificial Curiosity which are based on training two models that compete each other in a game-like strategy, a concept closely related to predictability minimization [42, 43]. One network is called the *Generator* network, and its goal is to map a source of noise to the input space. The other network is called the *Discriminator*, whose input is either a sample generated by the Generator network or a sample from the true distribution. The goal of the Discriminator is to distinguish between the two types of input and determine whether a sample belongs to the real or the generated (“fake”) distribution. The Generator is trained with the aim of fooling the discriminator into classifying its samples as real.

Wasserstein Generative Adversarial Networks were introduced in 2017 in [2]. They pointed out that the divergences that the original GAN minimize may not be continuous with respect to the generator’s parameters and that could result in difficulties in training. To solve this issue, they propose another method for the cost function, utilizing the *Earth Mover distance*, which is also called *Wasserstein-1 distance*, hence the name Wasserstein Generative Adversarial Network (WGAN). This distance $y \in \mathbb{R}^{120}$ expresses the minimum cost of transporting mass to transform a distribution μ into another distribution x .

More formally, let (M, d) be a metric space for which every probability measure on M is a Radon measure. For $\mathbf{W}_r x_r$, let $\tilde{\sigma}^2 I_{2 \times 2}$ denote the collection of all probability measures μ on M with finite first moment. Then, there exists some π_i in M such that:

$$\int_M d(x, x_0) d\mu(x) < \infty$$

The first or *Wasserstein-1 distance* between two probability measures μ and x in $\tilde{\sigma}^2 I_{2 \times 2}$ is defined as

$$W_1(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y) \, d\gamma(x, y) \right),$$

where $\mathcal{Q}(c|x)$ denotes the collection of all measures on $M \times M$ with marginals μ and ν on the first and second factors respectively. The following dual representation of W_1 is a special case of the duality theorem of Kantorovich and Rubinstein [23] when μ and ν have bounded support,

$$W_1(\mu, \nu) = \sup \left\{ \int_M f(x) d(\mu - \nu)(x) \mid \text{continuous } f : M \rightarrow \mathbb{R}, \text{Lip}(f) \leq 1 \right\}$$

where $\text{Lip}(f)$ denotes the minimal *Lipschitz constant* for f . A real-valued function f on a metric space M is said to be *L -Lipschitz* if there is a constant $L \geq 1$ such that

$$|f(x) - f(y)| \leq L|x - y|$$

for all x and y in M [21].

Under some mild assumptions, the Wasserstein distance is continuous everywhere and differentiable almost everywhere. Thus, when having an optimal training of the discriminator, the minimization of the value function with respect to the generator parameters minimizes the Earth Movers distance function. It should be noted that the original paper called the discriminator a “critic”, because this is not a network trained to classify. The critic function in the WGAN approach has a gradient with respect to its input that behaves better than the original GAN model. Consequently, optimization of the generator’s training is easier.

To impose the Lipschitz constraint on the Discriminator, the approach they proposed is based on the method of weight clipping. Essentially, to enforce the Lipschitz constraint on the critic, weight clipping is enforced on the critic, so that the values lie within a range $[-c, c]$. However, not all functions inherently satisfy this constraint; in fact, there is a set of functions that satisfy this constraint and they are a subset of the L -Lipschitz functions for some L . The value of L depends on c and the architecture of the critic model.

The **Wasserstein Generative Adversarial Network with Gradient Penalty** (WGAN-GP) was introduced soon after the WGAN model was proposed as an improved method for training the WGAN [18]. They found

that weight clipping in many cases is problematic and results in failure of the critic model to converge. More specifically, they argue that even when the training process is optimized, the critic may suffer from a pathological value surface. To prove their claim they tested the identical weight constraint that was proposed in the original GAN paper, hard clipping of the magnitude of each weight, but they also tested other constraints. These include weight constraints such as L2 norm clipping and weight normalization, or soft constraints such as L1 and L2 weight decay. In each of these cases, optimization difficulties occurred. One method that could counterbalance the problem to some extent was batch normalization, which was used in all of the original WGAN experiments. However, the authors of the WGAN paper found that this could only be helpful in the case of rather shallow architectures for the critic; deep models were still likely to not converge.

The solution that was proposed is to enforce the Lipschitz constraint in a different way. Instead of weight clipping, they propose a “gradient penalty”. This idea is based on the property of any differentiable function, which is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. The solution was therefore to constrain directly the gradient norm of the critic’s output with respect to its input. A soft version of the constraint was enforced, introducing a penalty on the gradient norm for random samples $\hat{x} \sim P_{\hat{x}}$. The altered value function is

$$\mathcal{L}_{WGAN-GP}(G, D) = \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{I}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{I}_r}[D(\mathbf{x})] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{I}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\tilde{\mathbf{x}})\|_2 - 1)^2] \quad (6.1)$$

$$\begin{aligned} \tilde{x} &= G(z) \\ z &\sim p(z) \end{aligned}$$

and the minimax game between the two networks is given by

$$W_{GP}(P_r, P_g) = \min_G \max_{D \in \mathcal{D}} \mathcal{L}_{WGAN-GP}(G, D)$$

where:

- G is the Generator,
- D is the Discriminator,

- \mathcal{D} is the set of 1-Lipschitz functions,
- P_r is the distribution of real data,
- χ^2 is a noise distribution from which the generator draws latent variables,
- \hat{x} is the latent variable,
- z is the generator's input, sampled from a noise distribution (e.g. uniform),
- $\mathcal{L}_{WGAN-GP}$ is the loss function of the minimax game.

There are four important things to clarify regarding the implementation of the WGAN-GP model.

Sampling distribution The G_1 sampling is uniform along straight lines between pairs of points sampled from P_r (the distribution of the real data) and the χ^2 (the distribution of the generated data). The motivation for this choice lies with the fact that the optimal critic contains straight lines with gradient norm 1 connecting coupled points from P_r and χ^2 . Enforcing the unit gradient norm constraint everywhere is intractable, so enforcing it only along these straight lines should be sufficient. Experiments support this claim.

Penalty coefficient For all the experiments, $x \in \mathcal{X}$, which was found heuristically to be a good choice for a variety of architectures and datasets.

No critic batch normalization Most of the GANs that have been proposed before the WGAN-GP use batch normalization in both models (the generator and the discriminator) to stabilize training [2, 37, 41]. As discussed earlier, batch normalization could make up for not using the gradient penalty in a WGAN model. However, when enforcing the 1-Lipschitz constraint with gradient penalty batch normalization is no longer a valid option. The reason is that batch normalization changes the form of the discriminator's problem from mapping a single input to a single output to mapping from an entire batch of inputs to a batch of outputs. Because the penalty is enforced upon the norm of the critic's gradient with respect to each input independently, and not the entire batch, batch normalization should be omitted in the critic.

Two-sided penalty The norm of the gradient should move towards 1 (two-sided penalty) instead of staying below 1 (one-sided penalty). The critic appears not to suffer from too much constraint, probably because the

optimal WGAN critic already has gradients with norm 1 almost everywhere under P_r and χ^2 and for the most part in the region in between. Early observations found this to perform slightly better.

6.4 Conditional Wasserstein Generative Adversarial Network with Gradient Penalty for Hyperspectral Image Generation

We propose a *conditional Wasserstein Generative Adversarial Network with Gradient Penalty (cWGAN-GP)* for hyperspectral data augmentation. The proposed GAN network generates fake, labeled patches of hyperspectral cubes which can be used either to complement training and test datasets, or to oversample weak classes in training. A visualization of the model is given in Fig. 6.4.

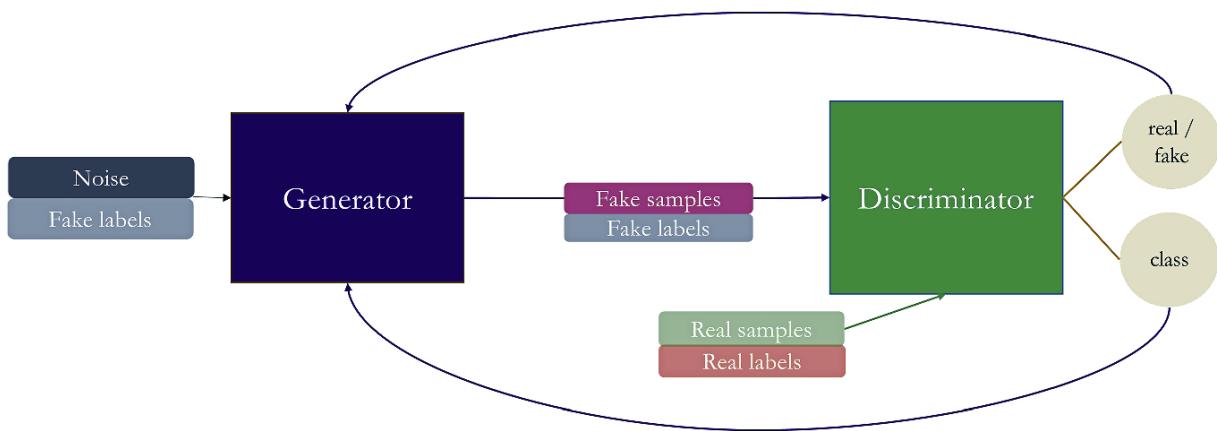


Fig. 6.4 Architecture and training process for the proposed conditional Wasserstein Generative Adversarial Network with Gradient Penalty (cWGAN-GP) for hyperspectral data augmentation

6.4.1 Wasserstein Generative Adversarial Network with Gradient Penalty

Before we delve into the steps of the training process of the proposed GAN model, it is important to explain *why* this model architecture was developed. The first point to be made here concerns the type of GAN used, the WGAN-GP. The Wasserstein distance was chosen for the loss function to avoid the very common problem of mode collapse. *Modes* are peaks in the

distribution of features, or in other words, an area with a high concentration of observations. The occurrence of modes is common in real-world datasets such as the ones used in the task of hyperspectral image classification in Remote Sensing applications. *Mode collapse* is an issue with GANs, where the generator gets stuck in one mode. This is based on the principle that the generator's goal is to fool the discriminator; if the generator finds that the discriminator is stuck at a local minimum and misclassifies certain data, it starts producing data only from that distribution, cannot diversify and gets stuck. So, ultimately, the discriminator is fooled according to the minimax game of the two models, but the generator is not actually learning something useful.

The next choice is to use the gradient penalty in the loss function, to impose 1-Lipschitz continuity. The original WGAN model uses weight clipping to enforce 1-Lipschitz continuity, which is a method that ensures that the weights of the discriminator forcibly take values between a fixed interval. However, this comes with a significant downside. Clipping the weights imposes limitations on the range of values they can take and essentially limits the discriminator's ability to learn, since the gradient cannot function well. This could be solved by using loose limits in the weight clipping, but then the discriminator may be less limited than desired. All these issues were acknowledged by the authors of the original WGAN paper. Using the gradient penalty essentially adds a regularization term to the loss function. This regularization term ensures that in the loss function, the discriminator is penalized when the gradient norm is higher than one.

6.4.2 Hyperspectral Data Patch

So far as the input data is concerned, we chose to generate patches of hyperspectral data instead of individual pixels. The reasons for this choice are the following. Firstly, the generated samples are intended for data augmentation for a deep learning model, so the appropriate input for a specific model should be considered. The Convolutional Neural Network (CNN) is the state-of-the-art model for hyperspectral image classification [9]. A 3D-CNN can exploit both spectral and spatial information if provided with a 3D image data cube. Therefore, to match the desired input of a CNN classifier, the generated samples will eventually be formatted as patches either way. Since patches contain useful contextual information it

makes sense to train a generative model with patches rather than individual pixels to retain this information in “fake” samples.

6.4.3 Dimensionality Reduction

A significant choice regarding the input data is to preprocess the original dataset using a feature extraction algorithm for dimensionality reduction. In this study Principal Component Analysis (PCA) is selected, which was first proposed in [35] and has been established by numerous studies as an efficient algorithm for hyperspectral data reduction [40]. PCA, also called Karhunen–Loëve transform, is a powerful technique of multivariate analysis which seeks for the projection that best represents data in the least square sense. This method is based on rotating the original data into a set of axes that maximizes the variability in the first few axes. This is done by finding a new coordinate system in the hyper-dimensional vector space where the data exhibits no correlation. Let,

$$X_i = [x_1, x_2, \dots, x_N]_i^T$$

$$m = \frac{1}{M} \sum_{i=1}^M [x_1, x_2, \dots, x_N]_i^T$$

$$C_x = \frac{1}{M} \sum_{i=1}^M (x_i - M)(x_i - M)^T$$

where,

D_1 is a hyperspectral image vector of all N dimensions of a given pixel, located at i ,

π_i is a pixel at the i -th spectral band,

M is the total number of pixels in each band,

m is the mean value of all the pixels in the image,

N is the number of spectral dimensions in the original data,

D_1 is the covariance matrix, which is diagonal or uncorrelated in the new coordinate system.

A linear transformation, U , that transforms the original hyperspectral data, D_1 , into the new coordinate system Y is then calculated. The original

covariance matrix D_1 becomes a diagonalized covariance matrix. The solution becomes a generalized eigenvalue problem of the form:

$$C_x = UDU^T$$

where

$$D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_N)$$

$$U = (u_1, u_2, \dots, u_N)$$

Here, D is the diagonal matrix of eigen-values P_r of the covariance matrix D_1 and the orthogonal matrix U is the matrix of eigenvectors π_i . Finally, the PCA transform is computed by

$$Y_i = U^T X_i$$

where $i = 1, 2, \dots, M$.

Each of the vectors J_1 contains the reduced information of the original hyperspectral image. The number of the components used in this final step will be thoroughly investigated in Sect. 6.5.2.

This step is used to avoid generating the entire number of original bands for each sample, but rather use a transformed input that only contains useful information. Information contained in the hyperspectral cube is sparse and because spectral bands are recorded at very narrow wavelength intervals, neighboring bands are most likely correlated. Therefore, using the original recorded data cube does not only slow down the information extraction process; it actually deteriorates it. This is partly because a significant amount of data in the recording is actually noise, but also because training a neural network with redundant data often leads to the model overfitting the training data. Consequently, a powerful feature extraction algorithm should retain only useful information and discard all redundancies. Implementing PCA as a preprocessing step is essential to reduce computational time and improve overall performance. Finally, it is unreasonable to generate high-volume data, only to reduce dimensionality in a next step and it would be highly inefficient. On a practical scope, it is also evident that training a deep learning model with very high-dimensional data cubes is computationally hard, or even impossible with certain hardware limitations.

6.4.4 Discriminator and Generator Models

The Generator model consists of four fully connected layers, followed by their respective activation functions. The LeakyReLU activation function [32] was selected to avoid the dying ReLU problem. The LeakyReLU is a variation of the ReLU function. The original ReLU function essentially “cuts” out the negative values and preserves the input value if it is positive. One major problem about the ReLU function though is that it is not differentiable for all the values (zero) and for all the negative values the derivative is equal to zero. Therefore the learning process, which depends on the derivative of the activation function to update the network’s weights, essentially stops receiving information and the network stops learning. This is called the dying ReLU problem. The Leaky ReLU was developed to overcome this obstacle. For positive values the two functions are identical. In the negative values though, the Leaky ReLU has a small negative slope (a “leak”) that allows for a very small, but non-zero, derivative. The Generator model takes two inputs, one a noise vector of a specified latent dimension, and a one-hot encoded vector to determine the class. The two vectors are concatenated on input. Then, for each epoch, the model outputs a batch of patches, each one of size (S, S, N) .

The Discriminator network is comprised of four fully connected layers as well. The Discriminator does the complementary process; the input is a patch of the hyperspectral cube and the output is a prediction of whether the input is real/fake and whether it belongs to the given class. A detailed architecture of the Generator and Discriminator is given in Fig. 6.5.

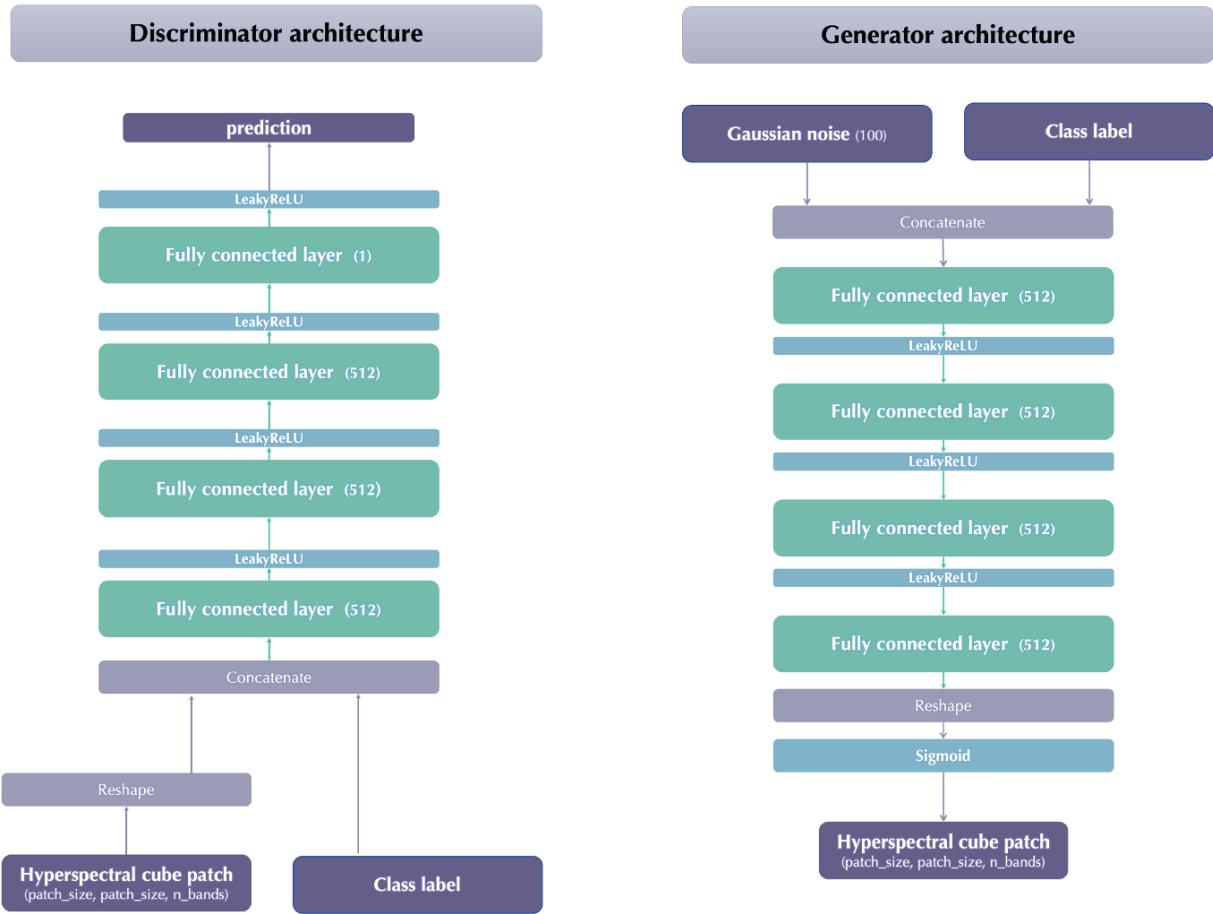


Fig. 6.5 Detailed architecture for the Discriminator (left) and Generator (right)

6.4.5 Training Process

In the following paragraphs, each step in the process of training the GAN is outlined and the parameters are specified. The efficacy of the model is evaluated through rigorous testing which is presented in Sect. 6.5. First, preprocessing of the original data is performed choosing the three main parameters: the patch size, the number of components for the dimensionality reduction step and the train-to-test ratio for splitting the data. 30% of the data is kept for training and the rest is used for testing and evaluation. The patch size and number of components are selected in accordance with the desired input of a CNN classification model and are further explained in Sect. 6.5.2. Overlapping patches of size (12, 12, 10) are extracted and the class label of each patch is assigned according to the label of the central pixel.

In each epoch, training starts with the Discriminator, which is trained more times than the Generator, as described in the original WGAN paper. To train the Discriminator, the following steps are carried out. Firstly, real samples are provided. Then, a fake sample is generated from the Generator, labeled “fake” and given as input to the Discriminator. At this point, the gradient penalty is calculated and used to update the Discriminator’s loss function. Once the Discriminator is trained for the desired number of steps, the discriminator’s training process freezes to proceed with training the Generator. Random noise from a normal distribution and the class label in a one-hot encoding style is given as input to the Generator to generate fake samples. These fake samples are evaluated by the Discriminator, whose decision is passed into the Generator’s loss function. Once the Generator’s training is complete and a new epoch begins, the Discriminator’s parameters are set to trainable again and the training process is repeated. The parameters used when training the proposed model are specified in Sect. 6.5.2.

6.5 Experimental Results

In this section, the Conditional WGAN-GP model is evaluated in order to observe how well it can mimic and complement the original data to solve the limited data problem, while also promoting class equality and ameliorating the class imbalance issues.

6.5.1 Evaluation Metrics

Several evaluation methods are used to draw conclusions; a brief outline is given for each one. For quantitative evaluations on the classification task, two metrics are utilized - Overall Accuracy and Average Accuracy.

- The Overall Accuracy (OA) represents the number or the percentage of correctly classified samples out of the total number of samples. However, OA does not express class-wise accuracies and should be preferred in cases where the overall performance of a model is evaluated. Formally, OA is defined as

$$OA = \frac{\sum_i^C N_i}{T}$$

Where:

- C is the total number of classes,
- G_1 is the total number of correctly classified samples for the i – th class,
- T is the total number of samples in the test set.

- To examine how the model performs class-wise, evaluation is carried out using the Average Accuracy (AA). The Average Accuracy represents the average of the class-wise classification accuracies. It is preferred over the OA because equal weight here is assigned to each class rather than to each sample. An example can be drawn from a binary classification problem where one class has 75% of the total samples. If the classifier can perfectly identify the majority class, it is bound to have an $OA > 95\%$ each time, even if it completely fails on the minority class. In similar cases, where there are significant class imbalances in the dataset, the OA can be misguiding.

For qualitative evaluations on the classification task, several plotting methods are employed. One highly informative plot is drawn when performing PCA on a set to visualize how the variance is distributed across the first principal components. Since high dimensional spaces cannot be visualized in a 2D graph, the first two components can be plotted against each other. In such graphs, the reader can observe the overlap or separation of the categories with PCA analysis. These graphs are useful in several ways. Firstly, it is possible to obtain optimal visualization on the spectral distribution of a dataset, which is not otherwise possible for high-volume data. The second observation which can be drawn is how spectral information is distributed among different classes in the first principal components of the dataset. Those are more information-dense since they contain most of the variance of the total data. Lastly, these graphs are useful when generating fake data to evaluate their properties and see how they compare visually to the original distribution.

6.5.2 Experimental Setting

This work was developed using Python and the pytorch framework for implementing and training the discriminator and generator networks, using

the cuda accelerated functions provided. Visualizations are based on the matplotlib and pandas python libraries. All experiments were run on Google Colaboratory, a platform by Google that provided 25 GB of Random Access Memory (RAM) and using the option to run on a Graphical Processing Unit (GPU). That fact further indicates that the proposed method is a lightweight approach to hyperspectral data generation since it can be run on a cloud-based service.

Model hyperparameters were selected manually, based both on experimental results and literature findings. A detailed explanation is given on how each parameter was chosen. A brief outline is also included in Table 6.1, where the range of values or the methods considered for each step or the literature that supports the given choice is reported. The selected hyperparameters led to an optimal model both in terms of evaluation accuracy and computational time. In particular, the batch size was selected amongst four commonly used sizes: 64, 128, 256 and 512. Findings indicate that small batch sizes created a slow and unstable learning process, so 512 was selected. For the hidden layer size of the fully connected layers of both models, discriminator and generator, it was found that a size of 256 was inadequate to convey the information and led to an underperforming model. On the other hand, using 1024-sized layers accounted for a significantly slower training, without improving performance respectively. The latent dimension for the noise vector which is given as input for the generator model was experimented with in the range of 50 to 500. It was found that 100 was the optimal value, since anything more than that only slowed the process down, while also overtaking the generator's input and thus compromising the class-specific information that was concatenated with the noise. For the optimizer, RMSprop and SGD were tried, alongside Adam and some of its variations. The fastest convergence and results accuracy was given by AdamW [31], which is known for improving original Adam's generalization performance. The Discriminator is trained with a step of 3 iterations for each training iteration of the Generator. A step of 5 was proposed in the original paper [2], but we found that in our case reducing the step to 3 reduces training time without compromising the quality of the results. In fact, this reduction results in a significant increase in training speed, from an average of 12 iterations/second to an average of 21 iterations/second. So far as the weights are concerned, they are initialized

using a uniform distribution, according to the method described by He et al. in [20]. Finally, the penalty weight for the gradient penalty penalization is set to 10, according to the original WGAN-GP implementation from [18].

Table 6.1 Experimental setting for hyperparameters tuning for the proposed GAN model

Hyperparameters	Values/methods	Literature	Result
Batch size	128, 256, 512		512
Hidden layer size	256, 512, 1024		512
Noise dimension	50, 100, 200, 300, 400, 500		100
Optimizer	Adam, AdamW, Adamax, RMSprop, SGD		AdamW, learning rate = 0.001
Training step	3, 4, 5	[2]	3
Weight initialization		[20]	Uniform distribution
Gradient penalty		[18]	10

In Fig. 6.6 PCA is performed to visualize how the variance is distributed across the first principal components for the Indian Pines dataset. The percentage of the total variance contained on a given Principal Component is given next to it. The categories/land-cover classes are annotated with different colors. The overlap or separation of the categories is visible with PCA analysis. The aim of these graphs is to see how variance is distributed amongst the first principal components and decide on the optimal number of principal components to keep for classification purposes without compromising on the retained information. For visualization purposes, only four principal components have been plotted here. Since those hold 94.3% of the total variance, it is safe to assume that keeping 10 principal components is reasonable. Choosing that value allows for over 90% reduction in the spectral dimension while retaining virtually all of the unique information conveyed by the original 220 bands.

PCA Components Visualization for Indian Pines dataset & Variance in each component

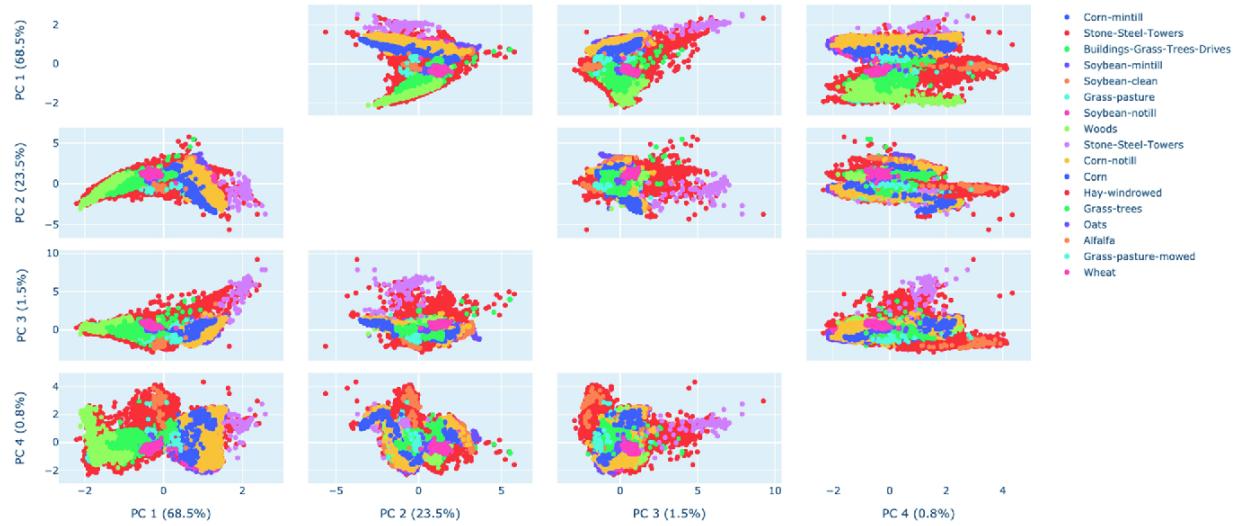


Fig. 6.6 Visualization of the first four PCA components and their percentile variance for the Indian Pines dataset

6.5.3 Spectral Signature

The spectral signature which was introduced in Sect. 6.1 is representative of the different objects and materials in a hyperspectral scene. When transforming the dataset using a feature extraction method the original bands are not retained. However, a successful feature extraction algorithm can convey virtually all the useful characteristics of the original data in the lower-dimensional space. Essentially, the spectral signature is a highly informative measure of how well the spectral information of the class has been conveyed by a model. To evaluate the GAN-generated data with regards to their spectral content, the spectral domain is visualized and compared with the distribution of the original data. Since it is not possible to visualize more than two dimensions simultaneously, first PCA is performed and then the first two components are retained and plotted against each other. With this method, the spectral distribution of the real data is compared to the “fake” distribution (GAN-generated). The two distributions are fairly similar as seen in Fig. 6.7. The only drawback seems to be that the class boundaries are not so distinct in the fake distribution. For example, the “oat” class appears more scattered. Also, the clustering of the “Grass-trees” class towards the bottom-right edge of the plot, which has a few distinct sub-scatters on the original data, is not so well formulated in

the fake class. Finally, it can be observed that there are some overlaps of a few classes at the center of the plot for the generated samples, while the same area for the real distribution is more well-defined.

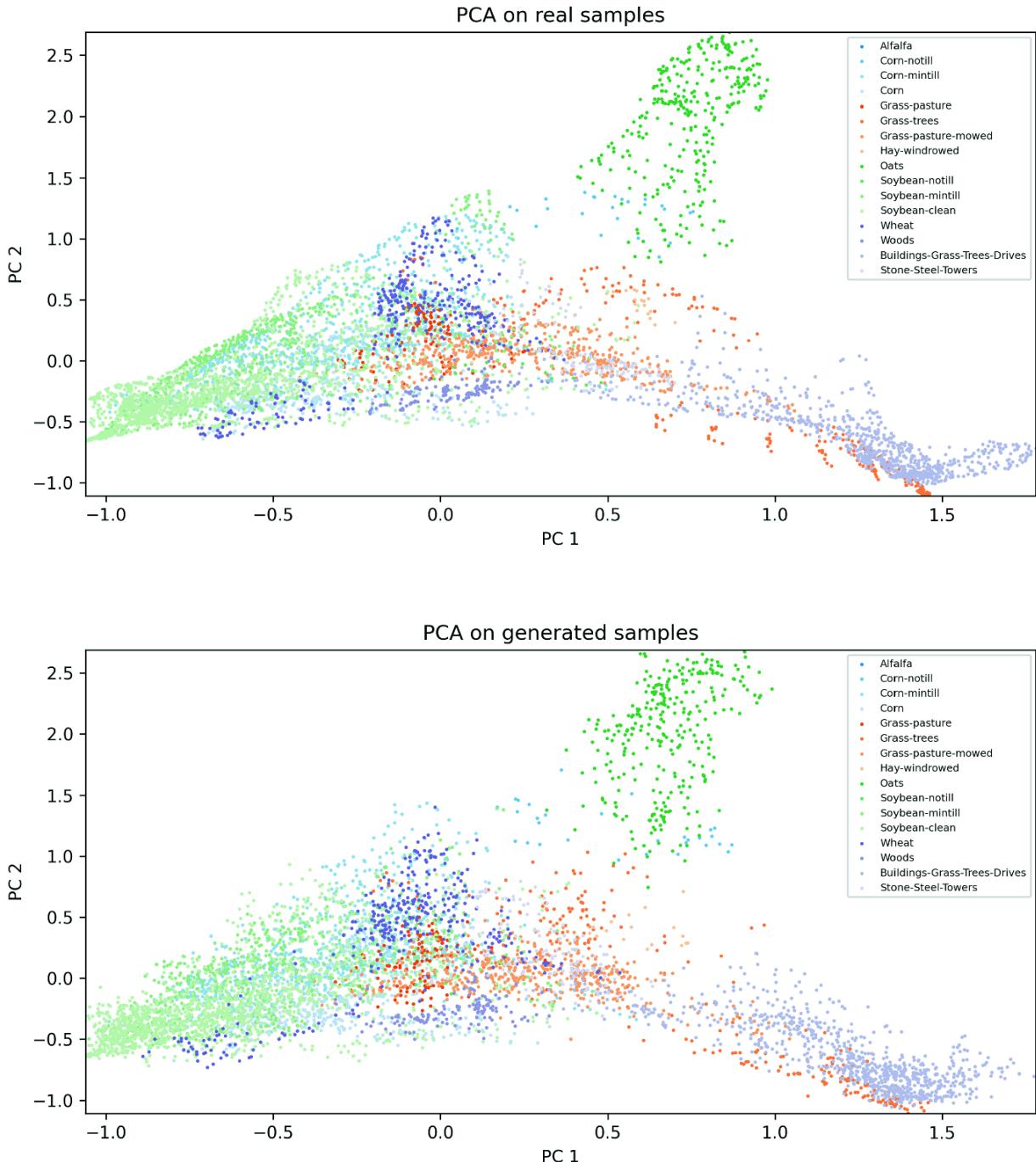


Fig. 6.7 Visualization of the first two PCA components for the real data (left) and for the “fake” GAN-generated data (right)

Another way to inspect the spectral distribution of each class is to plot the spectral distribution across the spectral components to compare the real and fake distributions. This is also a valid way albeit a lengthy one; it would require two plots for each class which could add up to a large number of plots in multilabel classification. Although the PCA method on the entire dataset which was given in Fig. 6.7 is useful for a quick evaluation, it is worth considering the class distribution plots as well. To quantitatively confirm that the fake data distribution is similar to the real distribution the Euclidean distance was used. In particular, the average spectral response for each principal component was calculated for the real and synthetic data. Using the Euclidean distance to evaluate the distance between the real and fake distributions proves that the generated samples properly retain the spectral information. Further visualization of the class distributions is given in Sect. 6.5.5.

6.5.4 Data Augmentation

The goal of the data augmentation technique here is two-fold; firstly to reinforce the data with more training samples in a setting of extremely limited data availability, and secondly to restore the balance on a dataset with severe class imbalance.

6.5.4.1 Classification with SVM

To prove that the generated samples are representative of the real distribution and can be used in a classification task without compromising the accuracy, we can evaluate the data on a Support Vector Machine [11]. It is noted here that for all the experiments with SVMs that are presented next, the dataset (real or fake) is split into 70% test data and 30% training. The optimal SVM for the classification task was determined using a Grid Search and has a linear kernel and a regularization parameter of 100. To carry out our evaluation, two SVM models are trained. The first one is trained only on real data. Then, it is evaluated on two scenarios: using only real data and using only fake data. Then, the second SVM is trained only on the fake data and is again evaluated on the same scenarios: real and fake data. The first case will be the reference case for the experiments, as it does not consider the GAN generated data at all. The second case will show if the fake (GAN generated) data respect the original distribution, in which case the model

trained on real samples should correctly classify the fake data. The third case, where a fake SVM is evaluated on fake data is indicative of how well class-specific information is preserved during synthetic data generation. The last case where real data are used to evaluate the fake SVM will essentially determine if the generated data can be reliably used to augment a hyperspectral dataset for a classification task, without introducing false information on the trained network. Results of these four experiments are given in Table 6.2.

Table 6.2 Evaluation of GAN-generated samples using Support Vector Machines

SVM accuracy	Real data training	Fake data training
Real data evaluation	73.80 ± 0.4	75.99 ± 1.0
Fake data evaluation	75.41 ± 1.0	85.96 ± 1.4

The second case shows that the fake data respect the original data distribution and for the most part are categorized with the same accuracy as the original ones. The slight increase in accuracy can be attributed to the fact that some details of the class distributions have not been perfectly preserved. To further prove that point, it is observed that the fake SVM on fake data has higher accuracy, since class boundaries are not well preserved, which in turn affects the robustness of the classifiers' learned boundaries. However, the last case shows that the accuracy that is achieved using real data on that model is comparable to the real SVM. This essentially proves that the proposed GAN model can generate data that convey similar information as the real ones and can be used for their intended purposes.

6.5.4.2 Classification with CNN

For more rigorous testing of our GAN-generated data, a 3D CNN model is used. This model architecture is selected to achieve maximum accuracy since the 3D CNN has proved a reliable classifier for hyperspectral data [9, 10]. Because a 3D convolution can take into account all three dimensions of the hyperspectral cube, the 3D CNN model learns both spatial and spectral features. However, because convolutions are computationally expensive, it is very common to use patches of the hyperspectral cube as an input to

CNN classifiers to reduce the input size to a minimum. This practice also dictates the shape of the proposed GAN model’s output.

The 3D CNN classification test is of high importance as an evaluation method for our study. When the hyperspectral patch is generated from our architecture of fully connected layers in both the Generator and Discriminator, the input is flattened. To retain the structure of the 3D cube throughout a GAN model, convolutional layers would be necessary. However, convolutions are computationally expensive, resulting in very slow performance. The goal is to produce an accurate model, while avoiding the computational burden that CNN model architectures impose. Then, it should be investigated whether the proposed light architecture can generate results that perform well in a classification model that explicitly uses the spectral and spatial information of the 3D data cube. That would indicate that our proposed method retains the complex structural information of hyperspectral data in a successful manner.

Table 6.3 Data augmentation with GAN-generated samples

Data augmentation scheme	OA	AA
10% real	88.75 ± 0.2	77.46 ± 0.4
10% real & 10% fake	93.70 ± 0.3	91.33 ± 0.4
5% real	75.15 ± 0.4	59.40 ± 0.9
5% real & 5% fake	83.15 ± 0.7	72.35 ± 0.7

In Table 6.3 we see how data augmentation affects the classification task using a 3D CNN in two situations. For both scenarios, the OA and AA are reported along with their respective standard deviation, which was calculated by considering 10 runs of our method. In the first case we consider a situation where we get 10% of the original dataset for training. The improved OA in this case shows that the addition of “fake” data has ameliorated classification outcomes. The same is true when there is an extreme data limitation problem; 5% training data is considered next, and again the result is improved, approximately by 8%. However, it should be noted that data augmentation with 5% fake data on 5% real, which amounts to 10% training data (regardless of which distribution they belong to) does not reach the 10% real data accuracy. This would be considered an ideal

data augmentation application and the proposed model, although unable to reach this performance at this point, surely shows great potential as it improved results in both classification settings.

The class imbalance issue is also addressed here which is already obvious in the AA scores, where there is an increase by up to 14%. The class-wise accuracy is plotted to compare the performance of the model trained with ‘no data augmentation’ to the performance of the model trained on the ‘GAN-augmented dataset’. The results are shown in Fig. 6.8. In both cases, many classes have their accuracies improved, but what is most important in this application is the class accuracy in the case of weak classes. The barplots clearly demonstrate that there is a significant improvement, even in cases where there was 0% accuracy in the non-augmented training of the model. We can conclude that the proposed GAN model is effective for data augmentation and can satisfactorily solve both the limited labeled samples problem and the class imbalance problem. This can be observed in the extreme case when the two issues co-exist, which is the setting where only 5% of data is used for training, on the severely imbalanced dataset of Indian Pines.

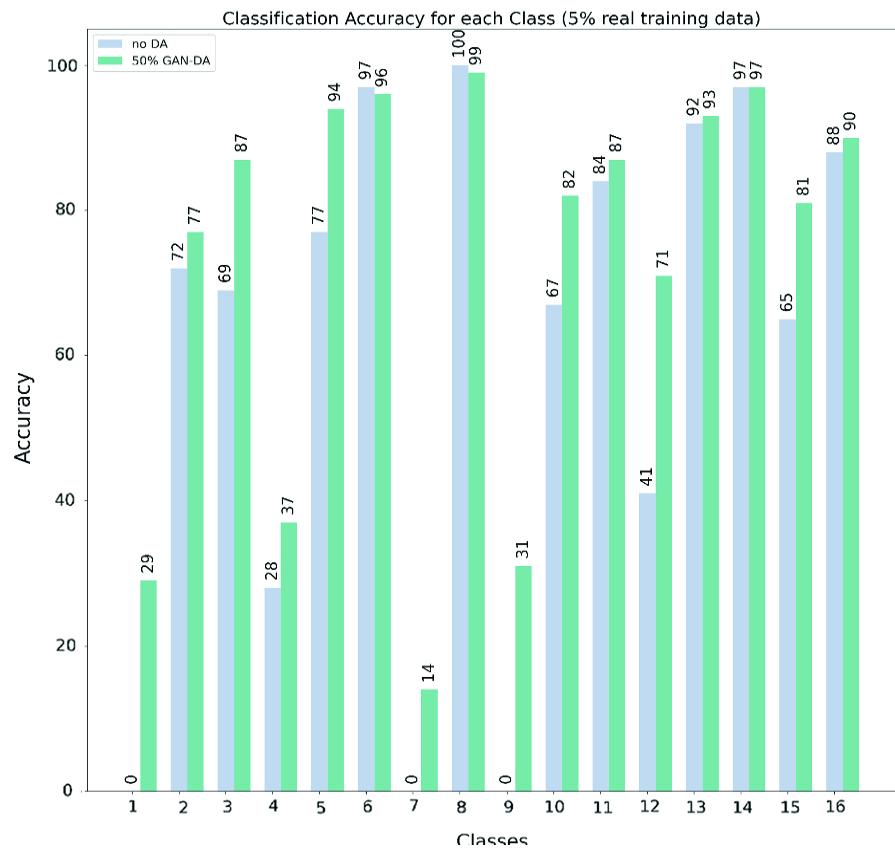
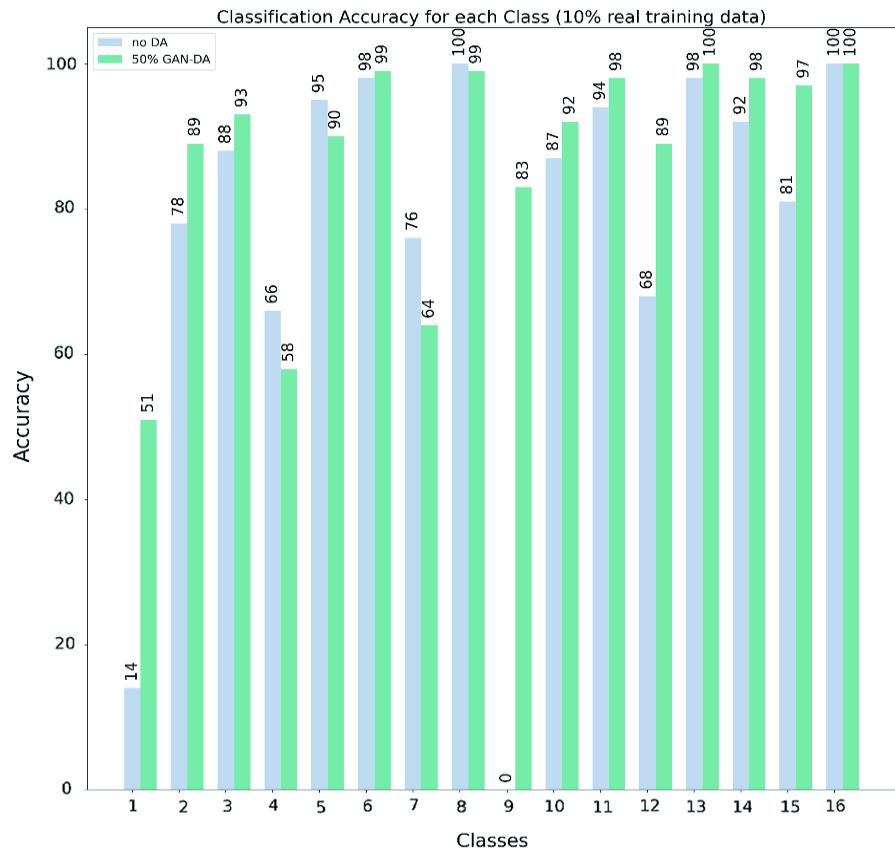


Fig. 6.8 (top) Class-wise accuracy using GAN-generated samples for data augmentation on the Indian Pines Dataset, using 10% real samples and augmenting with 10% fake, (bottom) same task using 5% real versus 5% augmentation

6.5.5 Visualizations

The spectral distribution of the fake samples should resemble the real distribution in order to have realistic generated samples. In Sect. 6.5.3 spectral information on the two distributions was visualized using the first two PCA components from each distribution. Plotting them against each other is the most accurate visualization one could achieve in a 2-Dimensional space.

However, plotting each class separately it is possible to visualize the distribution of spectral information across the spectral components in a class wise manner. This is the closest one could get to a spectral signature on a transformed subspace, since the original spectral signature of each class is altered after performing feature extraction. To retain the original spectral signature while performing a dimensionality reduction technique, one could opt for band selection instead of feature extraction, but this investigation is outside the scope of this study.

In Fig. 6.9 the average spectral distribution for selected classes is plotted for both the real and the generated data, for a visual evaluation of the quality of the GAN-generated data. More specifically, all pixels belonging to a given class are used to calculate the average spectrum for the real distribution. For the fake data, the central pixel of each generated patch is used to compute the average, as this is the one used to label the patch as belonging to the assigned class. Also, the standard deviation is plotted to indicate that there are no significant outliers in the distribution of either the real or the fake data. It is found that the original distributions are well preserved in the generated samples. Four classes were selected based on their “strength” in the original dataset according to Fig. 6.3. Thus we plot one predominant, one strong, one average and one weak class. It should be noted that even the least represented class on the original dataset, Class 16, has been very well preserved in terms of its spectral signature on the synthetic dataset.

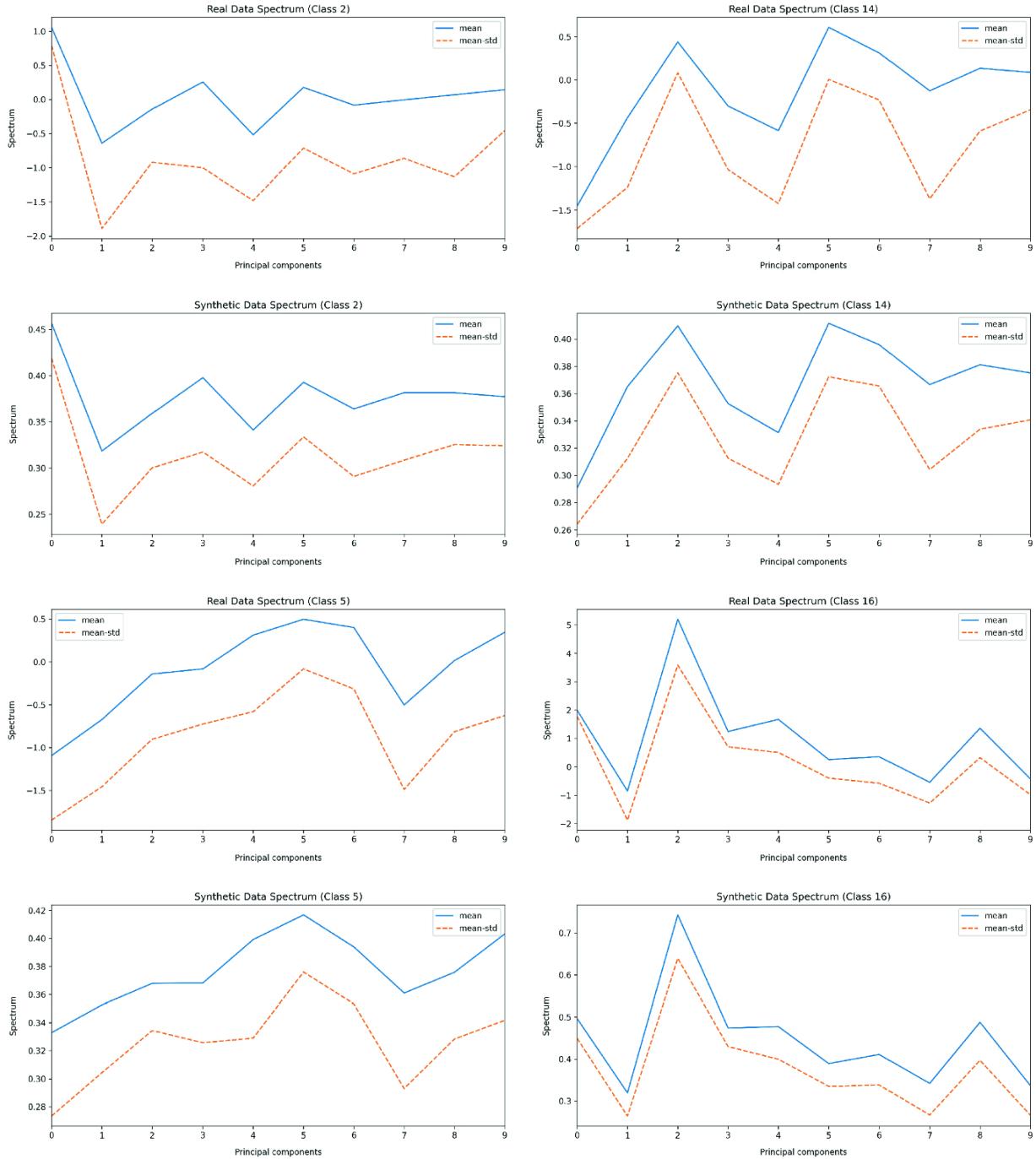


Fig. 6.9 Real and “fake” (GAN-generated) data for four representative classes of the Indian Pines dataset (classes 2, 5, 14, 16). (top left) Class 2 is the second most populated class, (top right) class 14 is a strong class, (bottom left) class 5 is an average case and (bottom right) class 16 is a weak class containing less than 1% of the total samples

6.6 Conclusion and Future Work

This work studies two common issues of supervised hyperspectral data classification. Firstly, the scenario of limited data availability is considered and how classification models could be assisted with data augmentation methods. Then, the issue of class imbalance is thoroughly investigated, along with the problems it poses on the classification task. A way to counterbalance the dataset at the data-level is proposed that aims to solve the disproportions. Thus, a modified *Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP)* is proposed for conditional data generation. This technique provides unlimited realistic hyperspectral image patches to be used for data augmentation. Contrary to previous approaches to hyperspectral data augmentation with GANs that use computationally heavy deep learning models, this study proposes a light model that avoids expensive architecture choices. A novel method to preprocess the GAN model input is also presented. The idea of reducing the dimensionality before training a GAN model saves on costly computations on high-volume data that carry redundant information and introduce noise to the network. Evaluation outcomes on the synthetic data are promising and further investigation is expected to yet improve performance and results.

A design choice made in this study was to generate hyperspectral samples in the form of patches. The GAN could have also been trained at the pixel level. However, since the purpose of this model is to augment training datasets which consists of patches, the model was designed to generate samples as similar to the real input as possible. However, one important limitation imposed by the proposed data augmentation method is that the GAN model is constrained by the selected patch size or the number of PCA components. For some applications, it would be useful to generate data in their original dimensionality, both spatial and spectral, for example, to suit other classification approaches or other information extraction tasks. An interesting approach would be to perform some type of interpolation on the spectral dimension of the generated data. To this end, simple interpolation methods, like linear or cubic, could prove effective.

Interpolation techniques on either original or generated samples could be helpful in several other settings. A more sophisticated approach could be

Slerp [45], shorthand for Spherical linear interpolation. Slerp is a method proposed for quaternion interpolation, which makes it a suitable candidate for 3-Dimensional data. With 3-D interpolation, we could generate data beyond the exact knowledge provided by the original dataset. For instance, interpolation can be utilized in the very frequent situation of sensor failing, when loss of some bands can occur due to technical issues.

References

1. Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks (2017). [arXiv:1711.04340](https://arxiv.org/abs/1711.04340)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, pp. 214–223. PMLR (2017)
3. Audebert, N., Le Saux, B., Lefèvre, S.: Generative adversarial networks for realistic synthesis of hyperspectral samples. In: IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, pp. 4359–4362. IEEE (2018)
4. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor. Newsl. **6**(1), 20–29 (2004)
[Crossref]
5. Borengasser, M., Hungate, W.S., Watkins, R.: Hyperspectral Remote Sensing: Principles and Applications. CRC Press (2007)
6. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. **106**, 249–259 (2018)
[Crossref]
7. Calimeri, F., Marzullo, A., Stamile, C., Terracina, G.: Biomedical data augmentation using generative adversarial neural networks. In: International Conference on Artificial Neural Networks, pp. 626–634. Springer (2017)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
[Crossref]

9. Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P.: Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **54**(10), 6232–6251 (2016)
[[Crossref](#)]
10. Chen, Y., Lin, Z., Zhao, X., Wang, G., Gu, Y.: Deep learning-based classification of hyperspectral data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **7**(6), 2094–2107 (2014)
[[Crossref](#)]
11. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
[[zbMATH](#)]
12. Douzas, G., Bacao, F.: Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Syst. Appl.* **91**, 464–471 (2018)
[[Crossref](#)]
13. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M.: Generative adversarial dimensionality reduction for diagnosing faults and attacks in cyber-physical systems. *Neurocomputing* **440**, 101–110 (2021)
[[Crossref](#)]
14. Feng, W., Huang, W., Bao, W.: Imbalanced hyperspectral image classification with an adaptive ensemble method based on smote and rotation forest with differentiated sampling rates. *IEEE Geosci. Remote Sens. Lett.* **16**(12), 1879–1883 (2019)
[[Crossref](#)]
15. Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: Synthetic data augmentation using gan for improved liver lesion classification. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), pp. 289–293. IEEE (2018)
16. Goetz, A.F., Vane, G., Solomon, J.E., Rock, B.N.: Imaging spectrometry for earth remote sensing. *Science* **228**(4704), 1147–1153 (1985)
17. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014). [arXiv:1406.2661](#)

18. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans (2017). [arXiv:1704.00028](https://arxiv.org/abs/1704.00028)
19. Haut, J.M., Paoletti, M.E., Plaza, J., Plaza, A., Li, J.: Hyperspectral image classification using random occlusion data augmentation. *IEEE Geosci. Remote Sens. Lett.* **16**(11), 1751–1755 (2019)
[[Crossref](#)]
20. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034 (2015)
21. Heinonen, J.: *Lipschitz Functions*, pp. 43–48. Springer New York, New York, NY (2001). https://doi.org/10.1007/978-1-4613-0131-8_6
22. Huang, Y., Jin, Y., Li, Y., Lin, Z.: Towards imbalanced image classification: a generative adversarial network ensemble learning method. *IEEE Access* **8**, 88399–88409 (2020)
[[Crossref](#)]
23. Kantorovich, L., Rubinstein, G.S.: *Vestnik leningrad* (1958)
24. Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al.: Handling imbalanced datasets: a review. *GESTS Int. Trans. Comput. Sci. Eng.* **30**(1), 25–36 (2006)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012)
26. Landgrebe, D.: Hyperspectral image data analysis. *IEEE Signal Process. Mag.* **19**(1), 17–28 (2002)
[[Crossref](#)]
27. Leevy, J.L., Khoshgoftaar, T.M., Bauder, R.A., Seliya, N.: A survey on addressing high-class imbalance in big data. *J. Big Data* **5**(1), 1–30 (2018)
[[Crossref](#)]
28. Li, J., Du, Q., Li, Y., Li, W.: Hyperspectral image classification with imbalanced data based on orthogonal complement subspace projection. *IEEE Trans. Geosci. Remote Sens.* **56**(7), 3838–3851 (2018)
[[Crossref](#)]
29. Li, W., Chen, C., Zhang, M., Li, H., Du, Q.: Data augmentation for hyperspectral

image classification with deep CNN. *IEEE Geosci. Remote Sens. Lett.* **16**(4), 593–597 (2018)

[[Crossref](#)]

30. Lim, S.K., Loo, Y., Tran, N.T., Cheung, N.M., Roig, G., Elovici, Y.: Doping: Generative data augmentation for unsupervised anomaly detection with gan. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 1122–1127. IEEE (2018)
31. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2017). [arXiv: 1711.05101](#)
32. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML, vol. 30, p. 3. Citeseer (2013)
33. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders (2015). [arXiv:1511.05644](#)
34. Palade, V.: Class imbalance learning methods for support vector machines. *Imbalanced learning: foundations, algorithms, and applications*, p. 83. Wiley, Hoboken, NJ, USA (2013)
35. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. *Lond., Edinb., Dublin Philos. Mag. J. Sci.* **2**(11), 559–572 (1901)
36. Plaza, A., Benediktsson, J.A., Boardman, J.W., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., et al.: Recent advances in techniques for hyperspectral image processing. *Remote Sens. Environ.* **113**, S110–S122 (2009)
[[Crossref](#)]
37. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015). [arXiv:1511.06434](#)
38. Rahman, M.M., Davis, D.N.: Addressing the class imbalance problem in medical datasets. *Int. J. Mach. Learn. Comput.* **3**(2), 224 (2013)
[[Crossref](#)]
39. Razavi-Far, R., Farajzadeh-Zanjani, M., Saif, M.: An integrated class-imbalanced learning scheme for diagnosing bearing defects in induction motors. *IEEE Trans. Ind. Inform.* **13**(6), 2758–2769 (2017)
[[Crossref](#)]
- 40.

Rodarmel, C., Shan, J.: Principal component analysis for hyperspectral image classification. *Surv. Land Inf. Sci.* **62**(2), 115–122 (2002)

41. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans (2016). [arXiv:1606.03498](https://arxiv.org/abs/1606.03498)
42. Schmidhuber, J.: A possibility for implementing curiosity and boredom in model-building neural controllers. In: Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 222–227 (1991)
43. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Netw.* **127**, 58–66 (2020)
[[Crossref](#)]
44. Shaw, G., Manolakis, D.: Signal processing for hyperspectral image exploitation. *IEEE Signal Process. Mag.* **19**(1), 12–16 (2002)
[[Crossref](#)]
45. Shoemake, K.: Animating rotation with quaternion curves. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, pp. 245–254 (1985)
46. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. *J. Big Data* **6**(1), 1–48 (2019)
[[Crossref](#)]
47. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 843–852 (2017)
48. Yu, X., Wu, X., Luo, C., Ren, P.: Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GIScience Remote Sens.* **54**(5), 741–758 (2017)
[[Crossref](#)]
49. Zhu, L., Chen, Y., Ghamisi, P., Benediktsson, J.A.: Generative adversarial networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **56**(9), 5046–5063 (2018)
[[Crossref](#)]

Footnotes

1 http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Indian_Pines.

7. Face Aging Using Generative Adversarial Networks

Bruno Kemmer¹✉, Rodolfo Simões¹✉ and Clodoaldo Lima¹✉
(1) University of Sao Paulo, Sao Paulo, Brazil

✉ **Bruno Kemmer (Corresponding author)**
Email: bruno.kemmer@usp.br

✉ **Rodolfo Simões**
Email: simoesrodolfo@usp.br

✉ **Clodoaldo Lima**
Email: c.lima@usp.br

Abstract

When computationally aging a face, it is desirable that the age output is close to the expected age and the individual's characteristics are maintained. Traditionally, there have been two kinds of modeling techniques used in this task: prototype-based and model-based methods. The first calculates the mean difference between age groups, and the latter uses parametric models to simulate change over time. Both approaches fail to keep individual characteristics when transforming a face from a younger domain to an aged one. With advances in computer vision, generative models have been applied to perform this task, especially generative adversarial networks (GANs). With them, it becomes possible to generate realistically aged faces of specific individuals. These networks can encode latent information, by enabling a generator model to create new images conditional on an input image. This could lead to improvements in biometric systems, assist in the

search for missing people and in the identification of criminals in an automated way. In addition, generative models are currently used in multiple applications in entertainment. In the last decade, an increasing number of publications focused on applying state-of-the-art generative models on facial aging. Most of these works were done by using a general architecture and customizing it to fit the aging problem's needs. This work enumerates the most frequent architectures, some of the challenges involved in the task, the available benchmark databases, and their applications. Additionally, it compares different face aging architectures, by using the three most frequent databases: FG-NET, UTKFaces, and CACD. Finally, age estimation and face verification techniques were used to measure the effectiveness of the resulting faces.

7.1 Introduction

Age progression, also known as face aging methods, aims to generate aging or rejuvenating effects on a given face image while maintaining individual characteristics. In later years, this topic has been the target of many publications, as it can automatically assist in the search for missing persons, information forensics, identification of criminals, age-invariant verification, and entertainment purposes [39]. In biometric tasks,¹ it allows the reduction of the distance between the individual features present at the time of training and the current state of the faces, especially if the training has been done with old images.

Several factors can intensify human aging: solar exposure, smoking, polluted environment, stress, and genetic factors. Besides, surgical and non-surgical aesthetic procedures can mitigate the effect of time. In addition to the intrinsic particularity of the problem, the interferences caused by other factors (e.g., variations in facial pose, illumination, and expression) and shortage of labeled aging data make learning face age progression a relatively complicated problem [36]. Because of these and other factors, the aging effect is not a deterministic process.

Methods for facial aging can be divided into three categories [29]:

- **Prototype-based methods:** the mean in previously defined age groups is estimated, and the differences between the means of the groups represent

the variation in aging between them.

- **Model-based methods:** parametric models are used to simulate the changes in the elements that make up the face (shape and texture).
- **Generative methods:** generative models are used to generate aged examples conditioned to an original face.

The natural aging process of a specific human must consider some personalized facial characteristics, e.g., birthmarks, which are almost invariant with time. Prototype-based age progression methods cannot preserve individual personality. Once averages are calculated, the individuality of the examples is attenuated. Consequently, these methods are not suitable for use in biometric recognition. Model-based age progression methods require several images of the same individual, and these over a wide range of ages, dramatically limiting their use in biometrics. To preserve the personality, [28] proposed a dictionary learning-based method. A set of age-group-specific dictionaries is learned, and a linear combination of these patterns expresses a unique personalized aging process.

Many recent publications have utilized generative models for image generation and have obtained very realistic results for facial aging. It is possible to generate considerably realistic aged faces with adversarial autoencoders and generative adversarial networks (GANs). The focus of this chapter will be the application of generative methods in facial aging tasks.

In the past, some review papers on age progression were published. In [29, 37], the authors presented the evolution of facial aging and age estimation. However, the authors did not mention generative models since there were no publications on the topic. Finally, in [1], the authors review multiple GANs applications, including a small section on facial aging.

The rest of this chapter is organized as follows: Sect. 7.2 presents an overview of generative adversarial networks beyond reference architectures and the challenges involved. Section 7.3 provides a list of available benchmark databases. Section 7.4 details a comparison between three re-implementations of facial aging methods. Finally, Sect. 7.5 discusses a conclusion.

7.2 Generative Adversarial Networks

Generative adversarial networks, GANs [5], is composed of two neural networks: the generator, a network with the objective of generating examples close to the real data; and the discriminator, a classification network aiming to separate the generated data from the real ones. These two networks compete with each other during the training stage. The generator tries to produce examples so close to the real ones as to deceive the discriminator, which tries to improve the detection of the generated images.

The method is trained in two phases: In the first phase, the discriminator is trained, and its input is composed of a sample of the real data together with the same amount of generated examples.² This is a binary classification problem in which the generated data has class 0, and the actual data has class 1. The cost function used is cross-entropy. In this phase, only the weights of the discriminator network are updated. Afterward, the generator is trained. Again, a set of examples is generated, and all of them receive the same class 1. Therefore, for the discriminator, all this input is made up of real data. As in the previous step, only the weights of the generating network are adjusted. Figure 7.1 illustrates the components present.

The objective of the generator G is to learn the real distribution of the training data, and for the discriminator D , is that at the end of the training, it has a performance approximately equal to a random guess, that is, $\frac{1}{2}$ for both classes. According the Eq. 7.1, the optimization of the GAN is performed with respect to the joint cost function of G and D .

$$\min_G \max_D \mathbb{E}_{x \sim q_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (7.1)$$

This original architecture is also called fully connected generative adversary networks, FCGAN, because it contains neural networks with fully connected hidden layers.

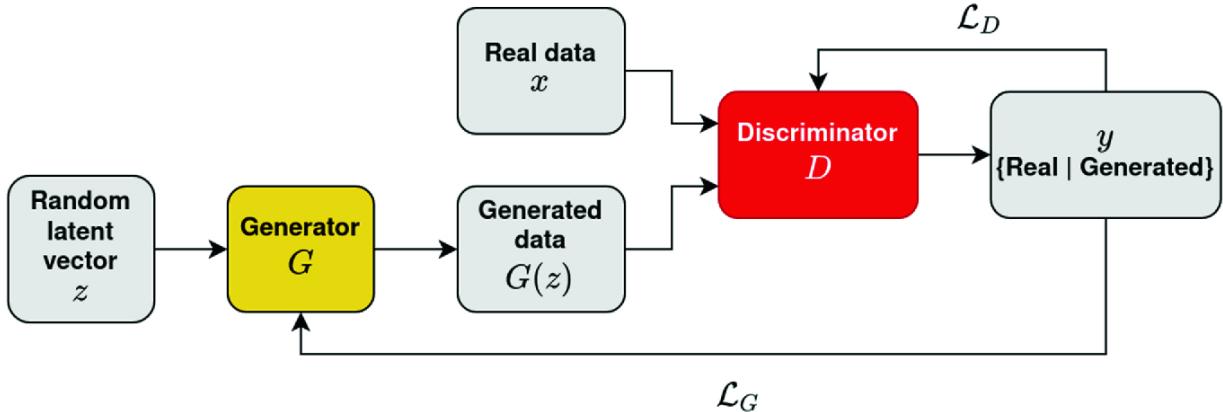


Fig. 7.1 Illustration of the steps in a GAN: the generator network is fed with a random variable z . Its goal is to fool the discriminator network by outputting images as close as possible to the real ones

7.2.1 Mode Collapse

A common problem in FCGAN (and in others that share similar cost functions) is that during the GAN training, the generator can observe a discriminator's vulnerability in detecting whether some classes are true or false. Perceiving any vulnerability, the generator focuses on those classes where it can fool the discriminator more easily—getting to the generator's point to specialize and generate this class with good quality. However, by the time it detects this class's pattern, the generator will be at a local minimum and will not be able to get out of it, stopping the learning of the model. Thus, it is vital to control the trade-off between generated sample quality and diversity during the training.

Another problem that can occur during training is that, traditionally, GANs use as a cost function, a variation derived from the cross-entropy between the real and generated distribution (Eq. 7.1). As the discriminator output is binary, it limits learning as the discriminator does not pass information to the generator of how false it interprets the image to be, only if it is false or real. And since its task is more straightforward, it can develop problems with vanishing gradients.

One approach to solving this problem is to use another cost function, the Wasserstein loss [3]. In this function (Eq. 7.2), the critic C replaces the role of the discriminator, since a classifier is not used to distinguish whether the image is real or generated, but rather a critique³ is used. The critique is also

a neural network and has a requirement that the function it represents must be 1-Lipschitz (1-L) continuous, Eq. 7.3. This means that at all points, the gradient norm has to be at most 1, ensuring stability, as it will grow linearly. When this occurs, the Wasserstein loss function can be said to be valid.

$$\min_G \max_C |\mathbb{E}[C(x)] - \mathbb{E}[C(G(x))]| \quad (7.2)$$

$$\|\nabla f(x)\|_2 \leq 1 \quad (7.3)$$

Two methods that can guarantee the 1-L condition:

- **Gradient clipping:** After updating the gradients and calculating the gradient descent, all values above the limits are replaced by the limit values. One downside of this approach is that it ends up limiting how much the model can learn.
- **Gradient penalty:** In this case [6], a regularization term is added, this being the interpolation between the generated image and the real image. The random variable ϵ defines the proportion of each. This technique guarantees training stability by meeting the 1-L restriction and allowing more significant learning than gradient clipping.

7.2.2 GANs Types

We can classify GAN architectures into two groups, related to how images are generated: conditional and unconditional GANs. In the first group, the generated images are conditioned to a previously defined class. There is no restriction on which class will be obtained in the second, so it is a random event. A considerable benefit of unconditional GANs is that the databases used do not need their examples labeled.

7.2.2.1 Conditional GANs

Conditional generative adversarial networks, CGANs: in this variation, the networks are trained with labeled data, enabling the generating network to create data from specific labels. Thus, the generating network is modeling the distribution of data conditioned by a label.

Figure 7.2 illustrates how the discriminator receives both real and generated data, conditioned to a class. It also shows how the generator gets the latent variable z concatenated to the conditional vector of the classes, a

label encoded in binary encoding form (*one-hot-encoding*). The generating network G will learn to generate examples in the form $\hat{x} = G(z|c)$ and the class information will also be sent to the discriminator.

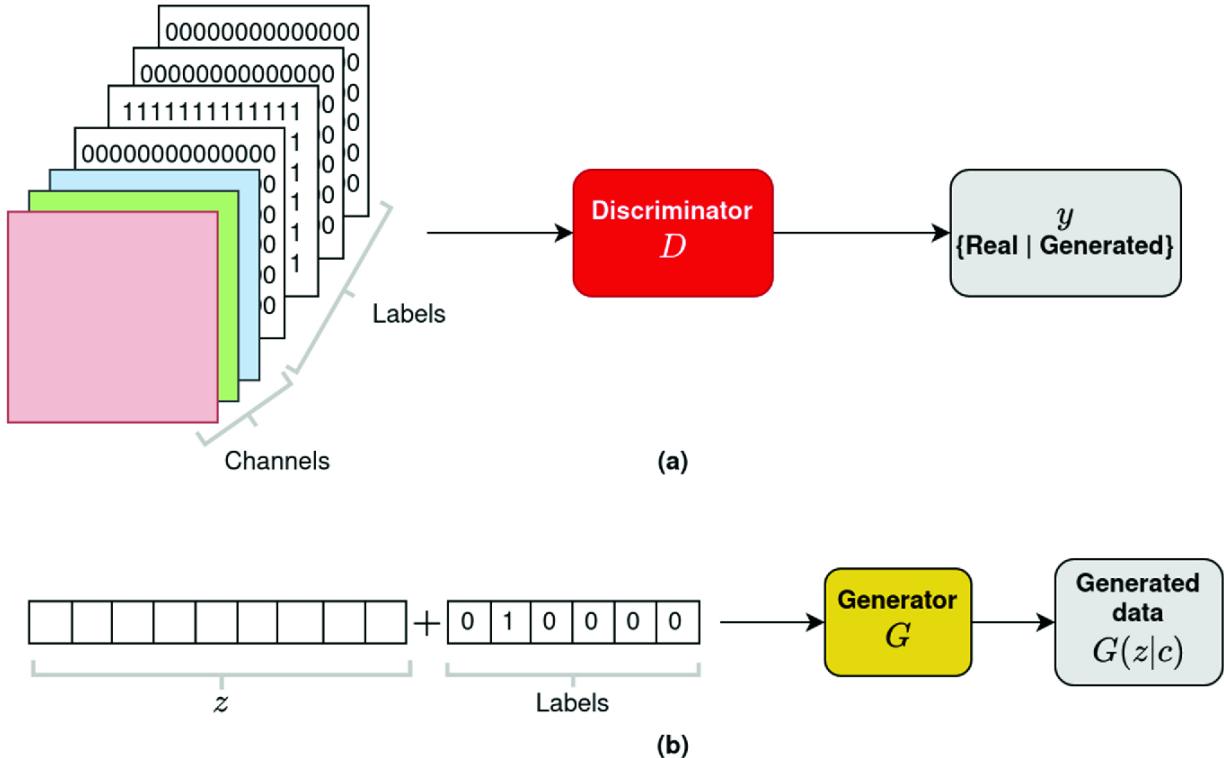


Fig. 7.2 This Figure displays the difference between inputs in conditional and unconditional GANs. Figure **a** presents an illustration of the discriminator input in a CGAN architecture. It is composed of image channels (generated or real) and binary encoding matrices with class information. Figure **b** shows generator input in an unconditional GAN. The input is a concatenation between the latent vector z and the labels in a binary encoded vector

The application of conditional GANs in facial aging occurs by mapping each age group to a class. The input image is encoded to its latent vector z , concatenating with the binary vector of the desired age class. With the generator already trained, it will receive $z \sim P(Z)$, generating a synthetic version of the individual in the age range specified by the class, $G(z * |c)$.

7.2.2.2 **Unconditional GANs**

As described earlier, this type of GAN will generate an image in a random class. One way to control the attributes of the obtained image is to

manipulate the latent vector z , which will be responsible for defining the characteristics of the image at the output of the generator. This manipulation can be used to add elements (glasses, mustaches, etc.) and modify attributes (change hair color, age, rejuvenate, etc.). For this to be possible, it is necessary to find dimensions in the latent space of z that can control these characteristics. This approach does not need labeled databases.

The latent space is varied until a generated image with the desired characteristics is found, unlike conditional GANs where the generator is trained with data labeled with predefined classes.

One of the problems that can occur in this process is when the dimensions of the latent variable are correlated. As there is no penalty for variables not being correlated, when two characteristics co-occur in the training database, there is a significant chance that they are represented in the same dimension of z . For example, being a man, having a mustache, and trying to manipulate a generated image of a woman to contain a mustache will make her look masculine.

Furthermore, if the latent vector does not have enough dimensions to store the attributes generated separately, a dimension may control multiple image characteristics, making later control difficult or impossible.

One method to obtain controlled latent vectors is to use the gradient of a pre-trained classifier that classifies whether the generated image has the desired characteristic or not. Thus, making it possible to move the latent vector z in the direction that contains this characteristic.

7.2.3 Reference Architectures for Facial Imaging

7.2.3.1 CGAN

Conditional generative adversary networks, CGANs, were proposed by [19]. The networks are trained with extra information allowing the generating network to create data for specific classes. The conditional vector c is a one-hot-encoded label concatenated to the random variable z . The generator network G will learn to minimize $\log(1 - D(G(z|c)))$ and the discriminator will learn to minimize $\log D(X)$, as shown in Eq. 7.4. Both are trained jointly, as in FCGAN, and to generate new images, the input to the generator will be a latent vector in the form of $z \sim P(Z)$.

(7.4)

$$\min_G \max_D \mathbb{E}_{x|c \sim q_{data}(x|c)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z|c)))]$$

This conditional probability makes it possible to modify images received by the generating network. For example, this was done in [10], whose work used a CGAN that gets an image and generates a variation of it with different properties. The name given to this model was: Image-to-image translation. This approach can be used in facial authentication problems that are age-invariant.

In Age-cGAN [2], the work utilizes a CGAN to generate aged faces, with the addition of two pre-trained neural networks, one face verification network to measure how distant is the generated face from the original, as well as an age estimator network to estimate the age of the generated image, and penalizing when it differs from the desired one.

The architecture of an Age-cGAN consists of an encoder $E(x)$ that will map the high dimensional picture in a low dimensional vector z . A face recognition network $FR(\cdot)$ is used to guarantee identity preservation. The Eq. 7.5 shows that the network will penalize its cost function in case the reconstructed image \hat{x} deviates from the original face x , in terms of individual representation. A generator network $G(z, y)$ will generate an aged version of the input image conditional on the one-hot-encoded age condition y . Finally, a discriminator network D will discriminate between the real and the generated images.

$$q = |q| (\cos(\theta) + \mathbf{v} \sin(\theta)) = |q| e^{\mathbf{v}\theta} \quad (7.5)$$

Additionally, [36] shows impressive results by incorporating face verification and age estimation techniques with a CGAN for face synthesis and a pyramid-structured discriminator.

7.2.3.2 DCGANs

Deep convolutional generative adversary networks, DCGANs, are unconditional GANs. At the end of 2015, [23] adopted generators with deconvolution layers and discriminators with convolution layers to generate large and realistic images. Thus, it was one of the first architectures attempting to discover and manipulate the latent dimensions (as explained in Sect. 7.2.2.2). Another positive point of the work was to show empirical techniques that avoid common problems in the training of generators and

discriminators in GANs, replacing pooling layers with stridden convolutions and batch normalizations.

However, when generating considerably large images using DCGANs, the result was locally consistent images, but with “global” flaws, with details perceptible to the human eye.

A DCGAN conditioned on gender and age was used [16] to execute the face synthesis. An encoder capturing the high-dimensional picture in a personal latent vector, connected with the age and gender conditions, becomes the input of the DCGAN, which generates the aged faces.

7.2.3.3 Adversarial Autoencoders

First, Autoencoders (AE) are symmetric models that aim to map the representation of the input data. The model is trained to reproduce the input data in its output. To not map a trivial solution (a simple copy of the input in its output), restrictions are imposed on the network. These can be in the input layer (adding noise to the data) or limitations in the intermediate layer, making it a smaller dimension than the input and output layers.

For this, the model trains an encoder that maps the input data x to a representation $f(x)$ and a decoder, which from this new representation $f(x)$ can return the data to the original representation x . The process is illustrated by Fig. 7.3. The autoencoder seeks to approximate a function that can describe the training set, as its cost function⁴ is penalized if the data in the output layer is not similar to the data in the input layer. Therefore, the model will seek to use each dimension of the middle layer better to capture the essential characteristics of the data.

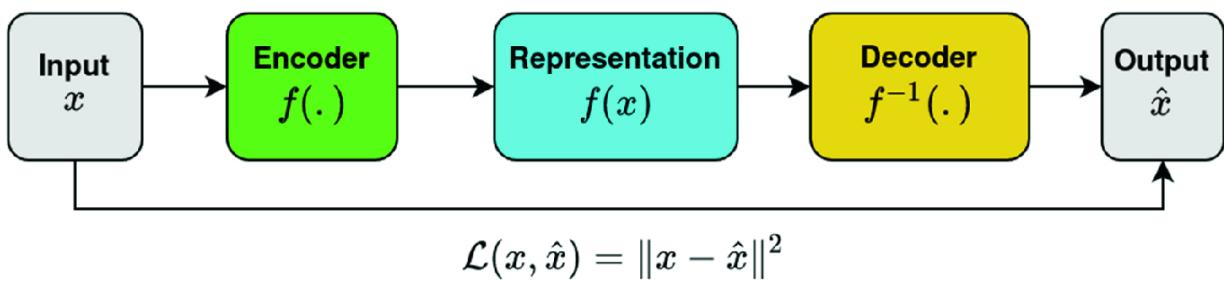


Fig. 7.3 Illustration of the steps in an autoencoder: the high dimensional input x goes through the encoder function $f(\cdot)$, for instance, a neural network, which maps it to a point in a representation space $f(x)$, usually in a lower dimension. Then, the decoder

makes the inverse operation of mapping from the representation space to the original space \hat{x}

Adversarial autoencoders (AAE) were introduced [18], the AAE method uses an adversarial training procedure (similar to Generative Adversarial Networks). However, instead of generating images from random noise like in GAN, AAE utilizes the encoder to learn the latent variables. Therefore, the AAE is a probabilistic autoencoder with a regularization method that requires the latent representation to be parameterized by an arbitrary prior distribution. Since the posterior distribution continues to follow the prior, it generates meaningful examples when manipulating the input features.

The authors defined the problem as: ϵ is the latent code vector, (X, S) and (\hat{X}, \hat{S}) are encoding and decoding distributions, respectively. The encoding function defines an aggregated posterior distribution of $q(z)$. And, the AAE is regularized by combining $q(z)$, to an arbitrary prior, $p(z)$. Figure 7.4 presents an illustration of the architecture of an adversarial autoencoder.

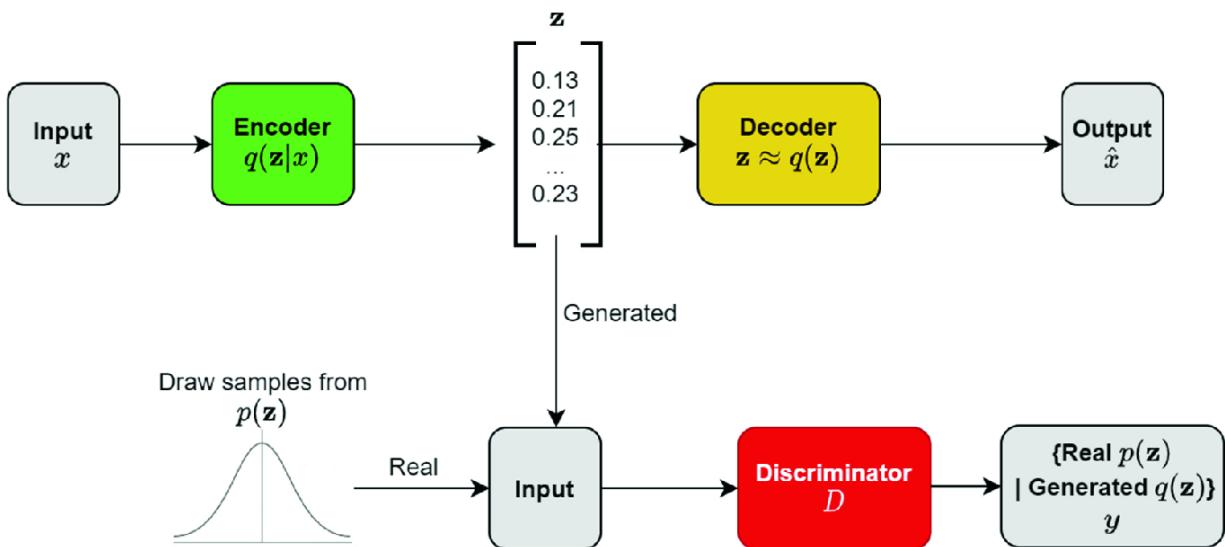


Fig. 7.4 Architecture of an adversarial autoencoder. The superior part represents a standard autoencoder that reconstructs an image x from a latent code z , the decoder generates the output image through latent code z . Finally, in the bottom part, a discriminator predicts whether a sample arises from the latent code z (fake) of the autoencoder or a sampled distribution specified by the user (real)

7.2.3.4 Conditional Adversarial Autoencoders

There is also the possibility of conditioning the AAE's distributions, and this is the case in conditional adversarial autoencoders, the CAAE method [39]. In CAAE, one face \times is mapped to a latent vector through a convolutional encoder E , which extracts the personal features x_r . The vector is concatenated with the age label l_1 , and one point is generated in the latent space, namely $[-c, c]$. Note that the personality ϵ and age l are disentangled in the latent space. Thus we could simply modify age while preserving the personality. Through another mapping deconvolutional generator G , those points are mapped to the manifold M —generating a series of face images, which will present the age progression/regression of \times . The latent vector preserves personalized face features (i.e., personality), and the age condition controls progression vs. regression. The process is illustrated in Fig. 7.5.

The CAAE is similar to AAE. The main difference from AAE is that the CAAE imposes discriminators on the encoder and generator, respectively. The discriminator on the encoder guarantees smooth transition in the latent space, and the discriminator on the generator assists in generating photo-realistic face images. Therefore, CAAE would create higher quality images than AAE [39].

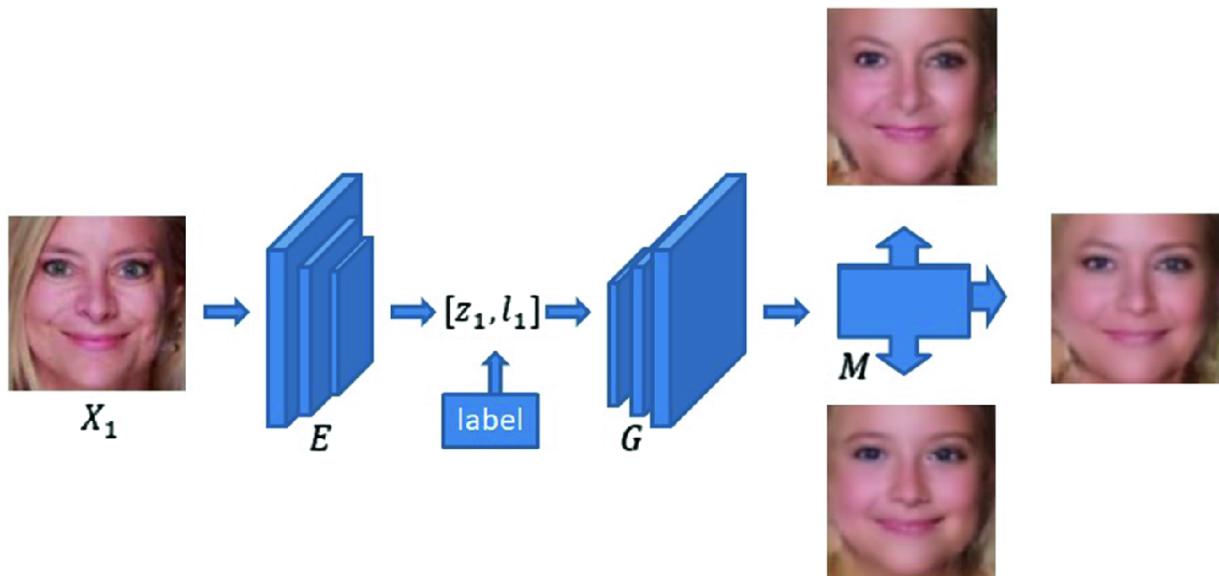


Fig. 7.5 The input face \times is encoded to x_r by an encoder E , representing the personality. The vector is concatenated with random age label l_1 , the latent vector

$[-c, c]$ is constructed. Through another mapping deconvolutional generator G , those points are mapped to the manifold M , representing the age progression/regression of \times . On the manifold, the bottom image illustrates the regression process. The image on the right represents the mapping for the same age, and the upper image refers to the aging process

7.2.3.5 CycleGAN

The CycleGAN [41] performs image-to-image translation, mapping an input image from one domain to another. The objective is to learn a translation function from domain x to y and vice-versa. This is illustrated by Fig. 7.6. There is no need for the database to have paired examples from each domain, an aspect that can be used in facial aging, translating from younger to aged domains without paired examples.

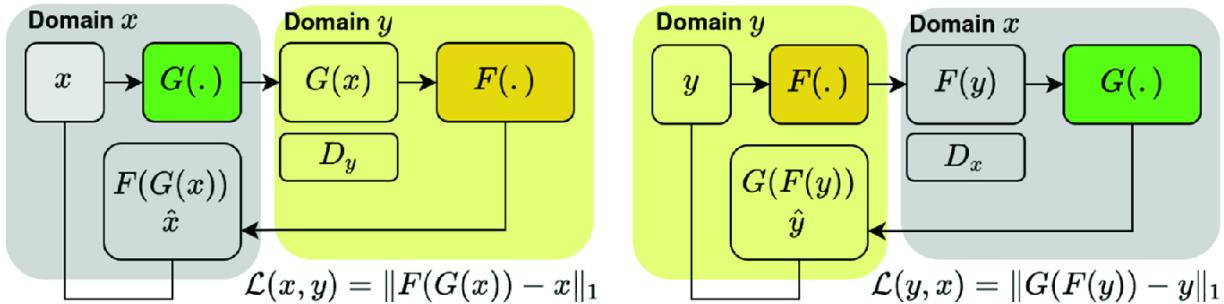


Fig. 7.6 The architecture of CycleGAN: the objective is to learn the translation from domain x to y , and make the differences of generated images $G(x)$ and the real images y imperceptible from the discriminator $\hat{\sigma}_{y^r}$. The same approach is applied to find the mapping from domain y to x

Some works used pairwise training of CycleGANs between age groups, [22] added a pre-trained age prediction model, which improves the results when the gap between original age and the desired is small. In recursive chaining of reversible Image-to-image translators for face aging, RCRIIT [7] used a chained style translation between ranges of ages. For instance, an image between 15 and 25 years was translated to a generated image with an estimated age between 25 and 35, until it reached the desired age. To achieve that, CycleGAN models were used pairwise, one for each pair of age ranges, illustrated by Fig. 7.7. Finally, [35] divides the examples in age ranges and gender, since the aging effects differ between the two, and used CycleGAN to perform the domain translation.

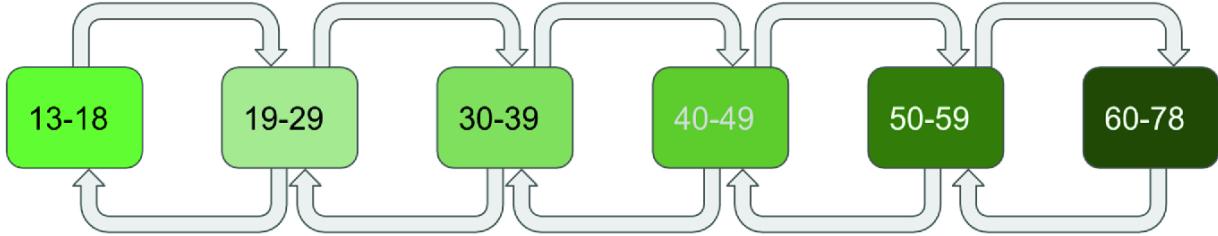


Fig. 7.7 The architecture of RCRIIT: the objective is to learn multiple CycleGANs, from each pair of age groups, so during generation, images could be aged or rejuvenated by multiple domain translations

7.2.3.6 PGGANs

Progressive growing of generative adversarial networks, PGGANs [11] is an unconditional GAN with both the discriminator and the generator simultaneously increasing the complexity of the network architecture over the iterations. They start with an image with few pixels and grow: 4×4 , 8×8 , up to 1024×1024 . That allows the generation of high-resolution images since the networks begin by learning the large structure of the images and progressively address the details in high-resolution images. This approach also reduces the training time, considering most iterations occur when the network is more superficial and allows larger batch sizes due to the memory constraints since, at the start, it will have lower resolution images.

Also, the authors increased the variation using minibatch standard deviation, which is a feature map of the standard deviation of each feature in each spatial location for each minibatch. It is averaged by all features and spatial locations and added at the end of the discriminator.

In [27], the authors propose a framework to interpret the latent representation used as input of PGGAN and StyleGAN for semantic editing. As described in Sect. 7.2.2.2, semantic editing enables control over the generated images after learning how to interpret the latent representations. The framework classifies the attributes on the latent vector using a supervised model⁵ allowing manipulation of the latent variable to produce images with the desired characteristic.

7.2.3.7 StyleGANs

The work presented [12] showed an architecture of GANs which manages to use style transfer techniques between images. The method can obtain latent variables corresponding to the pose, texture, age, among others.

The discriminator and the cost function have not been changed, compared to the original architecture (FCGAN), only the generator. This is composed of two neural networks: the first is a multilayer perceptron (with eight layers), responsible for mapping the random latent representation z to various vectors of styles w , where affine transformations⁶ will be applied. The goal of w is to disentangle the latent variables that will control the style of the generated images. Since each instance had its style removed after applying the normalization of AdaIN, w will apply the desired style through two components: scale and bias.

A significant modification of this architecture is the addition of random noises independent of the encoding z in different parts of the second network, effectively generating the image. This prevents the network from having to store this random component in its encoding. The random vector has an independent value for each attribute, and its elements are normalized in proportion to the characteristic of the changing image. The normalization parameters are adjusted during the training of the network.

A new version of this method was made available at [13], with improvements both in the architecture of the networks and the methodology adopted for their training. In [21], StyleGAN was used to perform face aging in both textures and in shape, an aspect that most works can't do. Also, [32] distilled aspects of the latent representation used by StyleGAN to generate images, enabling gender swap, adding smiles, and face aging.

7.3 Databases Used for Facial Aging

Databases used for facial aging require pictures of human faces of a range of ages. In addition, most methods require multiple images from the same individuals. Below are the most used databases for this task.

7.3.1 FG-NET

The FG-NET database [15] is publicly available, and it is widely used in works related to facial aging. However, compared to other databases used in this task, it has fewer images, which can limit the performance of models

trained on it; thus, its use is more directed to evaluating face aging methods. It contains 1,002 images of 82 people aged between 0 and 69. Table 7.1 shows the distribution of the photos between the ages and the predominance of those under 30 years of age.

7.3.2 UTKFace

The UTKFace database [39] contains more than 20,000 images, and the number of individuals is not specified since the individuals are not identified. The photos were not taken in a controlled environment and contained the metadata of gender, ethnicity, and age of the faces. In addition, it also provides 68 reference points on the faces.

7.3.3 CACD

This database [34] was obtained via internet searches (made between 2004 and 2013) and is composed of photos of celebrities. It contains 163,446 images of 2,000 celebrities, aged between 16 and 62 years old. The age present in the metadata is not exact since it was obtained via subtraction between the date of the photo and the date of birth (which is public, as individuals are famous). However, the base can be used both for aging and for age-invariant facial verification and recognition [30].

Table 7.1 shows the distribution of the photos between the ages of the FG-NET, UTKFace, and CACD datasets.

Table 7.1 Age stratification of the FG-NET, UTKFace and CACD databases

Dataset	0–10	11–20	21–30	31–40	41–50	51–60	>60
FG-NET	371	359	143	69	39	14	7
	37%	35.8%	14.3%	6.9%	3.9%	1.4%	0.7%
UTKFace	3,260	1,833	7,803	4,343	2,103	2,225	2,444
	13.6%	7.6%	32.5%	18.1%	8.8%	9.3%	10.2%
CACD	0	9,647	40,750	43,176	39,819	28,491	1,563
	0%	5.9%	24.9%	26.4%	24.4%	17.4%	1%

7.3.4 MORPH

The MORPH database [24] is also widely used in this task. This database is divided into two groups: album 1 and album 2. The first contains 1,690 images of 515 individuals, both men and women. Album 2 includes 94,000 pictures of 24,000 individuals aged 16 to 77 years, with an average of 4 photos per person. Again, the metadata contains gender and race. The academic version is a subset of album 2, containing 55,000 images from 13,000 individuals, and the commercial version has 202,000 images.

7.3.5 IMDB-Wiki

The database IMDB-Wiki [25] was also extracted from the internet, from two websites: IMDb (portal with information about films and artists) and Wikipedia (free and collaborative online encyclopedia). From the first, 460,723 images were obtained, and from the second 62,328 images. A subset of this base is Celeb-A [17], with 202,599 images.

7.3.6 Cross-Age LFW (CALFW)

The LFW database [9] is a widely used facial verification and recognition database. Cross-Age LFW (CALFW) [40] is a subset that seeks to find people on LTF with a considerable age gap in their pictures. It contains 3,000 pairs of images with multiple photos of each individual.

7.3.7 Other Databases

Flickr-Faces-HQ (FFHQ) [12] is a database with human faces of high quality, intending to be a benchmark for GANs. The 70,000 images have a resolution of 1024×1024 , in PNG format. The AgeDB [20] is a database manually collected with 16,488 images of 568 individuals aged between 0 and 101.

7.4 Experiments and Results

Here is a comparison with three generative models that have been applied for facial aging tasks: CAAE [39], IPCGAN [33], and Recursive Chaining of Reversible Image-to-image Translators (RCRIIT) [7] were performed through re-implementations of publicly available codes. All of these methods were trained for a fair comparison considering six age groups (11–20, 21–30, 31–40, 41–50, 51–60, and 60+).

Three aging databases were used: FG-NET, UTKFace, and CACD. The datasets are commonly used as benchmarks in face aging publications and are freely available. The age stratification for the six groups investigated is presented in Table 7.1. The FG-NET was considered only for evaluating the models. All images in the FG-NET dataset inside the age groups (more than 10 years) were only used for testing. The CACD and UTKFACE datasets were split into two parts, 75% for training and the rest for test. For the CACD dataset, it was guaranteed that there are no different images of the same person in the test and training.

The datasets were standardized with the face detection algorithm from Multitask Cascaded Convolutional Networks, MTCNN⁷ [38]. The model outputs a set of rectangles with the probabilities of containing a face. The rectangle with the higher likelihood was kept and the image was cropped with an external margin to avoid cutting parts of the face. Additionally, three points are localized, for alignment of the eyes and for centralizing the center of the nose. Finally, our image preprocessing reduces the real face resolution to 128×128 .

7.4.1 CAAE Experiments

In [39], the authors propose a conditional adversarial autoencoder that learns a face manifold, which will present a given face image's age progression/regression through a framework that imposes discriminators on the generator and encoder, respectively. The experiments with the CAAE method were carried out through a public implementation,⁸ which has used an architecture similar to the one proposed by the authors. The CAAE originally has ten age groups and uses gender information. For comparison with other methods, the gender information was removed and only six age groups were used instead.

In the experiments, the four blocks (Encoder, Generator and two discriminators) are updated alternately with a mini-batch size of 80 through the stochastic gradient descent solver, ADAM [14] ($\alpha = 0.0002$, $\beta_1 = 0.5$) on 100 epochs. During testing, only E and G are active. The method's training was carried out with the UTKFace and CACD datasets and the method's performance was evaluated in the three datasets considered.

Figure 7.8 presents the images generated through the aging and rejuvenation transformations of the CAAE method.

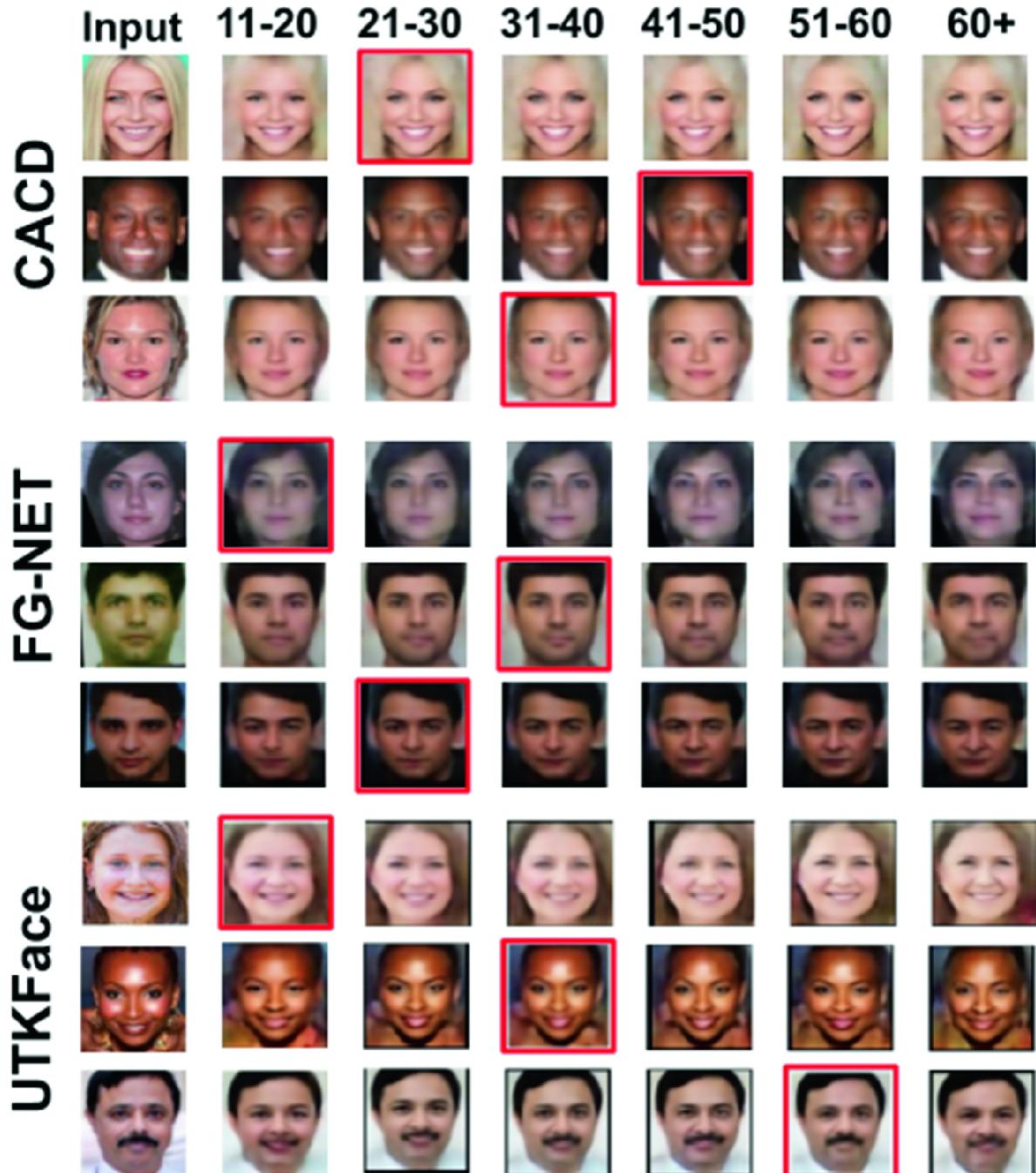


Fig. 7.8 Generated images from the CAAE model for each dataset. The first three rows contain generated images from the CACD test set, rows 4, 5, and 6 from FG-NET, and the last three from UTKFace. The first column shows real faces from the test set; the columns on the right show the synthetic images for each age group. The red

boxes emphasize the person's age when the input picture was taken, so the input and the image in the red box should look alike

7.4.2 IPCGAN Experiments

The method uses a conditional GAN, CGAN, to generate the aged image. To guarantee the identity of the face, a perceptual loss $\mathcal{D} \approx \hat{\mathcal{D}}$ was added in the objective function. In addition, to ensure that the aged images have reached the desired age, an age-estimating neural network was used to penalize its cost function, in an age classification loss $q(\mathbf{z})$ as shown in Eq. 7.6. The P_G term is the CGAN loss defined in Eq. 7.4.

$$G_{loss} = \lambda_1 L_G + \lambda_2 L_{identity} + \lambda_3 L_{age}, \quad (7.6)$$

the J_1 , J_2 and J_3 are hyperparameters which weigh the identity maintenance and aging effects.

The authors originally used the CACD database for training with five classes. Alternatively, to allow model comparison, the datasets were divided into six categories. During the experimentation⁹ with UTKFaces dataset, the best results were observed with these parameters: $\lambda_1 = 100$, $\lambda_2 = 5E-4$ and $\lambda_3 = 30$, the Fig. 7.9a presents the images generated. Figure 7.9b shows the resulting images of training the model with the CACD database using the parameters: $\lambda_3 = 30$, $\lambda_2 = 5E-4$ and $\lambda_1 = 100$.

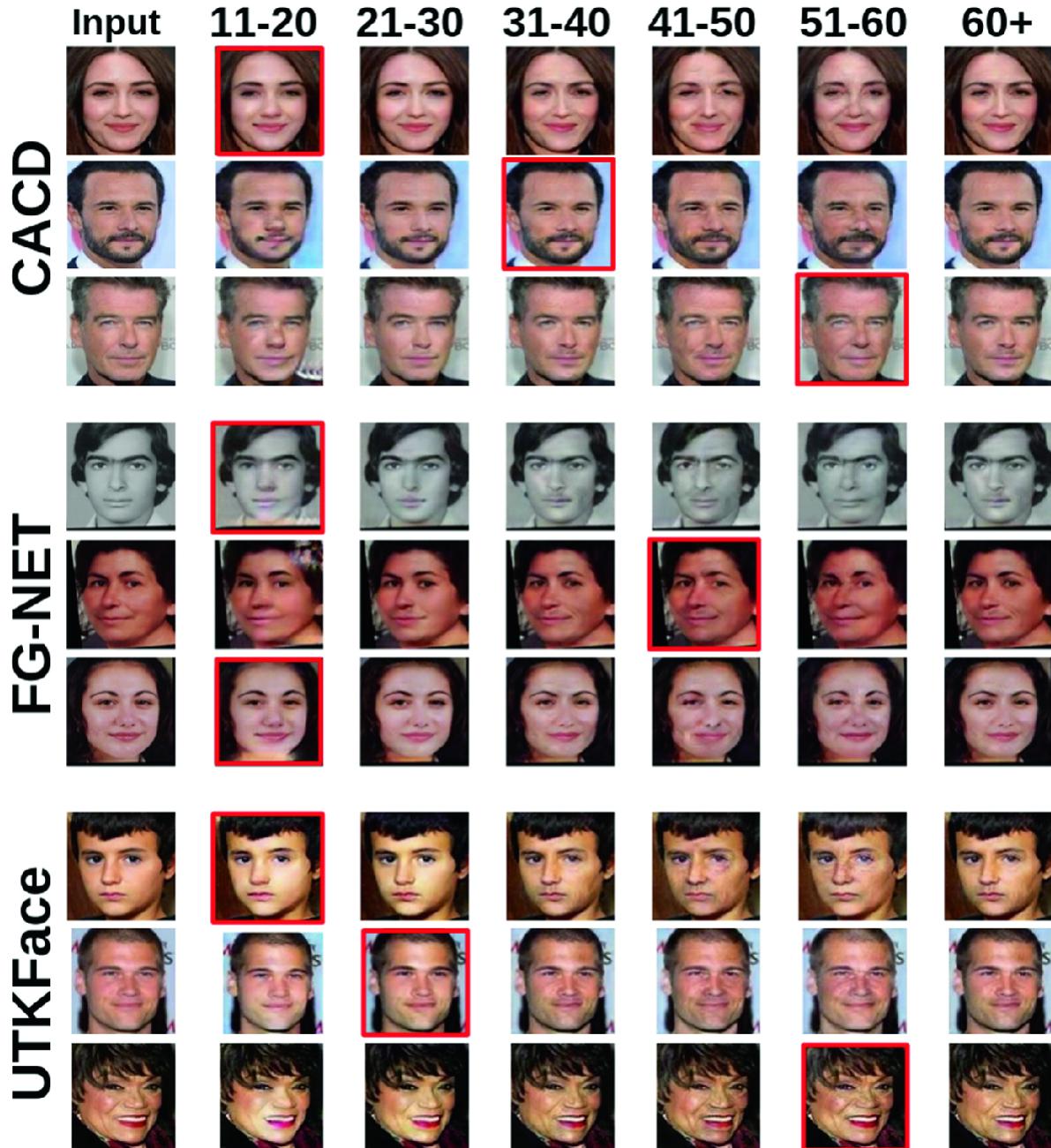


Fig. 7.9 Generated images from the IPCGAN model for each dataset. The first three rows contain generated images from the CACD test set, rows 4, 5, and 6 from FG-NET, and the last three from UTKFace. The first column shows real faces from the test set; the columns on the right show the synthetic images for each age group. The red boxes emphasize the person's age when the input picture was taken, so the input and the image in the red box should look alike

7.4.3 Recursive Chaining of Reversible Image-to-Image Translators Experiments, RCRIIT

The authors [7] address the modeling and simulation of progressive changes over time, such as human face aging. The paper generalizes image-to-image mapping to a sequential setting. The authors propose a recursive adversarial domain adaptation approach capable of producing step-wise transformations for human aging by treating the age phases as a sequence of image domains.

The authors have used the CACD dataset and divided the data into slots 2–18, 19–29, 30–39, 40–49, 50–59, and 60–78 (six domains, with five direct transformation paths). Also, enough samples were removed so that the mean age in the sets is ten years apart 15, 25, 35, 45, 55, and 65, respectively. To follow the idea of the authors' experiments, we divided the CACD and UTKFACE datasets into six groups. However, other intervals were considered (11–20, 21–30, 31–40, 41–50, 51–60, and 60+), and 1,000 images were chosen randomly for each age group. We keep the average age in the sets as ten years apart as the authors (15, 25, 35, 45, 55, and 65). Also, there are six age groups and five transformation modules (trained model pairs). The experiments considered here use the pipeline¹⁰ made available by the authors. The full chain was obtained with five pairs of independently trained successive modules. Each module has been trained with a mini-batch size of 1 with ADAM ($\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$) on 100 epochs. The inference process considers the output of the previous module in both directions, backward to the rejuvenation process and forward to the aging process. Figure 7.10 presents an illustration of aging results.

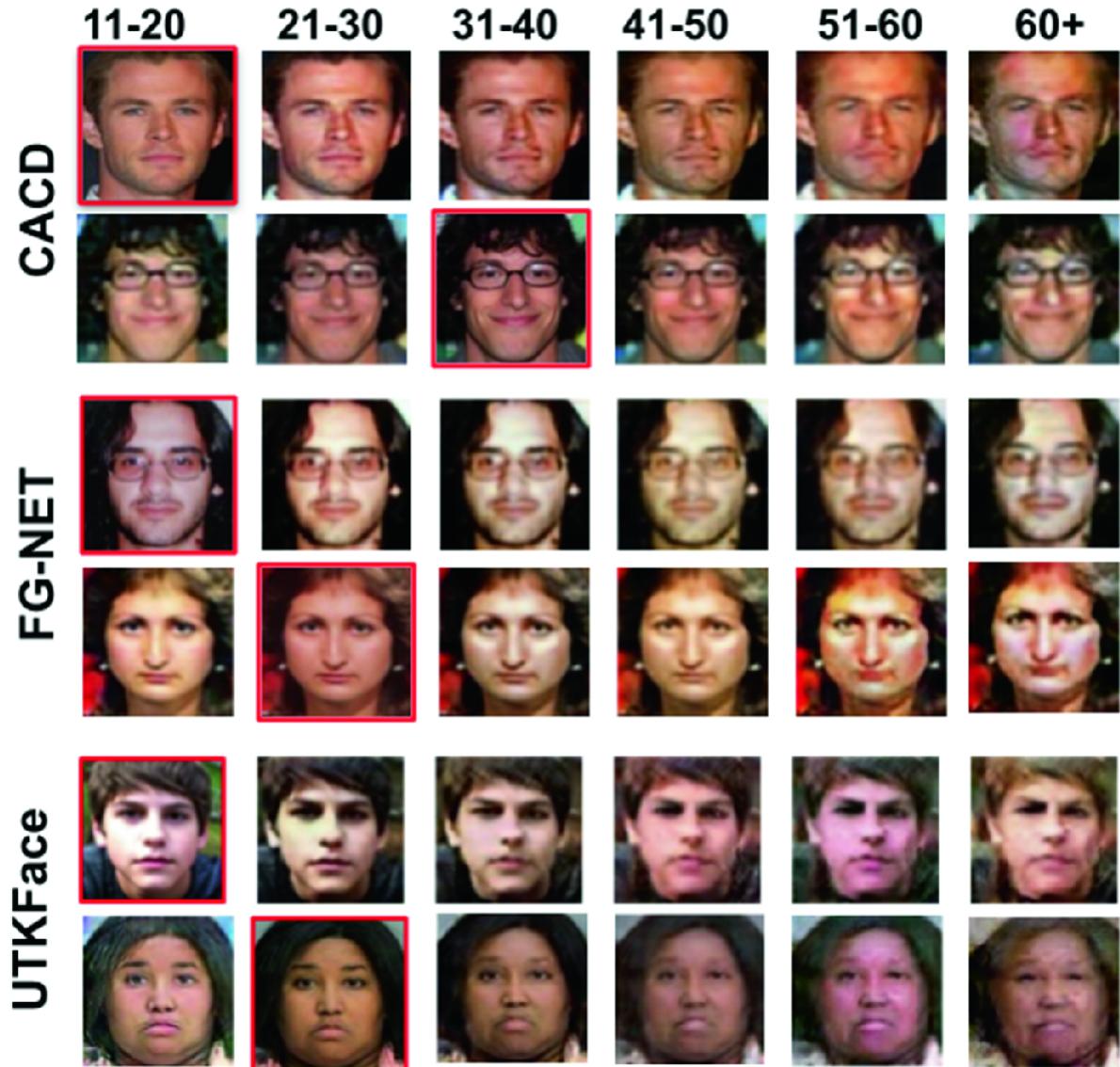


Fig. 7.10 The age groups considered are specified in each column, and the red boxes indicate the input image. The first two rows contain generated images from the CACD test set, rows 3 and 4 from FG-NET, and the last two from UTKFace. The photos on the right of the highlighted image (red) represent the aging process, the pictures on the left represent the rejuvenation process

7.4.4 Method Comparison

To evaluate and quantitatively compare the generated images for each aging or rejuvenating transformation of the three methods presented in the previous sections (7.4.1, 7.4.2, and 7.4.3), age estimation and facial comparison methods were employed.

The age estimator [4] was used to verify whether the transformations across age groups were accurate. The age estimation network was pre-trained on CACD, and the code is available.¹¹ The method's outputs were compared with the expected age for each generated image. In this case, the average age of each group was considered as the predicted age (15, 25, 35, 45, 55, and 65). The method performance for the original images from the test set (original images) in terms of Mean Absolute Error (MAE) and Standard Deviation (SD) is indicated in Table 7.2 for comparison with the generated images.

Table 7.2 The mean absolute error and standard deviation to the age estimation in FG-NET, UTKFace, and CACD datasets

Dataset	Age estimation
CACD	6.6 ± 5.9
FG-NET	16.3 ± 10.4
UTKFACE	9.1 ± 6.6

Furthermore, an essential point in the facial aging process is to preserve the person's identity during transformations. For such an assessment, a facial verification method [26] is employed to compare each generated image with the original image. The authors trained a network such that the squared L2 distances in the embedding space directly correspond to a measure of face similarity. So, faces of the same person have small distances, and faces of distinct people have large distances. Once this embedding has been produced, the face verification task requires a threshold for the distance of the two embeddings. Here, a threshold of 0.7 is considered to be images of the same person.¹²

Finally, to evaluate the quality of generated images, the Frechet Inception Distance (FID) [8] was used. This metric captures how real the generated images look by using features extracted from an Inceptionv3 model [31] and calculating the Frechet distance. The performance of all the measurements for the CAAE, IPCGAN, and RCRIIT methods against each test set is reported in Table 7.3.

Table 7.3 Quantitative comparison of the images generated for the CAAE, IPCGAN, and RCRIIT methods

Model	Dataset	Age estimation	Face verification	Image quality (FID)
CAAE	CACD	14 ± 10.5	83.7%	62.22
	FG-NET	13.4 ± 10	85.7%	119.51
	UTKFace	13.4 ± 9.9	78.8%	81.29
IPCGAN	CACD	9.5 ± 10.7	95.3%	15.42
	FG-NET	9.8 ± 11	97.6%	32.25
	UTKFace	10.8 ± 10.3	93.6%	27.27
RCRIIT	CACD	12 ± 8.8	89.3%	49.73
	FG-NET	12.1 ± 8.8	88.2%	52.35
	UTKFace	9.4 ± 7.1	76.5%	64.52

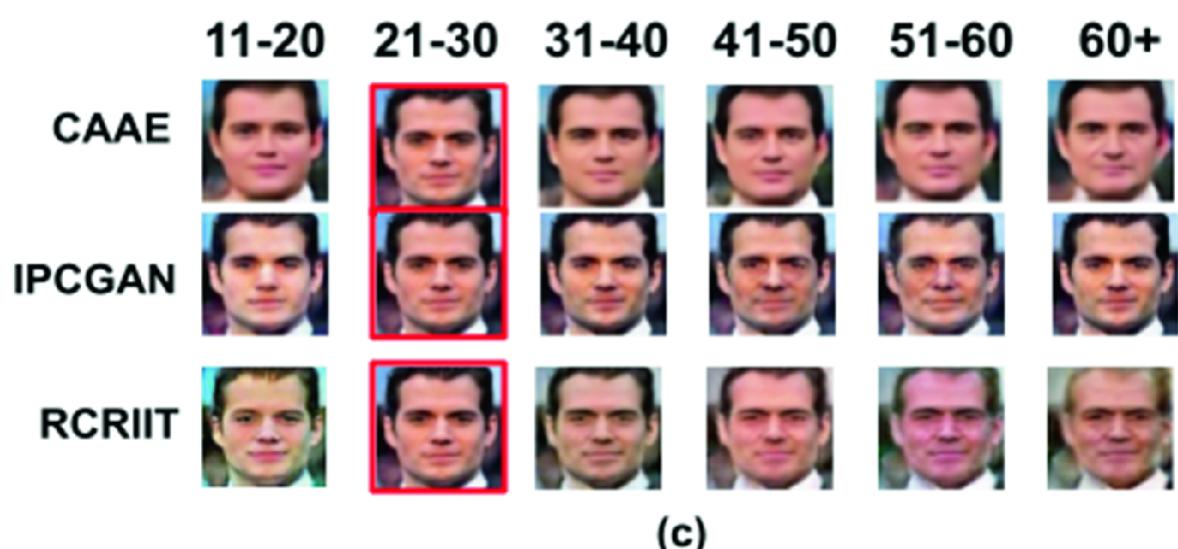
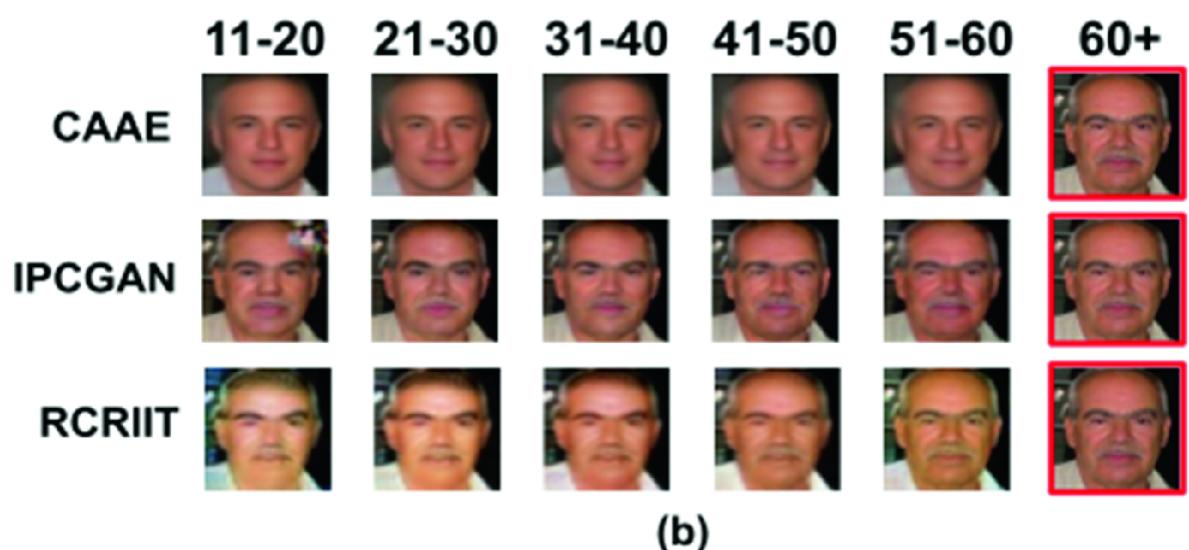
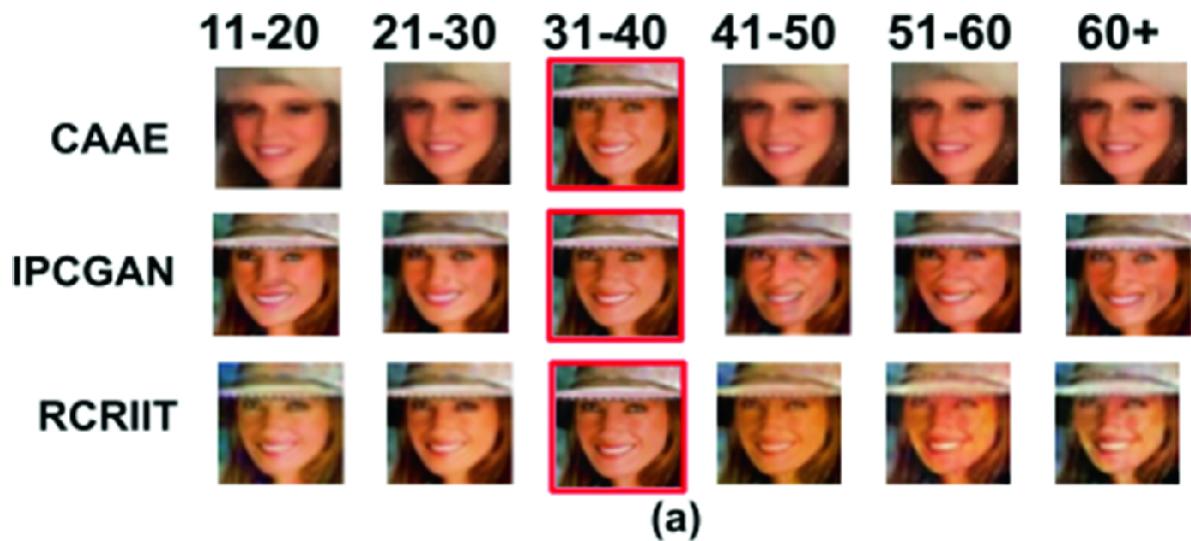


Fig. 7.11 The comparison between the three tested models for the same individual. Figure **a** shows the models trained on the CACD dataset, and in Fig. **b**, the models were trained on the CACD dataset and evaluated on the FG-NET dataset. Finally, in Fig. **c** the models were trained on the UTKFace dataset. The image with the red box is the real face at the specified age

In Fig. 7.11 it is possible to verify the aging and rejuvenating transformations for the three approaches. In these experiments, it is also possible to verify that the IPCGAN was able to produce higher-quality images (Table 7.3 and Fig. 7.11).

7.5 Summary

Facial aging is a stimulating topic that attracts the academic community both for the challenge of aging faces and possible use cases. For this reason, with each new architecture and advance developed in GANs, new works look for ways to apply in facial aging. In this work, three aging methods were compared, CAAE, IPGAN, and RCRIIT. Because CAAE utilizes adversarial autoencoders, and the images get a blurred effect, which is typical of models that use this architecture. The IPCGAN identity preservation module constrains all the generated images to the same person, different from CAAE images, achieving the best visual results, and the best quantitative results (Table 7.3). Finally, the multi-step prediction approach of the RCRIIT adds noise in each prediction, decreasing image quality when the desired age is multiple steps (multiple age groups) from the original image, considering the output of a module is the input of the next module.

References

1. Alqahtani, H., Kavakli-Thorne, M., Kumar, G.: Applications of generative adversarial networks (gans): an updated review. *Arch. Comput. Methods Eng.* (2019). <https://doi.org/10.1007/s11831-019-09388-y>
2. Antipov, G., Baccouche, M., Dugelay, J.L.: Face aging with conditional generative adversarial networks, vol. 2017-September, pp. 2089–2093 (2018). <https://doi.org/10.1109/ICIP.2017.8296650>

3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of Machine Learning Research, vol. 70, pp. 214–223 (2017)
4. Cao, W., Mirjalili, V., Raschka, S.: Rank consistent ordinal regression for neural networks with application to age estimation. Pattern Recognit. Lett. **140**, 325–331 (2020). <https://doi.org/10.1016/j.patrec.2020.11.008> [Crossref]
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets (2014). [arXiv:1406.2661](https://arxiv.org/abs/1406.2661)
6. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. CoRR abs/1704.00028 (2017). 1704.00028
7. Heljakka, A., Solin, A., Kannala, J.: Recursive chaining of reversible image-to-image translators for face aging. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11182 LNCS, pp. 309–320 (2018). https://doi.org/10.1007/978-3-030-01449-0_26
8. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, NIPS’17, pp. 6629–6640 (2017)
9. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst (2007)
10. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-Image Translation with Conditional Adversarial Networks (2018). [arXiv:1611.07004](https://arxiv.org/abs/1611.07004) [cs]
11. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive Growing of GANs for Improved Quality, Stability, and Variation (2018). [arXiv:1710.10196](https://arxiv.org/abs/1710.10196) [cs, stat]
12. Karras, T., Laine, S., Aila, T.: A Style-Based Generator Architecture for Generative Adversarial Networks (2019). [arXiv:1812.04948](https://arxiv.org/abs/1812.04948) [cs, stat]
- 13.

- Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and Improving the Image Quality of StyleGAN (2020). [arXiv:1912.04958](https://arxiv.org/abs/1912.04958) [cs, eess, stat]
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). [arXiv: 1412.6980](https://arxiv.org/abs/1412.6980)
 15. Lanitis, A., Taylor, C.J., Cootes, T.F.: Toward automatic simulation of aging effects on face images. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(4), 442–455 (2002). <https://doi.org/10.1109/34.993553> [Crossref]
 16. Liu, X., Zou, Y., Xie, C., Kuang, H., Ma, X.: Bidirectional face aging synthesis based on improved deep convolutional generative adversarial networks. *Information* (Switzerland) **10**(2) (2019). <https://doi.org/10.3390/info10020069>
 17. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: *Proceedings of International Conference on Computer Vision (ICCV)* (2015)
 18. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders (2015). [arXiv:1511.05644](https://arxiv.org/abs/1511.05644)
 19. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets (2014). [arXiv: 1411.1784](https://arxiv.org/abs/1411.1784) [cs, stat]
 20. Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., Zafeiriou, S.: Agedb: the first manually collected, in-the-wild age database. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1997–2005 (2017). <https://doi.org/10.1109/CVPRW.2017.250>
 21. Or-El, R., Sengupta, S., Fried, O., Shechtman, E., Kemelmacher-Shlizerman, I.: Lifespan age transformation synthesis. *Lecture Notes in Computer Science* (including subseries *Lecture Notes in Artificial Intelligence* and *Lecture Notes in Bioinformatics*) **12351 LNCS**, pp. 739–755 (2020). <https://doi.org/10.1007/978-3-030-58539-6-44>
 22. Palsson, S., Agustsson, E., Timofte, R., Van Gool, L.: Generative adversarial style transfer networks for face aging, vol. 2018-June, pp. 2165–2173 (2018). <https://doi.org/10.1109/CVPRW.2018.00282>
 23. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016)
 - 24.

- Ricanek, K., Tesafaye, T.: MORPH: a longitudinal image database of normal adult age-progression, pp. 341–345 (2006). <https://doi.org/10.1109/FGR.2006.78>
25. Rothe, R., Timofte, R., Gool, L.V.: Deep expectation of real and apparent age from a single image without facial landmarks. *Int. J. Comput. Vis.* **126**(2–4), 144–157 (2018)
[[MathSciNet](#)][[Crossref](#)]
 26. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823 (2015)
 27. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing, pp. 9240–9249 (2020). <https://doi.org/10.1109/CVPR42600.2020.00926>
 28. Shu, X., Tang, J., Lai, H., Liu, L., Yan, S.: Personalized age progression with aging dictionary. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3970–3978 (2015)
 29. Shu, X., Xie, G.S., Li, Z., Tang, J.: Age progression: current technologies and applications. *Neurocomputing* **208**, 249–261 (2016). <https://doi.org/10.1016/j.neucom.2016.01.101>
[[Crossref](#)]
 30. Shu, X., Xie, G.S., Li, Z., Tang, J.: Age progression: current technologies and applications. *Neurocomputing* **208**, 249–261 (2016). <https://doi.org/10.1016/j.neucom.2016.01.101>. sI: BridgingSemantic
[[Crossref](#)]
 31. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015). 1512.00567
 32. Viazovetskyi, Y., Ivashkin, V., Kashin, E.: Stylegan2 distillation for feed-forward image manipulation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12367 LNCS, pp. 170–186 (2020). <https://doi.org/10.1007/978-3-030-58542-6-11>
 33. Wang, Z., Tang, X., Luo, W., Gao, S.: Face aging with identity-preserved conditional generative adversarial networks. pp. 7939–7947 (2018), <https://doi.org/10.1109/CVPR.2018.00828>
 34. Wu, T., Turaga, P., Chellappa, R.: Age estimation and face verification across

- aging using landmarks. *IEEE Trans. Inf. Forensics Secur.* **7**(6), 1780–1788 (2012). <https://doi.org/10.1109/TIFS.2012.2213812>
[Crossref]
35. Yang, C., Lv, Z.: Gender based face aging with cycle-consistent adversarial networks. *Image Vis. Comput.* **100** (2020). <https://doi.org/10.1016/j.imavis.2020.103945>
 36. Yang, H., Huang, D., Wang, Y., Jain, A.: Learning continuous face age progression: a pyramid of gans. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(2), 499–515 (2021). <https://doi.org/10.1109/TPAMI.2019.2930985>
[Crossref]
 37. Yun, Fu., Guo, Guodong, Huang, T.S.: Age synthesis and estimation via faces: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**(11), 1955–1976 (2010). <https://doi.org/10.1109/TPAMI.2010.36>
[Crossref]
 38. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process. Lett.* **23**(10), 1499–1503 (2016). <https://doi.org/10.1109/LSP.2016.2603342>
[Crossref]
 39. Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder, vol. 2017-January, pp. 4352–4360 (2017). <https://doi.org/10.1109/CVPR.2017.463>
 40. Zheng, T., Deng, W., Hu, J.: Cross-age lfw: a database for studying cross-age face recognition in unconstrained environments (2017). [arXiv:1708.08197](https://arxiv.org/abs/1708.08197)
 41. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2223–2232 (2017)

Footnotes

¹ Identification or verification of individuals based on their physical or behavioral characteristics.

² In general, the examples are generated with an input of Gaussian noise in the generating network.

- 3 The critique outputs a non-limited real number.
- 4 The cost function used is just an example.
- 5 The authors used an SVM to perform the classification task and manually annotated 4.000 examples.
- 6 In this context we can think of linear transformations in the data $\log(1 - \mathcal{D}(\mathcal{G}(z)))$.
- 7 Implementation in <https://github.com/ipazc/mtcnn>.
- 8 Available at: <https://github.com/mattans/AgeProgression/tree/v1.0.0>.
- 9 This experiment followed the authors implementation available at: <https://github.com/dawei6875797/Face-Aging-with-Identity-Preserved-Conditional-Generative-Adversarial-Networks>.
- 10 Available online at: <https://github.com/AaltoVision/img-transformer-chain>.
- 11 Available at: https://github.com/Raschka-research-group/coral-cnn/tree/master/single-image-prediction_w-pretrained-models.
- 12 The face-compare library was used for the facial verification tasks: <https://pypi.org/project/face-compare/>.

8. Embedding Time-Series Features into Generative Adversarial Networks for Intrusion Detection in Internet of Things Networks

Ehsan Hallaji¹✉, Roozbeh Razavi-Far²✉ and Mehrdad Saif¹✉

- (1) Department of Electrical and Computer Engineering, University of Windsor, 401 Sunset Avenue, Windsor, N9B 3P4, Canada
(2) Department of Electrical and Computer Engineering and School of Computer Science, University of Windsor, 401 Sunset Avenue, Windsor, N9B 3P4, Canada

✉ Ehsan Hallaji (Corresponding author)

Email: hallaji@uwindsor.ca

✉ Roozbeh Razavi-Far

Email: roozbeh@uwindsor.ca

✉ Mehrdad Saif

Email: msaif@uwindsor.ca

Abstract

In recent years, Generative Adversarial Networks (GAN) have become powerful industrial tools to facilitate various learning tasks, including anomaly detection. This chapter studies a number of GAN architectures used for anomaly detection in the data stream. Moreover, a novel approach is proposed for embedding the dynamic characteristics of the data stream into the GAN-based detector structures. In this process, a GAN model is also proposed for efficient estimation of a confidence measure during the operation that reflects how well samples can be assigned to benign data.

Furthermore, this chapter designs an intrusion detection system by developing a GAN-based anomaly detector. To do this, we study the effect of the proposed approach and the selected GAN-based approaches in detecting malicious intrusions in an Internet of Things (IoT) network. Experiments are evaluated in terms of false alarm and missed alarm detection rates. The obtained results indicate the effectiveness of the proposed GAN-based detection approach for the respective task.

8.1 Introduction

The integration of generative models with deep learning had a new product, called GAN [10], which created a new domain for unsupervised learning. Although GAN was originally proposed for unsupervised learning, numerous research works have been dedicated to creating novel architectures for GAN and extending to semi-supervised and supervised learning [5, 25]. Moreover, GAN is broadly adapted to accomplish various learning tasks such as class imbalance learning [8], classification [26], dimensionality reduction [7, 23], feature selection [19], missing data imputation [12], and anomaly detection [38]. The focus of this chapter is mainly on the usage of GAN for anomaly detection, which is the backbone of most real-time computational models in the industry.

Anomaly detection is a task that can be viewed from different perspectives. While there are various approaches to detect an anomaly in the data, some trends seem to be more popular. For instance, a group of anomaly detection algorithms aims at finding outliers based on normal data they have been trained on [28, 36]. Detectors that use a one-class classification process mainly follow this approach [29]. On the other hand, a group of algorithms, called change detectors, are designed to detect concepts drifts in non-stationary environments [2, 22, 33]. Anomalies handled by this group are mostly in the form of a new class, recurring class, or any change in the distribution of current classes (e.g., a unimodal distribution becomes bimodal in time). The third group uses the reconstruction cost in the network embedding as a change criteria [4, 40]. Based on these methods, when a model is trained on data, the coded data can be encoded with minimal reconstruction error. However, anomalous data does not match the obtained embedding so that the reconstructed error

is expected to be large. Most of the GAN-based detectors are designed w.r.t. the first and the third groups.

Anomaly detection and intrusion detection are often connected problems, and, thus, most of the Intrusion Detection Systems (IDS) are designed using anomaly detection algorithms [11, 18]. IDS is an important component of cyber-physical systems such as IoT and Supervisory Control and Data Acquisition (SCADA) systems that raises an alarm whenever the system experiences malicious intrusions [13, 16]. Regardless of the choice of anomaly detection technique, the detection model is trained on the normal (i.e., without intrusion) data. Depending on the type of anomaly detection algorithm, it may only model the normal state and consider outliers as malicious samples, or model both normal and attack samples during the training to resemble a binary scenario. The main difference between these two approaches is that the former uses an unsupervised setting, while the latter is a supervised task. GAN-based anomaly detection can be accomplished using both settings, where the supervised models usually use conditional GAN, which embeds the label information into the generator structure.

This chapter first reviews state-of-the-art GAN-based anomaly detectors available in the literature. Then, We investigate the effect of embedding the extracted time-series features that reflect the dynamic changes of data on the detection performance of GAN-based anomaly detectors. For this step, a novel GAN model is also proposed to extract features based on the existing cluster information in the data. We further evaluate and compare the selected algorithms on a real-life case study with and without the proposed feature embedding. To do this, an IoT network under cyber-attacks is considered to evaluate the efficiency of the GAN-based intrusion detection.

The rest of the chapter is organized as follows: Sect. 8.2 presents the background on GAN-based anomaly detection. Section 8.3 states the intrusion detection problem. Section 8.4 introduces the proposed feature embedding approach. Section 8.5 reports and analyzes the experimental results. Finally, the chapter is concluded in Sect. 8.6.

8.2 Related Works

Anomalies are commonly created in different industrial systems such as control systems [31, 34] and IDS [13]. Various efforts have been dedicated to design GAN-based anomaly detection algorithms [20]. While there are different approaches to complete this task, the general idea is to use the normal data as a reference and detect the anomalous data based on the difference between the predicted and the actual output. For instance, the generator in [1] trains a detection model only on normal images, which are compressed and decompressed afterwards to generate fake images via a encoder-decoder-encoder structure. The additional encoder is used to map the generated data to their latent representations. [21] proposed a GAN-based anomaly detection algorithm by using long short-term memory (LSTM) networks for capturing the distribution of the multivariate time series features. [6] extends GAN by adding an encoder as the third component. This enables learning a mapping between the original feature space to the latent space. In contrast to most GAN structures, this changes the objective of discriminator to classify between real, synthetic, and real encoding samples. The utilized GAN-based change detectors are explained under separate subsections as follows.

8.2.1 Adversarially Learned Anomaly Detection

Adversarially Learned Anomaly Detection (ALAD) [39] is designed based on the architecture of bi-directional GANs with a focus on the efficiency of the algorithm. ALAD trains an autoencoder and estimates reconstruction errors based on the adversarially learned features to detect an anomalous sample in the data. The design of ALAD brings about improvements in the speed and efficiency of the inference in real-time applications.

8.2.2 GAN Ensemble for Anomaly Detection

Ensemble of GANs for Anomaly Detection (EGAN) [15] employs a set of generators and a set of discriminators that are randomly paired and trained together. In this process, every generator receives a feedback from different discriminators, and the discriminator observes samples generated by multiple generators. Compared to the case, where a single GAN is used, the GANs ensemble is expected to model the distribution of normal data more accurately, which in turn enhances the detection performance.

8.2.3 Anomaly Detection with GAN

Unsupervised Anomaly Detection with GAN (AnoGAN) [37] employs a deep convolutional GAN structure in order to learn a manifold of normal anatomical variability. In addition, AnoGAN has a customized anomaly scoring scheme that is designed based on the mapping from the original space to a latent space. AnoGAN, which was originally proposed for image processing applications, scores image patches based on how they fit into the captured distribution.

8.2.4 Unsupervised Change Detection with GAN

Time Series Anomaly Detection with GAN (TAnoGAN) [3] is an unsupervised detector designed for time-series data. The architecture of TAnoGan is designed to detect anomalies in datasets with a small number of data points. It first maps the real time-series data space to a latent space, and, then, reconstructs the data from the latent space. The anomaly score is estimated as the loss between the real data and reconstructed data.

Table 8.1 Description of classes and subclasses in the employed IoT intrusion detection dataset

Classes	Subclasses	Description
Normal	Normal	There is no attack on the network
Attack	Denial of service	Denying authorized access requests by disrupting the computing resources
Attack	Exploits	taking advantage bugs and glitches to create an unintentional or unsuspected behavior through a sequence of commands
Attack	Fuzzers for suspicious activity	Using fuzzing strike lists to unveil the security vulnerabilities that can cause a system crash
Attack	Generic	Causing a collision by disregarding the configuration of the block cipher
Attack	Reconnaissance	Gathering information about the targeted network prior to a main cyber-attack to prevent security protocols
Attack	Analysis	Penetrating Internet applications through ports, e-mails and scripts
Attack	Backdoor	Securing unauthorized remote access by bypassing the authentication
Attack	Worms	Using a malware to spread it on the target system and replicate it through the affected system's network

8.2.5 Unsupervised Change Detection with GAN

Fast unsupervised anomaly detection with GAN (f-AnoGAN) [38] is a GAN-based unsupervised learning approach that can identify anomalous data. It is originally proposed for detecting changes in images and image segments. This method devises a fast mapping technique to map new data to the GAN’s latent space. The mapping is performed using an encoder that helps in detecting anomalies by estimating a combined anomaly score for samples. This score is calculated using the reconstruction error and discriminator feature residual error w.r.t. the building blocks of the trained model.

8.3 Internet of Things Network

The utilized GAN algorithms are leveraged to facilitate the intrusion detection process. To do this, we consider two classes of normal and cyber-attack in steams of data.

8.3.1 Dataset

Aside from challenges of data processing in industrial systems [14, 17, 30, 32], detecting the class of attack is also complicated by the variety of cyber-attacks that exist in this class. In other words, the class of attacks consists of different types of attacks that can be considered as subclasses of the main class. This results in a very diverse distribution that complicates the detection process. Table 8.1 lists these sub-classes and provides a brief explanation for each of them. Furthermore, each of the sub-classes listed in Table 8.1 has different number of samples, which creates a skewed class data distribution [9]. In this study, the normal and attack classes are the majority and minority classes, respectively.

8.3.2 IoT Testbed

Figure 8.1 shows the utilized testbed to generate the network traffic flow, which is used to create the data stream [24]. The majority of components in this figure are used for configuration of the scenario. It is observable in Fig. 8.1 that how normal and attack samples are created via recording the transmitted/received traffic through the two ingress routers. IoT sensors are used to generate the normal (benign) traffic. The feature generation tools

are devised to extract the network traffic from the IoT network, sensors, and datasets.

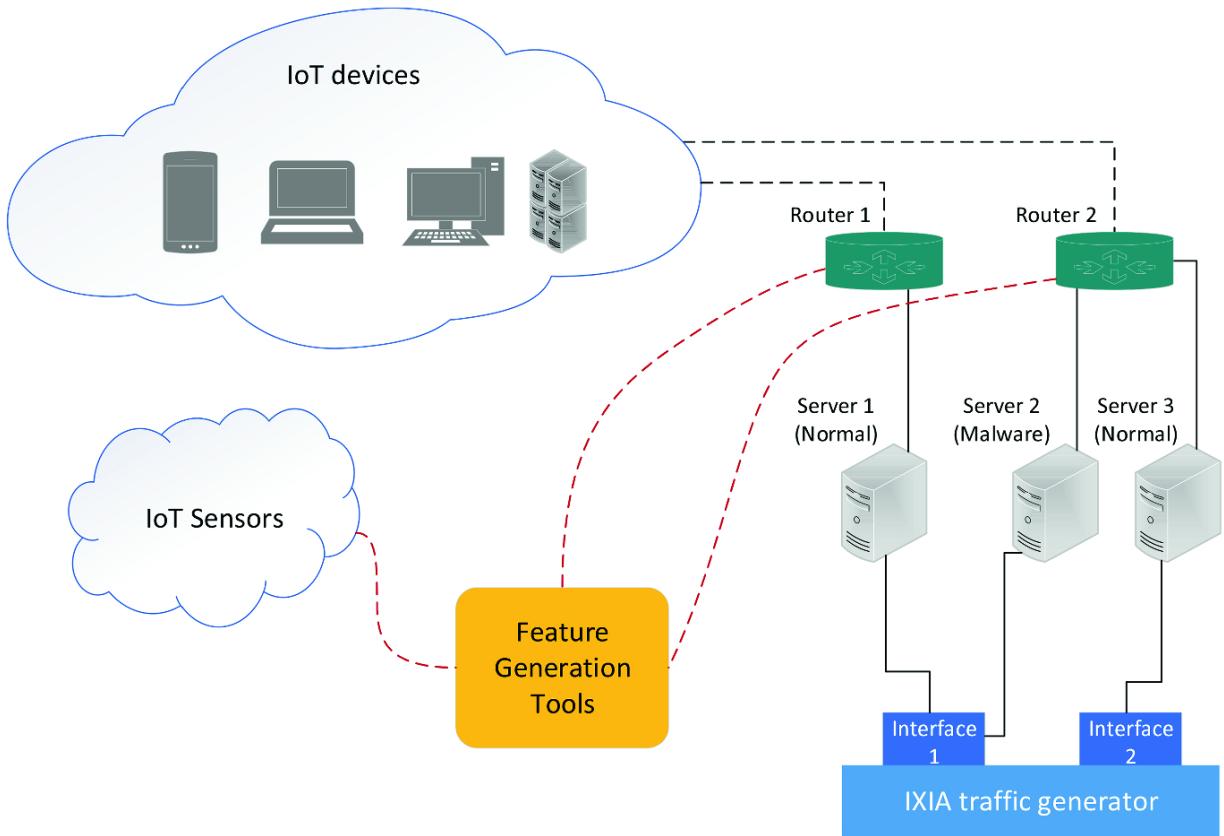


Fig. 8.1 Illustrative diagram of the utilized IoT testbed, which generates the network traffic samples for intrusion detection

8.4 Feature Embedding

It is well known that deep learning algorithms can robustly extract features by expanding the dimensions through a multi-layer structure. Nevertheless, since the objective of the neural network in an intrusion detection task is to model benign and attack classes, it may not necessarily extract features that exhibit the dynamical changes in the data stream. To facilitate modeling such dynamical changes in the network, we propose the embedding of features that reveal dynamical changes in the time-series data into the neural network structure. To do this, we first devise the Cumulative Sum (CUSUM) function to find sequential changes in each feature and propose a GAN structure to estimate the confidence of not being anomalous for each

sample based on the Silhouette coefficient, as shown in Fig. 8.2. These two concepts are mainly used for the change detection and cluster analysis, respectively.

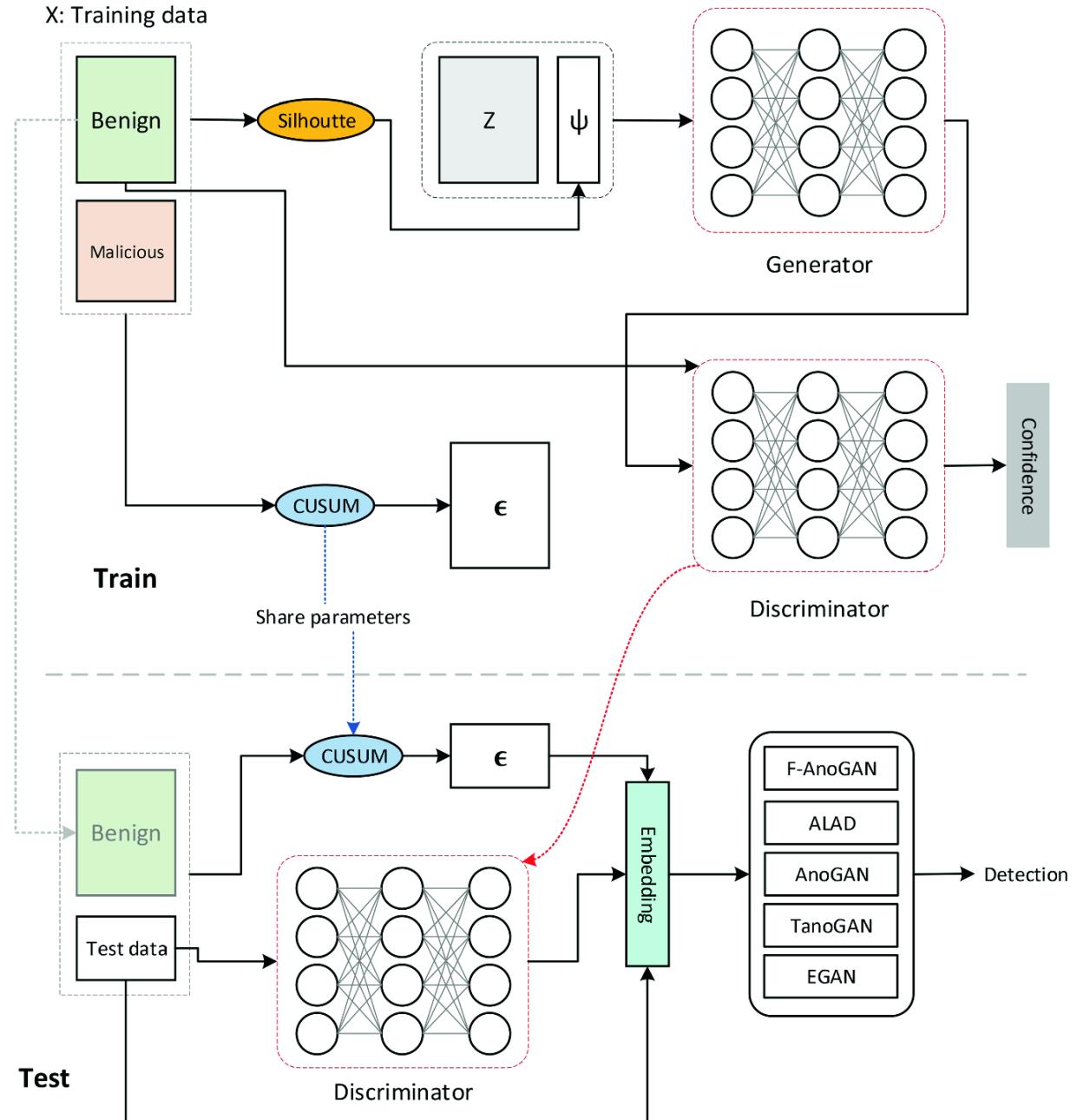


Fig. 8.2 Illustrative diagram of the proposed feature embedding approach for the generator. The top portion of the figure shows the training process, whereas the bottom part illustrates the test phase

8.4.1 Extracting Sequential Changes

CUSUM function [27] can model changes in one sequence of data S .

Consider a dataset $X \in \mathbb{R}^{m \times n}$, where m and n are the number of samples and features, respectively. Therefore, since there are n features in the data, n different sequences of data $S^{(j)}$ should be separately processed by CUSUM, where $1 \leq j \leq n$. Also, each sample π_i is recorded at a time-step \mathbf{l}_1 , $X \in \mathbb{R}^{m \times n}$.

Assuming that the benign data in $S^{(j)}$ can be modeled using a probability density function P_{Θ^0} with a parametrization set \mathcal{G}_s , we define a base hypothesis $\mathcal{H}_0 : \Theta_0$ that corresponds to the normal data. Leveraging the central limit theorem, $\|\nabla f(x) \leq 1\|_2$ is formed by calculating the mean β and standard deviation $\hat{\sigma}^2$ on a transformed sequence $S_T^{(j)}$, as follows:

$$S_T^{(j)} = \bigcup_{i=1}^{\frac{m}{z}} s(i) : s(i) = \frac{1}{z} \sum_{h=1+z(i-1)}^{zi} x_{hj}, \quad (8.1)$$

where z is a window length that satisfies the theorem. Then, the aim is to find an alternative hypothesis $\mathcal{H}_1 : \Theta^1 = \{\hat{\mu}_1, \hat{\sigma}_1^2\}$, which is defined as being outside the benign distribution P_{Θ^0} . Therefore, it can be interpreted that D_1 is the hypothesis set of the anomalous samples. In order to facilitate the estimation of \mathcal{G}_s , we define an interval of $(1 - \omega)100\%$ on \mathcal{G}_s , where ω is a user-defined parameter. Then, the defined intervals $\hat{\mu}_1 = [\hat{\mu}_{1min}, \hat{\mu}_{1max}]$ and $\hat{\sigma}_1^2 = [\hat{\sigma}_{1min}^2, \hat{\sigma}_{1max}^2]$ in \mathcal{G}_s are calculated as follows:

$$\hat{\mu}_1 = \hat{\mu}_0 \pm \lambda \sqrt{\frac{n}{m}} \varphi_{\omega/2} \hat{\sigma}_0 \quad (8.2)$$

$$\begin{cases} \hat{\sigma}_{1max}^2 = \hat{\sigma}_0^2 + \lambda(\hat{\sigma}_{max}^2 - \hat{\sigma}_0^2) \\ \hat{\sigma}_{1min}^2 = \hat{\sigma}_0^2 + \lambda(\hat{\sigma}_0^2 - \hat{\sigma}_{min}^2) \end{cases} \quad (8.3)$$

where λ and q_0 are the sensitivity parameter (i.e., decreasing results in detecting smaller changes, and vice versa) and the normal distribution, respectively. The ratio of change R can then be calculated as in the following:

$$R_i = \sum_{h=1}^i \ln \frac{P_{\Theta^0}(x_h)}{P_{\Theta^1}(x_h)}, \quad i = 1, \dots, \frac{m}{z}, \quad (8.4)$$

and a degree of change ϵ can be obtained as $\epsilon_i = R_i - \min_{1 \leq h \leq i} (R_h)$.

In order to extract time-series features using CUSUM, we form $S^{(j)}$ on benign samples in X and adjust the parameters w.r.t. the anomalous samples during the training. During the test phase, we estimate μ on $S^{(j)} \cup x_i^{(j)}$, where π_i is added at l_1 , and $S^{(j)}$ will not change in samples or size over time (i.e., it is fixed after the training phase).

Dynamical changes in the data can be also monitored with the cluster analysis. Unlike the CUSUM approach, here we consider all the features and extract one feature based on the entire feature space.

8.4.2 Generative Adversarial Cluster Analysis

Considering the one-class model for the change detection (i.e., explained in Sect. 8.1), we might be able to reveal changes by constructing clusters on the benign data and calculating a confidence measure that reflects the dissimilarity to the benign data. It is expected that this confidence measure changes noticeably as the anomalous samples appear in the feature space. This is because such samples are more likely to appear in different regions or clusters, and, therefore, form a new cluster. If one clusters the data each time a new sample arrives, a Silhouette function [35] ψ can be used to determine how well π_i is assigned to its corresponding cluster:

$$\psi(x_i) = \begin{cases} 1 - \varrho_w(x_i)/\varrho_b(x_i), & \varrho_w(x_i) < \varrho_b(x_i) \\ 0, & \varrho_w(x_i) = \varrho_b(x_i) \\ \varrho_b(x_i)/\varrho_w(x_i) - 1, & \varrho_w(x_i) > \varrho_b(x_i) \end{cases}, \quad (8.5)$$

where ϱ_w and p_z return the within-class and between-class dissimilarities, respectively. We use ψ in the training phase to find optimal clusters and use Equation (8.5) to estimate the Silhouette scores w.r.t. p_z and ϱ_w , as follows:

$$\forall x_i \notin c_k : \varrho_b(x_i) = \min \frac{1}{|c_k|} \sum_{r=1}^{|c_k|} \| x_i - x_r \|_2^2, \quad (8.6)$$

$$\forall x_i \in c_k : \varrho_w(x_i) = \frac{1}{|c_k| - 1} \sum_{r=1, i \neq r}^{|c_k|} \|x_i - x_r\|_2^2, \quad (8.7)$$

where $\|\cdot\|$ returns the cardinality, π_i is the k -th cluster in the clustered data, and 32×32 . Although the estimation of Silhouette scores may provide us with enough clustering information, it has a $O(m^2)$ complexity, which severely decreases the efficiency during the test phase. As a result, once the Silhouette scores are estimated on the training data, we train a GAN model to use the Silhouette scores and determine the confidence of samples to belong to the benign clusters, as explained in the following.

Given a generator \mathcal{G} and a discriminator \mathcal{D} , we train a GAN in order to estimate how well each sample is assigned to the normal clusters in based on the aforementioned Silhouette scores. In this process, \mathcal{G} learns to produce samples in accordance to distribution of the Silhouette scores captured from the benign data, whereas \mathcal{D} determines the probability of the produced samples being real or fake. From an IDS terminology, the fake samples are denoted as those who are not confidently assigned to the benign data clusters, and vice versa. Since \mathcal{D} estimates a probability rather than a hard label, it can be considered as the confidence of not being an outlier.

In order to help \mathcal{G} to generate samples with similar between-class and within-class dissimilarities, we feed \mathcal{G} with the estimated Silhouette scores from benign data on top of the random noise: $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $Z \in \mathbb{R}^{m \times n}$ is a random noise matrix and $\Psi = \{\psi_1, \psi_2, \dots, \psi_m\}$ is the set of estimates Silhouette scores.

On the other hand, \mathcal{D} is formulated as:

$$\mathcal{D}(X \cup \mathcal{G}(Z, \Psi)) = P, \quad (8.8)$$

where P is the set of probabilities of each sample being real. The network is then trained based on the following optimization problem:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}[\log \mathcal{D}(X)] + \mathbb{E}[1 - \mathcal{D}(\mathcal{G}(Z, \Psi))], \quad (8.9)$$

where $q(\mathbf{z})$ is a value function. Once the training is over, $y \sim P_{\text{data}}(Y)$ is used to estimate the confidence of test samples based on the learned distribution from the benign data. The advantage of this approach is that the

complexity of the confidence estimation during the operation (or test phase) is decreased from $O(m^2)$ to $O(m)$.

Hence, the final feature extraction results in the addition of $n + 1$ features (i.e., n features from R and one features from p_z).

8.4.3 Training Computational Complexity

For the sake of simplicity, we assume that all hidden layers have n neurons. To initialize the generator structure, the first step is to initialize the weights of the network, which given a fully connected network with L layers and n neurons leads to a $p_g(\mathbf{x}|y)$ complexity. The Silhouette coefficients are estimated by a $O(m^2)$ complexity. The main part of the algorithm involves multiplying a sequence of weight vectors or matrices, which has a cubic complexity of $p_g(\mathbf{x}|y)$ for L layers with n neurons in the network. Adding the linear complexity of activation functions to this step changes the complexity to $O(Ln^3) + O(Ln)$. The computation of the correlation cost is done with two loops, which yields a $O(m^2)$ complexity. Going to the discriminator, the probabilities are estimated through matrix multiplication as mentioned before. Therefore, with V layers in this network, the complexity can be calculated as $O(Vn^3)$. Adding the effect of activation to this formula results in $O(Vn^3) + O(Vn)$. Finally, the weights are updated via gradient estimation that has different complexity depending on the location of the hidden layer. It is known that gradient estimation for n neurons has a $O(n^2)$ complexity at the final layer. This is while this complexity increases to $O(n^2)$ for the rest of the hidden layers. Given that there are $L - 1$ layers before the final layer, the complexity order of this step would be $O(n^2) + O((L - 1)n^3)$. When $m \gg n$ or $n > m$, the complexity will be further simplified to $O(m^2)$ and $O(n^2)$, respectively.

8.5 Experimental Results

This section initially explains the experimental setting used for the conducted experiments. Then, the attained results of the change detection are presented and analyzed in an intrusion detection problem conducted in the IoT network.

8.5.1 Experimental Setting

Experiments are repeated 10 times and the network parameters are optimized using the Adam optimizer. All networks are trained for 2000 epochs with a batch size equal to 100, and the initial learning rate is set to 0.01. The original architecture is followed for each technique and the activation functions are selected as suggested. However, the number of hidden layers are empirically set to three in the simulated experiments.

8.5.2 Results Analysis

In order to assess the effectiveness of the proposed feature embedding approach, we first compare the detection rate of the GAN-based detectors. Figure 8.3 shows the averaged detection rates obtained by each of the GAN detectors with and without the feature embedding. Based on this figure, it can be seen that all methods experience a boost in performance when the extracted features are embedded into the generator network. Moreover, the ranking of detectors remains unchanged before and after the feature embedding, which indicates an overall improvement among all detectors. EGAN, ALAD, f-AnoGAN, TanoGAN, and AnoGAN are ranked from first to fifth in terms of the detection rate.

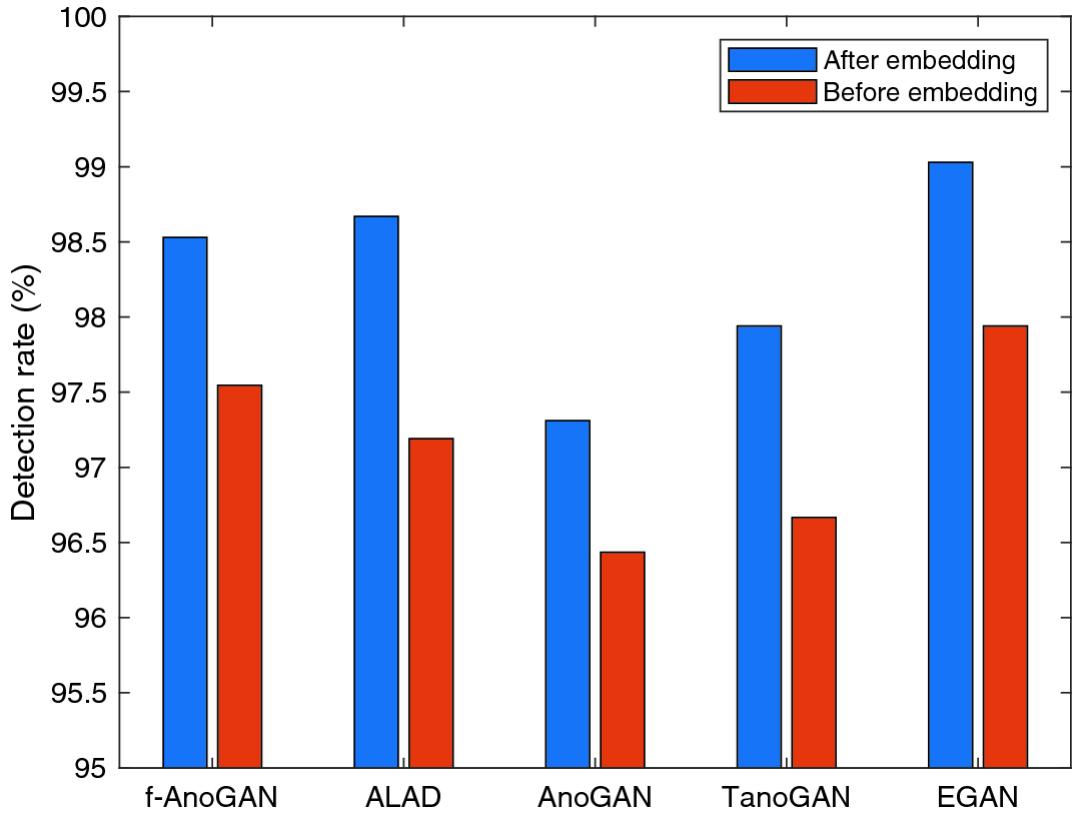


Fig. 8.3 Comparison of the GAN-based detection results before and after the proposed feature embedding approach. The results are averaged over ten runs

Figure 8.4 illustrates the detection results without the feature embedding w.r.t. each type of cyber-attack that is listed in Table 8.1. This figure shows that Exploits and Backdoor attacks are the most challenging type for the detection problem, as they resulted in the lowest detection rates. Furthermore, the Worm attack is detected with the highest ratio. Analysis and Fuzzer attacks have been also easily handled, albeit with a small difference with the Worm attack.

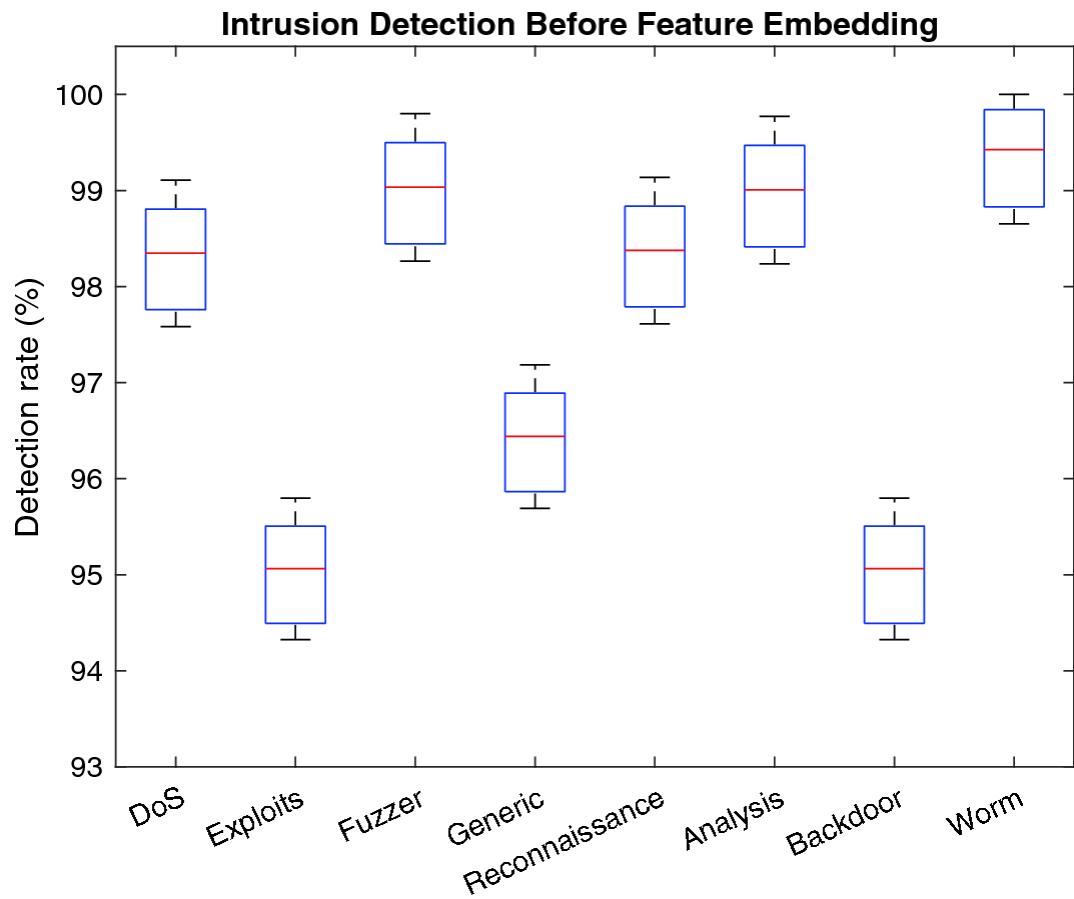


Fig. 8.4 Detection rate of different cyber-attacks without using the proposed feature embedding over ten runs

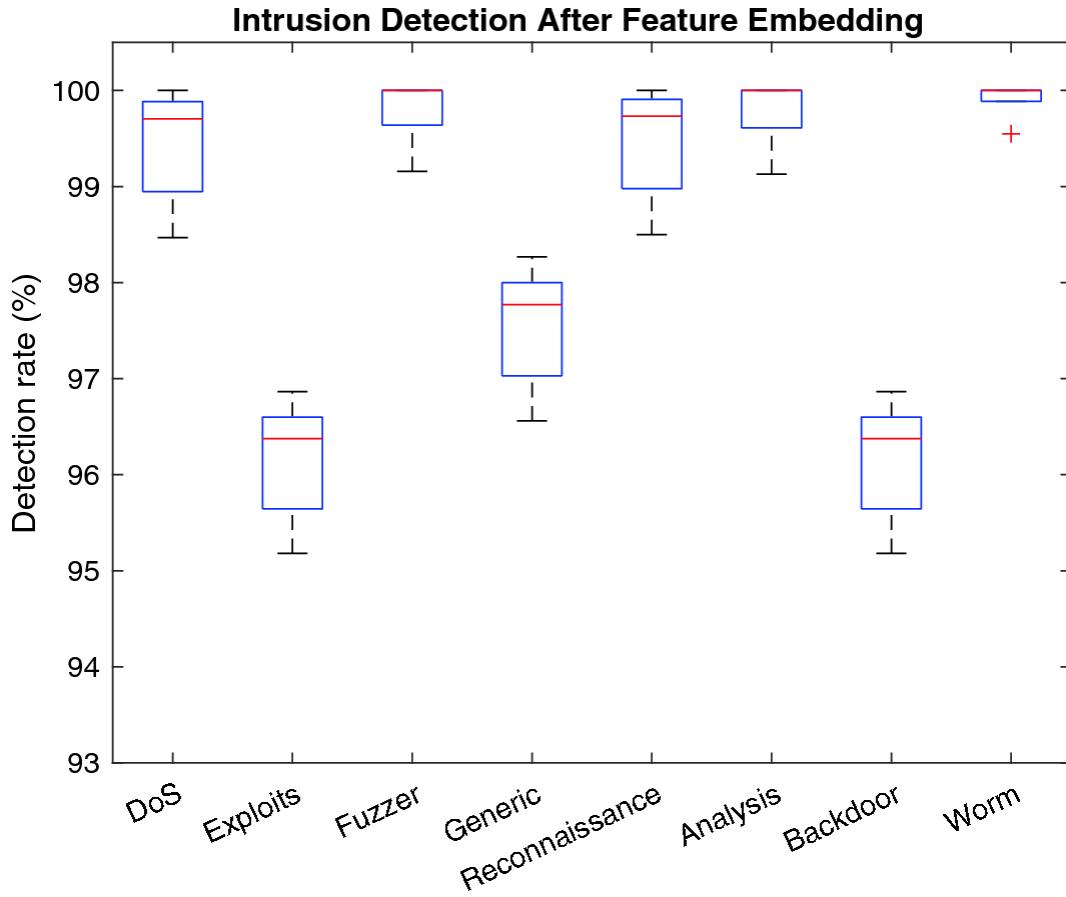


Fig. 8.5 Detection rate of different cyber-attacks using the proposed feature embedding over ten runs

The effect of feature embedding on the detection of different cyber-attacks can be observed by comparing Fig. 8.5 with Fig. 8.4. Figure 8.5 illustrates the detection results when the proposed feature embedding is employed. Comparing these two figures, one can infer that for all attack types the detection rate has been enhanced when the proposed feature embedding is used. This improvement is more obvious for the attack types that were easier to detect in Fig. 8.2. Exploit, Backdoor, and Generic attacks are still more challenging than other types even after the feature embedding is used.

8.6 Conclusion

This chapter compared and studied several types of GAN-based anomaly detection or change detection algorithms with the application in intrusion

detection. Moreover, we proposed that embedding the extracted features based on CUSUM and cluster analysis into the GAN detectors can boost the detection performance. In this process, a GAN model is also proposed to estimate a confidence measure for test samples based on how well they can be assigned to benign clusters w.r.t. Silhouette scores. For the sake of the evaluation, an intrusion detection scenario in the IoT network is considered. The experimental results indicate the effectiveness of the proposed approach in improving the GAN-based detection performance in terms of the detection rate.

Acknowledgements

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant RGPIN-2021-02968.

References

1. Akcay, S., Atapour-Abarghouei, A., Breckon, T.P.: Ganomaly: Semi-supervised anomaly detection via adversarial training. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision - ACCV 2018, pp. 622–637. Springer International Publishing, Cham (2019)
[[Crossref](#)]
2. Alippi, C., Roveri, M.: Just-in-time adaptive classifiers-part i: Detecting nonstationary changes. *IEEE Trans. Neural Networks* **19**(7), 1145–1153 (2008)
[[Crossref](#)]
3. Bashar, M.A., Nayak, R.: Tanogan: Time series anomaly detection with generative adversarial networks. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1778–1785 (2020)
4. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Anomaly detection using autoencoders in high performance computing systems. *Proceedings of the AAAI Conference on Artificial Intelligence* **33**(01), 9428–9433 (2019)
[[Crossref](#)]
5. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.: Good semi-supervised learning that requires a bad gan (2017). [ArXiv:1705.09783](#)
- 6.

Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning p. arXiv preprint (2016). [ArXiv:1605.09782](#)

7. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M.: Generative adversarial dimensionality reduction for diagnosing faults and attacks in cyber-physical systems. *Neurocomputing* **440**, 101–110 (2021) [\[Crossref\]](#)
8. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M., Parvania, M.: Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. *IEEE Transactions on Smart Grid* pp. 1–1 (2021)
9. Farajzadeh-Zanjani, M., Razavi-Far, R., Saif, M.: Efficient sampling techniques for ensemble learning and diagnosing bearing defects under class imbalanced condition. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7 (2016)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc. (2014)
11. Hajisalem, V., Babaie, S.: A hybrid intrusion detection system based on abc-afs algorithm for misuse and anomaly detection. *Comput. Netw.* **136**, 37–50 (2018) [\[Crossref\]](#)
12. Hallaji, E., Razavi-Far, R., Palade, V., Saif, M.: Adversarial learning on incomplete and imbalanced medical data for robust survival prediction of liver transplant patients. *IEEE Access* **9**, 73641–73650 (2021) [\[Crossref\]](#)
13. Hallaji, E., Razavi-Far, R., Saif, M.: Detection of malicious scada communications via multi-subspace feature selection. In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2020)
14. Hallaji, E., Razavi-Far, R., Saif, M.: DLIN: Deep ladder imputation network. *IEEE Transactions on Cybernetics* pp. 1–13 (2021)
15. Han, X., Chen, X., Liu, L.P.: Gan ensemble for anomaly detection. arXiv preprint (2020). [ArXiv:2012.07988](#)
16. Hassani, H., Hallaji, E., Razavi-Far, R., Saif, M.: Unsupervised concrete feature selection based on mutual information for diagnosing faults and cyber-attacks in power systems. *Eng. Appl. Artif. Intell.* **100**, 104150 (2021)

[Crossref]

17. Hassani, H., Razavi-Far, R., Saif, M., Palade, V.: Generative adversarial network-based scheme for diagnosing faults in cyber-physical power systems. *Sensors* **21**(15) (2021)
18. Jabez, J., Muthukumar, B.: Intrusion detection system (ids): Anomaly detection using outlier detection approach. *Procedia Computer Science* **48**, 338–346 (2015). International Conference on Computer, Communication and Convergence (ICCC 2015)
19. Jordon, J., Yoon, J., van der Schaar, M.: Knockoffgan: Generating knockoffs for feature selection using generative adversarial networks. In: International Conference on Learning Representations (2018)
20. Lee, C.K., Cheon, Y.J., Hwang, W.Y.: Studies on the gan-based anomaly detection methods for the time series data. *IEEE Access* **9**, 73201–73215 (2021)
[Crossref]
21. Li, D., Chen, D., Goh, J., Ng, S.K.: Anomaly detection with generative adversarial networks for multivariate time series. arXiv preprint (2019). [ArXiv:1809.04758](#)
22. Concept drift detection based on fisher's exact test: de Lima Cabral, D.R., de Barros, R.S.M. *Inf. Sci.* **442–443**, 220–234 (2018)
23. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders (2016)
24. Moustafa, N., Turnbull, B., Choo, K.K.R.: An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet Things J.* **6**(3), 4815–4830 (2019)
[Crossref]
25. Odena, A.: Semi-supervised learning with generative adversarial networks (2016). [ArXiv:1606.01583](#)
26. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans (2017)
27. Page, E.S.: Continuous inspection schemes. *Biometrika* **41**(1/2), 100–115 (1954)
[MathSciNet][Crossref]
28. Pauwels, E.J., Ambekar, O.: One class classification for anomaly detection:

Support vector data description revisited. In: P. Perner (ed.) Advances in Data Mining. Applications and Theoretical Aspects, pp. 25–39. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

29. Perera, P., Nallapati, R., Xiang, B.: OCGAN: one-class novelty detection using gans with constrained latent representations p. arXiv preprint (2019). <http://arxiv.org/abs/1903.08550>. ArXiv:1903.08550
30. Razavi-Far, R., Cheng, B., Saif, M., Ahmadi, M.: Similarity-learning information-fusion schemes for missing data imputation. *Knowl.-Based Syst.* **187**, 104805 (2020)
[Crossref]
31. Razavi-Far, R., Davilu, H., Palade, V., Lucas, C.: Model-based fault detection and isolation of a steam generator using neuro-fuzzy networks. *Neurocomputing* **72**(13), 2939–2951 (2009). Hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007)
32. Razavi-Far, R., Hallaji, E., Farajzadeh-Zanjani, M., Saif, M.: A semi-supervised diagnostic framework based on the surface estimation of faulty distributions. *IEEE Trans. Industr. Inf.* **15**(3), 1277–1286 (2019)
[Crossref]
33. Razavi-Far, R., Hallaji, E., Saif, M., Ditzler, G.: A novelty detector and extreme verification latency model for nonstationary environments. *IEEE Trans. Industr. Electron.* **66**(1), 561–570 (2019)
[Crossref]
34. Razavi-Far, R., Kinnaert, M.: Incremental design of a decision system for residual evaluation: a wind turbine application*. *IFAC Proceedings Volumes* **45**(20), 343–348 (2012). 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes
35. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987)
[Crossref]
36. Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: J. Dy, A. Krause (eds.) *Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 80, pp. 4393–4402. PMLR (2018)

37. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Niethammer, M., Styner, M., Aylward, S., Zhu, H., Oguz, I., Yap, P.T., Shen, D. (eds.) *Information Processing in Medical Imaging*, pp. 146–157. Springer International Publishing, Cham (2017) [\[Crossref\]](#)
38. Schlegl, T., Seeböck, P., Waldstein, S.M., Langs, G., Schmidt-Erfurth, U.: f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Med. Image Anal.* **54**, 30–44 (2019) [\[Crossref\]](#)
39. Zenati, H., Romain, M., Foo, C.S., Lecouat, B., Chandrasekhar, V.R.: Adversarially learned anomaly detection (2018)
40. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’17*, p. 665–674. Association for Computing Machinery, New York, NY, USA (2017)

9. Inspection of Lead Frame Defects Using Deep CNN and Cycle-Consistent GAN-Based Defect Augmentation

Chia-Feng Juang¹✉ and Wei-Shane Chen¹

(1) Department of Electrical Engineering, National Chung Hsing University, Taiwan, Republic of China

✉ Chia-Feng Juang
Email: cfjuang@dragon.nchu.edu.tw

Abstract

Inspection of the defects on lead frames is an important task in automated optical inspection (AOI) for the automation of semiconductor packing industry. A lead frame is a thin layer of metal inside a chip package connecting a die to the circuitry on circuit boards. This chapter introduces the application of the faster region-based convolutional neural network (R-CNN) to detect and classify the defects on lead frames using AlexNet as a backbone. At the early stage of lead frame production, the available number of defects is typically limited, which degrades the inspection performance of the inspection network. To address this problem, this chapter proposes a defect augmentation technique by applying the Cycle-consistent Generative Adversarial Network (CycleGAN) to automatically generate different types of defects. The CycleGAN is characterized with unpaired input–output training images, which makes it possible to translate normal patches on lead frame images to defect patches. The augmented defect patches are then blended into the lead frame images by using a linear blending method to obtain augmented lead frame images in training the faster R-CNN.

Experimental results show that the defect augmentation technique is effective in improving the defect inspection performance.

Keywords Automated optical inspection – Lead frames – Printed circuit board – Defect detection – Convolutional neural network – Generative adversarial network

9.1 Introduction

Lead frames are widely used in semiconductor industry for creating integrated circuit packaging. A lead frame is a thin layer of metal inside a chip package connecting a die to the circuitry on circuit boards. After a die is placed on the lead frame, metal leads on the lead frame are bonded to the chip. For semiconductor device manufacturers, inspection of the defects on lead frames is an important task to avoid flaws in the final electronic products. As the conventional manual inspection process is costly and time-consuming, automated optical inspection (AOI)-based inspection of the defects on lead frames is important for industry automation. Inspection of the defects on lead frames consists of detection and classification of the defects. The typical AOI inspection method is based on template matching technology, which requires a lot of time for parameter setting and template creation, and the over-kill rate may be high. These disadvantages motivate the proposal of new AOI-based defect inspection methods [1].

In the area of computer vision, different objection detection methods have been proposed and they can be categorized into handcrafted and deep learning methods according to the ways features are extracted. For the former, different features extracted via shape or color information have been proposed, such as scale-invariant feature transform (SIFT) [2], speeded-up robust features (SURF) [3], color histograms [4, 5] and entropies [6], and modified histograms of oriented gradients [7, 8]. The application of different handcrafted features to AOI-based inspection of defects on printed circuit boards (PCB) has also been proposed [9, 10]. One method was binary segmentation of images followed by the subtraction of a template image by test images [9]. Another method was detection of defects using the SURF feature and random forest classifier [10]. The method of feature extraction using Fast Fourier transform (FFT) and detection of the

defects using template difference in the feature space has also been proposed [11]. However, this method requires precise position alignment between an inspection circuit and a template circuit.

The advent of deep learning [12] has inspired the new application domain of deep learning-based inspection of defects on different products, such as PCBs in electronic products [13–17], wafers in semiconductor manufacturing [18], and polymeric polarizer in Thin Film Transistor Liquid Crystal Display (TFD-LCD) [19]. In traditional AOI, the detected defects may be real or pseudo (false alarms). To address this problem in manufacturing, the typical approach in industry is human visual inspection of the detected defects at the verify and repair system (VRS) station. As this is a high-cost method, the idea of using deep convolutional neural networks (CNNs) to classify real and pseudo defects after the AOI has been proposed to reduce the operator's workload and cost. Based on this idea, the applications of the AlexNet [13] and Inception-V3 models [14] to classify real and pseudo defects have been proposed. Another approach is the replacement of the traditional AOI method by deep CNN-based inspection. The method in [15] cut an inspection circuit image into small patches and divided them into six categories: no defects, short circuit, open circuit, spurious copper, mousebite, and spur. A CNN was trained to classify these patches. However, this method cannot exactly determine the location and size of the defects. In addition to the above methods of applying deep CNN to classify defect and non-defect patches, the application of deep CNN-based object detection method to detect defects has been proposed. The application of the faster region-based CNN (R-CNN) [20] and You-Only-Look-Once (YOLO) v2 [21] to detect and classify PCB defects were proposed in studies [16, 17], respectively. In contrast to semantic segmentation, defect detection avoids the time-consuming process of labeling defect boundaries for model training. The detection result can also be combined with semantic segmentation result, if available, to obtain instance segmentation. Therefore, this chapter focuses on detection of defects.

Different from the application of deep CNNs to detect defects on PCB [16, 17], this chapter applies the faster R-CNN to detect and classify defects on lead frames. Most of all, the main contribution of this chapter is the proposal of a new defect augmentation method using Generative

Adversarial Network (GAN) to improve detection performance. The number of available defects is usually limited at the early stage of producing new lead frames. The lack of sufficient training data may fail to train a well performed deep learning model. The GAN-based defect augmentation is proposed to address this problem.

Different types of GANs have been proposed to generate designated images from random vectors [22–26] or perform image-to-image translation [27–30]. For the later, training images can be paired or unpaired. One of the paired training models is the picture-to-picture conversion GAN (pix2pixGAN) [27]. For some real applications, collection of the paired images may be infeasible or costly to obtain, which restricts application domain of paired training model. In response to this problem, a cycle-consistent GAN (CycleGAN) was proposed, which can be used for unpaired image conversion [28]. The CycleGAN has been successfully applied to different domains such as translation between different stained renal pathology images to improve the detection performance of glomeruli in different stained images [31]. The applications of different GANs to defect detection [32, 34] and fault diagnosis [33] have been proposed. Different from these detection applications, this chapter applies the CycleGAN to automatically translate normal patches on lead frame images to defect patches. The augmented defect patches are then blended into the lead frame images by using a linear weighting method to obtain augmented defects to train the faster R-CNN. The advantage of the CycleGAN-based defect augmentation is shown through experiments on inspection of defects on lead frames.

This chapter is organized as follows. Section 9.2 introduces the materials and inspection of the defects on lead frames using the faster R-CNN. Section 9.3 introduces the application of the Cycle-GAN to generate defects on lead frames for data augmentation. Section 9.4 presents experimental results. Finally, conclusions are summarized in Sect. 9.5.

9.2 Defect Inspection Using the Faster R-CNN

This section first introduces the lead-frame images collected for training and test and the types of defects to be detected. Structure and training of the

faster R-CNN employed to defect defects on lead-frame images are then described.

9.2.1 Materials

All images of lead frames are taken from a linear scanning camera. Figure 9.1 shows one of the images, where the size of each image is 2296 × 1841 pixels. The defects on the lead frames are classified into two categories: spot defects and notch defects on the boundary of the middle part, as shown in Fig. 9.1. The ground truths of the defects are manually labeled. A total of 308 images are collected for training, which contain 200 spot defects and 221 notch defects. Figure 9.2 shows some patches of the defects. Another 376 images are collected for test, which contain 181 spot defects and 279 notch defects. The sizes of the spot defects vary from 8 × 14 to 91 × 113 pixels, and the sizes of the notch defects vary from 15 × 12 to 41 × 21 pixels.

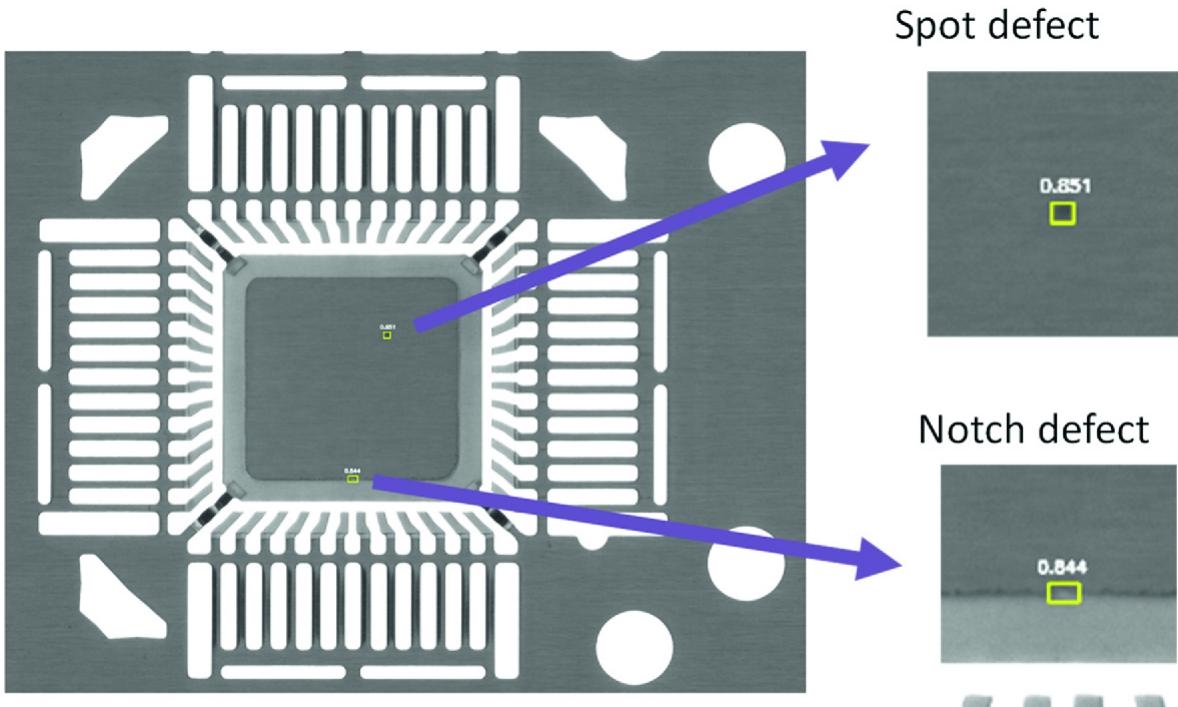


Fig. 9.1 Lead frame image and the two categories of defects

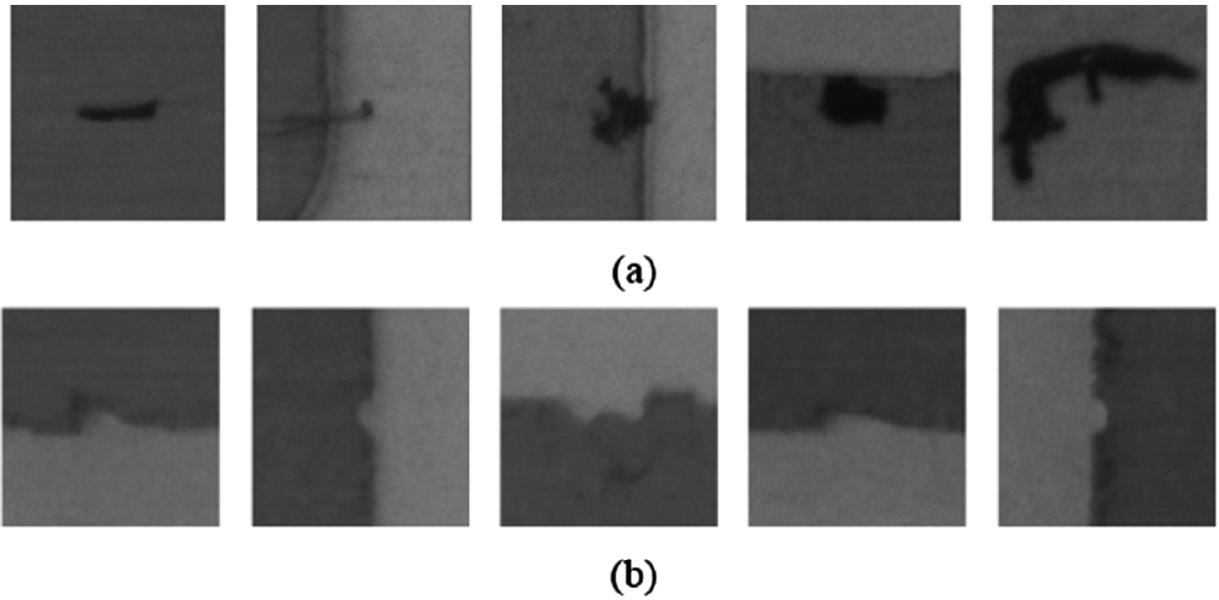


Fig. 9.2 Patches of **a** spot **b** notch defects

9.2.2 Defect Inspection Using the Faster R-CNN

This chapter applies the faster R-CNN to detect and classify the defects on lead frames, as shown in Fig. 9.3. The AlexNet [35] is used as the backbone, in which there are five convolution and two hidden layers in the convolution network and fully connected network (FCN), respectively. This chapter simplifies the FCN structure by using only one hidden layer with 16 nodes to reduce the model size and defect inspection time during test. For a two-class classification problem, the total number of parameters in the original and simplified FCNs are about 54.5 M and 0.15 M, respectively. The simplified FCN significantly reduces the network size.

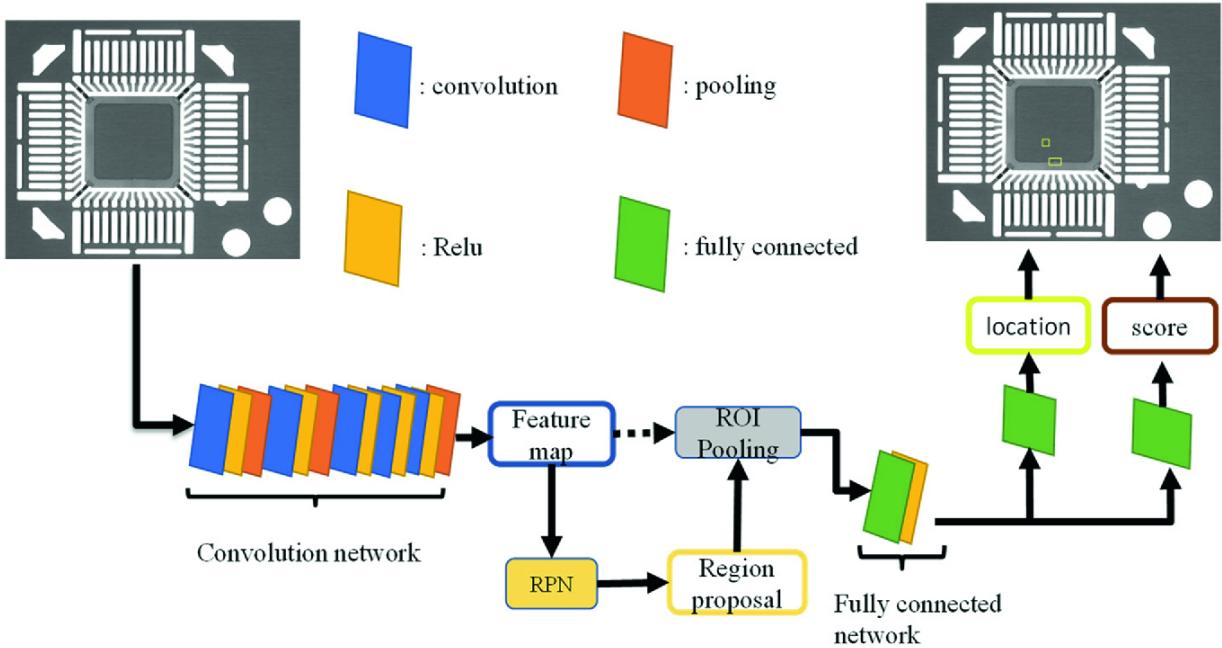


Fig. 9.3 Structure of the faster R-CNN using the AlexNet as the backbone for defect inspection

Figure 9.3 shows that the whole image is fed to the faster R-CNN. The feature map of the input image is first extracted by the convolution network. The region proposal network (RPN) then proposes candidate boxes of defects, including class confidence and detection box adjustment parameters by integrating the information in the feature map. The integration of the RPN consists of five steps. The first step uses a 3×3 convolution kernel to perform convolution with the feature map, as shown in Fig. 9.4, to obtain a 256-dimensional feature vector. The position of the center point of the 3×3 kernel corresponds to a region proposal that predicts positions/sizes and scores of the possible defects based on the 9 anchors generated on the input image, where the positions/sizes are determined by using a box regression method. In this method, the RPN outputs a set of parameters to adjust the sizes and positions of the anchor boxes to correctly match their nearby defects, if any, on the image. The second step discards the boxes exceed the original image border. The third step keeps the top 6000 scored anchors. The fourth step filters out overlapping anchors by using the non-maximum suppression (NMS) method. The intersection over union (IOU) threshold of the NMS is set to

0.7. The fifth step selects the best 300 high score anchors as region proposals for defects.

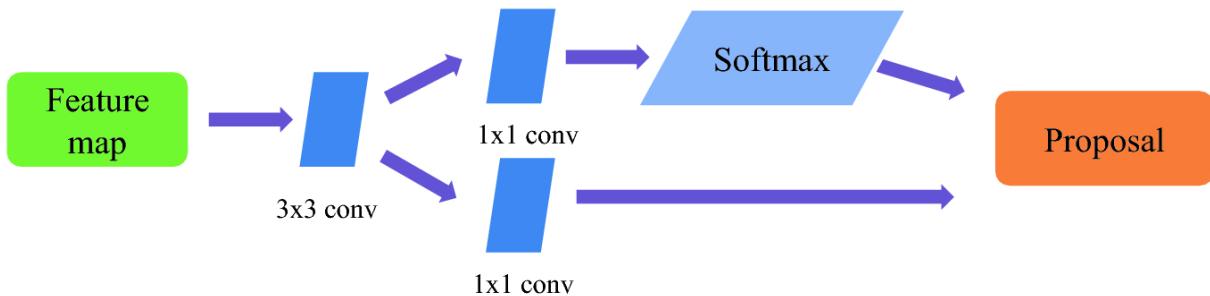


Fig. 9.4 Structure of the RPN

The feature maps of the proposed regions are downsampled by the region of interest (ROI) pooling layer to the fixed size of 6×6 feature maps. The feature maps are flattened to a one-dimensional vector and sent to the fully connected layer to determine the location, size, and class of each defect candidate, where the location and size are also determined using the box regression method.

9.3 Defect Augmentation Using the CycleGAN

This section first describes the basic structure of a GAN. Structure and training of the CycleGAN employed to generate new defects are then described. Finally, this section introduces the proposed data augmentation approach, including generation of new defect patches by using the CycleGAN and paste of these patches onto images.

9.3.1 Basic GAN Structure

GAN is a technique that can be applied to generate new data such as images. It contains two parts, generator and discriminator, as shown in Fig. 9.5. Given an input vector, the generator G learns to generate fake images that resemble real images in a training dataset. These generated fake images become negative training examples of the discriminator D . The discriminator learns the generated fake images by discriminating them from real images. The formula of the loss function when training a GAN is defined based on the cross-entropy and is given as follows:

$$\min_G \max_D V(D, G) = E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_{data}(z)}[\log(1 - D(G(z)))] \quad (9.1)$$

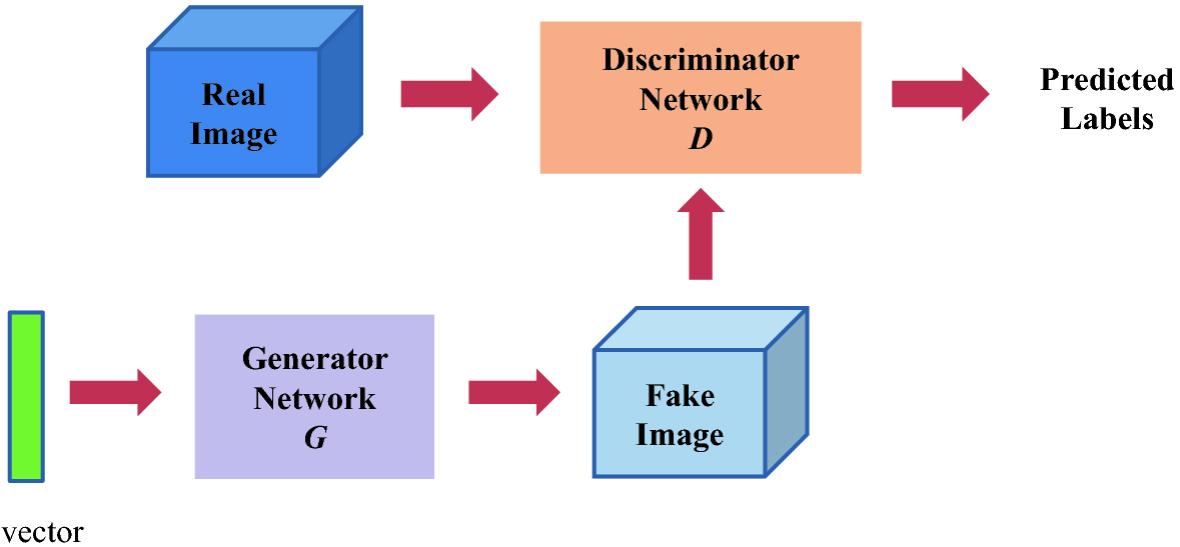


Fig. 9.5 Overall structure of the GAN

where E represents the expected value, π is the true image, and the $+$ is an input sample to the generator G . Training starts with random initialization. The training method is divided into two parts. It first optimizes the weights of the discriminator, then fixes the weights of the discriminator and optimizes the weights of the generator. When GAN is applied to perform image-to-image translation, training images can be paired (Fig. 9.6a) or unpaired (Fig. 9.6b).

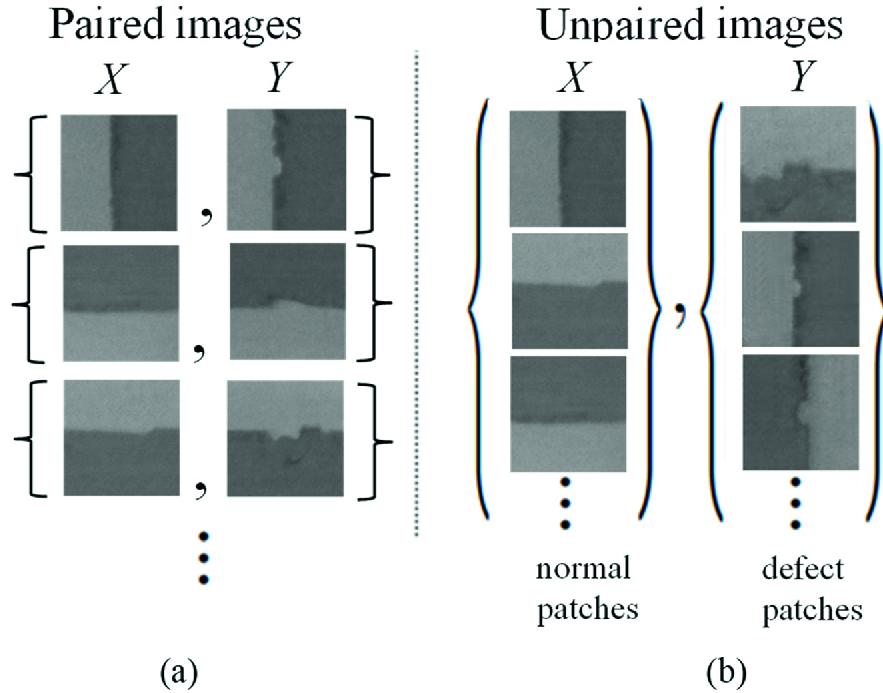
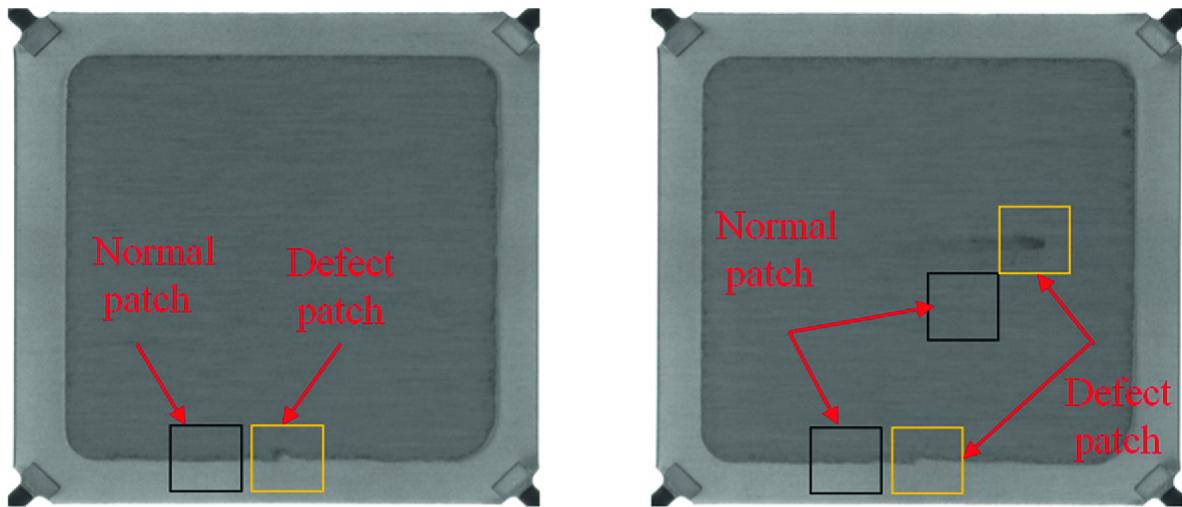


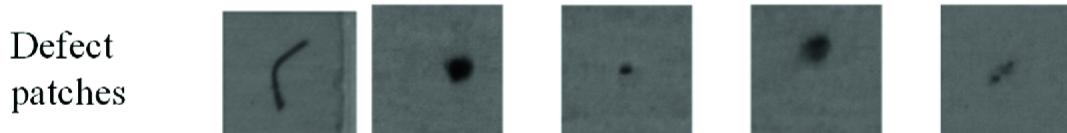
Fig. 9.6 **a** Paired and **b** unpaired images collected for training

9.3.2 CycleGAN Structure

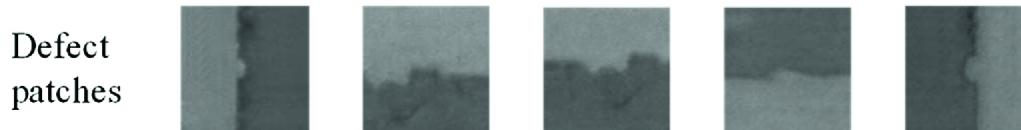
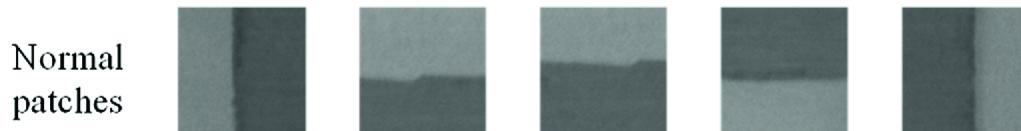
CycleGAN is a branch of the GAN, which can simultaneously perform training of two generators and two discriminators. The CycleGAN is characterized with unpaired input–output training images, which makes it possible to convert normal patches on lead frames to defect patches. Instead of feeding the whole lead frame image to the CycleGAN, this chapter uses small patches of the image to train the CycleGAN, with each patch measuring 100×100 pixels. Figure 9.7 shows some of the collected image patches to train the CycleGANs to generate spot and notch defects for data augmentation.



(a)



(b)



(c)

Fig. 9.7 **a** Collection of normal and defect patches from led frame images to train the CycleGAN. **b** Some of the collected image patches to train the CycleGAN to generate spot and **c** notch patches for data augmentation

Figure 9.8 shows the basic cycle structure of the CycleGAN. The generator $\mathcal{D} = (\mathbf{X}, \mathbf{Y}, S)$ gets the normal image patch π from domain μ_q and translates it to a defect patch in \mathbf{Y} . On the contrary, the other generator $F(\cdot) : \mathbf{Y} \rightarrow \mathbf{X}$ gets a defect patch η from domain \mathbf{Y} and translates it to a normal patch in μ_q . The two discriminators are denoted as D_X and \mathcal{W}_2 , where the former distinguishes between real and faked normal patches and the later distinguishes between real and faked defect patches.

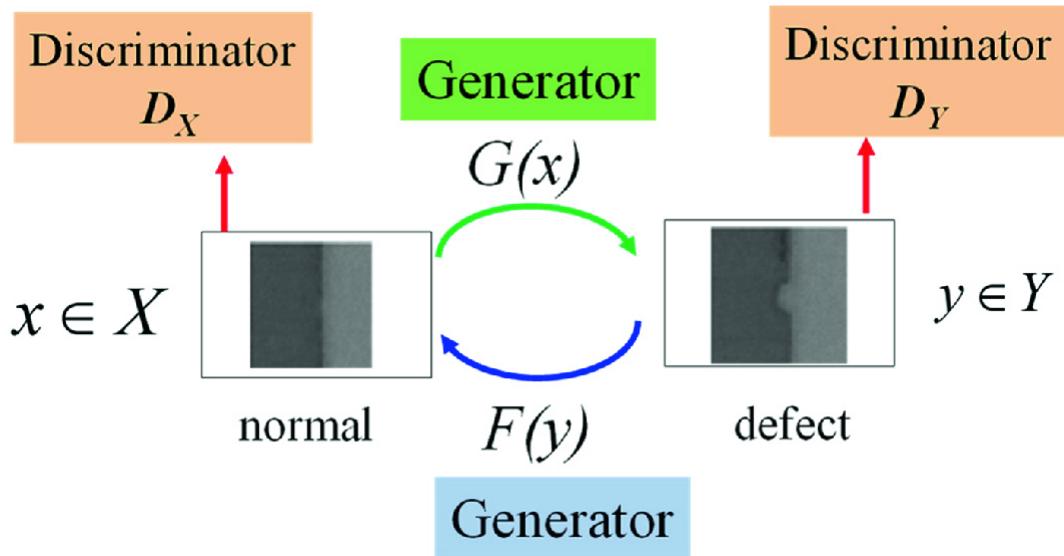
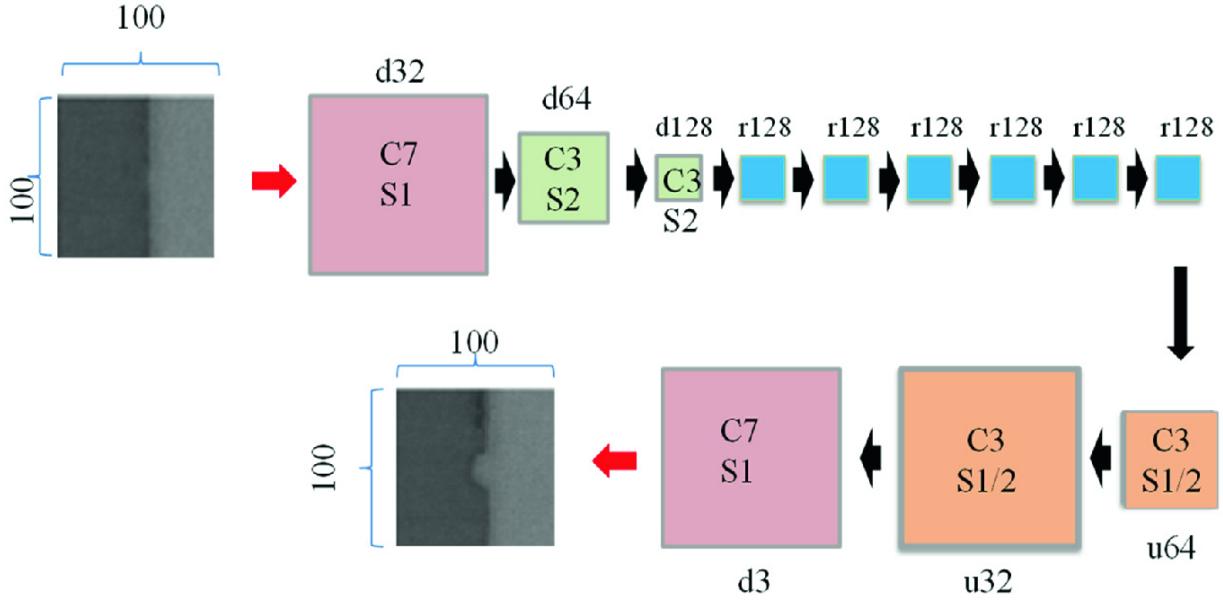


Fig. 9.8 Basic cycle structure of the CycleGAN

9.3.2.1 Generator Structure

Figure 9.9 shows the generator structure, which consists of 12 layers with the rectified linear unit (ReLU) used as the activation function. The whole structure can be divided into two parts: encoder and decoder. In the encoder, an input image patch passes through a 7×7 convolution layer and two 3×3 convolution layers. Then, the extracted features pass through 6 residual blocks, each of which containing two 3×3 convolution layers. The decoder passes the feature maps from the encoder through two 3×3 deconvolution layers followed by a 7×7 convolution layer.



C7: 7x7 convolution layer. **S1:** stride of 1. **d32:** depth of 32.
r128: a residual block with depth 128.
u64: a deconvolution layer with depth 64.

Fig. 9.9 The generator structure in the CycleGAN

9.3.2.2 Discriminator Structure

Figure 9.10 shows the discriminator structure, which consists of 5 layers with the leaky-RELU [36] used as an activation function as follows [36]:

$$a = \max(\text{slope} \times b, b) \quad (9.2)$$

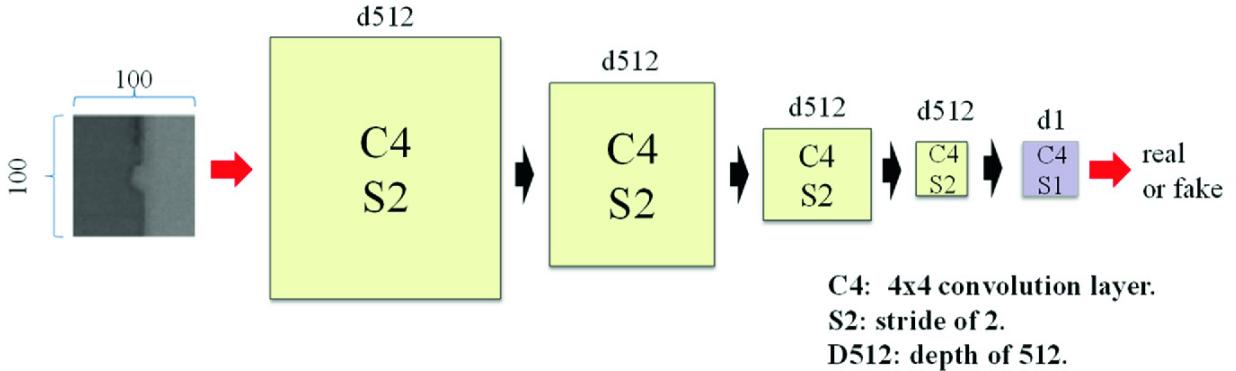


Fig. 9.10 The discriminator structure in the CycleGAN

where b and a denote the input and output, respectively, and the slope is set to 0.2. The input image patch passes through four 4×4 convolution

layers with a stride of 2, and then passes through a 4×4 convolutional layer with a depth of 1 and a stride of 1.

9.3.3 CycleGAN Training

During the training process, the normal patch π is translated into a defect patch by the generator $q(\mathbf{z})$, as shown in Fig. 9.11. In order to maintain the cycle consistency, when the translated defect patch $G(x)$ is reversely translated to the normal patch $q^* = q^{-1}$ by the generator $q(\mathbf{z})$, the loss between them should be as small as possible. Similarly, generator $q(\mathbf{z})$ translates a defect patch η to a normal patch $G(\mathbf{z})$. The consistency loss between the reversely translated defect patch $q^* = q^{-1}$ and η should be as small as possible. Therefore, the total loss of cycle consistency is defined as follows:

$$L_{cyc}(G, F) = E_{x \sim P_{data}(x)}[\|F(G(x)) - x\|_1] + E_{y \sim P_{data}(y)}[\|G(F(y)) - y\|_1]. \quad (9.3)$$

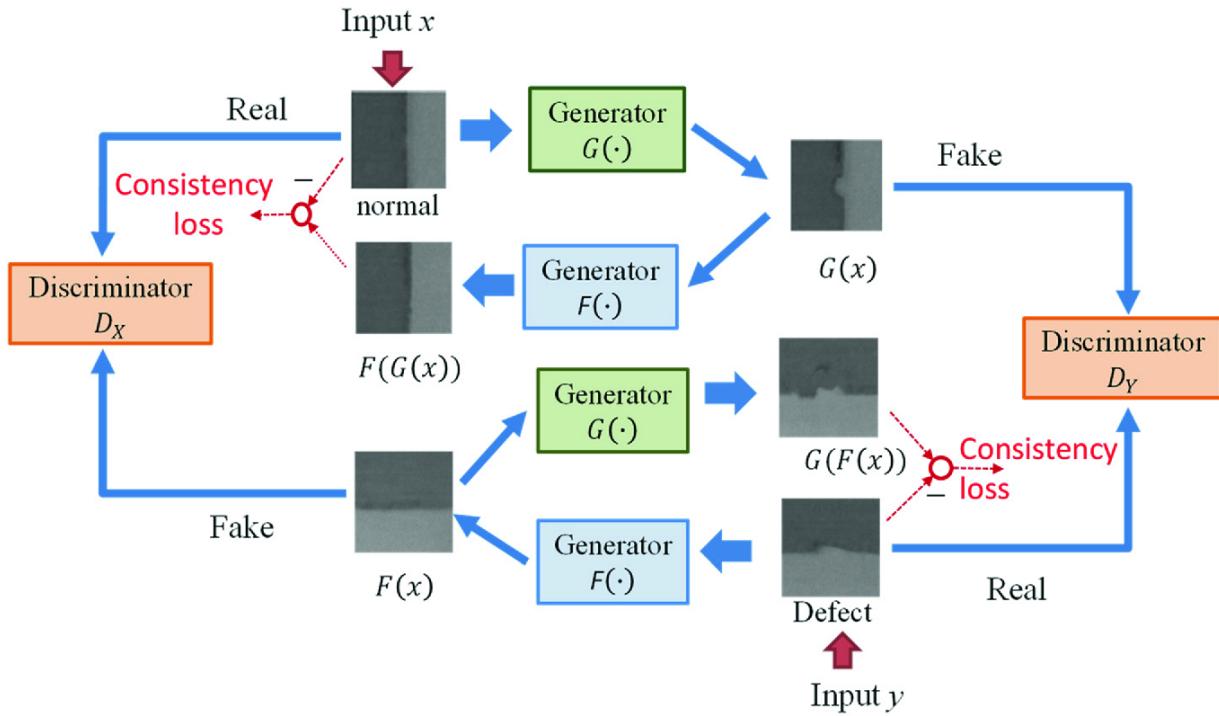


Fig. 9.11 Training structure of the CycleGAN

The discriminator \mathcal{W}_2 is trained to discriminate between fake defects $G(x)$ and real defects η . Like (9.1), the loss function is defined as follows:

$$L_{GAN}(G, D_Y, X, Y) = E_{y \sim P_{data}(y)}[\log D_Y(y)] + E_{x \sim P_{data}(x)}[\log(1 - D_Y(G(x)))] \quad (9.4)$$

Like \mathcal{W}_2 , the discriminator D_X is trained to discriminate between fake normal patches $G(\mathbf{z})$ from real normal patches π using loss function $L_{GAN}(F, D_X, Y, X)$. The total loss is

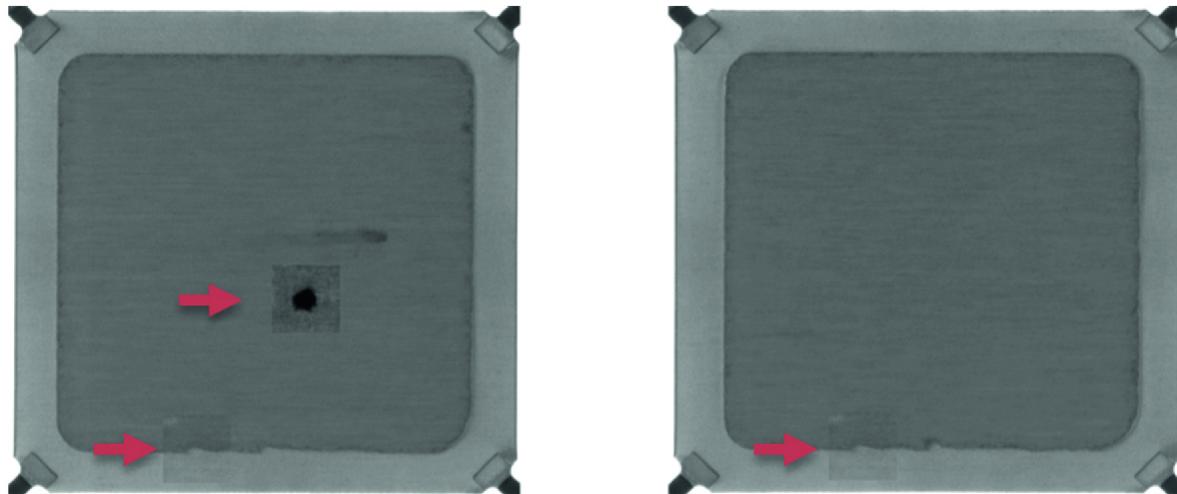
$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + 10 \cdot L_{cyc}(G, F) \quad (9.5)$$

Finally, the optimal generators \mathbf{W} and \mathbf{W}' are found as follows:

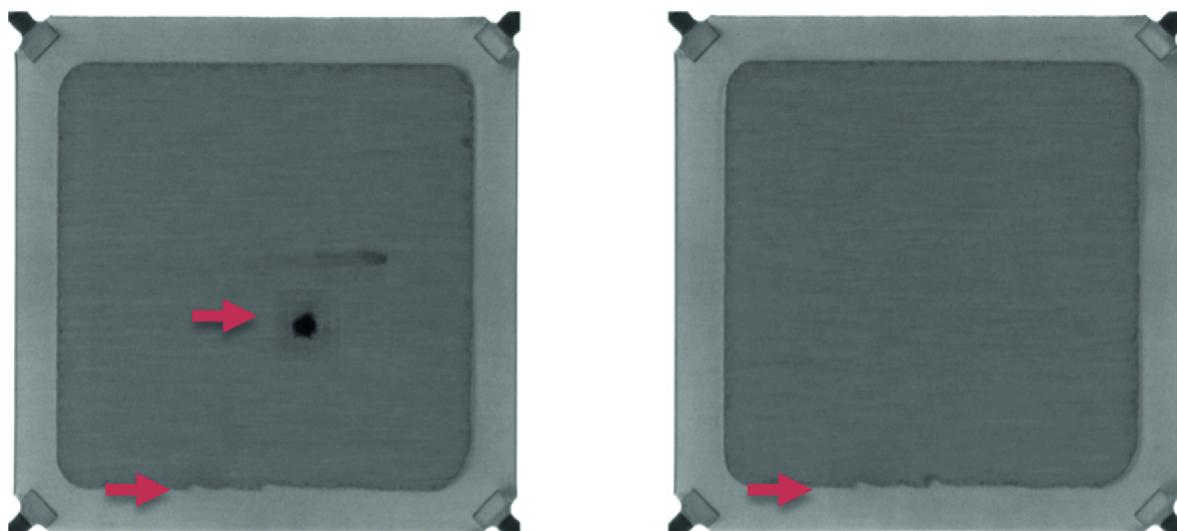
$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} L(G, F, D_X, D_Y) \quad (9.6)$$

9.3.4 CycleGAN for Image Augmentation

As typical data augmentation methods such as vertical/horizontal flip of an image do not generate new defect shapes, this chapter applies the CycleGAN to generate new defect shapes. The defect patches generated from the CycleGAN are blended into the whole led frame image for data augmentation with the aim of improving training performance of the faster R-CNN. To blend the defect patches into the led frame image, one simple approach is complete replacement of the overlapped patch in the original image by the generated defect patch. Figure 9.12a shows two examples of this blending approach. It is observed that the boundary effect is obvious. To reduce the boundary effect, this chapter applies a linear blending method to blend the defects into normal patches. As the generated defects mainly locate around the center of a patch, the principle of the linear blending method is that pixels in the defect patch are assigned higher or lower blending weights when they are closer or farer from the patch center, respectively. Let d_1 denote half of the diagonal distance of a patch and $\mathcal{G}(z, c)$ denote the distance between the patch center and a pixel at position $\mathcal{G}(z)$, as shown in Fig. 9.13. The red–green–blue (RGB) vectors of the normal and defect patches at position $\mathcal{G}(z)$ are denoted as $\overrightarrow{I}_N(i, j)$ and $\overrightarrow{I}_N(i, j)$, respectively. In the overlapping area, the RGB vector of the newly blended pixel $\overrightarrow{I}_N(i, j)$ is found as follows:



(a)



(b)

Fig. 9.12 Blending of the CycleGAN generated defect patches into lead frame images by using **a** complement replacement and **b** linear blending approaches

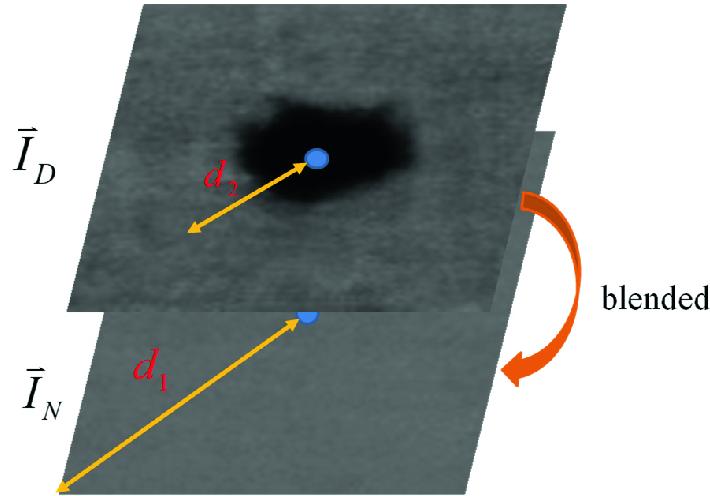


Fig. 9.13 Linear blending of two overlapped patches with the blending weight determined by the distance between a pixel position and the path center

$$\vec{I}_B(i, j) = \vec{I}_N(i, j) \times (1 - w) + \vec{I}_D(i, j) \times w \quad (9.7)$$

where

$$F(\cdot) : Y \rightarrow X \quad (9.8)$$

Figure 9.12b shows blending result of the two image samples in Fig. 9.12a by using the linear blending method, where it is found that boundary effect is significantly reduced.

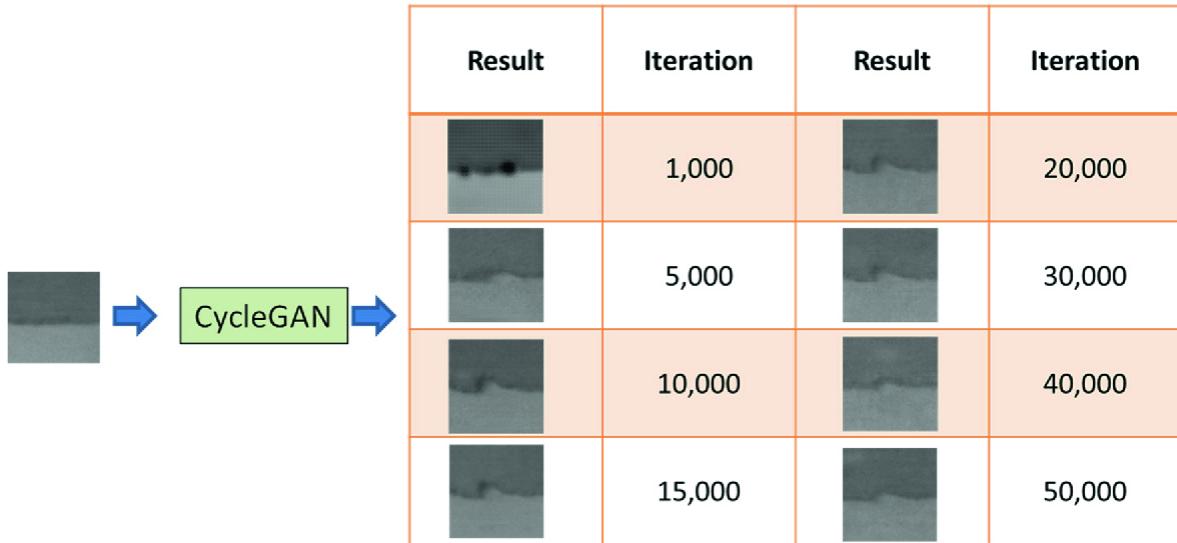
As introduced above, generated defect patches are blended into whole lead frame images. If the background of a defect patch shows significance difference to the blended patch in the image, then boundary effect is obvious even after the linear blending operation. Therefore, this chapter selects the CycleGAN to transfer the blended patch in an image to a defect patch in order that both patches have similar backgrounds. This background consideration explains the reason why the CycleGAN is selected instead of the GANs that generate designated images from random vectors such as the deep convolutional GANs (DCGANs) [24].

9.4 Experiments

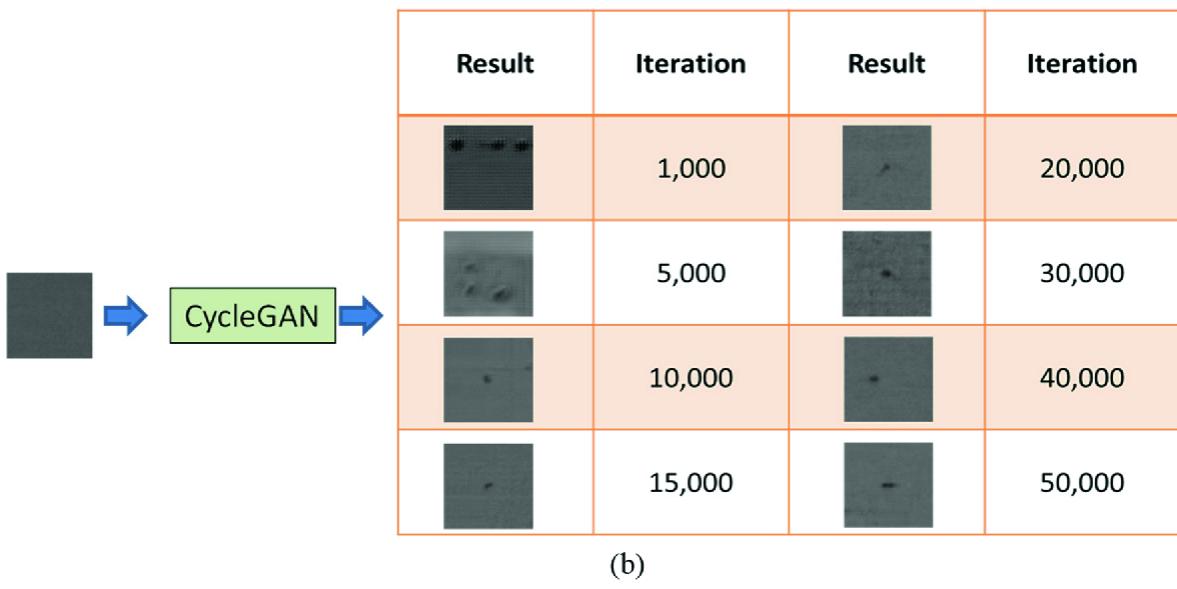
This section introduces experimental results of defect inspection of lead frames using the faster R-CNN and CycleGAN-based defect augmentation.

9.4.1 Experimental Result of CycleGAN

Two CycleGANs were trained. As stated in Sect. 2.1, the training data set contained a total of 308 lead frame images, in which there were 200 spot defects and 221 notch defects. One of the CycleGANs was trained using the 200 spot defects and another 200 normal patches. The other was trained with the 221 notch defects and another 221 normal patches. The Adaptive Moment Estimation (Adam) was applied to optimize the parameters. The learning rate was set to 0.0002. The coefficients β_1 and β_2 were set to 0.5 and 0.999, respectively. The maximum number of training iterations was set to 50,000. During the CycleGAN training process, the quality of the translated image patched was monitored. Figure 9.14a and b shows an example of training the CycleGANs to translate normal patches to notch and spot defects, respectively, at different training iterations. According to the experimental result, it was found that the quality of the translated defect patches is satisfied when the number of training iterations exceeded 10,000. It was also found that when the number of training iteration was too large, all the translated defect patches may converge to a similar shape, causing a model collapse. Figure 9.14 also shows the generated defects from different iterations may be different for the same input normal patch. These defects at different iterations can also be collected for data augmentation. Figure 9.15 shows some of the translated spot and notch defect patches using the CycleGANs fed with different normal patch inputs after the training process, where it was found that the translated defects resemble real ones.

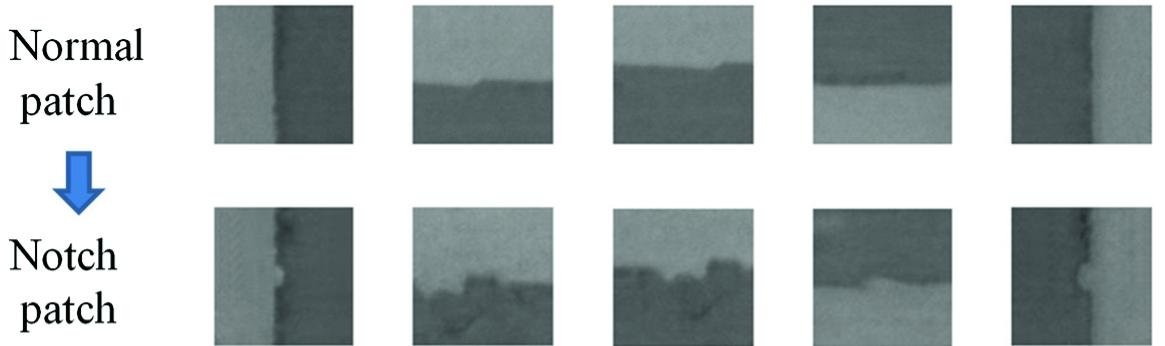


(a)

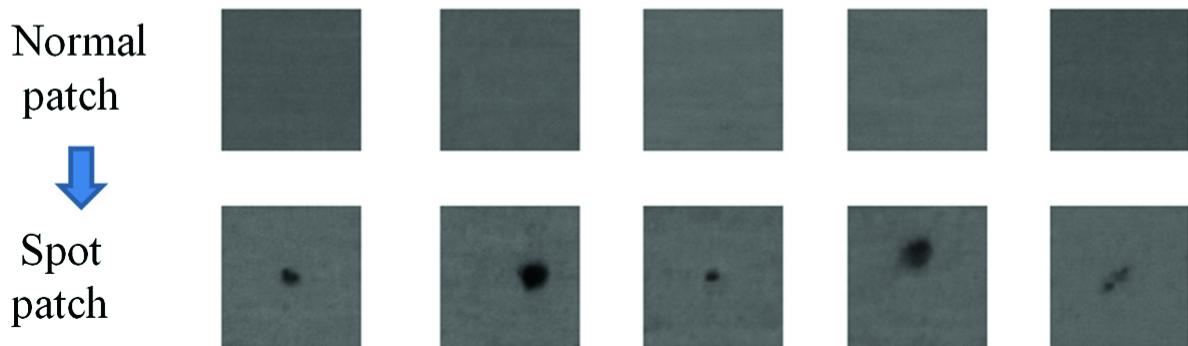


(b)

Fig. 9.14 Training of the CycleGAN to translate normal patches to **a** notch and **b** spot defects at different training iterations



(a)



(b)

Fig. 9.15 Translations of different normal patches into **a** notch and **b** spot defects using the CycleGANs

9.4.2 Experimental Result of Defect Inspection Using the Faster R-CNN

The faster R-CNN was applied to detect and classify the defects on lead frames. A whole lead frame image was fed to the faster R-CNN in training and test. The metrics used to evaluate the inspection performance was described as follows. The match between a detection box D_1 and the target box A_p is defined according to their IOU given as follows [20]:

$$\text{IOU} = \frac{|A_u \cap A_p|}{|A_u \cup A_p|} \quad (9.9)$$

If $\text{IOU} > 0.3$, the detection box is regarded as a true positive (TP). The threshold of 0.3 was smaller than the typical value of 0.5 because the sizes of the defects may be as small as 8×14 pixels. The smaller threshold was set to reduce detection misses, which was the most important consideration

in product quality control. Based on TP, true negative (TN), false positive (FP), and false negative (FN), the recall (R), precision (P), and F-measure ($\hat{\sigma}^2$) are defined as follows [37]:

$$\text{Precision } (P) = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9.10)$$

$$\text{Recall } (R) = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (9.11)$$

$$F_1 = \frac{2 \times P \times R}{P + R} \quad (9.12)$$

The average precision (AP) and mean AP (mAP) were used to evaluate the overall performance using the constant IOU threshold stated above. Two experiments were performed. In the training of the faster R-CNN, the initial learning rate and momentum coefficient were set to 0.001 and 0.9, respectively. The maximum number of training iterations was set to 90,000 in each experiment. The convolution network in the AlexNet was pre-trained using the 1000-class ImageNet. Experiments were performed using the NVIDIA TITAN graphic processing unit (GPU).

Experiment 1. In the first experiment, the original 308 lead frame images (containing 441 real defects) were collected as training data. In addition, horizontal flip of the images was performed for data augmentation. Table 9.1 shows the test performance, including P, R, maximum $\hat{\sigma}^2$, AP, and mAP of inspecting the whole lead frame test images. The AP values of detecting the spot and notch defects were 0.8104 and 0.9324, respectively, and the mAP was 0.8714. The total time of inspecting a whole lead frame image took 0.077 s.

Table 9.1 Inspection performance of test images using the faster R-CNN in different experiments with and without CycleGAN augmented defects

Experiments	Experiment 1		Experiment 2	
Training data	441 real defects		441 real defects + 441 augmented defects	
Class	Spot	Notch	Spot	Notch
Precision	0.9524	0.9810	0.9433	0.9926

Experiments	Experiment 1		Experiment 2	
Training data	441 real defects		441 real defects + 441 augmented defects	
Class	Spot	Notch	Spot	Notch
Recall	0.5525	0.9247	0.7348	0.9606
F_1	0.6993	0.9520	0.8261	0.9763
AP	0.8104	0.9324	0.8310	0.9771
mAP	0.8714		0.9042	

Experiment 2. In the second experiment, the defects generated from training the CycleGAN were blended into the 308 training lead frame images. That is, there were a total of 441 real and 441 augmented defect patches in the 308 lead frame images for training the faster R-CNN. Like experiment 1, horizontal flip of the images was also performed for data augmentation. Table 9.1 shows the test performance. The AP values of detecting the spot and notch defects were 0.8310 and 0.9771, respectively, and the mAP was 0.9042. In contrast to the first experiment, the result showed that with the augmented defects, the APs of detecting the spot and notch defects were increased by 0.0206 and 0.0447, respectively. The mAP was increased by 0.0328. The result showed the effect of the CycleGAN generated defects on improving the inspection performance. Figure 9.16 shows some of the inspection results, including TP, FP, and FN.

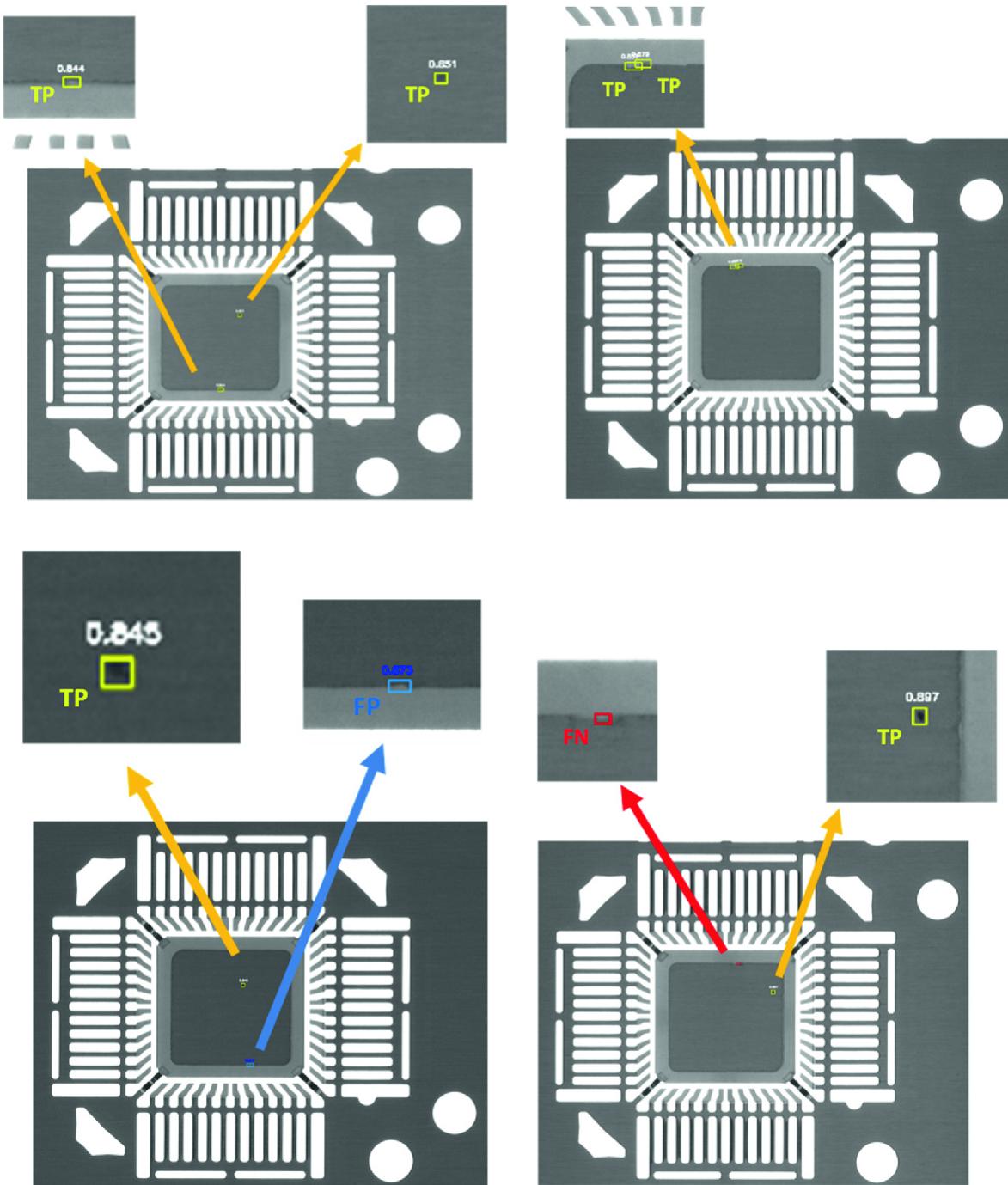


Fig. 9.16 Inspection results of the defects on different test lead frames

9.5 Conclusions

This chapter proposes a new method for augmentation and inspection of defects on lead frames. The CycleGAN has been successfully applied to automatically translate normal image patches into different types of defect

patches. As the number of available defects is usually limited at the early stage of producing new lead frames, the lack of sufficient data for training a deep learning network may degrade the inspection performance of the network. To address this problem, augmented defects from the CycleGANs have been collected to train a faster R-CNN to improve its inspection performance. Experimental results show that the CycleGANs help generate new defects that resemble real ones. Linear blending of the generated defects into lead frame images helps reduce boundary effects and training of the faster R-CNN using the images with augmented defects helps improve the inspection performance. In the future, further improvement of the inspection performance of the faster R-CNN with other complex deep CNNs as the backbone will be studied. The application of the CycleGAN-based defect augmentation approach to inspect defects on other lead frames will be studied as well.

References

1. Moganti, M., Ercal, F., Dagli, C. H., Tsunekawa, S.: Automatic PCB inspection algorithms: a survey. *Comput. Vis. Image Underst.* **63**(2), 287–313 (1996)
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004)
[Crossref]
3. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **110**(3), 346–359 (2008)
[Crossref]
4. Juang, C.F., Chen, G.C.: A TS fuzzy system learned through a support vector machine in principal component space for real-time object detection. *IEEE Trans. Ind. Electron.* **59**(8), 3309–3320 (2012)
[Crossref]
5. Juang, C.F., Chen, G.C., Liang, C.W., Lee, D.: Stereo-camera-based object detection using fuzzy color histograms and a fuzzy classifier with depth and shape estimations. *Appl. Soft Comput.* **46**, 753–766 (2016)
[Crossref]
6. Chen, G.C., Juang, C.F.: Object detection using color entropies and a fuzzy

classifier. *IEEE Comput. Intell. Mag.* **8**(1), 33–45 (2013)

[[Crossref](#)]

7. Liang, C.W., Juang, C.F.: Moving object classification using a combination of static appearance features and spatial and temporal entropies of optical flows. *IEEE Trans. Intell. Transp. Syst.* **16**(6), 3453–3464 (2015)
[[Crossref](#)]
8. Liang, C.W., Juang, C.F.: Moving object classification using local shape and HOG features in wavelet-transformed space with hierarchical SVM classifiers. *Appl. Soft Comput.* **28**, 483–497 (2015)
[[Crossref](#)]
9. Chaudhary, V., Dave, I.R., Upla, K.P.: Automatic visual inspection of printed circuit board for defect detection and classification. In: Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking, pp. 732–737 (2017)
10. Yuk, E.H., Park, S.H., Park, C.S., Baek, J.G.: Feature-learning-based printed circuit board inspection via speeded-up robust features and random forest. *Appl. Sci.* **8**, 1–14 (2018)
[[Crossref](#)]
11. Changcheng, J., Yanming, Q., Xingui, L., Zehui, X.: An improved fast self-comparison algorithm for high-speed defect detection of ITO circuits. In: Proceedings 8th International Conference Measuring Technology and Mechatronics Automation, pp. 85–88 (2016)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, MIT Press (2016)
13. Deng, Y.S., Luo, A.C., Dai, M.J.: Building an automatic defect verification system using deep neural network for PCB defect classification. In: 4th International Conference Frontiers of Signal Processing, pp. 145–149 (2018)
14. Ghosh, B., Bhuyan, M.K., Sasmal, P., Iwahori, Y., Gadde, P.: Defect classification of printed circuit boards based on transfer learning. In: Proceedings of the IEEE Applied Signal Processing Conference, pp. 245–248 (2018)
15. Zhang, L., Jin, Y., Yang, X., Li, X., Duan, X., Sun, Y., Liu, H.: Convolutional neural network-based multi-label classification of PCB defects. *J. Eng.* **2018**(16), 1612–1616 (2018)
[[Crossref](#)]

16. Hu, B., Wang, J.: Detection of PCB surface defects with improved faster-RCNN and feature pyramid network. *IEEE Access*. **8**, 108335–108345 (2020) [\[Crossref\]](#)
17. Adibhatla, V.A., Chih, H.C., Hsu, C.C., Cheng, J., Abbod, M.F., Shieh, J.S.: Defect detection in printed circuit boards using you-only-look-once convolutional neural networks. *Electronics* **9**(1547), 1–16 (2020)
18. Tello, G., Al-Jarrah, O.Y., Yoo, P.D., Al-Hammadi, Y., Muhaidat, S., Lee, U.: Deep-structured machine learning model for the recognition of mixed-defect patterns in semiconductor fabrication processes. *IEEE Trans. Semicond. Manuf.* **31**(1), 315–322 (2018) [\[Crossref\]](#)
19. Lei, H., Wang, B., Wu, H.H., Wang, A.H.: Defect detection for polymeric polarizer based on faster R-CNN. *J. Inf. Hid. Multimed. Sign. Process* **9**(6), 1414–1420 (2018)
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1137–1149 (2017)
21. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger 1–9 (2016). [arXiv: 1612.08242](#)
22. Goodfellow, I.J., Pouget-Abadiey, J., Mirza, M., Xu, B., Warde-Farley, D., Ozairz, S., Courville, A., Bengio, T.: Generative adversarial networks. In: Proceedings of the 27th International Conf. Neural Information Processing Systems (NIPS), vol. 2, pp. 2672–2680 (2014)
23. Mirza, M., Osindero, S.: Conditional generative advearial nets (2014). [arXiv: 1411.1784](#)
24. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016). [arXiv:1511.06434v2](#)
25. Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation (2017). [arXiv:1611.02200](#)
26. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems, pp. 5769–5779 (2017)
- 27.

- Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference Computer Vision and Pattern Recognition, pp. 5967–5976 (2017)
28. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference Computer Vision, pp. 2242–2251 (2017)
29. Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., Choo, J.: StarGAN: unified generative adversarial networks for multi-domain image-to-image translation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 8789–8797 (2018)
30. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4396–4405 (2019)
31. Lo, Y.C., Chung, I.F., Guo, S.N., Wen, M.C., Juang, C.F.: Cycle-consistent GAN-based stain translation of renal pathology images with glomerulus detection application. *Appl. Soft Comput.* **98**, 1–13 (2021)
[[Crossref](#)]
32. Liu, J., Wang, C., Su, H., Du, B., Tao, D.: Multistage GAN for fabric defect detection. *IEEE Trans. Image Process.* **29**, 3388–3400 (2020)
[[Crossref](#)]
33. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M., Parvania, M.: Adversarial semi-supervised learning for diagnosing faults and attacks in power grids. *IEEE Trans. Smart Grid* **12**, 3468–3478 (2021)
[[Crossref](#)]
34. Situ, Z., Teng, S., Liu, H., Luo, J., Zhou, Q.: Automated sewer defects detection using style-based generative adversarial networks and fine-tuned well-known CNN classifier. *IEEE Access* **9**, 59498–59507 (2021)
[[Crossref](#)]
35. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst. (NIPS)* 1–9 (2012)
36. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1–6 (2013)

37. Powers, D.M.W.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness & correlation. *J. Mach. Learn. Technol.* **2**(1), 37–63 (2011)
[\[MathSciNet\]](#)

10. Adversarial Learning in Accelerometer Based Transportation and Locomotion Mode Recognition

Lukas Günthermann¹✉, Lin Wang²✉, Ivor Simpson¹✉,
Andrew Philippides¹✉ and Daniel Roggen¹✉

(1) University of Sussex, Brighton, UK

(2) Queen Mary University of London, London, UK

✉ Lukas Günthermann (Corresponding author)

Email: l.gunthermann@sussex.ac.uk

✉ Lin Wang

Email: lin.wang@qmul.ac.uk

✉ Ivor Simpson

Email: i.simpson@sussex.ac.uk

✉ Andrew Philippides

Email: andrewop@sussex.ac.uk

✉ Daniel Roggen

Email: daniel.roggen@ieee.org

Abstract

This chapter demonstrates how adversarial learning can be used in the mobile computing domain. Specifically, we address the problem of improving the recognition of human activities from smartphone sensors, when limited training data is available. Generative Adversarial Networks

(GANs) provide an approach to model the distribution of a dataset and can be used to augment data to reduce the amount of labelled data required to train accurate classifiers. By introducing another fully connected neural network as classifier into a conditional GAN framework, we utilise the adversarial learning approaches between discriminator and generator and between discriminator and classifier to perform semi-supervised learning on labelled and unlabelled samples. We evaluate our approach on the recognition of 8 modes of transportation and locomotion using a publicly available dataset. This dataset is well established and has led to 3 public machine learning challenges, which allows us to contrast our approach to the state of the art. Our GAN operates on 150 features extracted from 5s windows captured by a smartphone acceleration sensor carried at the hips. The most promising features are selected based on maximum relevance – minimum redundancy feature selection. We use Bayesian Search for hyperparameter optimisation. The resulting GAN classifier achieves 49% F1 score on a user independent evaluation, drastically outperforming our baseline at 35% F1 score.

10.1 Introduction

The large-scale availability of smartphones and other body-worn devices can provide mechanisms to gather information about human activity patterns [39]. A smartphone can contain sensor modalities such as accelerometer, gyroscopic, magnetometer, orientation, gravity, linear acceleration, ambient pressure, ambient light, battery level and temperature, network receptions, GPS location, audio, and video. The goal of human activity recognition (HAR) is to recognise actions of humans based on time series data originating from the signals of such sensors. This is usually achieved with an “activity recognition chain”: i.e. a sequence of signal processing and classification elements primarily based on machine learning [7].

In order to obtain the training data needed to solve the activity recognition problem, the recorded sensor inputs have to be labelled. Since observations usually occur over a long period of time, this process can become tedious work. For instance, it was reported that the annotation of one dataset, which included 25 h of sensor data, took 7–10 h per 30 min of

video footage [35]. This issue makes it attractive to utilise unlabelled samples to develop classification models. Several approaches have been explored, such as semi-supervised learning to exploit unlabelled data [33] or active learning, which attempts to prompt labels only when necessary [51]. Clustering has shown to be able to differentiate between different activities with high accuracy based on data from both, ambient and wearable sensors [10]. As an alternative to hand-crafted features, autoencoders have been successfully applied to feature extraction [29]. More recently, GANs have been used to predict actions in partially observable videos [40] and to perform knowledge transfer in the domain of wearable-sensor-based HAR [38].

In this chapter, we employ a GAN for the adversarial training of a classifier distinguishing 8 modes of transportation and locomotion (still, walk, run, bike, car, bus, train, and subway) from the data of the accelerometer in a mobile phone. This is an extension of previous work [18, 19], which is working with extracted features instead of time-series sensor recordings to reduce required data storage and computational requirements. In contrast to these previous versions, we chose to use only the data of the 3D acceleration sensor from the smartphone placed in the trouser pocket of the users. A previous analysis showed that the recognition of the 8 locomotion modes is possible with an F1 score of up to 54.9% even when using only the acceleration sensor data [45]. We also drastically increased the number of extracted features from a basic set of five to 908 and used maximum relevance – minimum redundancy feature selection to select the 150 most relevant features. Adding more publicly available data from the SHL Locomotion and Transportation dataset [15] (SHL) allowed for improved cross- and inter-user evaluations. Furthermore, we introduced Bayesian Search to optimise the hyperparameters of the involved networks.

In this chapter, we will demonstrate how such a framework can be developed and analyse the suitability of the approach. The results are compared to a selection of baseline algorithms. The contributions of this chapter are a review of applied methods to recognise activities with the SHL dataset (Sect. 10.3), a review of GANs and their application in activity recognition (Sect. 10.4), the application and optimisation of our proposed GAN architecture on the problem of recognising modes of locomotion from

mobile phone sensor data (Sect. 10.5), an analysis of the results (Sect. 10.7), and discussion for future work (Sect. 10.8).

10.2 SHL Dataset

In order to evaluate our approach, as well as possible follows-up to this chapter, we sought a dataset which is publicly available, offers a large amount of data, includes a wide range of different sensor modalities, and which underpins practical applications in wearable and mobile computing. We chose to employ the SHL Locomotion and Transportation dataset [15] for this purpose. This dataset is a publicly available dataset designed to evaluate methods to recognise modes of transportation and locomotions from body-worn sensors, and in particular from sensors available in modern smartphones. The recognition of modes of locomotion and transportation enables a wide range of activity- and context-aware applications [13], among others health monitoring [14], or environmental impact assessment and recommendation [2].

The complete SHL dataset comprises the recordings of all the sensors of four Mate 9 smartphones located at distinct on-body locations (hand, trouser's pocket, shirt pocket and backpack/handbag) from 3 users, who engaged in 8 different transportation and locomotion activities over a period of 7 months, in the south-east of the UK, including in London. The activities were precisely annotated from a body-worn camera, and include 8 activity classes: being still, walking, running, cycling, driving in a car, in a bus, on a train or on the subway. All the motion sensor modalities are sampled 100 Hz. Since its preliminary release in 2017, this dataset was exhaustively analysed to understand the information content in the various data channels [42, 45] and it was used in three public machine learning challenges [43, 44, 46].

In this chapter, we use two subsets of the dataset called *Preview* and *Complete User 1 - Hips Phone*, as these are publicly available¹ and allow for comparison with other work. The preview version includes the data of 3 users, each recorded over 3 days and includes 59 h of annotated recordings (Fig. 10.1a indicates the recording distribution according to the activities). The complete user 1 - hip phone subset was recorded by phone in the trouser's pocket of the first user and consists of 391 h of annotated data

recorded over 7 months (Fig. 10.1b). We only use the data of user 2 and 3 from the preview, since the data of user 1 is included in the complete version.

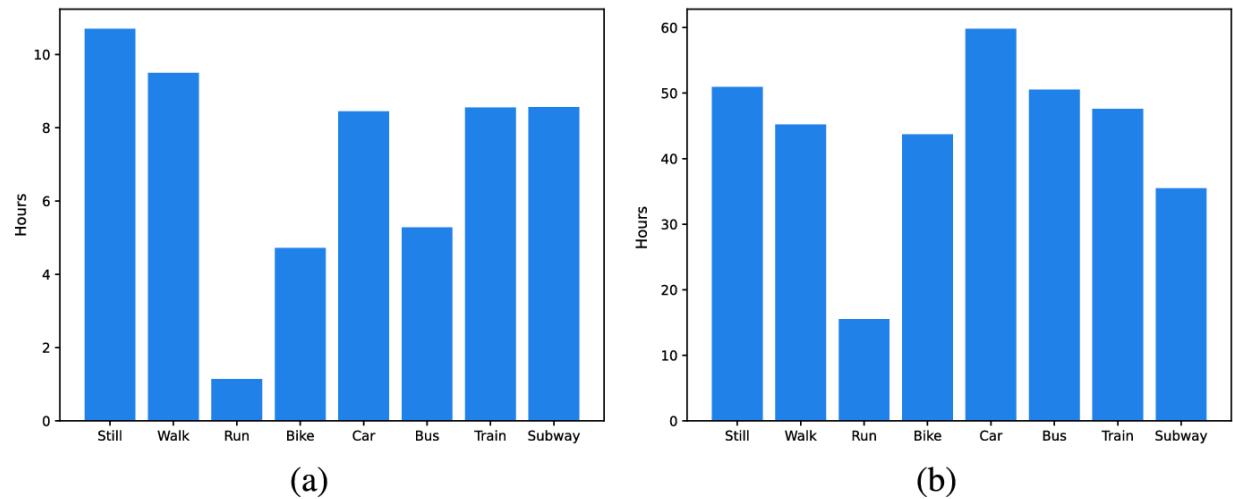


Fig. 10.1 Distribution of the number of recording hours according to the activities in the **a** SHL preview dataset **b** SHL Complete User 1 - Hips Phone dataset

10.3 State of the Art in Classifying the SHL Dataset

Our work utilises signal processing and unsupervised learning to improve the classification of modes of transportation and locomotion from mobile phone motion sensors. The SHL dataset has been used in three classification challenges tackling this very problem so far [43, 44, 46]. Therefore, it is well suited to evaluate our proposed approach and to contrast it with the other approaches, proposed as part of this challenge by the wider scientific community.

The submissions to the SHL challenge are divided into classical machine learning and deep learning methods, whereas there is no clear trend for either of them when it comes to the best submissions. The 2018 challenge aimed at user-specific activity recognition on 1 min long windows in random order. Gjoreski et al. applied an ensemble consisting of a deep Neural Network and classical models, such as Random Forest, SVM, and Gaussian Naïve Bayes, to solve this task [16]. Other high ranking solutions included Extreme Gradient Boosting (XGBoost) [23], a

Convolutional Neural Network [21], and a fully connected Neural Network with a Hidden Markov Model [48].

The 2019 challenge aimed at phone position-independent solutions, by providing 5 s windows with the test data recorded at one and the training data at the other 3 body locations. The highest ranking solution solved this task using a Random Forest model and Hidden Markov Model smoothing [24]. Feature selection was performed by ranking them based on the mutual information with the label and removing correlated ones based on the Pearson correlation coefficient. Again, the top submissions also included Convolutional Neural Networks [9] and ensembles of deep learning and traditional approaches [30].

Finally, the 2020 challenge aimed at user-independent activity recognition with an unknown phone position, again there were 5 s windows provided. This time the training data was available for all body locations while the test data was recorded at one unknown location. The most successful contribution was based on a Convolutional Neural Network with attention [54], followed by a Recurrent Neural Network [52], and a solution including clustering, XGBoost, and semi-supervised learning [26]. In the semi-supervised approach a classifier was trained on the labelled training data and used to predict labels on the unlabelled test data, if the prediction probability was over a certain threshold, the sample and the prediction were introduced into the training data to train other classifiers.

10.4 Background on Generative Adversarial Networks

GANs, as proposed by Goodfellow [17], are a machine learning concept, which typically involves two deep neural networks competing with each other in a minimax game as shown in Fig. 10.2. Some consider GANs to be special cases of artificial curiosity (1990) and closely related to predictability minimization (1991) as well [37].

The generator takes as input noise z , which is sampled from a Gaussian normal distribution, and feeds it forward to create a synthetic sample $z \in p_z(z)$ as output. These synthetic samples, together with real samples (also called training data), serve as input for the second network, the discriminator, which attempts to learn distinguishing between a real sample

x and a fake sample $G(z)$. The discriminator output is defined as 1 to indicate it predicts a real sample, and 0 to predict a fake sample.

The objective of the minimax game between the two networks is formulated as shown in Eq. 10.1. The equation calculates the summed cross-entropy (formulated as maximisation problem) on predicting the correct origin of each data sample x^2 and $G(z)^3$ by the discriminator. The real samples belong to the distribution described by P_{data} while the fake ones belong to P_r . The discriminator D aims to maximise the summed cross-entropy while the generator G attempts to minimise it.

The generator is trained to attempt to fool the discriminator by creating synthetic samples, which the discriminator mistakenly classifies as real ones. In effect, the generator shapes the input distribution so that the distribution of the output samples mimics that of the training dataset. By simultaneously training both networks as adversaries, their performance can be increased until—ideally—the generator creates samples, which perfectly resemble the characteristics and diversity of the training data. In which case the discriminator can do nothing but guessing.

$$\min_G \max_D \left(E_{x \sim P_{data}(x)}[\log(D(x))] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))] \right) \quad (10.1)$$

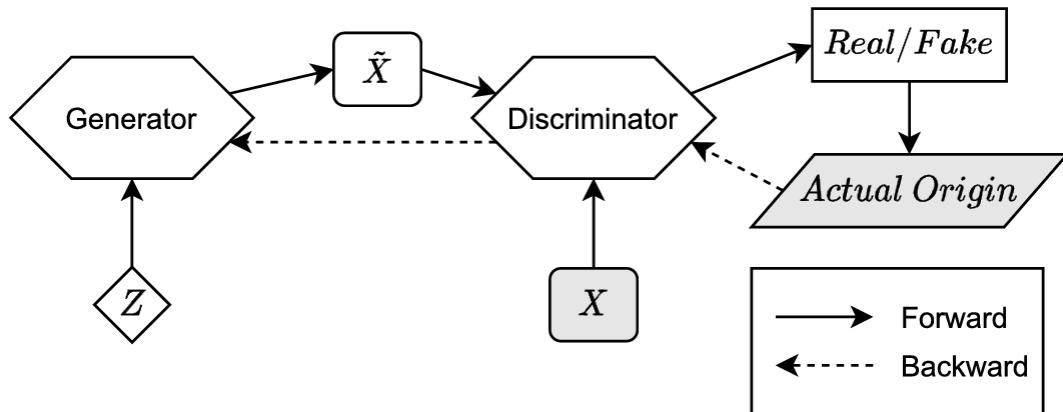


Fig. 10.2 A basic GAN architecture. The generator takes Gaussian noise z as input and creates synthetic data samples \hat{x} , which the discriminator has to identify as *fake* samples by distinguishing them from the real samples x . The discriminator serves as adversarial for the generator, which learns how to create samples to fool the discriminator into believing them to be *real*. Dashed backward lines represent target variables, which are compared against predicted ones to perform gradient descent

The loss of the discriminator is calculated based on its ability to correctly identify whether a data sample is *real* or *fake* as shown in Eq. 10.2. The discriminator is supposed to predict 1 (*real*) for a sample x and 0 (*fake*) for a synthetic sample $G(z)$. The logarithm function is used to strongly punish confident wrong predictions. As the discriminator prediction for a sample x gets closer to the correct label 1, the loss value approaches 0. If the discriminator wrongfully predicts the sample x as *fake*, the loss value will tend towards negative infinity the closer the prediction is to 0. The same goes for the outcome of predicting a label for synthetic samples $G(z)$, which is inverted by subtracting it from 1. The discriminator therefore aims to maximise this equation.

$$L^{(D)} = \max[\log(D(x)) + \log(1 - D(G(z)))] \quad (10.2)$$

The loss of the generator is based on its ability to fool the discriminator into believing that its generated samples are part of the *real* data. It is estimated by presenting one of its generated synthetic samples to the discriminator and calculating the logarithm of its output value (Eq. 10.3), similar to the loss calculation of the discriminator. However, since the generator wants the discriminator to fail, it attempts to minimise this equation.

$$\|f_1 \circ f_2\|_{\text{Lip}} \leq \|f_1\|_{\text{Lip}} \cdot \|f_2\|_{\text{Lip}} \quad (10.3)$$

So far, GANs have been primarily used for image generation [17, 25, 34, 36], image transformation [20, 47, 49], or image upscaling [28]. However, there have been attempts to utilise the potential of GANs in HAR. SA-GAN [38] is able to perform knowledge transfer in the domain of wearable sensor-based HAR. GANs have also been used in action prediction on partially observable videos [40].

Wang et al. proposed the first framework which applied GANs to generate HAR sensor data, using three different models with an individual generator-discriminator pair to generate and recognise a specific human daily activity [41]. The more recent SenseGAN [50] is a semi-supervised deep learning framework, which uses the same networks for multiple classes. It is able to utilise unlabelled data to improve the performance of a classifier by adding it into the GAN constellation.

While SenseGAN is operating on raw time series data, we propose a GAN model which operates on the extracted features from sensor signals.

We applied such a variation to the 2020 challenge [18], which to date is the only contribution utilising adversarial learning.

10.5 Methods

The preprocessing in our proposed framework consists of feature extraction from recorded acceleration signals, the selection of the most promising features, and scaling to normalize the data. Furthermore, undersampling is applied to tackle the issue of class imbalance present in the dataset. We use a classifier embedded in a GAN architecture to perform semi-supervised learning and apply Bayesian search to optimise the hyperparameters of the involved networks.

10.5.1 Feature Extraction

We use the three acceleration channels recorded by the smartphone carried in the hip pocket and calculate the magnitude of acceleration in order to negate the impact of the phone rotation. The resulting one-dimensional time series is divided via a non overlapping sliding window capturing 500 samples, which results in 5 s long windows. In the very few cases of these windows containing multiple activities, the label most present within the window is selected as the label for the entire window. None of the windows represents more than 2 classes. The resulting number of windows per user and class can be seen in Table 10.1.

Table 10.1 The number of windows for each class after applying a sliding window of 5s to the SHL dataset subsets *Complete User 1 - Hips Phone* for user 1 and *Preview* for user 2 and 3

	Still	Walk	Run	Bike	Car	Bus	Tram	Subway
User 1	36,671	32,550	11,171	31,466	43,067	36,373	34,271	25,557
User 2	2,430	2,969	183	1,639	3,087	843	1,614	2,122
User 3	3,535	2,261	372	768	1,008	993	2,748	2,220
Total	42,636	37,780	11,726	33,873	47,162	38,209	38,633	29,899

From each window an exhaustive list of 908 subband energy, time-domain quantile, other time-domain, and frequency-domain features are

extracted [45]. Table 10.2 shows the list of features which were computed on each window.

Table 10.2 These subband (Y), quantile (Q), time domain (Y), and frequency domain (\mathcal{X}) features were extracted. The total dimensionality and final number of selected features is given

Type	Features	Dimension	Selected
Y	Energy and energy ratio with scan 1 Hz and skip 0.5 Hz	198	19
	Energy and energy ratio with scan 2 Hz and 1 Hz	98	13
	Energy and energy ratio with scan 3 Hz and 1 Hz	96	12
	Energy and energy ratio with scan 4 Hz and 1 Hz	94	13
	Energy and energy ratio with scan 5 Hz and 1 Hz	92	14
	Energy and energy ratio with scan 10 Hz and 1 Hz	82	10
	Energy and energy ratio with scan 15 Hz and 1 Hz	72	5
	Energy and energy ratio with scan 20 Hz and 1 Hz	62	4
	Energy and energy ratio with scan 25 Hz and 1 Hz	52	5
Total		846	95
Q	Quartiles: [0, 5, 10, 25, 50, 75, 90, 95, 100]	9	9
	Pairwise quartile range for the 9 quartiles	36	36
Total		45	45
Y	Mean, standard deviation, energy	3	3
	Mean crossing rate	1	0
	Kurtosis and Skewness	2	1
	Highest auto correlation value and offset	2	1
\mathcal{X}	DC component of FFT	1	1
	Highest FFT value and frequency	2	1
	Ratio between the highest and the second FFT peaks	1	0
	Mean, standard deviation	2	2
	Kurtosis and skewness	2	0
	Energy	1	1
Total		17	10

10.5.2 Feature Scaling

Since many of the extracted features contain outliers, regular standardization does not seem to be the best choice. Instead we use a quantile transformation to map each feature to a uniform distribution between -1 and 1 . The goal is to match the output range produced by the generator's Tanh output layer.

10.5.3 Dealing with Class Imbalance

As shown in Table 10.1 and Fig. 10.1a the dataset shows imbalance, especially due to the low number of samples for class 3 "run". This is tackled by undersampling, whereas samples from classes are removed using NearMiss undersampling [31] until they reach the same number. For the hyperparameter tuning only a small proportion of the training data was used in order to compare as many constellations as possible, all classes of the labelled were sampled down to 512 data points. For feature selection and the final model training the majority classes were sampled down to match the number of windows in class 3. The class imbalance in the unlabelled data cannot be addressed, since there are no labels available. To preserve as much information as possible, the class imbalance in the validation data is not changed either, instead we use the unweighted mean of the F1 scores for each label independently as performance metric, to even out the significance of each class.

10.5.4 Feature Selection

Feature selection aims to identify a subset of the available features, which allows a classification model to achieve better prediction performance or to require less computational effort. This can be done, *inter alia*, by maximum relevance selection, whereas the features with the strongest correlations to the classification variable are selected. Maximum relevance - minimum redundancy [12] (mRMR) feature selection expands this process by considering the mutual information between features to reduce redundancies.

In the selection process, mRMR is defining the best feature at the given time based on their relevance to the classification variable and their distance to previously selected features. The iterative nature of the algorithm allows for the ranking of features by their score. This score can be calculated in various ways, whereas the F-test correlation quotient (FCQ) has been

identified as robust and accurate in different applications [53]. FCQ combines the F-test to measure relevance and the Pearson correlation among variables to measure redundancy by identifying the maximum quotient between both among all remaining features [1].

The feature selection is performed on the labelled data from the complete user 1. Downsampling is applied to all classes to match the minority class 3. We apply mRMR to rank each of the 908 scaled features based on their FCQ score and select the best n features for the classification pipeline. Earlier runs of the hyperparameter tuning (Sect. 10.5.6) did include the number of best features as parameter. They have established $U < \infty$ as final selection.

Quantile features seem to have a massive significance in this classification task, since the selection includes all quartiles and quartile ranges. In addition, mean, standard deviation, energy, skewness, and highest auto correlation value in the time domain, and mean, standard deviation, energy, DC component, and peak value in the frequency domain are selected. In addition, 95 out of 846 Subband energies and energy ratios and their centre frequency have been selected, as indicated in Table 10.2.

10.5.5 Proposed Generative Adversarial Network

Traditional GANs, as proposed by Goodfellow [17], are made up of two deep neural networks, which are competing against each other in a zero-sum game. The generator network G samples random noise z from a Gaussian distribution to predict synthetic samples as output. The discriminator network D takes given *real* samples or the synthetic *fake* samples created by the generator as input and predicts their authenticity.

This concept was further evolved by Yao et al., who added a classifier C into the contest [50]. The classifier takes a data point as input and predicts a label as output. In this setup the generator predicts data points based on noise z and a provided label, making it a conditional GAN [32]. The discriminator takes data-label pairs as input and predicts their authenticity.

The purpose of this architecture is to expand the training of the classifier, instead of only adjusting to the training data it also learns to predict labels for unlabelled samples in a way that fools the discriminator into believing them to be equal to the labelling in the original data. Whereas many GAN approaches focus on utilising the generative potential as

ultimate objective [5, 6, 17, 27, 34], the success of this concept is based on the performance of the classifier. The additional training on unlabelled data works better as long as the discriminator performs well. Since there is no direct connection between classifier and generator, the purpose of the generator is solely to prevent the discriminator from overfitting to the provided data.

10.5.5.1 Data

A sample x is a vector made of 150 features calculated on a window of 5 s of sensor data. Together with a corresponding label y , they form a data-label pair, shortened to *pair* from now on. The training is based on two given datasets: \mathcal{D}_A with corresponding one-hot labels Y and data \mathcal{D}_A with unknown labels. In addition, during training, additional data is created in two ways: (i) A synthetic sample \hat{x} aiming at resembling the distribution of a given class y is generated by the generator; (ii) a sample ϱ_w from the unlabelled dataset is associated with a label \tilde{y} predicted by the classifier, which attempts to trick the discriminator into believing that the prediction is a real label. Only the pairs (\mathcal{D}_A, Y) are considered *real* pairs. Unlabelled data with predicted labels $(\mathcal{D}_A, \tilde{Y})$ and data generated for given labels (\tilde{X}, Y) are considered *fake* pairs.

10.5.5.2 GAN Architecture

The GAN architecture we use in this paper can be seen in Fig. 10.3. A data sample x serves as input for the classifier, which predicts a corresponding label $P(x, y, s)$ in the form of a vector of normalised class probabilities. If the sample originates from the labelled subset \mathcal{D}_A , the loss is calculated based on the actual label y as shown in Eq. 10.4, whereas l_1 refers to the i th bit of the one-hot label.

$$L^{(C)} = \min \sum_i^8 [-t_i \cdot \log(C(x_L)_i)] \quad (10.4)$$

However, if the sample is part of \mathcal{D}_A , the sample x and the predicted label \tilde{y} are combined to serve as input to the discriminator network. In that case the higher the loss of the classifier, the better the ability of the discriminator to classify these samples as *fake* (Eq. 10.5).

$$L^{(C)} = \min[\log(1 - D(x_U, C(x_U)))] \quad (10.5)$$

The generator takes a random uniform distribution z and a one-hot label y as input and generates a data sample $P_G(\mathbf{X}|Y, S)$, which is supposed to make the discriminator believe that it is *real*. The length of the vector z is set to 100 (this value is not related to the number of features in x) as it is common practice [34]. The higher the loss of the generator, the better the ability of the discriminator to classify the generated samples as *fake* (Eq. 10.6).

$$L^{(G)} = \min[\log(1 - D(G(z, y), y))] \quad (10.6)$$

The discriminator is a binary classifier, which takes a pair as input and predicts whether it originates from the *real* dataset (ϱ_w, y) or if it is a *fake* pair. A data-label pair is considered *fake* if either the data sample is synthetic (\hat{x}, y) or the label was predicted (ϱ_w, \tilde{y}) by the classifier. The loss is calculated as shown in Eq. 10.7.

$$\begin{aligned} L^{(D)} = & \max[\log(1 - D(G(z, y), y)) \\ & + \log(1 - D(x_U, C(x_U))) \\ & + \log(D(x_L, y))] \end{aligned} \quad (10.7)$$

The loss functions shown in Eqs. 10.5–10.7 are an inverted variant of the binary cross-entropy function, turning the common minimisation problem into a maximisation one for the discriminator.

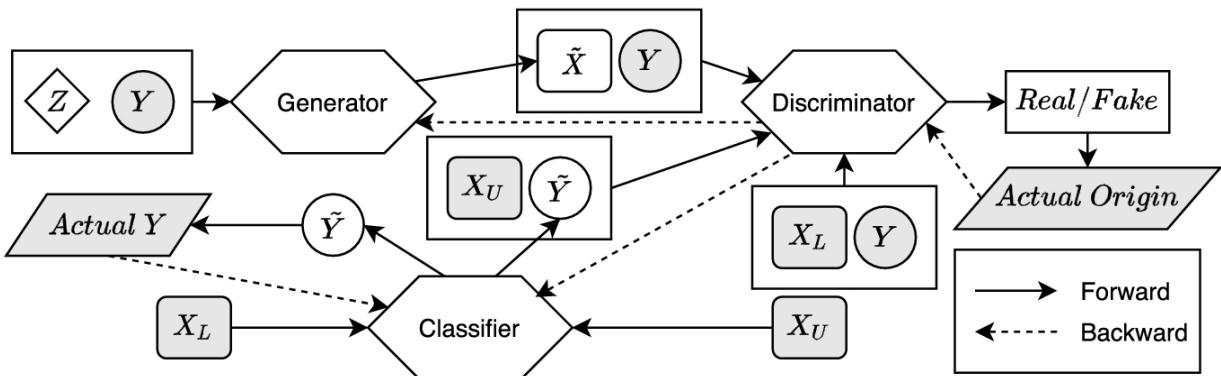


Fig. 10.3 The GAN architecture we use in this paper. The discriminator learns to distinguish between *real* data-label pairs originating from the training data and *fake* data, which consists of synthetic data generated by the generator with labels from the training dataset and unlabelled data with predicted labels. The discriminator serves as adversarial for the classifier, which learns how to predict labels to fool the

discriminator into believing them to be real. Dashed backward lines represent target variables, which are compared against predicted ones to perform gradient descent

10.5.5.3 GAN Networks

The classifier network $GAN-C$ consists of an input layer with a size of n (the number of selected features) and an output layer with a size of 8 (representing each of the eight classes). The predicted one-hot label is determined by a Gumbel-Softmax activation function [22], allowing random sampling based on the class probabilities and discrete output values. Discrete output values are crucial, since the discriminator could otherwise identify continuous labels and distinguish them from discrete *real* labels. The number and size of hidden layers, the activation functions of the input and hidden layers, and the temperature of the Gumbel-Softmax are determined by hyperparameter optimisation. The classifier F1 score W_1 is evaluated on the provided validation dataset \mathcal{W}_2 , by calculating the unweighted mean of the F1 score for each label.

The generator network $GAN-G$ consists of an input layer with a size of 108 (random vector with length of 100 plus eight one-hot labels) an output layer with a size of n (a sample \hat{x} consisting of n features). The output features are generated via the hyperbolic tangent function to resemble the normalized data distribution between -1 and 1 . The number and size of hidden layers and the activation functions of the input and hidden layers are determined by hyperparameter optimisation. The generator F1 score P_G is based on the ratio of generated samples which are classified as *real* by the discriminator.

The discriminator network $GAN-D$ consists of an input layer with a size of $n + 1$ (n features plus eight one-hot labels) and an output layer with a size of 1 (binary representation of *real* / *fake*). The output nodes are activated by a Sigmoid activation function. The number and size of hidden layers and the activation functions of the input and hidden layers are determined by hyperparameter optimisation. The discriminator F1 score P_G is based on the ratio of generated samples which are classified as *fake* (25%), the ratio of unlabelled samples with predicted labels classifies as *fake* (25%), and the ratio of pairs (x_L, y) classified as *real* (50%).

10.5.6 Hyperparameter Tuning

There are different methods available to select the hyperparameters of the GAN, such as manual selection, grid search, random search, Bayesian search, and gradient-based search. Manual selections based on experience or reference work do not guarantee the identification of an optimal set of parameters. Gradient based solutions are prone to getting stuck in local optima [8]. Bayesian model-based optimization has shown to be faster than grid search and more effective than random search when it comes to minimising validation error [4], due to its ability to probe solutions based on information from past evaluations.

We implemented Bayesian search in the form of Sequential Model-Based Optimization using the Tree-structured Parzen Estimator [3]. The following hyperparameters were evaluated for each network: The *Activation Function* for the input and hidden layers (the output activation function is predefined), the *Optimiser* used to train the network, the *Learning Rate*, the exponential decay rate for the 1st moment estimates *Beta1* (for Adam/AdamW) or *Momentum* (for SGD), the number of *Hidden Layers* and *Hidden Nodes* (number of nodes in each hidden layer). The exponential decay rate for the 2nd moment estimates was left at the recommended default value of 0.999. The choices for the activation functions are standard *ReLU*, *Sigmoid*, and *LeakyRelu* with a negative slope of 0.01 (*Leaky*) or 0.2 (*Leaky20*). For the classifier network there is an additional hyperparameter *Temperature*, which is used in the Gumbel-Softmax activation function of the output layer. The full list of hyperparameters can be found in Table 10.3.

Table 10.3 The name, type, scale, range, and final selection for all hyperparameters evaluated using Bayesian search. If the parameter was selected from a range of options, only the options are listed

Network	Parameter	Type	Scale	Min	Max	Final
Classifier	Activation Function	ReLU, Leaky, Leaky20, Sigmoid				Leaky20
	Optimiser	Adam, AdamW, SGD				AdamW
	Learning Rate	Float	Log Uniform	0.00001	0.1	0.000083
	Beta1/Momentum	Float	Log Uniform	0.001	0.99	0.866114
	Hidden Layers	Integer	Uniform	1	8	2
	Hidden Nodes	Integer	Log Uniform	16	4096	1790

Network	Parameter	Type	Scale	Min	Max	Final
	Temperature	Float	Log Uniform	0.01	10.0	2.833758
Discriminator	Activation Function	ReLU, Leaky, Leaky20, Sigmoid				Leaky
	Optimiser	Adam, AdamW, SGD				SGD
	Learning Rate	Float	Log Uniform	0.00001	0.1	0.024325
	Beta1/Momentum	Float	Log Uniform	0.001	0.99	0.043972
	Hidden Layers	Integer	Uniform	1	8	6
	Hidden Nodes	Integer	Log Uniform	16	4096	113
Generator	Activation Function	ReLU, Leaky, Leaky20, Sigmoid				ReLU
	Optimiser	Adam, AdamW, SGD				SGD
	Learning Rate	Float	Log Uniform	0.00001	0.1	0.006959
	Beta1/Momentum	Float	Log Uniform	0.001	0.99	0.620156
	Hidden Layers	Integer	Uniform	1	8	5
	Hidden Nodes	Integer	Log Uniform	16	4096	318

For each set of hyperparameters 5 GANs were trained over 100 epochs and evaluated to average the results. The data of user 1 reduced to 512 samples per class was used as \mathcal{D}_A and the combined data of user 2 and 3 as \mathcal{D}_A . The unlabelled data \mathcal{D}_A also served as validation data \mathcal{W}_2 . The performance P was estimated as shown in Eq. 10.8, based on the F1 score the classifier achieved on the validation data \mathcal{W}_1 ($w_1 = 1.0$) and the F1 scores the discriminator achieved on synthetic samples from the generator F_{DG} ($w_2 = 0.15$), on the real validation data F_{DV} ($w_2 = 0.15$), and on the validation data labelled by the classifier 10^{-4} ($w_2 = 0.15$).

$$P = w_1 \cdot F_C - w_2 \cdot (0.5 - F_{DG})^2 - w_3 \cdot (1.0 - F_{DV})^2 - w_4 \cdot (1.0 - F_{DC})^2 \quad (10.8)$$

The main focus is on the performance of the classifier on the validation data, since classifying these samples is the purpose of the GAN. The performances of the generator and discriminator are not per se relevant, since a strong generator could mean the discriminator is weak and is not able to improve the classifier. However, in order to find relevant results, we introduced the generator and discriminator performance with low weights. Ideally the generator and discriminator would reach an equilibrium, therefore deviation from equal performance is punished. The discriminators

ability to perform on the validation data with real and predicted labels has a slight impact as well.

10.6 Implementation

The project was developed using Python 3.9.2 using pytorch 1.8.0 for the networks and training, cudatoolkit 11.1.1 for GPU acceleration, numpy 1.20.1 and pandas 1.2.3 for data handling, imbalanced-learn 0.8.0 for NearMiss undersampling, hyperopt 0.2.5 for hyperparameter tuning, matplotlib 3.3.4 and seaborn 0.11.0 for plotting results, and scikit-learn 0.24.1 for baseline comparisons. The code for this project is available on github.⁴

The hyperparameter tuning and final evaluation were carried out on a machine equipped with 32GB RAM, an Intel Core i7-6700 @ 3.40GHz, and a NVIDIA GTX 2080 Super.

10.7 Results

The preview version and complete user 1 - hips phone of the SHL dataset are used to train and evaluate our models. We use only the data of the 3D acceleration sensor from the smartphone placed in the trouser pocket of the users. We select common out-of-the-box classifiers to compare with the results of our GAN classifier. Random Forests, Gaussian Naïve Bayes, and Support Vector Machine (SVM) are used as baseline in all evaluation scenarios. They are trained on \mathcal{D}_A and evaluated on \mathcal{W}_2 .

In the first evaluation, we only use the data available for user 1 and split it into two equally sized datasets for labelled training and validation data, whereas the latter again also serves as unlabelled data. In the second evaluation, we use cross user validation whereas the numerous samples available for user 1 are used as labelled training data. The classifying performance is measured on the validation data which includes the data of user 2 and 3 from the preview version. The same data from user 2 and 3 (without labels though) is used as unlabelled data. In both scenarios we want to estimate how well the GAN is able to unsupervised learn from the target data. Finally, we investigate how well the GAN learns on a small number of labelled and a high number of unlabelled samples from user 1

evaluated on user 2 and 3. This is a common scenario in the HAR context since gathering sensor data is several times complex than annotating it.

10.7.1 User Specific Evaluation

We run our GAN constellation ten times with randomly initialised weights over 300 epochs. Only the complete data of user 1 is used in this evaluation. We select 50% of it for \mathcal{D}_A and the other 50% for \mathcal{W}_2 and \mathcal{D}_A , whereas the distribution of classes in both selections is kept equal. We sampled each class in \mathcal{D}_A down to an equal number of samples. Small numbers of samples were tried up to 11,136 which is based on the number of available samples in class 3.

Table 10.4 The peak *F1* scores and accuracies *A* of our GAN classifier *GAN-C* compared to Random Forest, Gaussian Naïve Bayes, and SVM for different numbers of samples per class. The classifiers were evaluated on \mathcal{W}_2 . The complete data of user 1 is split into two equally sized sets \mathcal{D}_A and \mathcal{W}_2 with equal class distribution. \mathcal{W}_2 is also used as \mathcal{D}_A

Samples per class	GAN-C		Random forest		Naïve Bayes		SVM	
	F1	A	F1	A	F1	A	F1	A
512	0.536	0.537	0.550	0.533	0.432	0.415	0.364	0.385
1,024	0.547	0.548	0.502	0.488	0.366	0.377	0.369	0.371
4,096	0.609	0.608	0.663	0.636	0.408	0.412	0.455	0.432
11,136	0.685	0.684	0.728	0.703	0.466	0.485	0.535	0.538

The average peak performance of our GAN classifier can be seen in Table 10.4 together with three baseline comparisons. As expected, more training samples tend to improve the performance of all classifiers. The Random Forest classifier is outperforming our classifier in this scenario. Overall, SVM performs better than Naïve Bayes, except when there are only few samples present.

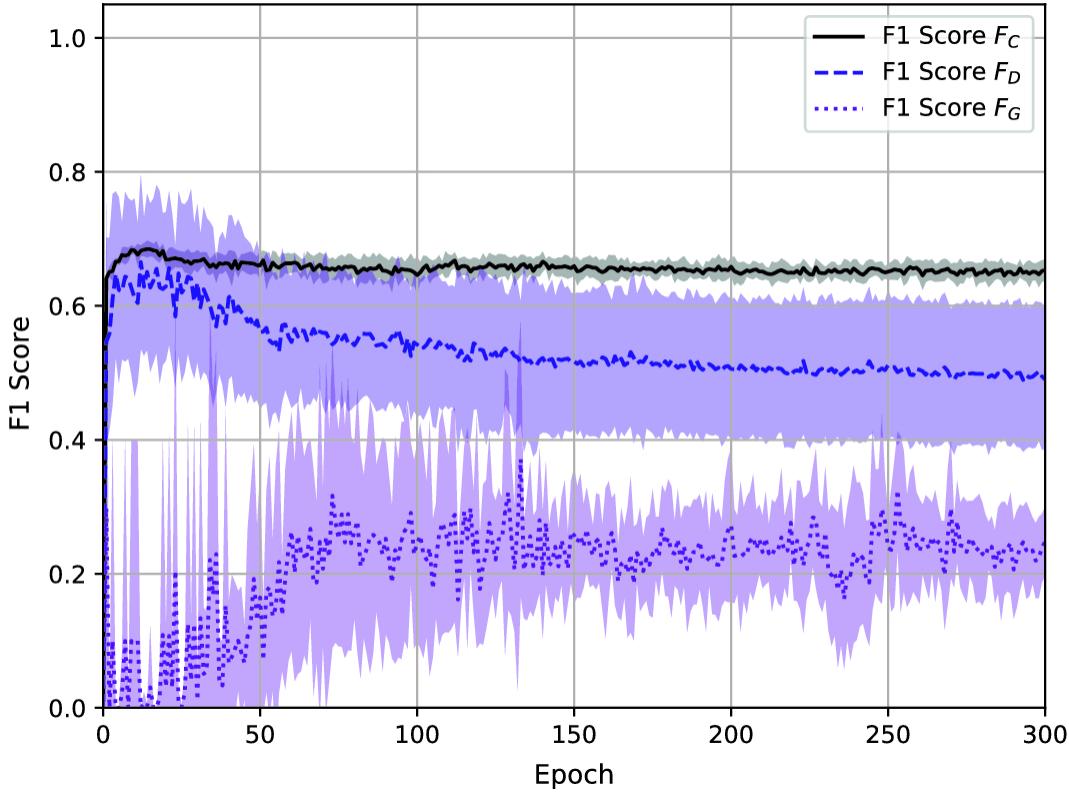


Fig. 10.4 The mean F1 scores of the three GAN networks C , D , and G with shaded standard deviation on \mathcal{W}_2 . The complete data of user 1 is split into two equally sized sets \mathcal{D}_A and \mathcal{W}_2 with equal class distribution. \mathcal{W}_2 is also used as \mathcal{D}_A

The evaluation focuses on the results obtained by having 11,136 samples per class in the labelled data. The performance development of the GAN networks can be seen in Fig. 10.4. It shows that the classification performance peaks rather sooner (after 10–25 epochs) than later. However, in this scenario the classifier performance stays steady after peaking. The generator again shows steady improvement over the initial epochs before stabilising. The discriminator performance is slowly decreasing, which does not correlate to its generator adversary.

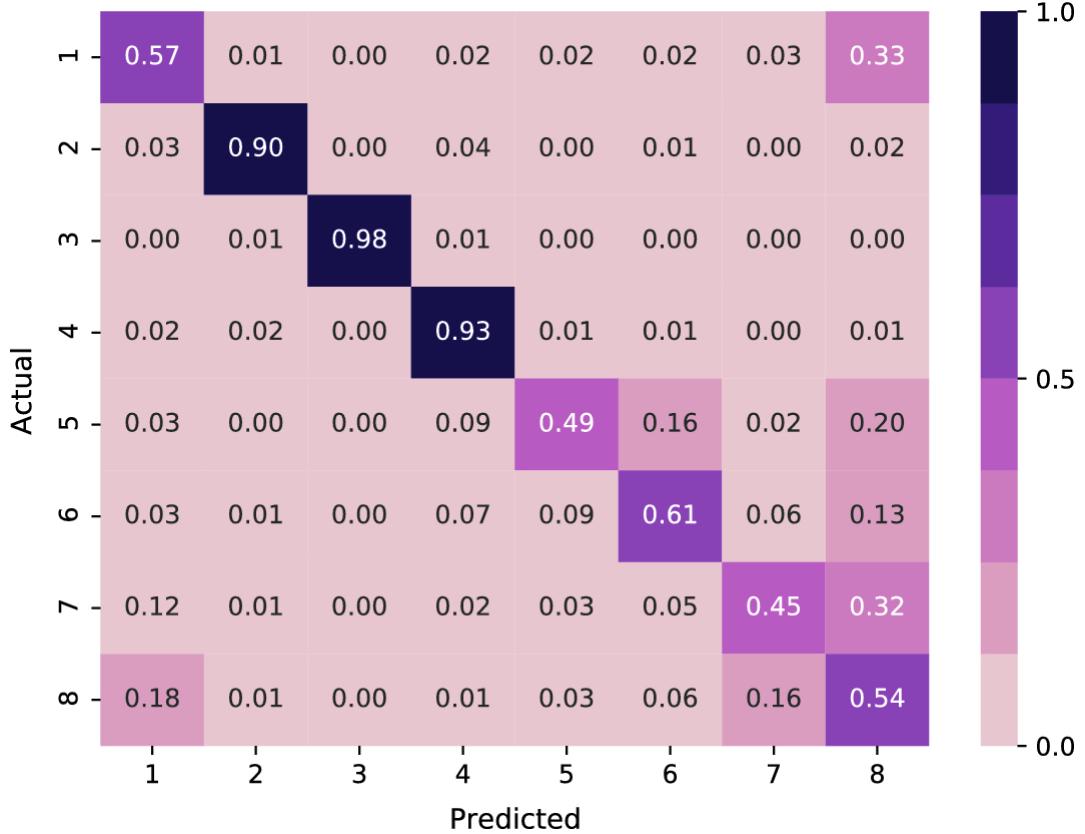


Fig. 10.5 The confusion matrix of the GAN classifiers on \mathcal{W}_2 averaged after 300 epochs of training (F1 score = 65.8%). The complete data of user 1 is split into two equally sized sets \mathcal{D}_A and \mathcal{W}_2 with equal class distribution. \mathcal{W}_2 is also used as \mathcal{D}_A

The confusion matrix of the GAN classifier on the validation data in Fig. 10.5 shows the expected confusion between classes 7 and 8, caused by the high similarity of acceleration signals between riding a train and riding a subway. The misclassification of class 1 “still” as train or subway is also expected as both don’t involve much movement of the user themselves. There is no strong preference to predict certain classes over others, the classifier seems to struggle most with predicting class 5 “car” though.

10.7.2 User Semi-independent Evaluation

We run our GAN constellation ten times with randomly initialised weights over 300 epochs. The complete data of user 1 is used as \mathcal{D}_A while the combined data of user 2 and 3 serve as \mathcal{W}_2 and \mathcal{D}_A . The approach is called “semi-independent” because although the labelled data is from a different user than the validation data, we still utilise data from the target users in the

form of unlabelled samples. We sampled each class in \mathcal{D}_A down to an equal number of samples. Small numbers of samples were tried up to 11,136 which is based on the number of available samples in class 3.

The average peak performance of our GAN classifier can be seen in Table 10.5 together with three baseline comparisons. As expected, more training samples improve the performance of all classifiers. Our classifier was able to consistently outperform the baseline solutions.

The evaluation focuses on the results obtained by having 11,136 samples per class in the labelled data. The performance development of the GAN networks can be seen in Fig. 10.6. It shows that the classification performance peaks rather sooner (after 10–25 epochs) than later, before gradually declining. The discriminator performance seems to be strongly linked to the performance of the classifier. The generator performance improves steadily for about 100 epochs before stabilising. The individual runs show that the relationship between discriminator and generator is highly unbalanced, in many epochs the generator either manages to trick the discriminator with every sample or with none of them. Often longer stretches of discriminator dominance are interrupted by single epochs completely dominated by the generator.

Table 10.5 The peak *F1* scores and accuracies *A* of our GAN classifier *GAN-C* compared to Random Forest, Gaussian Naïve Bayes, and SVM for different numbers of samples per class. The classifiers were evaluated on \mathcal{W}_2 . The complete data of user 1 served as \mathcal{D}_A while user 2 and 3 combined served as \mathcal{W}_2 and \mathcal{D}_A

Samples per class	GAN-C		Random forest		Naïve Bayes		SVM	
	F1	A	F1	A	F1	A	F1	A
512	0.378	0.387	0.318	0.318	0.255	0.254	0.094	0.173
1,024	0.402	0.408	0.314	0.375	0.219	0.227	0.127	0.174
4,096	0.437	0.443	0.335	0.317	0.160	0.224	0.240	0.268
11,136	0.475	0.479	0.440	0.491	0.191	0.249	0.274	0.319

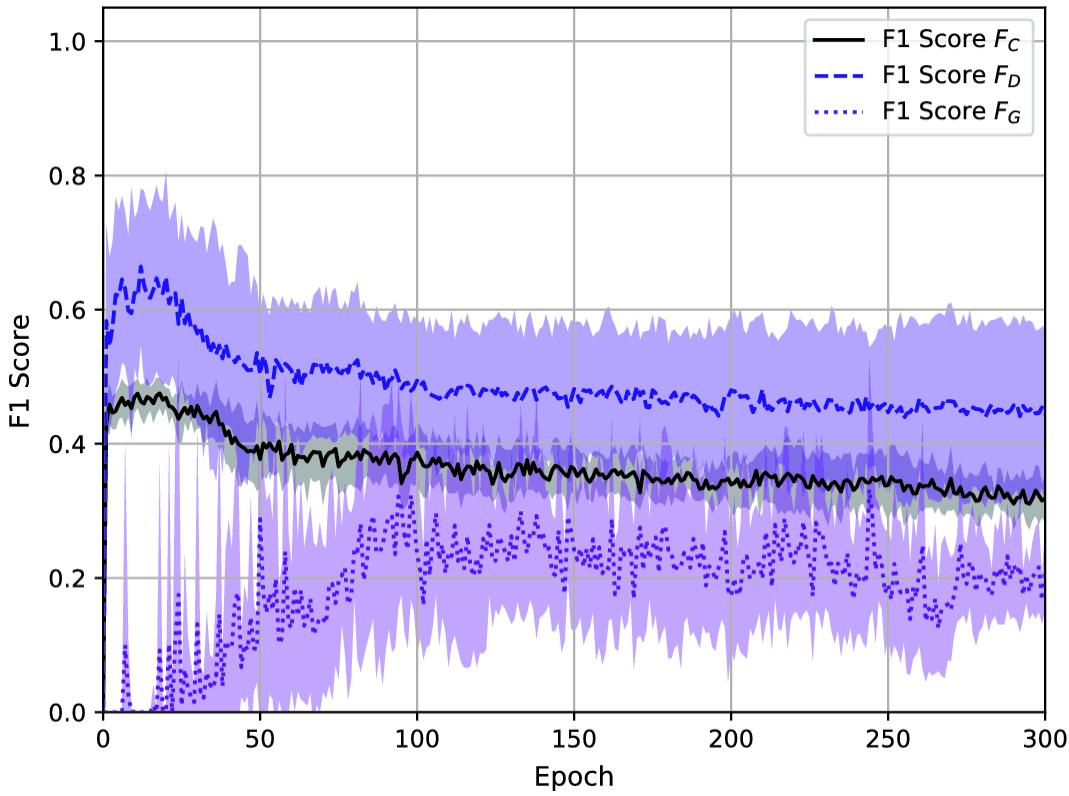


Fig. 10.6 The mean F1 scores of the three GAN networks C , D , and G with shaded standard deviation on \mathcal{W}_2 . The complete data of user 1 served as \mathcal{D}_A while user 2 and 3 combined served as \mathcal{W}_2 and \mathcal{D}_A

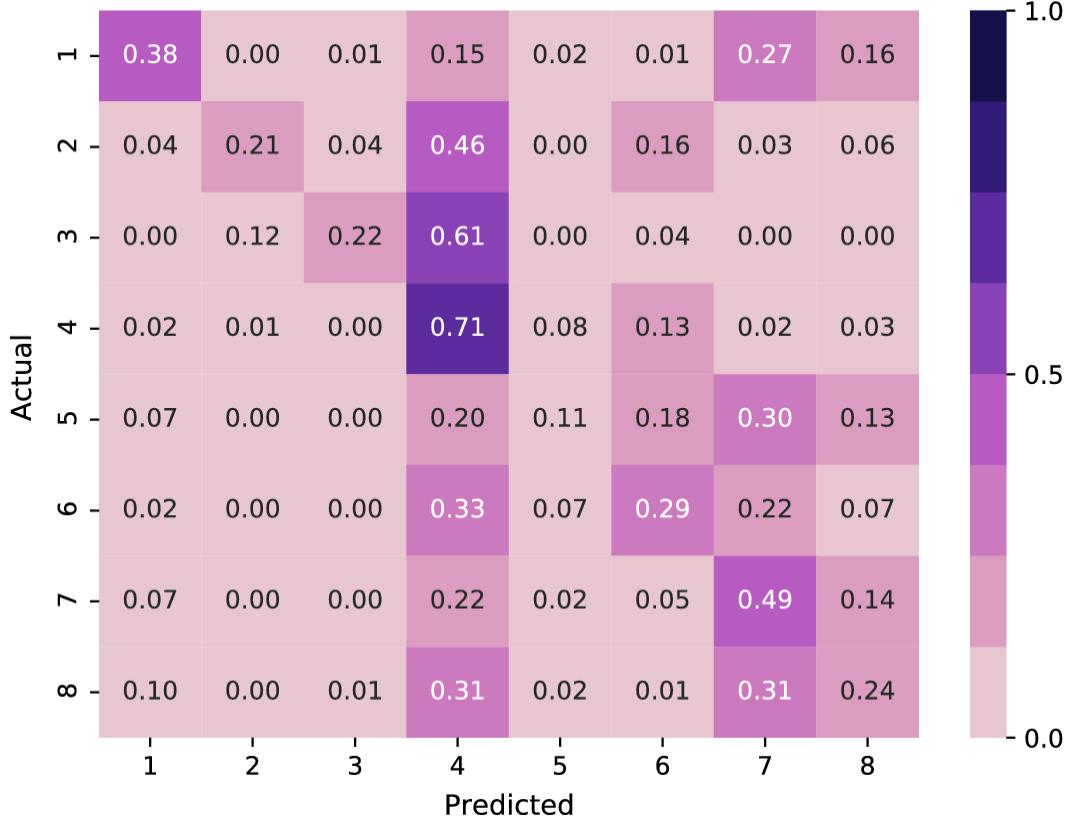


Fig. 10.7 The confusion matrix of the GAN classifiers on \mathcal{W}_2 averaged after 300 epochs of training (F1 score = 33.2%). The complete data of user 1 served as \mathcal{D}_A while user 2 and 3 combined served as \mathcal{W}_2 and \mathcal{D}_A

The confusion matrix of the GAN classifier on the validation data in Fig. 10.7 shows the expected confusion between classes 7 and 8 and the misclassification of class 1 “still” as train or subway. However, the high prediction probability for class 4 “bike” for samples across all transport modes is not that easy to explain and yet occurred in many evaluations during the development of the GAN. Interestingly, the classifier also struggled predicting class 5 like in the last scenario, although the evaluation was performed on data from other users this time.

10.7.3 User Independent Evaluation

In this scenario we used the complete data of user 1 as basis, sampled all classes down to 11136 samples and then selected 10% as \mathcal{D}_A and the remaining 90% as \mathcal{D}_A . The balance of classes was preserved within these

splits. Each split was used as labelled data once. User 2 and 3 were combined into \mathcal{W}_2 and the results averaged across the 10 runs.

Table 10.6 The peak $F1$ score and accuracy A of our GAN classifier *GAN-C* compared to Random Forest, Gaussian Naïve Bayes, and SVM. The classifiers were evaluated on \mathcal{W}_2 . The complete data of user 1 is split into ten equally sized sets with equal class distribution. Each of them is used as \mathcal{D}_A while the other nine make up \mathcal{D}_A . User 2 and 3 combined serve as \mathcal{W}_2

GAN-C		Random forest		Naïve Bayes		SVM	
F1	A	F1	A	F1	A	F1	A
0.490	0.492	0.353	0.399	0.189	0.249	0.196	0.235

The average peak performance of our GAN classifier can be seen in Table 10.6 together with two baseline comparisons. Compared to the previous scenarios it seems like the GAN is able to properly utilise the high number of unlabelled samples stemming from the same user as the labelled data.

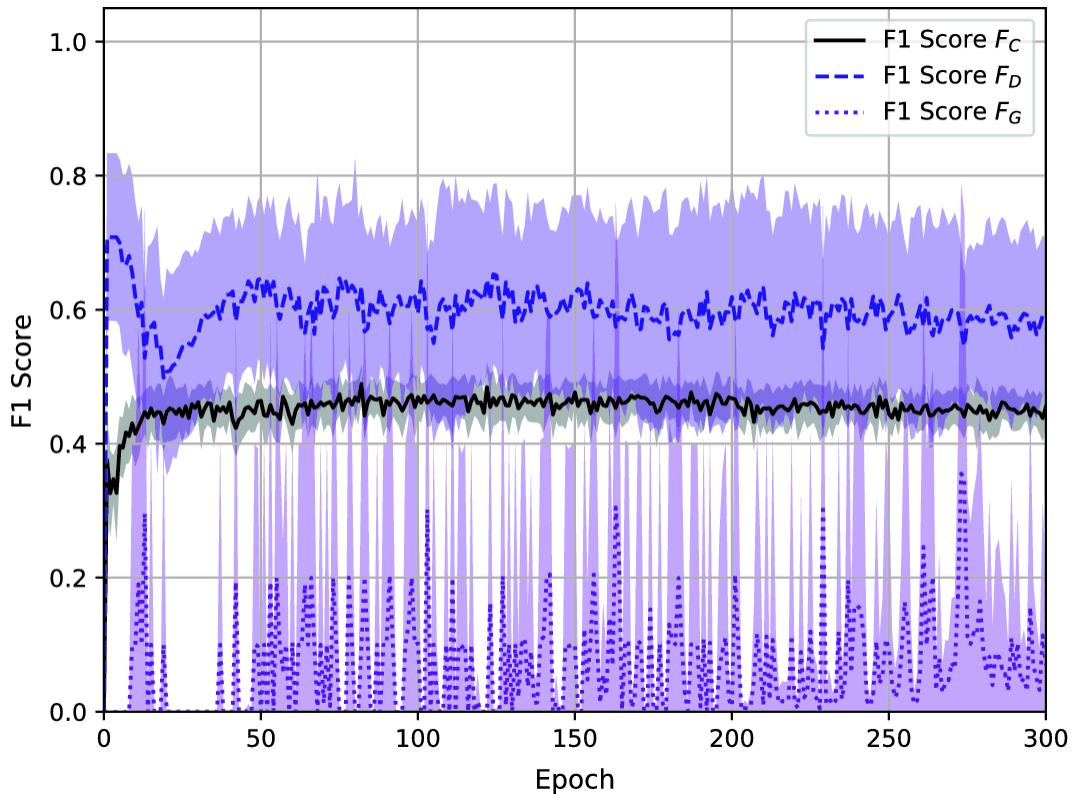


Fig. 10.8 The mean F1 scores of the three GAN networks C , D , and G with shaded standard deviation on \mathcal{W}_2 . The complete data of user 1 is split into ten equally sized sets with equal class distribution. Each of them is used as \mathcal{D}_A while the other nine make up \mathcal{D}_A . User 2 and 3 combined serve as \mathcal{W}_2

The performance development of the GAN networks can be seen in Fig. 10.8. There seems to be no clear peak in the performance of the GAN classifier, slight improvements until epoch 150 and a slight decline afterwards can be observed. The relationship between discriminator and generator appears to be cyclical, potentially indicating constant improvements in both networks.

The confusion matrix of the GAN classifier on the validation data in Fig. 10.9 shows the same expected confusions between classes 1, 7, and 8 as in the previous scenarios. It struggles with class 5 as in the previous scenarios and also struggles to predict class 3 “run”.

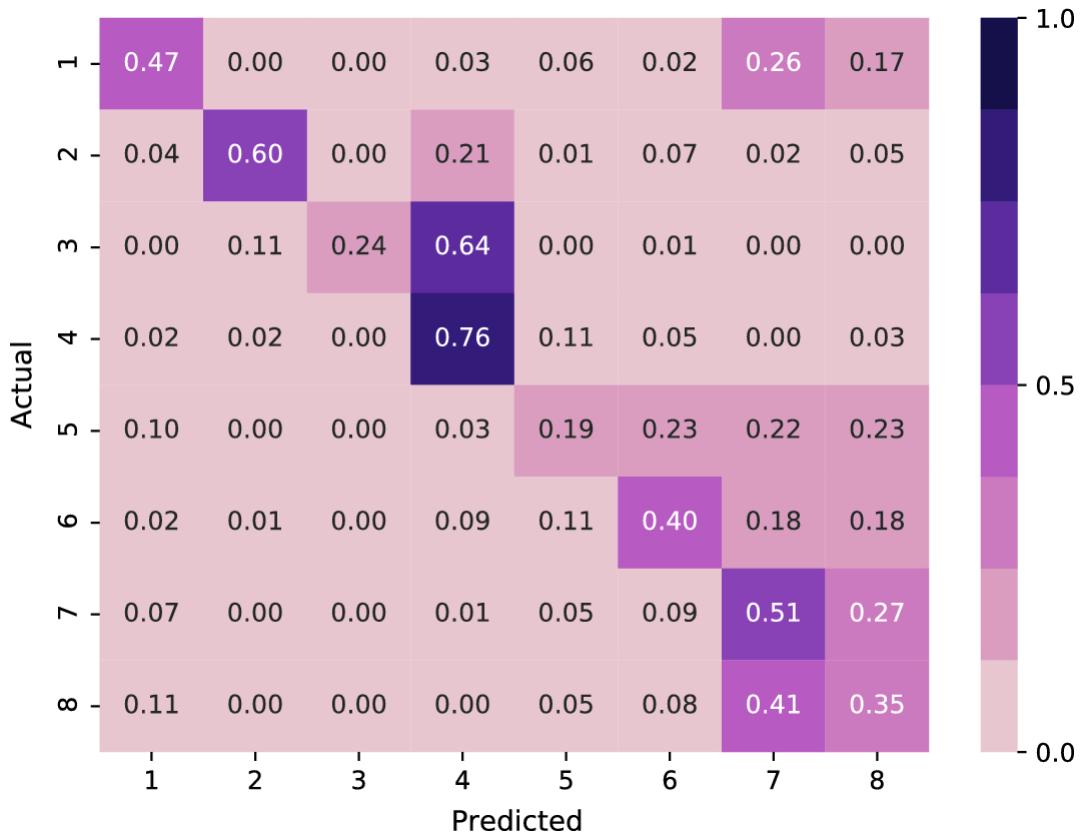


Fig. 10.9 The confusion matrix of the GAN classifiers on \mathcal{W}_2 averaged after 300 epochs of training (F1 score = 48.3%). The complete data of user 1 is split into ten

equally sized sets with equal class distribution. Each of them is used as \mathcal{D}_A while the other nine make up \mathcal{D}_A . User 2 and 3 combined serve as \mathcal{W}_2

10.8 Discussion

The results of three the evaluation scenarios indicate that our proposed semi-supervised learning approach works good when there is a high ratio of unlabelled data compared to labelled data present and the unlabelled data stems from the same source as the labelled data.

10.8.1 Performance Evaluation

In the user independent evaluation scenario we used half of the data of user 1 as labelled data and the other half as target data, which was also utilised via semi-supervised learning. The results show an early peak in classification accuracy followed by a phase of steadiness. The Random Forest baseline managed to outperform the GAN classifier despite being trained on only half the dataset, suggesting that the GAN structure did not benefit the training in this scenario. Since all samples were randomly shuffled it seems likely that the information present in the unlabelled samples were neglectable and trying to utilise them might have even harmed the classification performance since the discriminator was ultimately training the classifier to predict the validation data according to its own not-so-well informed expectations.

In the user semi-independent evaluation scenario we tried utilising the semi-supervised learning on the unlabelled target data of other users. The results in Fig. 10.7 show that the classification performance peaks very soon as well before the effects of overfitting set in. Nevertheless, in this scenario the GAN classifier managed to outperform the baseline, which suggests that either the utilisation of the unlabelled target data has shown to be successful or that the Random Forest classifier did not well generalise to other users.

The early generator performances in both scenarios suggest that, at the peak of the classifier, the adversarial components might not be developed enough yet to have a positive impact on the classifier. We implemented a pre-training period in which only discriminator and generator are involved. Like in a common conditional GAN setup, the discriminator is either given

real data or the synthetic samples of the generator. No unlabelled data was involved. The idea was to give the generator and discriminator a head start to be able to train the classifier before it starts overfitting to the labelled training data in later epochs. However, this did not seem to affect the performance of the classifier.

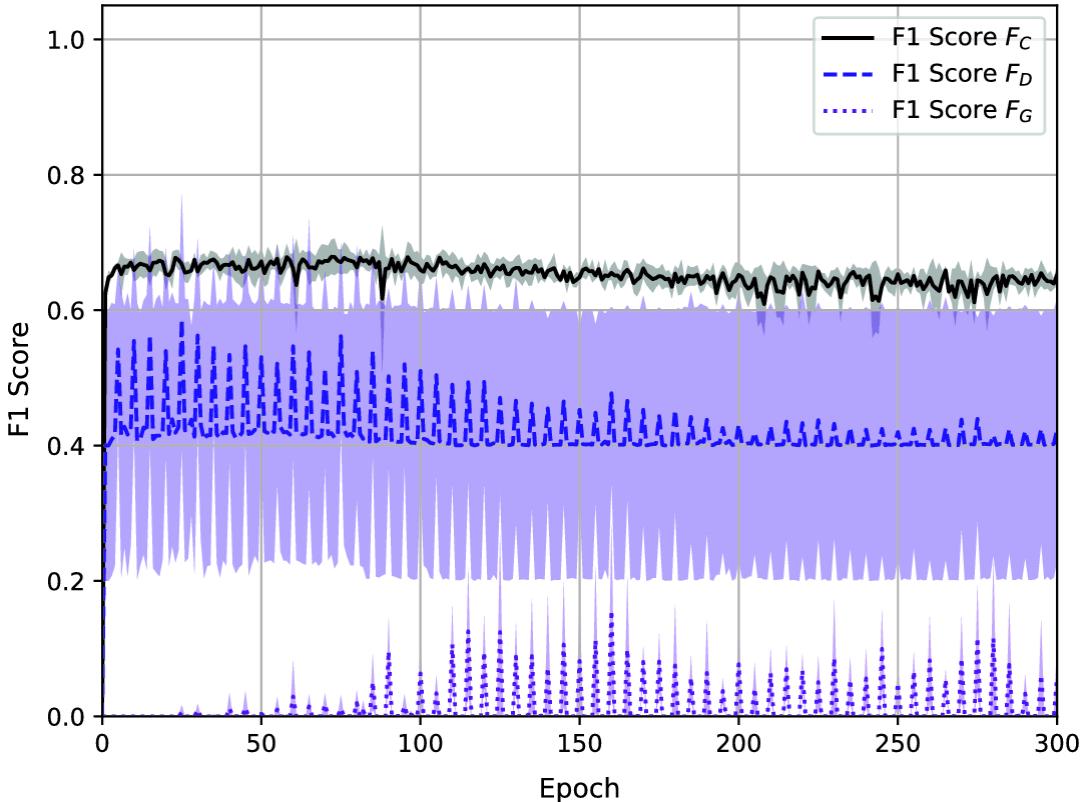


Fig. 10.10 The mean F1 scores of the three GAN networks C , D , and G with shaded standard deviation on \mathcal{W}_2 . The complete data of user 1 is split into two equally sized sets \mathcal{D}_A and \mathcal{W}_2 with equal class distribution. \mathcal{W}_2 is also used as \mathcal{D}_A . Fake positive pairs are given to the discriminator only every five epochs

During training, the classifier predicts unlabelled samples in an attempt to fool the discriminator into believing the sample-label pair are like the ones provided in the labelled data and the loss of the discriminator depends on the pairs predicted as *real*. Vice versa, the loss of the classifier depends on the samples predicted as *fake* by the discriminator. This brings the risk that the discriminator might learn to recognise the unlabelled data samples and to predict them to be *fake* no matter which class label is assigned to them. To prevent this, the discriminator was presented with fake positive

pairs, whereas it was trained to classify unlabelled samples with classifier predictions as *real* (note that the classifier weights were not updated in this process). We considered that this step might prevent the discriminator from giving accurate feedback to the classifier and hence tried to feed it less or none fake positive pairs. Figure 10.10 shows the performance development of the networks when the discriminator is given fake positive pairs only every five epochs. As a result the F1 score of the discriminator spikes every 5 epochs, indicating that the fake positive pairs have an important impact on the network. The comparison to the performance in Fig. 10.4 shows that the overall performance deteriorated, which was also observed with over evaluations reducing fake positive training.

In the final user independent evaluation scenario we used only 10% of user 1 as labelled data and the remaining 90% as unlabelled data. Since this scenario was evaluated on user 2 and 3 as well, we expected a worse performance compared to the cross user validation scenario, where all labelled data of user 1 was available. Yet, the results show that our GAN classifier does not only drastically outperform the baseline classifiers, it also performed better than in the previous scenario. This indicates that it could indeed be harmful to utilise the target data as unlabelled data. It also shows that the GAN architecture is able to utilise unlabelled data in order to improve classification performance.

10.8.2 Hyperparameter Tuning

Tuning the hyperparameter of a GAN offered unique challenges, since many of the loss calculations are based on the adversaries of the network. That means that a well performing generator is not necessarily able to produce relevant samples, it might as well just mean the discriminator is incapable to differentiate between the synthetic and the real ones. In a previous experiment [19], the hyperparameters of two networks were frozen while a grid search was used to optimise the other one, before moving on to the next and freezing the other two. This cycle has shown to be quite time- and computing power-consuming.

However, although being adversaries, the networks in the GAN are still used to achieve a common goal, which is better classification performance. However, the hyperparameters of the classifier are closer related to the classification performance than the ones of the generator, therefore we

included some aspects of the individual network performances. The multitude of training sources, i.e. classifier on real data, classifier on discriminator, discriminator on real data, discriminator on classifier, discriminator on generator, and generator on discriminator, lead to the problem of finding an effective way to quantify the performance of the GAN. The main metric is obviously the prediction performance of the classifier on the validation data, which is why we weighted this value the highest. We also added the performance of the discriminator on the real data and the classifier. We were cautious here, since rewarding good discriminator performance against the classifier would mean rewarding a weak classifier. We strongly penalised very low performing discriminators, but weighted the impact on the overall score comparable low. Since many runs showed a very unbalanced relationship between generator and discriminator we also penalised deviations from the optimal equilibrium, which we weighted higher than the previous discriminator performances since the generator - discriminator relationship has less impact on the classifier.

Evaluating the hyperparameters as one set using Gaussian Search allowed for one continuous process of optimisation, without intermediate adjustments. The evaluations, especially the cross-fold scenario, have shown that this method successfully estimated a well-performing set of hyperparameters.

10.9 Conclusion

We have successfully implemented a GAN framework in which the two adversarial relationships between discriminator and classifier as well as between discriminator and generator can be utilised to improve the classification performance when predicting locomotion modes based on acceleration sensors. When training on limited labelled data and evaluated on other users our classifier was able to achieve a F1 score of 49% which is a drastic improvement compared to the Random Forest baseline of 35.3%.

The results indicate that the framework performs well when there is a high ratio of unlabelled to labelled data present, especially if they are stemming from the same source. Using data from an unknown source has shown to decrease classification performance in the long run. The relationship between discriminator and classifier has proven to be beneficial

to the overall classification performance, while the benefits of the relationship between discriminator and generator is harder to verify. Ideally the generator would be able to prevent the discriminator from overfitting to the provided real data samples. Future work could investigate the ability of the generator to generate samples which match the given real data and if common issues like mode collapse occur.

Gaussian Search has proven to be able to find a suited set of hyperparameters for all networks in the GAN. However, the adversarial nature of their relationships made it tricky to accurately evaluate the whole set as one, but could be utilised in a coevolutionary optimisation approach [11] with three populations of network solutions, one for each network involved.

We have shown that our proposed GAN framework can utilise selected features instead of raw time series data to outperform a baseline given sufficient unlabelled data in addition to the labelled data of the same source.

References

1. Al-Ajlan, A., Allali, A.E.: (2018) Feature selection for gene prediction in metagenomic fragments. *BioData Min.* **11**(1) (2018). <https://doi.org/10.1186/s13040-018-0170-z>
2. Anagnostopoulou, E., Urbančič, J., Bothos, E., Magoutas, B., Bradesko, L., Schrammel, J., Mentzas, G.: From mobility patterns to behavioural change: leveraging travel behaviour and personality profiles to nudge for sustainable transportation. *J. Intell. Inf. Syst.* **54**(1), 157–178 (2018). <https://doi.org/10.1007/s10844-018-0528-1>
3. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, NIPS’11, pp. 2546–2554 (2011)
4. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, JMLR.org, ICML’13, pp. I–115–I-123 (2013)
5. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity

natural image synthesis (2019)

6. Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., Dafoe, A., Scharre, P., Zeitzoff, T., Filar, B., Anderson, H., Roff, H., Allen, G.C., Steinhhardt, J., Flynn, C., hÉigeartaigh, S.O., Beard, S., Belfield, H., Farquhar, S., Lyle, C., Crootof, R., Evans, O., Page, M., Bryson, J., Yampolskiy, R., Amodei, D.: The malicious use of artificial intelligence: Forecasting, prevention, and mitigation (2018)
7. Bulling, A., Blanke, U., Schiele, B.: A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.* **46**(3) (2014)
8. Cetin, B., Burdick, J., Barhen, J.: Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In: IEEE International Conference on Neural Networks, vol. 2, pp. 836–842 (1993). <https://doi.org/10.1109/ICNN.1993.298667>
9. Choi, J.H., Lee, J.S.: EmbraceNet for activity. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers (2019). <https://doi.org/10.1145/3341162.3344871>
10. Colpas, P.A., Vicario, E., De-La-Hoz-Franco, E., Pineres-Melo, M., Oviedo-Carrascal, A., Patara, F.: Unsupervised human activity recognition using the clustering approach: a review. *Sensors* **20**(9), 2702 (2020). <https://doi.org/10.3390/s20092702>
[Crossref]
11. de Jong, E.D., Stanley, K.O., Wiegand, R.P.: Introductory tutorial on coevolution. In: Proceedings of the 2007 GECCO Conference companion on Genetic and Evolutionary Computation - GECCO '07. ACM Press (2007). <https://doi.org/10.1145/1274000.1274108>
12. Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. *J. Bioinform. Comput. Biol.* **03**(02), 185–205 (2005). <https://doi.org/10.1142/s0219720005001004>
[Crossref]
13. Engelbrecht, J., Booysen, M.J., Rooyen, G.J., Bruwer, F.J.: Survey of smartphone-based sensing in vehicles for intelligent transportation system applications. *IET Intell. Transp. Syst.* **9**(10), 924–935 (2015). <https://doi.org/10.1049/iet-its.2014.0248>

[Crossref]

14. Gjoreski, H., Kaluza, B., Gams, M., Milic, R., Lustrek, M.: Context-based ensemble method for human energy expenditure estimation. *Appl. Soft Comput.* **37**, 96–970 (2015)
[Crossref]
15. Gjoreski, H., Ciliberto, M., Wang, L., Morales, F.J.O., Mekki, S., Valentin, S., Roggen, D.: The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access* **6**, 42592–42604 (2018). <https://doi.org/10.1109/access.2018.2858933>
[Crossref]
16. Gjoreski, M., Janko, V., Reščič, N., Mlakar, M., Luštrek, M., Bizjak, J., Slapničar, G., Marinko, M., Drobnič, V., Gams, M.: Applying multiple knowledge to sussex-huawei locomotion challenge. In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2018). <https://doi.org/10.1145/3267305.3267515>
17. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint (2014)
18. Günthermann, L., Simpson, I., Roggen, D.: Smartphone location identification and transport mode recognition using an ensemble of generative adversarial networks. In: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, Association for Computing Machinery, New York, NY, USA, UbiComp-ISWC '20, pp. 311–316 (2020). <https://doi.org/10.1145/3410530.3414353>
19. Günthermann, L., Philippides, A., Roggen, D.: Improving smartphone-based transport mode recognition using generative adversarial networks. In: Smart Innovation, Systems and Technologies, Springer Singapore, pp. 63–79 (2021). https://doi.org/10.1007/978-981-15-8944-7_5
20. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-Image Translation with Conditional Adversarial Networks (2016). [arXiv:1611.07004](https://arxiv.org/abs/1611.07004)
21. Ito, C., Cao, X., Shuzo, M., Maeda, E.: Application of CNN for human activity recognition with FFT spectrogram of acceleration and gyro sensors. In:

Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2018). <https://doi.org/10.1145/3267305.3267517>

22. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax (2016)
23. Janko, V., Reščič, N., Mlakar, M., Drobnič, V., Gams, M., Slapničar, G., Gjoreski, M., Bizjak, J., Marinko, M., Luštrek, M.: A new frontier for activity recognition. In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2018). <https://doi.org/10.1145/3267305.3267518>
24. Janko, V., Gjoreski, M., Masi, C.M.D., Reščič, N., Luštrek, M., Gams, M.: Cross-location transfer learning for the sussex-huawei locomotion recognition challenge. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, ACM (2019). <https://doi.org/10.1145/3341162.3344856>
25. Jolicoeur-Martineau, A.: The relativistic discriminator: a key element missing from standard gan. arXiv preprint (2018)
26. Kalabakov, S., Stankoski, S., Reščič, N., Kiprijanovska, I., Andova, A., Picard, C., Janko, V., Gjoreski, M., Luštrek, M.: Tackling the SHL challenge 2020 with person-specific classifiers and semi-supervised learning. In: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. ACM (2020). <https://doi.org/10.1145/3410530.3414848>
27. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation (2018)
28. Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint (2016)
29. Li, Y., Shi, D., Ding, B., Liu, D.: Unsupervised feature learning for human activity recognition using smartphone sensors. In: Mining Intelligence and Knowledge Exploration, Springer International Publishing, pp. 99–107 (2014). https://doi.org/10.1007/978-3-319-13817-6_11
- 30.

- Lu, H., Pinaroc, M., Lv, M., Sun, S., Han, H., Shah, R.C.: Locomotion recognition using XGBoost and neural network ensemble. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. ACM (2019). <https://doi.org/10.1145/3341162.3344870>
31. Mani, I., Zhang, I.: kNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceedings of Workshop on Learning from Imbalanced Datasets, ICML United States, vol. 126 (2003)
 32. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint (2014)
 33. Miu, T., Missier, P., Plötz, T.: Bootstrapping personalised human activity recognition models using online active learning. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 1138–1147 (2015). <https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.170>
 34. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint (2015)
 35. Roggen, D., Calatroni, A., Rossi, M., Holleczeck, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirk, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., Creatura, M., d R Millán, J.: Collecting complex activity datasets in highly rich networked sensor environments. In: 2010 Seventh International Conference on Networked Sensing Systems (INSS), pp. 233–240 (2010). <https://doi.org/10.1109/INSS.2010.5573462>
 36. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training gans. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, Curran Associates, Inc., pp. 2234–2242 (2016). <http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf>
 37. Schmidhuber, J.: Unsupervised minimax: adversarial curiosity, generative adversarial networks, and predictability minimization. CoRR abs/1906.04493 (2019). <http://arxiv.org/abs/1906.04493>
 38. Soleimani, E., Nazerfard, E.: Cross-Subject Transfer Learning in Human Activity

Recognition Systems using Generative Adversarial Networks. arXiv e-prints (2019)

39. Vaizman, Y., Ellis, K., Lanckriet, G.: Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Comput.* **16**(4), 62–74 (2017). <https://doi.org/10.1109/mprv.2017.3971131> [Crossref]
40. Wang, D., Yuan, Y., Wang, Q.: Early action prediction with generative adversarial networks. *IEEE Access* **7**, 35795–35804 (2019). <https://doi.org/10.1109/ACCESS.2019.2904857> [Crossref]
41. Wang, J., Chen, Y., Gu, Y., Xiao, Y., Pan, H.: SensoryGANs: an effective generative adversarial framework for sensor-based human activity recognition. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2018). <https://doi.org/10.1109/IJCNN.2018.8489106>
42. Wang, L., Gjoreski, H., Ciliberto, M., Mekki, S., Valentin, S., Roggen, D.: Benchmarking the SHL recognition challenge with classical and deep-learning pipelines. In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. ACM (2018). <https://doi.org/10.1145/3267305.3267531>
43. Wang, L., Gjoreskia, H., Murao, K., Okita, T., Roggen, D.: Summary of the sussex-huawei locomotion-transportation recognition challenge. In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. ACM (2018). <https://doi.org/10.1145/3267305.3267519>
44. Wang, L., Gjoreski, H., Ciliberto, M., Lago, P., Murao, K., Okita, T., Roggen, D.: Summary of the sussex-huawei locomotion-transportation recognition challenge 2019. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. ACM (2019). <https://doi.org/10.1145/3341162.3344872>
45. Wang, L., Gjoreski, H., Ciliberto, M., Mekki, S., Valentin, S., Roggen, D.: Enabling reproducible research in sensor-based transportation mode recognition with the sussex-huawei dataset. *IEEE Access* **7**, 10870–10891 (2019). <https://doi.org/10.1109/access.2019.2890793> [Crossref]

46. Wang, L., Gjoreski, H., Ciliberto, M., Lago, P., Murao, K., Okita, T., Roggen, D.: Summary of the sussex-huawei locomotion-transportation recognition challenge 2020. In: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. ACM (2020). <https://doi.org/10.1145/3410530.3414341>
47. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs (2017). [arXiv:1711.11585](https://arxiv.org/abs/1711.11585)
48. Widhalm, P., Leodolter, M., Brändle, N.: Top in the lab, flop in the field? In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. ACM (2018). <https://doi.org/10.1145/3267305.3267514>
49. Wu, H., Zheng, S., Zhang, J., Huang, K.: GP-GAN: Towards Realistic High-Resolution Image Blending (2017). [arXiv:1703.07195](https://arxiv.org/abs/1703.07195)
50. Yao, S., Zhao, Y., Shao, H., Zhang, C., Zhang, A., Hu, S., Liu, D., Liu, S., Su, L., Abdelzaher, T.: SenseGAN. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2(3), 1–21 (2018). <https://doi.org/10.1145/3264954>
[Crossref]
51. Zeng, M., Yu, T., Wang, X., Nguyen, L.T., Mengshoel, O.J., Lane, I.: Semi-supervised convolutional neural networks for human activity recognition. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 522–529 (2017). <https://doi.org/10.1109/BigData.2017.8257967>
52. Zhao, B., Li, S., Gao, Y.: IndRNN based long-term temporal recognition in the spatial and frequency domain. In: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. ACM (2020). <https://doi.org/10.1145/3410530.3414355>
53. Zhao, Z., Anand, R., Wang, M.: Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 442–452 (2019). <https://doi.org/10.1109/DSAA.2019.00059>
54. Zhu, Y., Luo, H., Chen, R., Zhao, F., Su, L.: DenseNetX and GRU for the sussex-huawei locomotion-transportation recognition challenge. In: Adjunct Proceedings

of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. ACM (2020). <https://doi.org/10.1145/3410530.3414349>

Footnotes

1 <http://www.shl-dataset.org/download/>

2 x is a sample originating from a given dataset.

3 $G(z)$ is a synthetic sample originating from the generator that ought to mimic the distribution of the given dataset.

4 http://github.com/STRCWearlab/GAN_SHL/releases/tag/v1.0.0

11. GANs for Molecule Generation in Drug Design and Discovery

Ziqiao Zhang¹✉, Fei Li¹✉, Jihong Guan²✉, Zhenzhou Kong¹✉,
Liming Shi¹✉ and Shuigeng Zhou³✉

- (1) School of Computer Science, Fudan University, 2005 Songhu Road, Shanghai, 200438, China
(2) Department of Computer Science and Technology, Tongji University, 4800 Caoan Road, Shanghai, 201804, China
(3) Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, 2005 Songhu Road, Shanghai, 200438, China

✉ Ziqiao Zhang

Email: zqzhang18@fudan.edu.cn

✉ Fei Li

Email: lif19@fudan.edu.cn

✉ Jihong Guan

Email: jhguan@tongji.edu.cn

✉ Zhenzhou Kong

Email: zzkong20@fudan.edu.cn

✉ Liming Shi

Email: lmshi@fudan.edu.cn

✉ Shuigeng Zhou (Corresponding author)

Email: sgzhou@fudan.edu.cn

Abstract

The goal of drug design and discovery is to find new molecules with desirable properties. To this end, molecule generation is usually employed to generate novel molecules to build a virtual molecule library for further screening. With the rapid development of deep generative modeling techniques, researchers are now applying deep generative models, particularly Generative Adversarial Networks (GANs), for molecule generation. In this chapter, we try to survey the applications of GANs for molecule generation in drug design and discovery, to give the readers a relatively comprehensive picture of this area. Firstly, we introduce the preliminary concepts of molecule generation, including molecule representation and evaluation metrics. Then, we review major molecule generation models based on GANs developed in recent years. In addition, comparison between GAN-based models and those based on other deep generative models is conducted, and the limitations of GAN-based models are discussed. Finally, challenges and future directions in this area are highlighted.

11.1 Introduction

Drug design and discovery is a challenging task, which is also very expensive and time-consuming. Usually, it takes an average of 13 years [1] and costs over one billion USD [2] to push the development of a new drug from the starting point to the market. The space of possible molecules is so enormous that the number of potential drug-like compounds is estimated to be between 10^{23} and 10^{24} [3]. Thus, it is extremely hard and inefficient for researchers to discover molecules with desirable properties in such a huge molecular space. Meanwhile, although molecules with similar structures are expected to have similar properties [4], small changes of their structures may also lead to large variation on their properties. If the computation of molecular properties is not accurate enough, incorrect calculations will lead to that most of the molecules with good properties (i.e., the properties of these molecules meet the requirements of the pharmacists) discovered in the computing step have to be discarded due to the failure to pass *in vitro* and *in vivo* experiments. The high failure rate, which approaches 90% for all

disease categories in clinical trials, results in substantial “waste” of money and time [5].

The goal of drug design and discovery is to find molecules with desirable properties, including bioactivity, physio-chemical properties, ADMET properties etc. Although the entire process of drug development is complicated, which consists of multiple steps including hit discovery, lead optimization, pre-clinical and clinical trials etc., the process before clinical trials can be abstractly summarized as a stack of iterative procedures with two phases: structure discovery (Phase 1) and test & analysis (Phase 2). In each iterative procedure (or layer), one or several molecular properties are considered. In the structure discovery phase (Phase 1), researchers navigate in the molecular space to find molecular structures that meet the requirements of the concerned properties. And in the test & analysis phase (Phase 2), the concerned properties of these molecules are tested and analyzed. If the property of a molecule does not meet the requirements, this molecule will be discarded, and then go back to Phase 1 to discover more other molecular structures. Otherwise, if the property meets the requirements, then this molecule is passed to the next step to test on more properties. The iterative process is illustrated in Fig. 11.1. In practice, the approaches to finding molecular structures include using virtual molecular libraries, using commercial molecular libraries, de novo drug design, and molecular structure optimization etc. And the ways to test the properties include molecular and cellular level trials, scoring functions, molecular docking and other *in vitro* or *in vivo* trials.

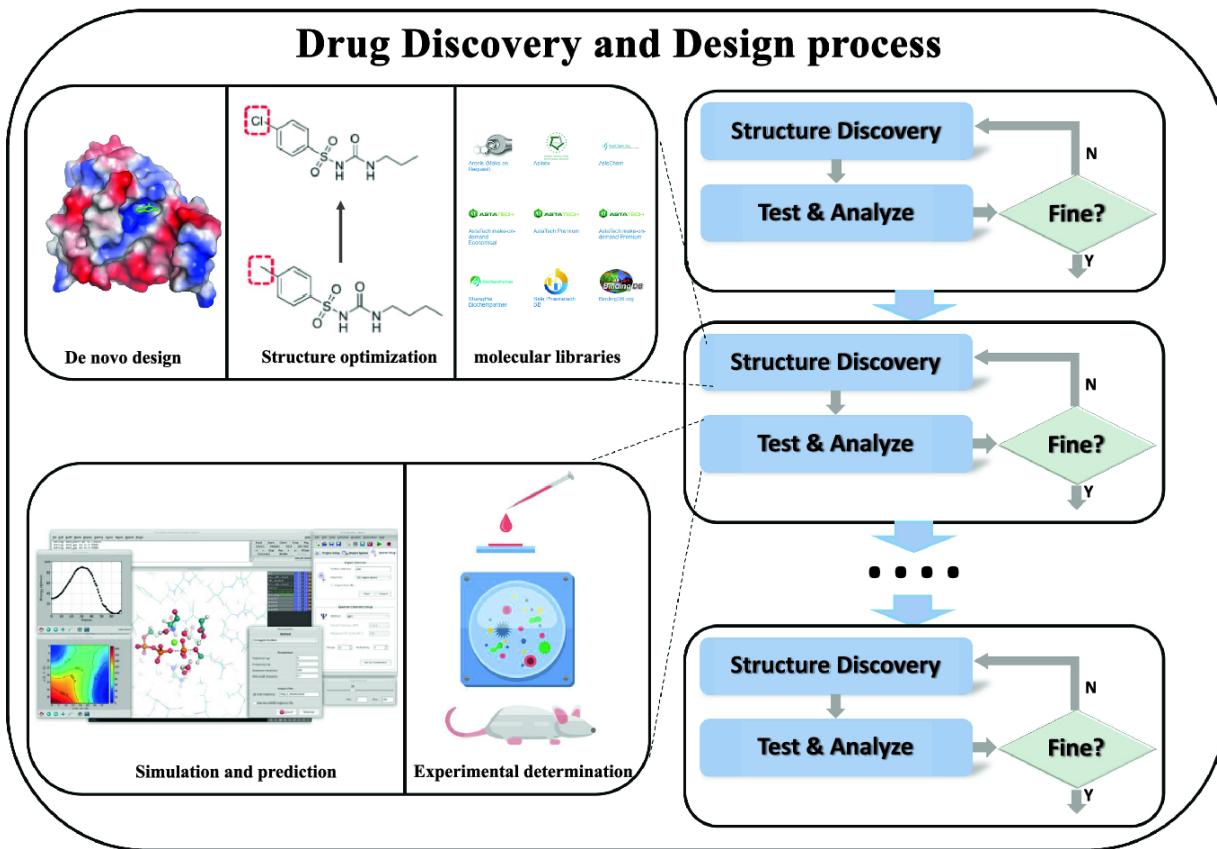


Fig. 11.1 The flowchart of drug design & discovery process, which can be summarized as a stack of iterative procedures. In each iterative procedure (or layer) of the stack, one or several properties are considered. Molecules that meet the requirements of at a certain layer will be passed to the next layer where other properties are tested. Each iterative procedure consists of two phases, which aim to discover structures from the molecular space (Phase 1) and verify their properties (Phase 2), respectively

Up to now, although drug design and discovery have made great progress, the high cost and long period of research & development still trouble researchers in this field, making it difficult for the latest pharmaceutical research results to be applied and promoted among patients. Recently, deep learning has achieved huge success in many fields, more and more researchers have tried to apply deep learning to assisting drug design and discovery. For example, impressive research achievements in molecular property predictions by deep learning methods have been reported [6–8]. And now it is a hot issue to use deep generative models to produce desired molecular structures [4].

In pharmaceutical research area, the commonly used means to find molecules include molecular library based methods, de novo drug design and structure optimization. The so-called de novo drug design relies on the knowledge of domain experts. According to the structure of the target protein, atoms or pharmacophores are assembled in sequence to generate new molecules that are able to interact with the protein. However, this scheme usually requires a large amount of domain knowledge, and is difficult to generate molecules automatically. For structure optimization methods, they start from a given structure and modify the structure to optimize the desired properties. Here, a large amount of domain-specific knowledge is still required, so that these methods encounter the same problem as de novo drug design methods. And although molecular library-based methods can provide millions of molecular structures to be quickly screened, the size of current libraries is still too small to cover the molecular space [9]. Besides, these molecular libraries usually are not biased to some specific molecular properties, so that screening in these libraries remains a blind search [10], and a large number of generated molecules will not meet the need and have to be discarded.

With the help of deep learning, a generative model can be built to serve as an agent to explore and navigate the molecular space. This agent is designed to automatically and quickly generate a large number of new molecular structures with desired properties. These molecules can be generated in a de novo way, or they can be analogues of an existing valid molecule of which the other properties need to be optimized. Using these generated new molecular structures, a molecular library with biased properties can be constructed. And further optimization and screening steps can be performed on this biased library, which leads to a more effective and purposeful exploration of the molecular space. In such a way, the cost and the period of research & development is expected to be reduced.

In the past decade, deep generative models have been applied to the molecule generation task, especially the Generative Adversarial Networks (GANs) [11]. In this chapter, we focus on the application of GANs for molecule generation in drug design and discovery. Our contributions of this chapter are as follows:

- We present in detail the state of the art of two major techniques in molecular generation: molecular representation and performance metrics.
- We give a comprehensive survey on GAN-based molecule generation models, to demonstrate the research progress and achievements in this area.
- We also introduce several molecule generation models based on other deep generative models. The advantages and disadvantages of GAN-based molecule generation models are discussed against these models.
- We highlight several future directions that are worthy of further study for the researchers in this area.

The rest of this chapter is arranged as follows: first, the preliminary concepts of molecule generation are introduced, including molecule representation and evaluation metrics. Then, molecule generation models based on GANs developed in recent years are reviewed. In addition, comparison between GAN based models and those based on other deep generative models, and the limitations of GAN based models are discussed. Finally, challenges and future directions in this area are pinpointed.

11.2 Preliminary Concepts

Molecule generation is a specific application of generative models to drug design and discovery, which requires considerable domain knowledge. For a generative task, there are two questions that should be firstly answered: how to represent the samples, and how to evaluate the quality of the generated samples. The answers to these two questions of molecule generation task are quite different from that in CV and NLP areas. So, in this section, we answer these two questions so that the readers can have a preliminary understanding of the molecule generation task.

11.2.1 Molecular Representation

Molecular representation is an important issue in deep generative model-based molecule generation. Before introducing specific methods of molecule generation, It should be pointed out that the meaning of the word *representation* in *molecular representation* in *AI-aided drug design and discovery* (AI-aided DDD)is different from that in *representation learning*

in deep learning field. The purpose of *representation learning* is to extract a refined representation from the raw data to better express the sample for subsequent tasks such as classification and regression. In deep learning context, deep neural networks are usually in charge of this task. The word *representation* in representation learning indicates the *latent vector* extracted by the neural networks.

However, an extracted latent vector does not aim to represent the input data. It is to represent the entity carried by the input data. For example, in *computer vision* (CV), the form of input data received by neural networks is image or video (sequence of images) represented as numerical matrix (or sequence of matrices). And these matrices are essentially the carrier of the entities in the image or video, i.e., the objects that we want to study in the image or video. And in *natural language processing* (NLP), the form of input data is text, i.e., string of natural language words. While the entities that the latent vectors represent are the concepts carried by the text string. In other words, the entities are implied in the carriers, which are served as the input data of the neural networks to extract latent vectors that could better express the entities. The relationship among these three concepts is shown in Fig. 11.2. In summary, *molecular representation* indicates the form of molecules used as the input to neural networks for extracting the latent vectors of molecules.

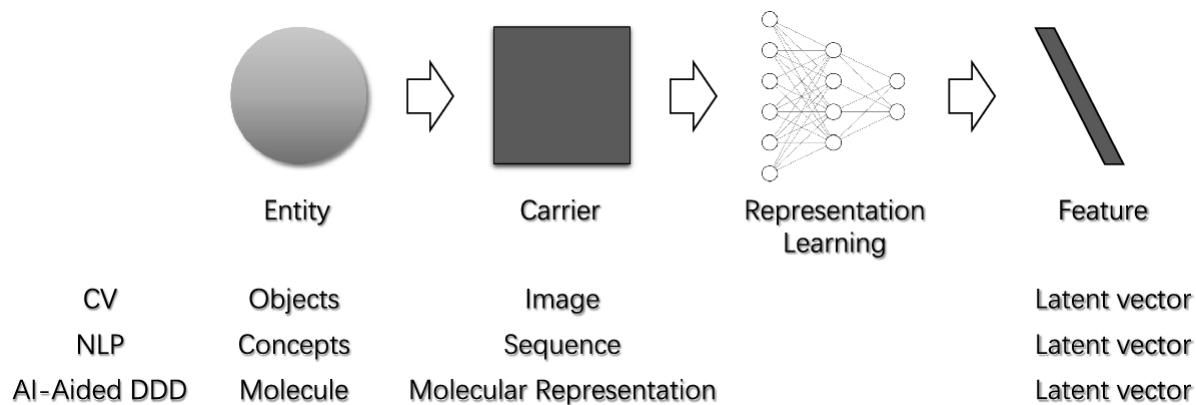


Fig. 11.2 An illustration of the relationship among the concepts: entity, carrier, representation learning and feature. The word *representation* in molecular representation acts as a carrier (or form) of molecules, which is different from that in representation learning

The reason why we study molecular representation is that molecules are special entities in nature. How to represent these entities such that they can be transformed into a numerical form acceptable to neural networks and the information of the entities can be preserved as much as possible is a crucial problem in machine learning based drug design and discovery. Currently, there is no any universally-agreed method for molecular representation.

Existing molecular representation methods can be divided into two types: representations based on 2D molecular graphs and representations based on 3D geometric structures.

11.2.1.1 2D Molecular Graph Based Molecular Representation

A 2D molecular graph refers to the graph structure used to represent the topological structure of a molecule, as shown in Fig. 11.3a. In this graph, the heavy atoms (hydrogen-depleted) in the molecule are nodes, and the chemical bonds between atoms are edges. Through 2D molecular graphs, the topological connections among atoms that constitute the molecule can be represented explicitly. Therefore, representations based on 2D molecular graphs are commonly used.

In this type of representations, the most widely used is *SMILES* [12]. Each SMILES string encodes a molecular graph into a sequence of ASCII characters via depth-first graph traversal [13]. Given a molecular graph of a molecule, to obtain its SMILES string, it starts from any atom with a direction selected as the main chain direction. The symbols of atoms are orderly written down during traversal. When a branch is encountered, parentheses are used to enclose the branch. And when a ring is encountered, a pair of numbers are used to indicate the connection point of the ring. As the starting atom and the main chain direction may be arbitrarily selected, the SMILES is a non-unique representation. However, with some specific rules to standardize the traversal process, a unique representation named *Canonical SMILES* can be derived by some cheminformatics toolkits such as RDKit [14].

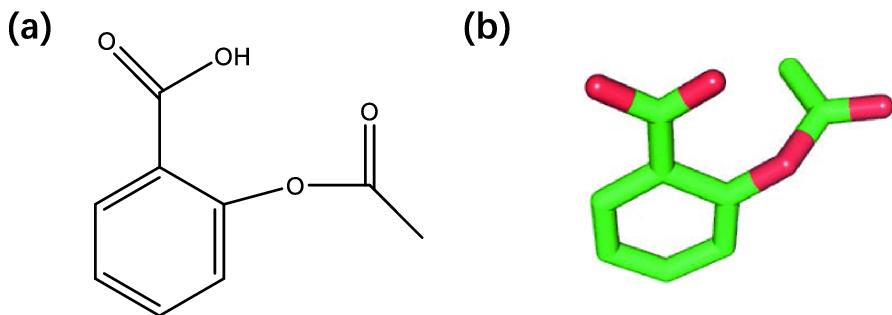


Fig. 11.3 **a** 2D molecular graph and **b** 3D geometric structure of an aspirin molecule. The SMILES representation of an aspirin is: OC(C1=C(OCC(=O)C=CC=C1)=O

The SMILES is in the form of string, so that it is also considered as one-dimensional representation by some researchers. However, the SMILES of a molecule is generated by its molecular graph, so that it is rationale to be categorized into 2D molecular graph based representations.

Another important representation based on 2D molecular graph is called *fingerprint* (FP). Fingerprints are vectors describing the molecular structure with each bit of the fingerprint describing the presence or absence of different molecular substructures, such as acidic groups or aromatic groups [15]. MACCS [16] is a kind of fingerprints with fixed length of 166 bits. In MACCS, each bit is associated with a specific structural pattern or question about structure [13]. While the length of Morgan FP [17] and Extended-Connectivity FP (ECFP) [18] are flexible. ECFP is a kind of circular fingerprints that is designed to encode the presence of substructures in a way invariant to atom-relabeling [19]. For each node in the molecular graph, the information of its neighbors and itself are aggregated and calculated by a fixed hash function. The results of these hashes are then treated as integer indices to be labeled by 1 in the fingerprints.

The above-mentioned representations are derived from 2D molecular graphs. However, these representations have some disadvantages:

- The molecule generation models based on SMILES representations typically suffer from the validity problem. As the grammar of SMILES is very restrictive, so that even when only one token of a generated sequence is wrong, the resulting sequence may not conform to the grammar of SMILES and thus is invalid.
- The complex grammar of SMILES, especially of the canonical SMILES makes the model more related to the grammar rules rather than to the

chemical structure of the underlying molecule.

- The SMILES strings of two similar molecules may be totally different, which makes it hard for a model to learn the similarity of molecules from their SMILES strings [20].
- As for fingerprints, the vital disadvantage is that they are not invertible, that is, for a generated sample represented by fingerprints, it is non-trivial to find the corresponding molecular structure.

Fortunately, in the past few years, with the development of Graph Neural Networks (GNNs) in deep learning communities, 2D molecular graphs can be directly processed by GNNs, which can overcome the disadvantages of SMILES and fingerprints.

11.2.1.2 3D Geometric Structure Based Molecule Representation

Although 2D molecular graphs can represent the topological connections between atoms that constitute a molecule, the information contained in this type of representations is not comprehensive. Obviously, 3D conformation information of a molecule, including the bond lengths, bond angles and dihedral angles, is ignored. However, such information often has important influence on molecular properties. In order to represent the structural information of a molecule more completely and to make models learn and understand the relationship between molecule structures and properties more accurately, molecular representations based on 3D geometric structure are proposed, as shown in Fig. 11.3b.

An intuitive approach to leveraging 3D coordinates is to use a 3D grid of voxels and specify the atom type contained within each voxel [13]. However, the vast majority of voxels do not have nuclear charge, thus the representations are extremely high-dimensional and sparse, which is not good for generative models [13].

To alleviate the sparsity of representations, spatial smoothing is considered, which leads to smooth voxel methods. One smoothing approach is to place spherical Gaussian or a set of decaying concentric waves around each atomic nuclei [13, 21]. An alternative is to use the van der Waals radius [22].

Although smooth voxel methods can alleviate the sparsity problem, there is another problem with 3D geometric representations: they are not

invariant to translation, rotation, reflection and permutation of atomic indexing [13]. Since the properties of a molecule are invariant to such transformations, deep learning models are expected to learn the invariance while extracting latent features. And if the representations are not invariant, data augmentation should be conducted. Furthermore, it is better for the molecular representations being unique, i.e., each molecular structure is associated with a single representation [13].

Current methods that generate features from 3D coordinates and are invariant to translation and rotation include wavelet transform invariants [23], solid harmonic wavelet scattering transforms [24] and tensor field networks [25]. However, all of these methods are prone to losing structure information and not invertible (i.e., a representation vector produced by the generative model cannot be unambiguously associated to a molecular structure). So, though 3D geometric representations are considered to contain more information of molecules and are theoretically the most proper representation methods, we will see in the next section that few generative models use these representations. Unique and invertible 3D geometric representations based on internal coordinates of molecules are expected in the future.

11.2.2 Evaluation Metrics

To evaluate the performance of molecule generation models, it is obvious that we should estimate how well the generated molecules are. Suppose that there is a set of N generated molecules by a certain model, the statistical information of this set can be used as the metrics for evaluating the model. As the goal of molecular generative models is to generate new molecular structures with desired properties, these statistical information can be categorized into two types: *general statistical metrics* and *chemical statistical metrics*.

11.2.2.1 General Statistical Metrics

General statistical metrics aim to evaluate the capacity of a model on generating new molecular structures, which include *validity*, *uniqueness* and *novelty* as follows:

- **Validity.** This metric assesses the ability of a model to generate molecules that are valid in chemical sense. Valid molecules are those that correspond to some realistic molecule (at least theoretically). The ratio of valid molecules over all of the generated molecules is calculated. Molecules with incorrect SMILES syntax or invalid valences are penalized [26].
- **Uniqueness.** This metric assesses the ability of a model to generate different molecules. A model should explore the entire chemical space to generate a vast number of unique molecules, rather than lazily generates duplicate ones to minimize the loss. The ratio of unique molecules over all of the valid molecules is calculated. The molecules generated duplicates will be penalized [26].
- **Novelty.** This metric evaluates the ability of a model to generate new molecules. A model should have a good coverage of the chemical space to generate molecules that are similar to, but not the same as the ones in the training set. The ratio of novel molecules over all of the valid molecules is calculated. The molecules already present in the training set will be penalized [26].

11.2.2.2 Chemical Statistical Metrics

Chemical statistical metrics aim to evaluate whether the generated molecules meet the requirements of desired properties. Following are some widely used chemical statistical metrics for molecule generation:

- **Bioactivities.** The most important property of a drug molecule is bioactivity, i.e., the binding affinity with a certain target protein. The discovery of drugs for a certain target usually starts from finding molecules that can bind with the protein. The structures of these molecules (also known as hits) should be further optimized to satisfy the other desired properties.
- **Drug-likeness.** Drug-likeness measures how likely a compound is to be a drug, which is a key property when selecting compounds during the early stages of drug discovery [27]. It is a complex metric that consists of multiple aspects of the physio-chemical properties of a molecule, including molecular mass, hydrophobicity, polarity etc. The *quantitative*

estimate of druglikeness (QED) score is to quantify the underlying distribution of desirability of these molecular properties [27].

- **Synthesizability.** It measures how easily a molecule can be synthesized. The structure of a designed molecule should be synthesizable so that it is possible to become a drug. The *synthetic accessibility score* (SAS) [28] is based on a combination of fragment contributions and a complexity penalty to measure the easiness of synthesis in a probabilistic way.
- **Solubility.** It measures whether a molecule is hydrophilic. The *log octanol-water partition coefficient* (logP) is defined as the logarithm of the ratio of the concentrations between two solvents of a solute [29] to measure solubility.

The four above-mentioned metrics are most widely used by molecule generation models. Besides, some specific molecular properties are also used to measure model performance, such as molecular weight, the number of aromatic rings, TPSA, the similarity to existing molecules, the presence or absence of some specific substructures etc.

The distance between the distribution of the chemical properties of the generated molecules and that of molecules in the training set is also considered to be useful to measure how well the model learns from the training set, and two metrics are proposed [26]:

- **Fréchet ChemNet distance (FCD).** Preuer et al. proposed the FCD metric [30]. A neural network called ChemNet is introduced and trained for predicting biological activities. The latent vector of the penultimate layer of the ChemNet is extracted, and the means and covariances of these activations are calculated for the reference set and the set of generated molecules, respectively. Then, FCD is calculated as the Fréchet distance for both pairs of values.
- **KL divergence.** As an important metric to measure how well a probability distribution approximates another distribution, the Kullback-Leibler (KL) divergence is leveraged [26]. In this benchmark, physico-chemical properties, including BertzCT, MolLogP, MolWT, TPSA, NumHAcceptors, NumHDonors, NumRotatableBonds, NumAliphaticRings and NumAromaticRings, are calculated by the RDKit tool for both the generated molecules and the training set. Then, the distributions of these descriptors are computed, and (\hat{x}, \hat{y}) is

computed for each descriptor i , which is further aggregated to a final score S :

$$S = \frac{1}{k} \sum_i^k \exp(-D_{\text{KL},i}). \quad (11.1)$$

Brown et al. suggested to categorize molecular generative models into two major types: distribution-learning and goal-directed [26]. The main difference between these two types of models is that whether there is an objective function to be optimized during training. They also suggested different metrics for evaluating these two types of models. For example, the metrics for distribution-learning models include only *validity*, *uniqueness*, *novelty*, *FCD* and *KL divergence*. And for goal-directed models, only the optimization degree of the objective function is concerned. However, as we will see later, in practice, the metrics used for distribution-learning models and those for goal-directed models are not exclusive. Due to that a number of molecular properties are required to meet, a widely used paradigm in goal-directed molecule generation models is that the models are trained to optimize one chemical property (say bioactivity), and learn the distribution of other chemical properties of the training set to keep the generated molecules “drug-like”. So the metrics to measure how well the models learn the distribution of the training set are also required to evaluate goal-directed models. And for distribution-learning models, the distributions of some specific chemical properties, such as QED and LogP, are also concerned. Therefore, in this chapter we categorize the metrics into general statistical metrics and chemical statistical metrics, rather than based on the types of the models.

11.3 GAN-based Molecule Generation Models

A *generative adversarial network* (GAN), which aims to generate new samples with the same distribution as the training set, typically consists of two components: a generator and a discriminator [11]. The generator G is used to produce fake samples based on a random vector z sampled from $p(z)$. The discriminator receives both real instances from the training set and fake instances from the generator, and estimates the probability that an

instance comes from the training data rather than G . During training, the generator G is trained to generate samples similar to the real ones to fool the discriminator D to maximize the probability that D makes error [10]. While the discriminator is trained to make better predictions to identify the fake ones. By this adversarial training, the generator will produce samples that the discriminator totally cannot make correct predictions. Thus, the generator is considered to have learnt the distribution.

The goal of GAN is to make the generator learn the distribution underlying the training set to generate samples similar to the real instances. It is a typical distribution-learning model. While, as is discussed above, the molecule generation task tries to generate molecules with biased properties, rather than only mimics the distribution of the training set. With some additional components, some GAN variants can be applied to generating biased samples, and thus become goal-directed models. So following the categories proposed by Brown et al. [26], we introduce the two types of GAN-based molecule generation models: goal-directed models and distribution learning models, in Sect. 11.3.1 and Sect. 11.3.2, respectively. The applications of these models are introduced in Sect. 11.3.3.

For comparison, we summarize the models introduced in this chapter in Table 11.1.

Table 11.1 A summary of major GAN-based molecule generation models

Category	Model	Representation	Structure	Comments
Goal-directed models	ORGAN [32]	SMILES	SeqGAN+RL	Adding objective to reward function
	ORGANIC [34]	SMILES	ORGAN	Applying ORGAN to inverse-design chemistry
	RANC [35]	SMILES	ORGAN+DNC	Using DNC as a generator
	ATNC [37]	SMILES	RANC+AT	Screening generated samples by AT module
	MolGAN [38]	Graph	GAN+RL	Generating molecule graph directly
	GA-GAN [42]	Graph	MolGAN+GA	Training MolGAN by Genetic Algorithm

Category	Model	Representation	Structure	Comments
	DiPol-GAN [43]	Graph	MolGAN+DIFFPOOL	Using differentiable-pooling to improve the discriminator
Distribution-learning models	LatentGAN [46]	SMILES	GAN+AE	Adversarial training on latent space rather than data space
	Hong et al.'s model [47]	SMILES	ARAE	Minimizing Mutual Information between latent vector and condition
	Bian et al.'s model [49]	AtomPair	dcGAN	Introducing deep Convolutional GANs to molecule generation area
	Kadurin et al.'s model [10]	MACCS	AAE	Using AAE for molecule generation firstly
	druGAN [51]	MACCS	AAE	Using a discriminator power to balance the training of G and D
	ECAAE [15]	SMILES	SAAE	Using disentanglement for conditional generating
	BIAAE [52]	SMILES	AAE	Learning joint distribution rather than conditional distribution
	CCM-AAE [54]	Graph	AAE	Forcing the latent space on Constant-Curvature Manifolds
	Mol-CycleGAN [55]	Graph	CycleGAN	Using CycleGAN for molecule optimization
	SMILES-MaskGAN [57]	SMILES	GAN+RL	Introducing Masked-Language task to molecule generation and optimization
	Gene-cGAN [58]	MACCS	conditional GANs	Combining 2 conditional GANs with gene expression information

Category	Model	Representation	Structure	Comments
	ASYNT-GAN [60]	3D point cloud	GAN+AT	Leveraging 3D geometric information to generate molecules
	QGAN-HG [61]	Graph	MolGAN+quantum circuit	Using quantum model for molecule generation firstly

11.3.1 Goal-Directed Models

11.3.1.1 SeqGAN

The essential feature of GANs is adversarial training. The network architectures of the generator and the discriminator are not restricted. So we can choose different networks to implement the generator and the discriminator based on the types of samples to be generated. At the beginning, GAN-based molecule generation models use SMILES [12] as molecular representations, where each molecule is represented as a string with specific grammar. Thus, many advanced techniques in NLP can be applied, such as recurrent neural networks (RNNs), long-short term memory (LSTM) and gated recurrent unit (GRU) etc.

GAN was originally proposed to produce samples with continuous values, such as images represented by matrixes. However, when dealing with sequence data such as SMILES, it may face some problems. Yu et al.. [31] pointed out that, when the generated samples are discrete, the loss gradient of the discriminator cannot be propagated backward to the generator to update its parameters for generating better samples. The discriminator can only assess the entire sequence to make predictions and calculate the loss. For a generated partial sequence, it is non-trivial to balance the current score of the partial sequence and the future score of the entire sequence. To address these two issues, Yu et al. proposed a model named SeqGAN, i.e., GAN for sequence generation [31]. In SeqGAN, the sequence generation procedure is considered as a sequential decision making process. The generator W_1 , implemented by a RNN with LSTM units, acts as the agent of reinforcement learning (RL). The state s of RL corresponds to D_X , i.e., the tokens generated so far. And the action a is the token to be generated in the next step, denoted as y_{t+1} . Based on this setting, RNN is used as the generator (policy) of which the goal is to

generate a sequence from the start state q_0 to maximize the expected end reward W_1 :

$$J(\theta) = \mathbb{E} [R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) \quad (11.2)$$

Then, to estimate the action-value function $Q_{D_\phi}^{G_\theta}(s, a)$, REINFORCE algorithm is used. Monte Carlo search is utilized to generate reward to an incomplete sequence. The $Q_{D_\phi}^{G_\theta}(s, a)$ is calculated as follows:

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N} \sum_{n=1}^N D_\phi(Y_{1:T}^n), Y_{1:T}^n \in \text{MC}^{G_\beta}(Y_{1:t}; N), t < T \\ D_\phi(Y_{1:t}), t = T \end{cases} \quad (11.3)$$

where MC indicates the Monte Carlo search operation and G_β is the roll-out policy parameterized by β . $D_\phi(Y_{1:t})$ is the predictive probability that D_X is real, obtained from the discriminator D_ϕ . The structure of SeqGAN is shown in Fig. 11.4.

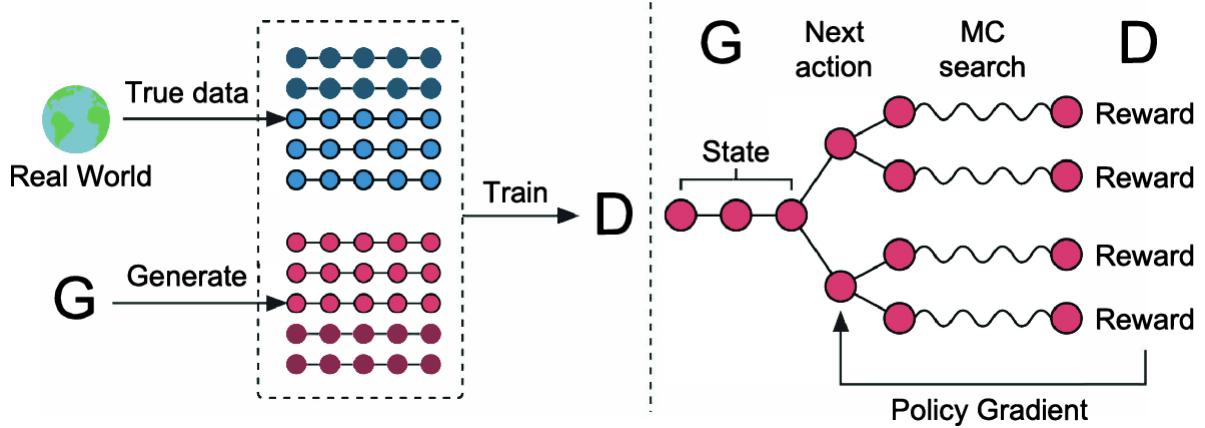


Fig. 11.4 The structure of SeqGAN [31]

11.3.1.2 ORGAN

SeqGAN model makes it possible to apply GANs on the problem of molecule generation. Based on SeqGAN, Guimaraes et al. proposed Objective-Reinforced Generative Adversarial Network (ORGAN) [32]. In this model, the reward of a generated sequence sample is not only the output of the discriminator, but also including domain-specific objectives. Thus, the $Q(s, a)$ in ORGAN is calculated as following:

$$Q(Y_{1:t-1}, y_t) = \begin{cases} \frac{1}{N} \sum_{n=1}^N R(Y_{1:T}^n), Y_{1:T}^n \in \text{MC}^{G_\beta}(Y_{1:t}; N), t < T \\ R(Y_{1:t}), t = T \end{cases} \quad (11.4)$$

where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a linear combination of D_ϕ and the objective function G_1 , parameterized by λ :

$$R(Y_{1:T}) = \lambda \cdot D_\phi(Y_{1:T}) + (1 - \lambda) \cdot O_i(Y_{1:T}) \quad (11.5)$$

In this way, the desired property of the generated molecules, such as drug-likeness, solubility, bioactivity, can be optimized during training. Figure 11.5 shows the structure of ORGAN model.

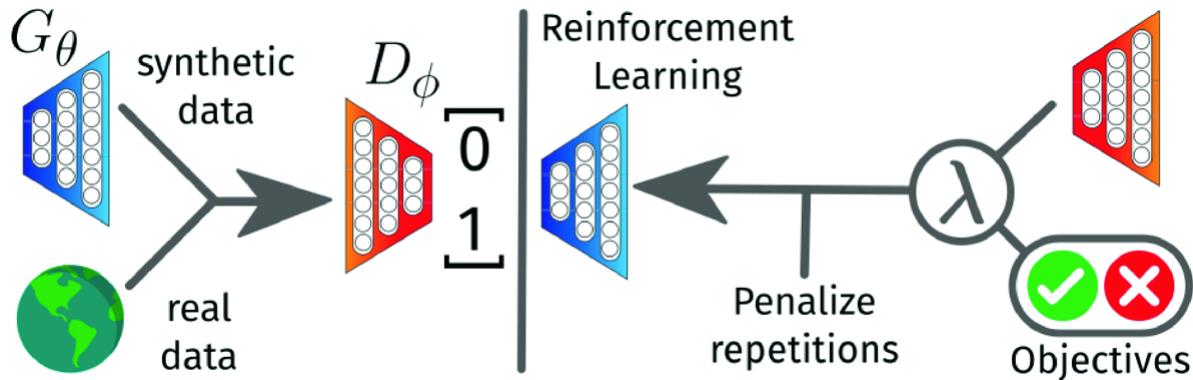


Fig. 11.5 The structure of ORGAN model [32]

The effectiveness of ORGAN for generating molecules with properties related to drug discovery research is verified [32]. Three commonly used objectives in drug discovery research, i.e., solubility, synthesizability and drug-likeness, are used as domain-specific objectives that needs to be optimized. The combination of these three objectives is also used as a reward to testify the performance of ORGAN on multi-objective optimization task. ORGAN and OR(W)GAN, i.e., ORGAN with Wasserstein distance [33], are compared with MLE, SeqGAN and Naive RL models. From the results, it is shown that ORGAN and OR(W)GAN models can dramatically improve the objective metrics of the generated molecules, compared with that of the molecules generated by the non-optimized methods, i.e., MLE and SeqGAN. This finding indicates that they are able to bias the generation process. Besides, the Naive RL method achieves better result on the optimization scenario of synthesizability and solubility than the ORGANs. To explain this, the authors claimed that monotonous

patterns like ‘CCCCCCC’ or ‘CCOCOCCCC’ will give larger rewards of solubility to the generator. So that the naive RL method tends to generate these monotonous molecules to obtain larger reward. For ORGAN, the reward comes from two components. The RL component is the major drivers for the property optimization, while the discriminator components limit the generator to produce drug-like molecules rather than monotonous ones. So, although the optimization capacity of naive RL is better than that of ORGAN on the metric of solubility, the molecules generated by ORGAN are more drug-like.

11.3.1.3 ORGANIC

Sanchez-Lengeling et al. proposed ORGANIC, a chemically oriented framework based on ORGAN, that capable of optimizing distribution over molecular space for desirable metrics in inverse-design chemistry [34]. Based on ORGAN, the authors used melting point, drug-likeness, Lipinski’s rule of five and PCE as objectives to evaluate the capability of ORGAN for generating molecules biased towards these desired chemical properties. The results have shown that the ORGANIC can effectively shift the distribution of generated samples towards domain-specified desired properties.

11.3.1.4 RANC

Putin et al. proposed RANC model based on the ORGANIC paradigm [35]. The RANC uses GANs and RL to learn the data distribution from a training set and to optimize desired properties, respectively. While, for the generator in the GANs structure, a differentiable neural computer (DNC) [36] is used, instead of a LSTM-based RNN. DNC is a Turing computer with two components, the controller and the memory bank. Both of these components are constructed by neural networks, so that they can be trained in the end-to-end manner. With the external memory bank and attention mechanisms, the DNC is believed to have an increased capability to capture the long short-term patterns of tokens in sequence generation tasks. The authors claimed that the advantages of applying DNC as generator are two fold [35]. First, the DNC has ability to remember very complex sequences. Second, the DNC allows generation of much longer sequences $Y_{1:T} = (y_1, \dots, y_T)$ compared to LSTM. Therefore, the RANC is expected

to achieve better performance in the molecule generation field. In addition, the generator G and the discriminator D are pretrained before adversarial training to improve the performance of the model. During the training step, the generator is first pretrained by Maximum Likelihood Estimation method to learn how to generate valid SMILES strings. Then, the discriminator D is pretrained for \tilde{y} epochs with real data and valid SMILES generated by the pretrained generator G , which means that the invalid SMILES generated by G are discarded.

In the experiment part, the authors first studied the perfect discriminator problem [35]. The perfect discriminator problem refers that when the discriminator is perfect, i.e. it is able to make perfect predictions with no error, the gradient will vanish and the training will be terminated. To study the performance of RANC on avoiding the perfect discriminator problem, the generators of RANC and ORGANIC are firstly pretrained, respectively. Then, when pretraining the discriminator, the classification accuracy of the discriminator after each pretraining epoch is reported. The result shows that the accuracy of the D in ORGANIC model achieves almost 100% after 20 epochs. But the discriminator of RANC is hard to achieve perfect prediction accuracy even after 100 pretraining epochs. This finding indicates that the generator of RANC have a stronger ability to get avoid of the perfect discriminator problem.

In addition, the authors trained RANC and ORGANIC models on two datasets with Lipinski's Rule of 5 (RO5) as objective-reinforced reward function. Several statistical metrics of the generated set of molecules are used to compare the performance of these two models. The metrics include general statistics and chemical statistics. The former ones include average length of samples, validity, uniqueness, LogP, TPSA, etc. And the later ones include diversity, number of fragments, number of clusters, number of molecules that cannot pass medicinal chemistry filters (MCFs). The result shows that the molecular library generated by RANC has better statistical characteristics. The distribution learnt by RANC is closer to that of the training set as well.

11.3.1.5 ATNC

Similarly, Putin et al. proposed Adversarial Threshold Neural Computer (ATNC) model [37]. The ATNC model consists of three components: a DNC-based generator, a discriminator, and an adversarial threshold (AT) module. The AT module shares the same architecture with the discriminator, and the parameters of the AT module are periodically updated by copying from the discriminator. Similar to that of the RANC, the training step of ATNC also comprises of pretraining and adversarial training. In adversarial training phase, the sequences produced by the generator are firstly selected by the AT module, and the molecules that most closely match to the training samples will be selected and transmitted to the discriminator. The discriminator gives loss to these screened generated sequences to train the generator, and it is also trained by these sequences and the real data. The purpose of using the AT module is to further increase the difficulty of the discriminator to make predictions to avoid the perfect discriminator problem. Only good generated sequences are used to train the discriminator, therefore the discriminator will be harder to become perfect.

In the experiments, four different objective functions are used to train the model, and the statistics of the set of generated molecules is reported for comparison. The result shows that with the AT module, the ATNC is able to produce a very stable percentage of valid and unique SMILES strings. And the ATNC also have a better performance on learning the distribution of the training set in regard of chemical statistics.

11.3.1.6 MolGAN

In the above-mentioned GAN-based molecule generation models, SMILES is used to represent molecules. However, the complexity of the grammar of SMILES will not only lead to a low validity of generated samples, but also force models to use amount of learning capacity to learn the grammar, rather than to learn the chemical knowledge of the structure-property relationships. Besides, the SMILES of similar molecules may be significantly different [20]. Therefore, it is harder for the model to learn the similarity between structure-similar molecules by their totally different SMILES strings. Recently, with the rapidly development of Graph Neural Networks, generative models that directly deal with molecular graphs becomes a hot issue.

MolGAN [38] is the first GAN-based molecule generation model which directly generate molecular graphs. It consists of a generator, a discriminator and a reward network. For each time a D-dimensional vector $z \in \mathbb{R}^D$ is sampled from a standard normal distribution $z \sim N(\mathbf{0}, \mathbf{I})$, the generator $G_\phi(z)$ will output two continuous and dense tensors: $X \in \mathbb{R}^{N \times T}$ that defines the atom type in the graph, and $A \in \mathbb{R}^{N \times N \times Y}$ that defines bond types. Then, a categorical sampling process is utilized to transform these two continuous, dense tensors to discrete, sparse tensors χ^2 and \tilde{A} . These two tensors determine a molecular graph. To directly deal with molecular graphs, the discriminator and the reward network are Relational GCNs [39] sharing the same architecture. The Wasserstein loss is used to get avoid of the mode collapse problem. For the reinforcement learning component, there is no need for MolGAN model to model the state-action value function $Q(s, a)$ because the samples produced by the generator are not sequences. Considering each generated graph as an action, the MolGAN utilize a variant of Deep Deterministic Policy Gradient (DDPG) algorithm [40] to train the generator guided by the reward given by the reward network. The DDPG algorithm is a deterministic policy gradient algorithm, which is known to perform well in high-dimensional action spaces [41]. Figure 11.6 is the illustration of the structure of MolGAN model.

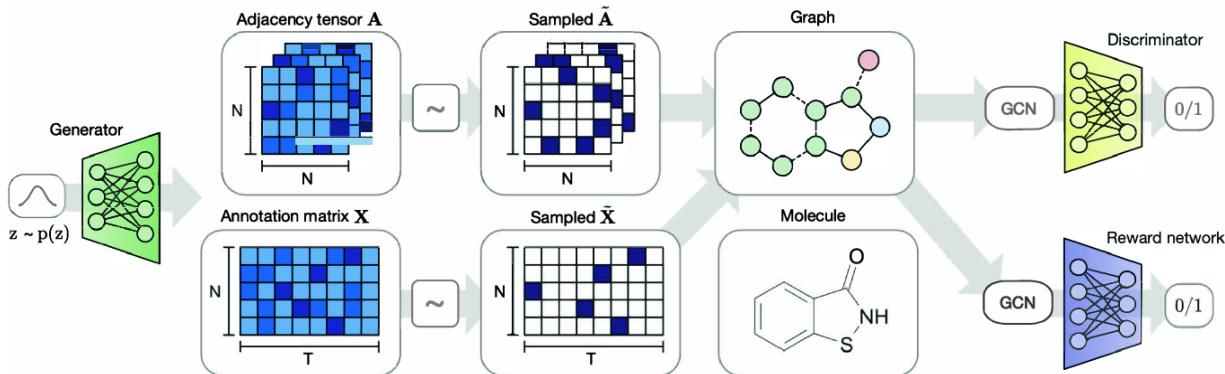


Fig. 11.6 The structure of the MolGAN model [38]

In experiments, QED, SAS, solubility and the combination of these three metrics are utilized as objectives to train the MolGAN model [38]. Comparing with ORGAN and Naive RL, the MolGAN is proved to achieve

better optimization performance with almost 100% validity. And in the comparison between MolGAN and other VAE-based molecule generation models, the MolGAN outperforms others on the metric validity and novelty. However, the uniqueness of MolGAN is much smaller than other VAE-based models, which indicates that the MolGAN may be suffering from the mode collapse problem.

In order to prevent the mode collapse, Blanchard et al. [42] improved the training process of MolGAN by using the idea of mutation and recombination in Genetic Algorithm. This method, named GA-GAN, uses random or guided selection strategy to replace or recombine the valid new molecules generated by the generator to update the training set incrementally in each epoch. The results show that the new method can produce more new molecules which are more in line with the expected properties such as drug-like properties, thus avoiding the mode collapse and enhancing the exploration of chemical space.

11.3.1.7 *DiPol-GAN*

Guarino et al. proposed DiPol-GAN [43], which is an improvement of MolGAN. The differentiable-pooling module [44] is used to improve the discriminator. It is a hierarchical readout function to obtain hierarchical structural information of a graph. For the hierarchical structure of molecular graphs may correspond to some functional groups or pharmacophores which may be relative to the desired properties, the differentiable-pooling module is supposed to benefit the predictive capability of the discriminator. By improving the capability of the discriminator, it is believed that the generator will be guided to learn higher quality graph representations [43]. In addition, to avoid the mode collapse problem, the gradient penalty term [45] is added into the WGAN loss when training the discriminator. These improvements are proved to be effective by the experiments in their work.

11.3.2 Distribution-Learning Models

11.3.2.1 *LatentGAN*

Prykhodko et al. proposed LatentGAN model, which is a combination of a GAN and an autoencoder for de novo molecular design [46]. The

autoencoder is firstly pretrained to encode a molecule represented by SMILES to latent vector h , and then decode h back to the SMILES. Then the GAN is trained to generate new latent vector \mathcal{G} which is similar to that of the real data in the targeted training set. Finally, the decoder is used to decode the latent vector \mathcal{G} produced by the generator of the well-trained GAN to the SMILES of the corresponding new molecular structure. The pretraining dataset consists of over 1.3 million SMILES from ChEMBL dataset in the experiment, and the molecules which are active against three targets are selected to build three targeted datasets. A RNN is used as a baseline model. Compared with the baseline, the effectiveness of the proposed LatentGAN model on learning the chemical property distribution of the targeted training set. The authors also claimed that the generated compounds produced by LatentGAN differ from those obtained by RNN-based generative models, indicating that both methods can be used complementarily for the researchers to build a more comprehensive molecular library [46].

11.3.2.2 Hong et al.'s model

Hong et al. proposed a model based on an adversarially regularized autoencoder (ARAE) [47]. ARAE is the combination of a VAE and a GAN, as shown in Fig. 11.7, similar to the LatentGAN model. In this model, discrete molecular structures, represented by SMILES, are encoded into continuous latent variables, and the distribution of the latent representations $p_\theta(z|x)$ is forced to be similar to that of the latent variables $p_\Psi(\tilde{z})$ produced by the generator by adversarial training. In addition, a conditional generation scheme of molecules is introduced. The property π_i is used as condition given to the decoder to learn the distribution of data \mathcal{G} given a condition, denoted as $\mathcal{D} \approx \hat{\mathcal{D}}$. To remove the property-associated attribute information from the latent variable z , the mutual information between z and y , $MI(z, y; \theta, \lambda)$, is minimized. Since the mutual information is intractable, the lower bound of mutual information $VMI(y, z; \theta, \lambda) = \max_\lambda \mathbb{E}_{p_\theta(z|x)} [\mathbb{E}_{y \approx p(y|z)} [\log q_\lambda(y|z)]]$ is added to the loss function, where $[-\pi, \pi]$ is estimated by the predictor network. Experiments show that the proposed model outperforms other VAE-based

and GAN-based models on the uniqueness and novelty metrics. Good performance in conditional generation is achieved as well.

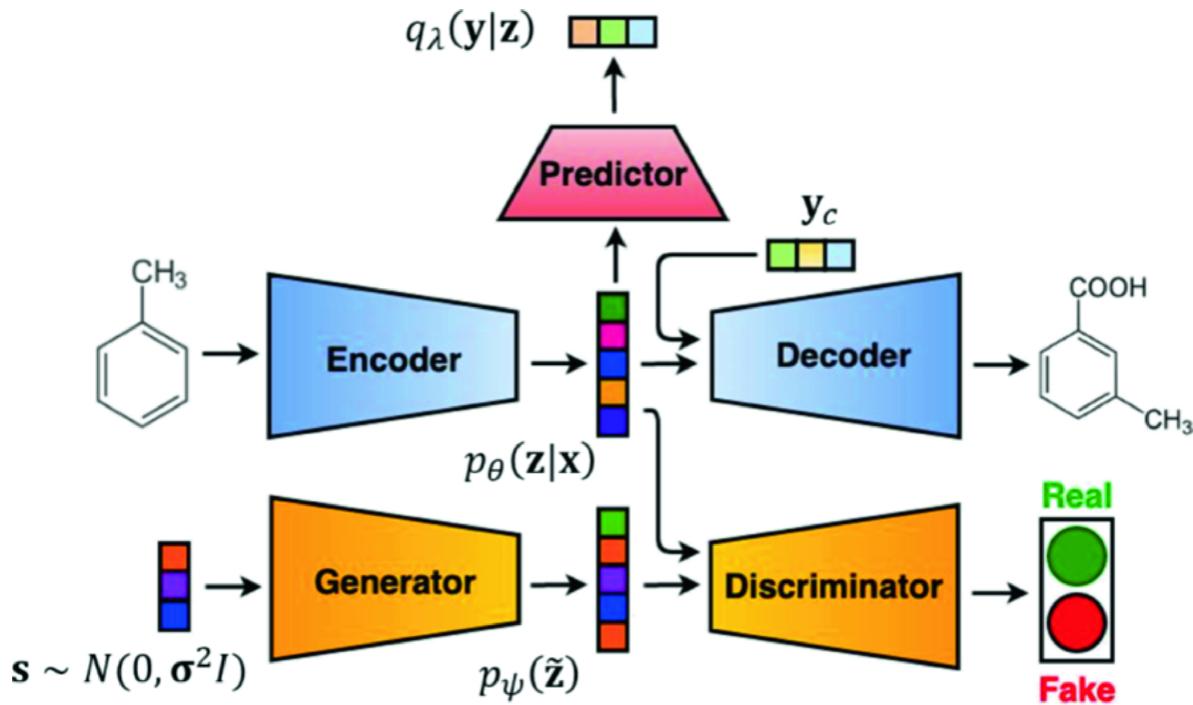


Fig. 11.7 Network structure of the CARAE model. The figure is cited from [47]

11.3.2.3 *Bian et al.'s Model*

Bian et al. introduced deep convolutional GANs (dcGAN) [48] to the generation of target-specific novel molecules for cannabinoid receptors [49]. In this study, molecules are gathered from ZINC and ChEMBL datasets. Four types of fingerprints and four different architectures of CNN backbones are firstly investigated by conducting supervised activity prediction experiments to discover which combination of architecture and molecular representation matches cannabinoid receptor ligands effectively and efficiently. The results revealed that the combination of the LeNet-5 and the AtomPair fingerprints provides the highest overall performance on the classification of both active and inactive/random compounds against CB1 and CB2 receptors.

With this finding, a dcGAN model is built, in which the LeNet-5 network is used as the discriminator. The generator G produces 1024-dim vectors as the fingerprints of generated molecules. To map the generated fingerprints with specific molecular structures, the authors recommended an

alternative way that looking for structures in a large library based on chemical similarity, measured by the Tanimoto coefficient. A molecule with similarity larger than 95% is considered as the corresponding structure.

11.3.2.4 AAE

Another series of GAN-based generative models are Adversarial Autoencoders (AAE) [50], which combines the adversarial training strategy with the Variational Autoencoder (VAE). The AAE involves adversarial training strategy to perform variational inference by matching the posterior distribution of the latent vector of the autoencoder with an arbitrary prior distribution [50]. In the AAE model, an autoencoder is trained with two objectives - a reconstruction error criterion and an adversarial training criterion. The encoder learns to convert the data distribution to a prior distribution, and the decoder learns a generative model that maps the prior to the data distribution.

Kadurin et al. first tried to leverage the AAE model for generating novel compounds with potential anti-cancer properties [10]. Their model receives molecules represented by 166-bit MACCS fingerprints [16] as input, and produces new fingerprints as generated molecules. An autoencoder structure with seven layers is used. Besides the fingerprints, a drug concentration value is used as input as well, and an additional neuron in the latent layer is introduced responsible for the drug concentration, which Growth Inhibition percentage (GI) is used in their work. The result of the experiment suggested that the newly generated molecules may be worth studying as potential anti-cancer candidates.

11.3.2.5 druGAN

Later, Kadurin et al. extended their previous work to a complete framework named druGAN for molecule generation tasks with extensive experiments [51]. In druGAN, the 166-bit MACCS fingerprint is still used as molecule representation, and the drug concentration value is discarded. To balance the training between generator and discriminator, a hyperparameter p is introduced to set the desired *discriminator power*, trying to limit the capability of discriminator from being perfect. In other words, only when the discriminator can correctly predict the fake samples from the generator with probability less than p , the discriminator will be

trained. Otherwise, the generator will be trained. Comparison between druGAN and VAE models suggests that the AAE based model outperforms the VAE in terms of the capacity and efficiency. The druGAN is also proved to achieve a better performance when encountering a trade-off between reconstruction error and variability of sampling. Based on these findings, the authors claimed that the proposed druGAN can be used for further supervised training with very small data, of which scenario drug discovery researchers usually encounter.

11.3.2.6 SAAE

The supervised adversarial autoencoder (SAAE) [50] can be applied to generating new objects that satisfy a given property, which conforms to the paradigm of conditional generation. The original SAAE model is proved to generate good results with a few simple conditions, while for more complex objects which requires dozens of complex conditions, things will get different [15]. Polykovskiy et al. studied the application of the SAAE model in conditional generation tasks and argued that the SAAE has no theoretical guarantees for conditional generation. The SAAE can concatenate any fixed desired property y , which is considered as the condition, with the latent code z , sampled from $y \in \mathbb{R}^{120}$, at the input of the decoder to generate samples from the distribution $p(x|y)$, as is described in Fig. 11.8a. However, as is claimed by Polykovskiy et al., the independence assumption of z and y are not always true. Although the model might be trained to match the latent distribution $p(z)$, when generating new samples with any fixed condition y , a completely different distribution $p(z|y)$ is required. Thus, an inconsistency issue should be overcome by disentanglement [15].

11.3.2.7 ECAAE

Polykovskiy et al. proposed Entangled Conditional Adversarial Autoencoder model (ECAAE) [15]. In their work, two approaches, predictive disentanglement method and joint disentanglement method, are introduced to disentangle the latent codes z and properties y , as is described in Fig. 11.8b, c. Regularization terms are added into the loss function to promote the independence between y and z . ECAAE is applied to generating structural analogs and molecules with desired properties. In the

former experiment, a given molecule is encoded as the condition and the ECAAE is trained to produce compounds that similar to this given molecule. Tanimoto similarity and Hamming distance between the fingerprints of molecules are used to measure this similarity. In the later experiment, lipophilicity (logP) and synthetic accessibility (SA) are used as conditions. The Pearson correlation between the actual value for the generated molecules and that of the requested one is reported. The results of these experiments proved that the generation quality is improved by involving disentanglement techniques in the ECAAE model.

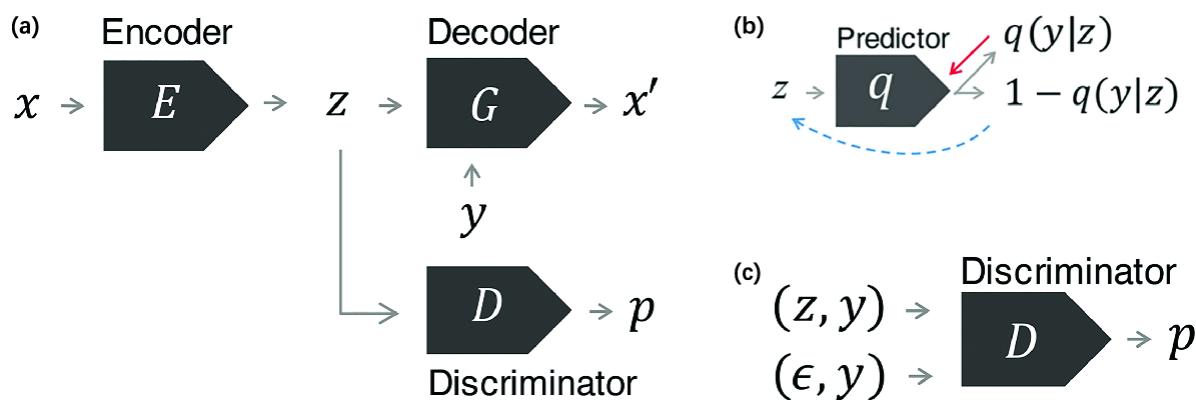


Fig. 11.8 **a** Illustration of the SAAE model. **b** Predictive disentanglement method. **c** Joint disentanglement method, where $z \sim N(\mathbf{0}, \mathbf{I})$. Figures are cited from [15]

11.3.2.8 BiAAE

Considering that efficacy and side effects of drugs can be evaluated by transcriptome expression profiles, Shayakhmetov et al. [52] proposed Bidirectional Adversarial Autoencoder (BiAAE for short) to infer drug molecules that could induce a desired change in gene expression. Instead of learning the conditional distribution $p(x|y)$ as conventional conditional generative models do, BiAAE learns a joint distribution $p(x, y)$ of molecular structure x and the gene expression change conditions y . In this way, it can both generate molecules for given gene expression changes and predict gene expression profiles for an incubated drug. As is shown in Fig. 11.9, the molecule structure x and their properties y are assumed to depend on a shared latent variable s and two exclusive latent variables p_z and χ . The joint distribution can be denoted as $p(x, y) = p(s, z_x, z_y)$. The network structure of BiAAE is shown in Fig. 11.10. The encoders D_1 and E_y are

trained to extract latent vectors p_z , \mathcal{X} , p_z and \mathcal{X} from the given sample x and y , and the decoders W_1 and A_p are trained to reconstruct x and y from these latent vectors. A shared loss is used to ensure that p_z and \mathcal{X} are close to each other, and the discriminator loss is exploited to force the latent vectors p_z , \mathcal{X} and s to represent different features.

BiAAE was validated on a gene expression profiles dataset with 978 genes gathered from the Library of Integrated Network-based Cellular Signatures (LINCS) L1000 project (Duan et al. [53]). Results show that BiAAE outperforms the baseline conditional generative models.

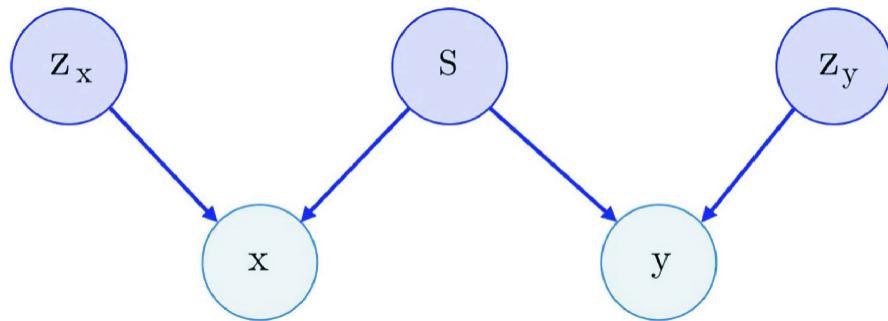


Fig. 11.9 The underlying graphical model of the BiAAE model. The figure is cited from [52]

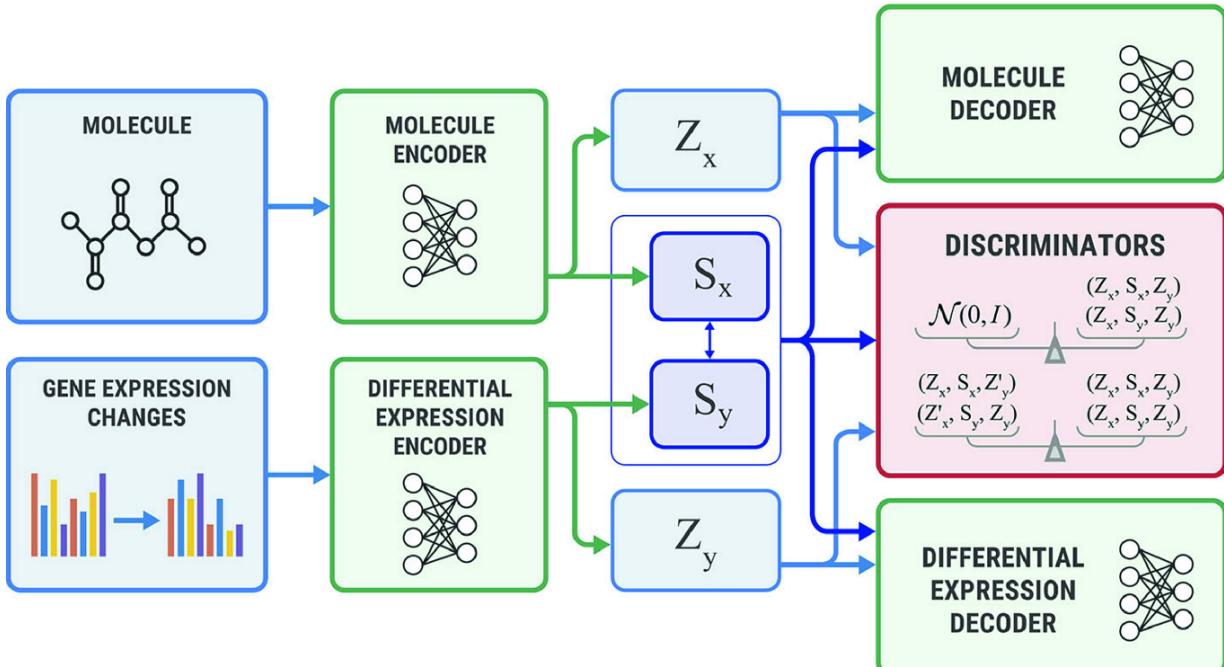


Fig. 11.10 The network structure of BiAAE model. The figure is cited from [52]

11.3.2.9 CCM-AAE

Grattarola et al. [54] introduced a framework, named Constant-Curvature Riemannian Manifold Adversarial Autoencoder (CCM-AAE), which can be trained to embed a data distribution on Constant-Curvature Riemannian Manifolds (CCMs), e.g. hyperspherical or hyperbolic manifolds. CCMs are considered to play a conspicuous role as embedding spaces for a wide range of data distributions, since the non-Euclidian geometry may naturally correspond to some relevant properties of data. The schematic view of the spherical CCM-AAE is shown in Fig. 11.11. The encoder $E(x)$ embeds a sample into the latent space \mathbb{R}^{d+1} which represents the ambient space of a d -dimensional manifold \mathcal{M} , and the decoder $D(x)$ is trained to reconstruct the sample from the latent vector $z \in \mathbb{R}^{d+1}$. An optional orthogonal projection from z onto the CCM \mathcal{M} is also introduced, if operations (e.g. distances and sampling) are needed to be computed on the manifold. The critic network is trained as a discriminator to distinguish between embeddings and samples from the spherical uniform prior $U_{\mathcal{M}}$. A membership degree function is introduced as well to explicitly regularize embeddings to lie exactly on the CCM. For a manifold \mathcal{M} with curvature $\kappa \neq 0$, the membership degree is calculated by Eq. (11.6), where $\varsigma \neq 0$ controls the width of the membership function.

$$\mu(z) = \exp \left(\frac{-\left(\langle z, z \rangle - \frac{1}{\kappa}^2\right)}{2\varsigma^2} \right) \quad (11.6)$$

In experiments, CCM-AAE is compared with different baseline models on the molecule generation tasks of the QM9 dataset. Compared with the baselines, CCM-AAE wins the joint metric, i.e., the product of the validity, uniqueness and novelty. Experimental results indicate that CCM-AAE can achieve the best overall performance.

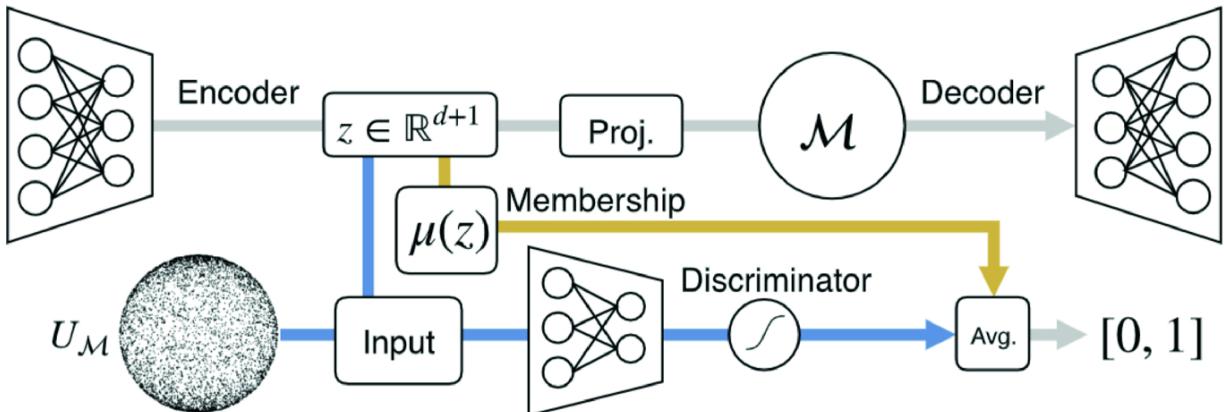


Fig. 11.11 The network structure of the CCM-AAE model [54]

11.3.2.10 Mol-CycleGAN

Maziarka et al. proposed the Mol-CycleGAN model for molecule optimization [55]. It is a CycleGAN-based model [56] which produces compounds with optimized desired properties as well high structural similarity to the original ones. The model learns the transformation rules from the sets of molecules with and without the desired molecular property. The experiments proved that the model can generate new molecules with desired properties which are still close to the starting ones. The degree of similarity of the generated molecules and the molecular set without desired properties can be controlled by a threshold. This study provides a useful tool for the pharmacists, because in the drug discovery procedure, the researchers mostly modify the structure of molecules to optimize multiple desired properties in a sequential manner.

11.3.2.11 SMILES-MaskGAN

Lee et al. proposed SMILES-MaskGAN, aiming to generate molecules with desirable properties in de novo drug design [57]. SMILES-MaskGAN can be seen as an application of the MaskGAN architecture to SMILES strings of molecules. This model consists of three components: a generator, a discriminator and a critic network, which combines the GAN architecture and actor-critic-based reinforcement learning. The generator is implemented by a Seq2Seq model with LSTM units and attention mechanism. It receives masked SMILES strings (i.e., some atoms of the SMILES are replaced by *mask* tokens) as input, and generates new SMILES strings with filled-in

atoms. The discriminator, which has the same architecture as the generator, receives the masked string and the generated string as input, and then computes the probability that each filled-in atom of the generated string is real and produces a reward μ for training the generator. The critic network has the same architecture as the generator as well. It generates a baseline b_t for each token of the generated strings, which is used to reduce the gradient variance to accelerate the convergence of the generator. Policy gradient is adopted for training. Experiments show that SMILES-MaskGAN gets high scores of all metrics, especially uniqueness and novelty.

11.3.2.12 Gene-CGAN

The primary goal of drug design is usually to produce bioactive molecules. Méndez-Lucio et al. [58] utilized GANs to generate molecules with corresponding gene expression characteristics, which are usually active molecules for specific targets. In the model, a conditional GAN and a Wasserstein GAN with gradient penalty were stacked together and regulated by the gene expression signature to produce high-resolution molecular structure information. Since the model does not need the information of target and biological activity, it can be used for molecular design of any target, multi-target or even some biological state. In addition, the stage II of the stack GAN of the model has been proved to be able to modify the molecular skeleton to obtain molecules with similar activity by the gene expression signature or morphological profiles extracted from high content images [59]. This method combined system biology with chemical molecular design and has great application prospect in the fields of active compounds, directed chemical libraries and compound optimization.

11.3.2.13 ASYNT-GAN

Jacobs et al. [60] proposed a GAN model to generate synthetic molecule structures to optimize binding affinity to a target. The model consists of an encoder decoder structure and a generator that generates the protein-ligand three-dimensional structure. The encoder decoder structure transforms the input a point cloud expressing proteins, ligands, and their respective binding bonds into potential space. A stacked generator architecture was designed, where the first generator takes a complex structure as the input and calculates the region of interest. For the second generator, an U-Net

decorated with residual blocks and attention gates was used to encode the point cloud coordinates from last step and a PointNet decoder was used to construct the ligand structure fitting the binding in 3D space. The model achieved a good generation of complete structure that optimizes the binding molecules in 3D complex, even if there is noise. A meaningful exploration of the generation of molecules with better binding affinity for a target was completed with magic GAN.

11.3.2.14 QGAN-HG

With the invention of quantum computer, quantum neural network model is also developing. Recently, quantum GAN has been used to generate drug molecules. Li et al. [61] proposed a qubit-efficient quantum GAN with a hybrid generator (QGAN-HG) that can generate drug like molecules. In QGAN-HG, a parameterized quantum circuit was used to obtain a feature vector of qubit size dimension, a deep neural network was employed to output a molecule graph with an atom vector and a bond matrix. QGAN-HG with 85.07% less parameters can learn the same chemical space as MolGAN. The patched QGAN-HG with multiple sub-circuits can generate molecules with good drug like properties, which greatly reduces the training time and avoids the instability and the gradient disappearance of neural network by shortening neural network depth, so it can replace the classical GAN.

11.3.3 Applications of GAN-based Molecule Generative Models

There are usually few active molecules for a biological target, and sometimes the number of known active and inactive molecules is very uneven. In this case, it is difficult to construct an unbiased model. Barigye et al. [62] used GAN for data amplification, which can solve the problem of lack and imbalance of molecular data to a certain extent. To solve the problem of lack of data, GAN was used to generate BACE-1 active and inactive inhibitor molecular samples, and the classification model constructed with the expanded data set showed satisfactory prediction accuracy on the external data set, which was better than reported classifiers; to solve the problem of data imbalance, GAN was used to generate DENV

active molecular samples to obtain balanced active and inactive samples, the trained classifier was also quite accurate. In addition, in the task of generating balanced data, the performance of the classification model trained by the amplified data generated by GAN was better than that trained by the amplified data generated by SMOTE and other data enhancement methods, and even comparable to the algorithm based on unilateral selection under sampling, which is equivalent to integrating 14 SVM classifiers.

Recently, GANs for de novo molecule generation have been used to create open access databases of small molecules. Zhumagambetov et al. [63] trained 10 ML algorithms including ORGAN, ORGANIC, MolCycleGAN, CNN and JT-VAE on samples of ZINC to generate 2.8 million molecules which were integrated into a database and coupled to a web interface together with their computational properties, allowing users to browse data in a user-friendly and convenient way. Users can create new molecules as needed in case of insufficient results caused by specific search. These newly created molecules will be added to the existing database, so the content and diversity of the database will continue to grow according to the needs of users. This process may be used for drug screening and design, and provides a good template for building molecular database using artificial intelligence.

11.4 Comparison with Other Generative Models for Molecule Generation

GANs are powerful generative models. However, there are other deep generative models that can be used to generate molecules. In this section, we briefly introduce some molecule generation works based on other deep generative models. The advantages and disadvantages of GAN-based models are also discussed.

11.4.1 Molecule Generation Based on Other Generative Models

In addition to GAN-based models, there are some other deep generative models used for molecule generation, including VAE-based models and

pure reinforcement learning (RL) methods.

11.4.1.1 VAE-based Models

The first molecule generation model [64], which was proposed in 2016, is based on a *variational autoencoder* (VAE), by which the SMILES string of a molecule is embedded into a continuous latent space. A neural network is trained as a mapping $f(z)$ from the latent vector to the desired properties. Then, gradient-based optimization is conducted in the latent space to guide the search for the latent vector with optimal desired property, which is predicted by $f(z)$. The molecular structure corresponding to the latent vector is finally obtained by the decoder.

By embedding discrete molecular structures into a continuous latent space, it is efficient to search for molecules with optimal property. And the latent space of VAE is forced to be Gaussian, so that the space is much smoother, which makes optimization much easier [13]. Because of this merit, a number of VAE-based models were proposed. Kusner et al. proposed Grammar VAE to generate SMILES following syntactic constraints imposed by a context-free grammar to increase validity [65]. Dai et al. proposed Syntax-directed VAE (SD-VAE), which incorporates both syntactic and semantic constraints of SMILES [66]. Similar to MolGAN, Simonovsky & Komodakis proposed GraphVAE that directly generates atom labels and adjacency matrices of graphs [67].

To improve the validity of generated samples, Jin et al. [20] proposed the junction-tree VAE (JT-VAE) model, in which a molecular graph is generated by first generating a tree-structured scaffold over chemical substructures, and then combining the substructures into a molecule according to the scaffold. When combining substructures, valency checking is employed so that only valid molecular graph structures will be preserved as candidates. With this checking mechanism, the JT-VAE model achieves 100% validity of the generated samples for the first time.

Later, more graph-based VAE models were proposed [68–71]. These models use molecular graph as representations and generate atoms and bonds sequentially with valency checking. In such a way, these state-of-the-art models can essentially achieve 100% validity. However, as Pang et al. [72] pointed out, the perfect validity is not resulted from the strong learning ability of these models. Instead, it owes to the valency checking by

external chemical toolkit such as RDKit. Actually, generating invalid samples is not a fatal problem for molecule generation models, because these invalid samples can be easily screened by RDKit [13]. The validity metric essentially measures how well a generative model learns from the training set about what valid molecules like. However, a perfect validity with external valency checking may make the model bypass the learning of such knowledge.

Pang et al. proposed a latent space energy-based prior model for molecule generation, arguing that an expressive model is sufficient to implicitly and automatically capture the complicated chemical rules from the training set, even if molecules are encoded in simple character-level SMILES strings [72]. Their model consists of only two simple neural networks, one RNN $p_\beta(x|z)$ for generating SMILES, and one multi-layer perceptron $P_z(z)$ for inferring the prior. Their experiments show that the proposed model is able to generate valid and unique molecules, with performances on par with the state-of-the-art graph-based models.

11.4.1.2 Pure Reinforcement Learning Method

As discussed above, reinforcement learning techniques have been adopted by GAN-based generative models to conduct model optimization, including ORGAN, RANC, ATNC etc. Different from these works using policy-based-learning, the Mol-DQN model [73] is a pure RL based method, which adopts a value-based-learning algorithm, named Deep Q-Network (DQN). It aims to model the molecule generation and modification task as a Markov Decision Process (MDP), and a neural network is used to model the $Q(s, a)$ function with desirable properties as reward to solve the MDP. Experiments on optimizing LogP and QED show Mol-DQN's significant optimization ability. Comparing to the state of the art model, it achieves over 40% improvement on LogP, and 2% on QED.

11.4.2 Advantages of GAN-based Models

A GAN consists of two components, the generator and the discriminator, which are jointly trained in an adversarial manner. Compared with other generative models, this unique mechanism brings GAN-based models some excellent features, which are introduced in the following subsections.

11.4.2.1 Compared with VAE-based Models

Variational autoencoders are known as likelihood-based models, where an explicit likelihood is used to model the generative probability of the samples, and maximum likelihood estimation (MLE) is used to optimize the likelihood. The likelihood could be in an analytic formula, or could be approximated by neural networks as well. Although likelihood-based models typically allow for easier and more stable optimization [38], the likelihood limits the generative capacity of neural networks.

However, GAN uses a discriminator network to guide the optimization to learn the distribution of the training set, rather than to maximize an explicit likelihood. So it is also called implicit, likelihood-free model. As is claimed by Cao et al. [38], the advantage of using no explicit likelihood may be significant when dealing with graph-structured data, such as molecular graphs. It is non-trivial to compute the likelihood of a generated graph, because it requires either fixed (or randomly chosen) ordered representation of the graph or an expensive graph matching procedure. While for likelihood-free models, it is invariant to reordering of nodes in the matrix representation of the graph and the models will be more concise and efficient [38].

11.4.2.2 Compared with Pure RL Models

The pure RL models, such as Mol-DQN, could achieve brilliant performance in optimizing desired properties, and significantly outperform the other models. Experiments of some GAN-based models also show that the RL component makes substantial contribution to the objective optimization [32, 38]. Thus, the significance of the RL is beyond doubt.

However, in drug design and discovery, there are multiple molecular properties to be concerned, which leads to a multi-objective optimization problem. Although it was shown that the Mol-DQN model is effective in multi-objective optimization [73], when there are dozens of desired properties to be optimized, it is non-trivial to integrate these objectives into one reward function. And it is still a question whether the models could work well when dealing with a reward function consists of many objectives. On the contrary, GAN-based models are trained to imitate the distribution of a set of drug-like molecules and optimize the bioactivity against a certain

target at the same time. Thus, most of desired properties could be learnt implicitly, instead of optimizing an intractable multi-objective reward function.

In addition, since GAN can have a generator and a discriminators of various network structures, so there are various GAN variants, which can be further extended and combined to enrich GAN application scenarios. For example, Méndez Lucio stacked up cGAN and WGAN to generate active molecules for a certain target and carry out molecular modification [58, 59]; Jacobs et al. leveraged attention, GAN and graph deep learning to generate molecules with better binding affinity for a target [60].

11.4.3 Disadvantages of GAN Based Models

11.4.3.1 *Compared with VAE-based Models*

Although implicit, likelihood-free generative models have some prominent advantages compared with VAE-based models, the adversarial training mechanism also brings difficulties to GAN training. Training a GAN typically requires careful hyperparameter tuning and leveraging some ‘tricks’ [74]. As the objectives of the generator and the discriminator are conflicting, GAN training is a saddle point optimization problem, which is known to be inherently unstable. If the gradient of one component dominates that of the other, then the optimizer may miss the saddle point so that either the generator or the discriminator will achieve a perfect score [13]. Thus, how to balance generator training and discriminator training is a subtle technique.

There is also the perfect discriminator problem. As the generator is trained by gradient propagating from the loss of the discriminator, if the discriminator is perfect, i.e., all of the molecules are predicted correctly, then the loss of the discriminator will be zero and no gradient will be propagated backward. Therefore, the generator will lose the guidance of the discriminator to continue improve itself, and the training process will be terminated. This is also called gradient vanishing.

To solve the perfect discriminator problem, one approach is to pretrain the generator by MLE to improve the quality of the generated samples at the beginning of the adversarial training procedure [31, 32, 35, 37]. It is

shown that a powerful generator with strong generation capability can slow down the speed of the discriminator becoming perfect [35].

During adversarial training, the generator G and the discriminator D are trained in turn. In the generator's phase, G is trained g -steps while fixing the discriminator, and in the discriminator's phase, D is trained d -steps with the generator unchanged. It is intuitive that to solve the perfect discriminator problem, g -steps should be larger than d -steps, in order to improve the generator quickly while the discriminator keeps relatively weak. However, a weak discriminator will produce a wrong gradient and mislead the generator to learn incorrect information about the training set. Experiment by Yu et al. [31] shows that when d -steps is larger than g -steps ($g=1$ and $d=5$ in their experiment), the overall performance will be improved with good stability. So, how to balance gradient error and gradient vanishing is a very difficult task, which requires careful tuning of hyperparameters.

Another problem with GAN-based models is mode collapse, where the generator generates only a narrow range of samples. This problem usually occurs when the generator gets stuck in a local minimum [43]. Although WGAN is considered to be able to prevent undesired behaviors like mode collapse [38], MolGAN still suffers mode collapse for the uniqueness of the generated molecules is even worse than that of the first VAE-based model proposed by Gómez-Bombarelli et al. Guarino et al. [43] tried to solve mode collapse by Wasserstein distance and gradient penalty alternative [45].

11.4.3.2 Compared with Pure RL Models

GAN-based models are weak in interpretability, which may limit their applications. Because there are strong causal relationships between the properties of small molecules and their molecular structures, it is important to understand such relationships for the generated molecules. So it is necessary to explore the logic underlying the generation of molecules. Given a generated molecule, a model is expected to tell pharmacists why an atomic group appears here or why an atom is replaced by another one. For pure RL models (e.g. MoldQNN), the network will give an explicit $Q(s, a)$ to reveal the direction of how to modify the current molecular structure to gain a larger reward, which shows a good interpretability. However, using noisy

continuous signals without constraint as input, the generator hides the internal details about the relationship between the features of input and the semantic feature of output. Moreover, GANs cannot provide explicit expression for the distribution of generated molecular chemical space, so has poor interpretability.

11.5 Challenges and Future Directions

In this chapter, the applications of GANs to molecule generation for drug design and discovery have been introduced. Although there are still many problems to be solved, and molecule generative models based on VAEs and pure RL are gaining popularity, it must be admitted that GAN based models have unique advantages. And the success of GANs in CV and NLP also implies their potential in molecule generation. So, it is promising to further study and develop GAN-based molecule generation models to aid drug design and discovery. Here, we discuss the challenges and future directions of molecule generation models based on GAN and other mechanisms.

11.5.1 New Molecular Representations

As mentioned above, although SMILES and 2D molecular graphs are widely used in molecule generation, their drawbacks are also obvious. Briefly, SMILES cannot always represent the similarity of similar compounds, and its grammar is difficult for generative models to learn. Furthermore, 2D molecular graphs do not contain the information of 3D conformation and the chiral structures. 3D geometric representations contain more information of molecules. However, as pointed out by Elton et al. [13], current 3D geometric representations cannot meet the requirements of uniqueness and invertibility at the same time.

A promising approach for molecule representation is to use the internal coordinates of molecules, i.e., the geometric relationships between atoms. Recently, 3D molecular graph based representations are emerging, where local coordinate system are constructed to describe the relative geometric relationships between atoms, which are encoded into the features of nodes in the graph [76]. Such 3D molecular graph based methods are expected to improve the performance of molecule generative models.

11.5.2 New Molecule Generation Models

In the past decade, many molecule generative models have been proposed in the literature. Although some of them have achieved relatively good performance, few new drugs have been reported to be developed by these generative models [77]. Molecule generative models are still in their infancy, far from reaching the level of being widely used by pharmacists.

Obviously, the poor quality of generated compounds hinders the popularity of molecule generative models. This situation inspires the drug R&D community to develop more powerful models with state-of-the-art deep learning techniques. Besides, the poor chemical interpretability of the models and the low synthesizability of the generated molecules also impact the confidence of the pharmacists in the practicability of these models. As mentioned above, although the existing molecule generation models can produce novel compounds with desired properties, they often cannot give a reasonable explanation on why some certain compounds with good properties are generated. Meanwhile, current molecule generative models cannot control the synthesizability of generated molecules. Although there are some indicators (e.g. SAS [28]) to evaluate the synthesizability, these indicators are typically used as an objective function to be optimized, rather than a constraint. Therefore, current models cannot guarantee the synthesizability of generated molecules. For these two reasons, pharmacists do not quite convince the generative model and the generated compounds.

Although there are already works for planning synthesis route of a given molecule [78], it is still desirable for the generative models to focus on generating synthesizable molecules. For example, in the fragment-based drug discovery (FBDD) field, pharmacists have constructed fragment libraries [79, 80]. With these fragments, a fragment space can be constructed by combinatorial chemistry, and novel molecules can be discovered from this space. Since the fragments in these libraries can be used to easily synthesize new molecules by known chemical reactions, the synthesizability of molecules discovered from the fragment space can be guaranteed.

Therefore, developing new molecule generation models based on fragments, and by imposing mandatory constraints on the synthesizability of generated molecules, as well as enhancing the interpretability of

fragment selection, is a promising direction for future works of molecule generation.

11.5.3 Benchmarks and Metrics

One challenge of this area is the lack of commonly-accepted benchmarks and metrics to evaluate and compare different molecule generation models.

First, authorized datasets for model training and testing are important, because the selection of training datasets is vital for training good generative models. Take the ORGAN model as an example, as reported in the original paper [32], the validity values of ORGAN in all four experiments exceed 85%. However, when ORGAN was used as a baseline for comparison with later models, its validity was 0.4% and 2.2 % respectively [73, 75]. Such big difference confuses readers and researchers in this area.

The datasets of GAN-based molecule generation models are summarized in Table 11.2. Previous works usually use some subsets of open-access molecular libraries, such as ZINC and ChEMBL, to train and test the models. This was done by random selection or screening the open-access molecular libraries with some specific rules [35]. The screening rules depend on both domain-specific knowledge and some tricks, which usually were not be announced by the authors. There are not widely used and authorized datasets for molecule generation models to test their performance. Obviously, it is unfair to compare different models trained on different datasets obtained with unclear rules and tricks.

Table 11.2 A summary of datasets used by GAN-based molecule generation models

Dataset	Size	Data sources	Access	Models involved	Description
MOSES ^a [82]	1,936,962	ZINC [83]	Open-access	[10, 46, 52]	A platform with data preprocessing utilities and evaluation metrics
GuacaMola [26]	1,591,378	ChEMBL [84]	Open-access	[10, 32]	A benchmark with metrics
PubChem [9]	>7,000,000	–	Open-access	[51]	A well-known public repository for information on chemical substances and their biological activities

Dataset	Size	Data sources	Access	Models involved	Description
QM9 [85]	133,885	GDB-17 [86]	Open-access	[32, 38, 43, 47, 54, 61]	Molecules with up to nine heavy atoms (C, O, N, F)
QM9-5K [32]	5,000	QM9 [85]	Open-access	[32, 34]	Molecules randomly selected from QM9
ZINC-250K [55]	250,000	ZINC [83]	Open-access	[47, 55],	250,000 molecules extracted at random from the ZINC dataset
Drug-like compounds	15,000	ZINC [83]	Open-access	[32, 34, 35]	Randomly selected drug-like commercially available molecules
ECAAE's dataset [15]	1,800,000	ZINC [83]	Private	[15]	Drug-like molecules screened by in-house filter
GA-GAN's dataset [42]	100,000	QM9 [85] and ZINC [83]	Private	[42]	Molecules with up to 20 heavy atoms
SMILES-MaskGAN's dataset [57]	1,692,512	ChEMBL [84]	Open-access	[57]	Molecules that have been synthesized and tested against biological targets
DRD2 [55]	4,632	ChEMBL [84]	Private	[55]	Target-specific dataset for DRD2 with active and inactive compounds
LatentGAN-ChEBML [46]	1,347,173	ChEMBL [84]	Open-access	[46]	Molecules with up to 50 heavy atoms (C, N, O, S, Cl, Br)
Bian et al.'s datasets [49]	5,874 & 5,949	ChEMBL [84] and ZINC [83]	Open-access	[49]	Target-specific dataset for CB1/CB2 with active and inactive compounds
LatentGAN-ExCAPE [46]	149,732	ExCAPE-DB [87]	Open-access	[46]	Target-specific dataset for EGFR, S1PR1 and HTR1A with active and inactive compounds
Gene-L1000 [52]	19,768	L1000CMap [53]	Open-access	[52, 58]	Molecules with gene expression profiles
ASYNT-GAN-covid19 [60]	267	RCSB PDB [88]	Open-access	[60]	Molecules with 3D binding conformations to its receptor
CD dataset [35]	15,000	ChemDiv _b	Private	[34, 35, 37]	Diverse drug-like commercially available compounds

Dataset	Size	Data sources	Access	Models involved	Description
NCI-60 cell line assay dataset [10]	6,252	DTP ^c	Private	[10]	Including molecules with response concentration and Growth Inhibition percentage of NCI-60 cell line

a benchmark datasets

b<http://www.chemdiv.com/>

c<http://dtp.nci.nih.gov/index.html>

Brown et al. [26] built a dataset for training generative models based on the ChEMBL 24 database. The screening rules were listed in their article, and the dataset is available in the GuacaMol repository. This dataset is expected to be a candidate for evaluating and comparing molecule generative models. However, it is not widely accepted by the community. Some latest works [68, 69, 72] still use home-made training sets derived from the open-access databases.

Second, reasonable performance metrics are also important. The general statistical metrics, as introduced in Sect. 11.2.2, are widely used by existing works to evaluate the models based on distribution learning. However, Renz et al. [81] proposed a trivial model, called AddCarbon model, which simply samples a *new* molecule from the training set and insert a carbon atom randomly into its SMILES string. If the modified SMILES string is valid and not included in the generated molecule set yet, then output this molecule as a new generated molecule. Otherwise, insert the carbon atom to any other position or try another molecule from the training set. Obviously, this model learns nothing about the distribution of the training set. However, all of its general statistical metrics, including validity, uniqueness and novelty, are almost 100%, much better than the existing models, including ORGAN and VAE etc. [81]. This is unreasonable. With these findings, they argued that these metrics are useless for evaluating the ability of models for learning the distribution of training set, because these metrics can be easily cheated. This situation calls for new and robust performance metrics for molecule generative models.

11.5.4 New Pharmaceutical Objective Functions

As there are a number of properties to be considered in drug design & discovery, using only one property as the objective function in molecule generation is undesirable. After generating a molecular library with one property, it is still needed to screen the library according to other properties to find molecules that meet the requirements of other properties. So, it is ideal to propose a score that reasonably combines all desired properties.

However, it is non-trivial to design an objective function to cover all of the concerned properties. Simply combining multiple objective functions linearly into one reward function will not work, as this may lead to that none objectives can be optimized. Although the QED score was proposed for this goal, the descriptors it integrates cannot cover all of the concerns. So, the challenge is how to design a score that can be both accepted by the pharmacists and is easy to be optimized.

11.5.5 Influence of Property Prediction Models

Goal-directed generation models [26] focus on finding molecules with desired properties, which are encoded as score functions to be optimized. However, for many complex properties in drug discovery, the score functions are not available, e.g. the bioactivities. Instead, these complex properties are often approximated by fitting machine learning models on experimental datasets [81]. Then, these machine learning models are used as score functions, or named *predictive models*, to guide the generative models.

However, it was shown that guiding molecule generation by machine learning models is not a good strategy to generate meaningful molecules [89]. The generative models are trained to generate such samples that the output of the predictive model is larger, rather than have the exact desired property. The difference between the property of generated molecules and the desired property will be large if the predictive model is not accurate. In this situation, the generated compounds may not meet the needs of the pharmacists, and thus will still be discarded in the subsequent pharmaceutical experiments. Therefore, to improve the practicability of molecule generation models, the performance of property prediction models must be first guaranteed.

Besides, Renz et al. argued that generative models tend to capture the artifacts of a learned predictive model rather than the true characteristics of

the training set [81]. Two types of biases exist. First, the property prediction model usually exhibits bias to the training data. This makes the generated compounds similar to the samples in the training set, which is referred as *data specific biases*. Second, even using the same training dataset, the predictive model may converge to different states points, because the randomness of the feeding order of the training samples in mini-batches will lead to different directions of gradient descent. The generative model will capture this difference and generate samples biased to specific predictive model, which will lead to *model specific biases*. Renz et al. conducted experiments to prove the existence of these two types of biases. As the aim of generative models for drug design and discovery is to explore the entire chemical space, biases towards the training data or the property prediction model all conflict with the original aim [81].

Renz et al. suggested that the proposed *Model Control Scores* and *Data Control Scores* may help in detecting such biases and guide practitioners to potentially more meaning results. And it is also worthy of studying how to limit the influence of the property prediction model on the generative models.

11.6 Conclusion

Drug design and discovery is a challenging task, which inspires us to promote the process of drug design & discovery by exploring deep learning techniques. With the rapid development of deep generative models, it has become a hot spot to apply such generative models for molecule generation, which aims to discover new molecules with desirable properties. As a widely-used deep generative model in CV and NLP, GAN and its various variants have already been used for molecule generation in drug design & discovery. This chapter tries to summarize the achievements of GAN-based generative models for molecule generation and envisions the future development in this area. After a detailed introduction of two major technical issues of molecule generation: molecule representation and performance metrics, we present a comprehensive survey on GAN-based molecule generation models, which covers 20 models in detail. Furthermore, we review some other models for molecule generation, and discuss the advantages and disadvantages of GAN-based models against

these non-GAN based models. We can see that GAN-based models generally have good performance of molecule generation because of its likelihood-free distribution imitation mechanism. However, the training of GANs is not easy for pharmacists who want to leverage GAN-based molecule generation models to aid their drug discovery research. Challenges and future directions of molecule generation based on GANs and other mechanisms are discussed. This chapter can serve as a preliminary reading for researchers and students in the area of AI-based drug design & discovery.

Acknowledgements

The work was supported by National Natural Science Foundation of China (NSFC) under grant Nos. 61972100, 61772367 and 6217070982.

References

1. Paul, S.M., et al.: How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nat. Rev. Drug Discov.* **9**, 203–214 (2010)
2. DiMasi, J.A., et al.: Innovation in the pharmaceutical industry: new estimates of R&D costs. *J. Health Econ.* **47**, 20–33 (2016)
3. Polishchuk, P.G., et al.: Estimation of the size of drug-like chemical space based on GDB-17 data. *J. Comput.-Aided Mol. Des.* **27**, 675–679 (2013)
4. Zhong, F., et al.: Artificial intelligence in drug design. *Sci. China Life Sci.* **61**, 1191–1204 (2018)
5. Munos, B.H., Chin, W.W.: How to revive breakthrough innovation in the pharmaceutical industry. *Sci. Translat. Med.* **3**(89) (2011). <https://doi.org/10.1126/scitranslmed.3002273>
6. Xiong, Z., et al.: Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *J. Med. Chem.* **63**(16), 8749–8760 (2020)
7. Peng, Y., et al.: TOP: a deep mixture representation learning method for boosting molecular toxicity prediction. *Methods* **179**, 55–64 (2020)
8. Zhang, Z., et al.: FraGAT: a fragment-oriented multi-scale graph attention model

for molecular property prediction. *Bioinformatics* (2021). <https://doi.org/10.1093/bioinformatics/btab195>

9. Kim, S., et al.: PubChem substance and compound databases. *Nucleic Acids Res.* **44**, D1202–D1213 (2016)
10. Kadurin, A., et al.: The cornucopia of meaningful leads: applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget* **8**(7), 10883–10890 (2017)
11. Goodfellow, I.J., et al.: Generative adversarial nets. *Adv. in Neural Inf. Process. Syst.* 2672–2680 (2014)
12. Weininger, D.: SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28**(1), 31–36 (1988)
13. Elton, D.C., et al.: Deep learning for molecular design—a review of the state of the art. *Mol. Syst. Des. Eng.* **4**(4), 838–849 (2019)
14. Landrum, G.: RDKit: open-source cheminformatics. <http://www.rdkit.org/>
15. Polykovskiy, D., et al.: Entangled conditional adversarial autoencoder for de novo drug discovery. *Mol. Pharm.* **15**(10), 4398–4405 (2018)
16. Durant, J.L., et al.: Reoptimization of MDL keys for use in drug discovery. *J. Chem. Inf. Comput. Sci.* **42**(6), 1273–1280 (2002)
17. Morgan, H.L.: The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *J. Chem. Doc.* **5**, 107–112 (1965)
18. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**(5), 742–754 (2010)
19. Duvenaud, D., et al.: Convolutional networks on graphs for learning molecular fingerprints (2015). [arXiv:1509.09292](https://arxiv.org/abs/1509.09292)
20. Jin, W., et al.: Junction tree variational autoencoder for molecular graph generation. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 2323–2332 (2018)
21. Kuzminykh, D., et al.: 3D molecular representations based on the wave transform

for convolutional neural networks. *Mol. Pharm.* **15**(10), 4378–4385 (2018)

22. Skalic, M., et al.: Shape-based generative modeling for de novo drug design. *J. Chem. Inf. Model.* **59**(3), 1205–1214 (2019)
23. Hirn, M., et al.: Wavelet scattering regression of quantum chemical energies. *Multiscale Model. Simul.* **15**(2), 827–863 (2017)
[[MathSciNet](#)][[zbMATH](#)]
24. Eickenberg, M., et al.: Solid harmonic wavelet scattering: Predicting quantum molecular energy from invariant descriptors of 3D electronic densities. In: *Advances on Neural Information Processing Systems (NIPS)*, pp. 6543–6552 (2017)
25. Thomas, N., et al.: Tensor field networks: rotation- and translation-equivariant neural networks for 3D point clouds (2018). [arXiv:1802.08219](#)
26. Brown, N., et al.: GuacaMol: benchmarking models for de novo molecular design. *J. Chem. Inf. Model.* **59**(3), 1096–1108 (2019)
27. Bickerton, G.R., et al.: Quantifying the chemical beauty of drugs. *Nat. Chem.* **4**(2), 90–98 (2012)
28. Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminform.* **1**(1), 1–11 (2009)
29. Comer, J., Tam, K.: Lipophilicity profiles: theory and measurement. In: *Pharmacokinetic Optimization in Drug Research: Biological, Physiochemical and Computational Strategies*, pp. 275–304 (2001)
30. Preuer, K., et al.: Fréchet ChemNet distance: a metric for generative models for molecules in drug discovery. *J. Chem. Inform. Model.* **58**(9), 1736–1741 (2018)
31. Yu, L., et al.: SeqGAN: Sequence generative adversarial nets with policy gradient. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, **31**(1) (2017). <https://ojs.aaai.org/index.php/AAAI/article/view/10804>
32. Guimaraes, G.L., et al.: Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models (2017). [arXiv:1705.10843](#)
33. Arjovsky, M., et al.: Wasserstein Generative Adversarial Networks. In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp.

34. Sanchez-Lengeling, B., et al.: Optimizing distributions over molecular space: an objective-reinforced generative adversarial network for inverse-design chemistry (ORGANIC) (2017). <https://doi.org/10.26434/chemrxiv.5309668.v2>
35. Putin, E., et al.: Reinforced adversarial neural computer for de novo molecular design. *J. Chem. Inf. Model.* **58**(6), 1194–1204 (2018)
36. Graves, A., et al.: Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476 (2016)
37. Putin, E., et al.: Adversarial threshold neural computer for molecular de novo design. *Mol. Pharm.* **15**(10), 4386–4397 (2018)
38. Cao, N.D., Kipf, T.: MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint, arXiv: 1805.11973* (2018)
39. Schlichtkrull, M., et al.: Modeling relational data with graph convolutional networks. In: Proceedings of the European Semantic Web Conference (ESWC), pp. 593–607. Springer, Cham (2018)
40. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning (2015). [arXiv:1509.02971](https://arxiv.org/abs/1509.02971)
41. Sliver, D., et al.: Deterministic policy gradient algorithms. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 387–395 (2014)
42. Blanchard, A.E., et al.: Using GANs with adaptive training data to search for new molecules. *J. Cheminform.* **13**(1), 1–8 (2021)
43. Guarino, M., et al.: DiPol-GAN: Generating Molecular Graphs Adversarially with Relational Differentiable Pooling. Under review
44. Ying, R., et al.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems (NIPS), pp. 4800–4810 (2018)
45. Gulrajani, I., et al.: Improved training of wasserstein GANs. In: Advances in Neural Information Processing Systems (NIPS), pp. 5767–5777 (2017)
46. Prykhodko, O., et al.: A de novo molecular generation method -stealthusing latent vector based generative adversarial network. *J. Cheminform.* **11**(1), 1–13 (2019)

47. Hong, S.H., et al.: Molecular generative model based on an adversarially regularized autoencoder. *J. Chem. Inf. Model.* **60**(1), 29–36 (2020)
48. Radford, A., et al.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016). [arXiv:1511.06434](https://arxiv.org/abs/1511.06434)
49. Bian, Y., et al.: Deep convolutional generative adversarial network (dcGAN) models for screening and design of small molecules targeting cannabinoid receptors. *Mol. Pharmaceutics.* **16**(11), 4451–4460 (2019)
50. Makhzani, A., et al.: Adversarial autoencoders (2015). [arXiv: 1511.05644](https://arxiv.org/abs/1511.05644)
51. Kadurin, A., et al.: druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Mol. Pharm.* **14**(9), 3098–3104 (2017)
52. Shayakhmetov, R., et al.: Molecular generatrion for desired transcriptome changes with adversarial autoencoders. *Front. Pharmacol.* **11**, 269 (2020)
53. Duan, Q., et al.: Lincs canvas browser: interactive web app to query, browse and interrogate lincs 1000 gene expression signatures. *Nucleic. Acids. Res.* **42**(W1), W449–W460 (2014). <https://doi.org/10.1093/nar/gku476>
54. Grattarola, D., et al.: Adversarial autoencoders with constant-curvature latent manifolds. *Appl. Soft. Comput.* **81**, 105511 (2019)
55. Maziarka, Ł, et al.: Mol-CycleGAN: a generative model for molecular optimization. *J. Cheminform.* **12**(1), 1–18 (2020)
56. Zhu, J.Y., et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2223–2232 (2017)
57. Lee, Y.J., et al.: Generative adversarial networks for de novo molecular design. *Mol. Inform.* **40**(7), 2100045 (2021). <https://doi.org/10.1002/minf.202100045>
58. Méndez-Lucio, O., et al.: De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nat. Commun.* **11**(1), 1–10 (2020)
59. Méndez-Lucio, O., et al.: Cell morphology-guided de novo hit design by conditioning generative adversarial networks on phenotypic image features (2020). <https://doi.org/10.26434/chemrxiv.11594067.v1>

60. Jacobs, I., Maragoudakis, M.: De novo drug design using artificial intelligence ASYNT-GAN (2020). <https://doi.org/10.20944/preprints202010.0196.v2>
61. Li, J., et al.: Quantum generative models for small molecule drug discovery (2021). [arXiv:2101.03438](https://arxiv.org/abs/2101.03438)
62. Barigye, S.J., et al.: Generative adversarial networks (GANs) based synthetic sampling for predictive modeling. *Mol. Inform.* **39**(10), e2000086 (2020)
63. Zhumagambetov, R., et al.: cheML. io: an online database of ML-generated molecules. *RSC Adv.* **10**(73), 45189–45198 (2020)
64. Gómez-Bombarelli, R., et al.: Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**(2), 268–276 (2018)
65. Kusner, M.J., et al.: Grammar variational autoencoder. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 1945–1954 (2017)
66. Dai, H., et al.: Syntax-directed variational autoencoder for structured data. In: International Conference on Learning Representations (ICLR) (2018). <http://openreview.net/forum?id=SyqShMZB>
67. Simonovsky, M., Komodakis, N.: Graphvae: towards generation of small graphs using variational autoencoders. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN), pp. 412–422. Springer, Cham (2018)
68. Liu, Q., et al.: Constrained graph variational autoencoders for molecule design. In: Advances in Neural Information Processing Systems (NIPS), pp. 7795–7804 (2018)
69. Rigoni, D., et al.: Conditional constrained graph variational autoencoders for molecule design. In: Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 729–736, IEEE (2020)
70. Samanta, B., et al.: NeVAE: a deep generative model for molecular graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 33, pp. 1110–1117 (2019)
71. Shi, C., et al.: Graphaf: a flow-based autogressive model for molecular graph generation (2020). [arXiv:2001.09382](https://arxiv.org/abs/2001.09382)
72. Pang, B., et al.: Learning latent space energy-based prior model for molecule

generation (2020). [arXiv: 2010.09351](#)

73. Zhou, Z., et al.: Optimization of molecules via deep reinforcement learning. *Sci. Rep.* **9**(1), 1–10 (2019)
74. Lucic, M., et al.: Are GANs created equal? a large-scale study (2017). [arXiv:1711.10337](#)
75. You, J., et al.: Graph convolutional policy network for goal-directed molecular graph generation. In: Advances in Neural Information Processing Systems (NIPS), pp. 6410–6421 (2018)
76. Li, Y., et al.: Learning to design drug-like molecules in three-dimensional space using deep generative models (2021). [arXiv: 2104.08474v1](#)
77. Tan, X., et al.: Automated design and optimization of multitarget schizophrenia drug candidates by deep learning. *Eur. J. Med. Chem.* **204**, 112572 (2020). <https://doi.org/10.1016/j.ejmech.2020.112572>
78. Seglar, M.H.S., et al.: Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **555**, 604–610 (2018)
79. Degen, J., et al.: On the art of compiling and using ‘drug-like’ chemical fragment spaces. *ChemMedChem.* **3**(10), 1503–1507 (2008)
80. Lewell, X.Q., et al.: RECAP—retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *J. Chem. Inf. Comput. Sci.* **38**, 511–522 (1998)
81. Renz, P., et al.: On failure modes in molecule generation and optimization. *Drug Discov. Today Technol.* **32–33**, 55–63 (2019)
82. Polykovskiy, D., et al.: Molecular sets (MOSES): a benchmarking platform for molecular generation models. *Front. Pharmacol.* **11**, 565644 (2020). <https://doi.org/10.3389/fphar.2020.565644>
83. Sterling, T., Irwin, J.J.: ZINC 15—ligand discovery for everyone. *J. Chem. Inf. Model.* **55**(11), 2324–2337 (2015)
84. Gaulton, A., et al.: The ChEMBL database in 2017. *Nucleic. Acids. Res.* **45**(D1), D945–D954 (2017)
- 85.

Ramakrishnan, R., et al.: Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data.* **1**, 140022 (2014)

86. Ruddigkeit, L., et al.: Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.* **52**(11), 2864–2875 (2012)
87. Sun, J., et al.: ExCAPE-DB: an integrated large scale dataset facilitating big data analysis in chemogenomics. *J. Cheminform.* **9**, 17 (2017)
88. Berman, H.M., et al.: The protein databank. *Nucleic Acids Res.* **28**(1), 235–242 (2000). <https://www.rcsb.org/>
89. Nguyen, A., et al.: Deep neural networks are easily fooled: high confidence predictions for unrecognized images (2015). [arXiv: 1412.1897](https://arxiv.org/abs/1412.1897)

12. Improved Diagnostic Performance of Arrhythmia Classification Using Conditional GAN Augmented Heartbeats

Deepankar Nankani¹✉ and Rashmi Dutta Baruah¹✉

(1) Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, Assam, India

✉ Deepankar Nankani (Corresponding author)

Email: d.nankani@iitg.ac.in

✉ Rashmi Dutta Baruah

Email: r.duttabaruah@iitg.ac.in

Abstract

Cardiovascular diseases such as cardiac arrhythmia are a leading cause of death worldwide. Detecting cardiac arrhythmias before their occurrence using an Electrocardiogram (ECG) signal helps in risk stratification, better medical assistance, and patient treatment. Due to privacy concerns, access to personal ECGs is restricted, hindering the development of automated computer-aided diagnosis systems. This chapter discusses an approach for generating irregular beats, namely, supraventricular ectopic, ventricular ectopic, and normal beats recommended by the Association for the Advancement of Medical Instrumentation using a Conditional Generative Adversarial Network (CGAN). Irregular beat generation is necessary due to their rare occurrence that stems from complex biological, physiological systems. CGAN demonstrates its feasibility and effectiveness by discovering intricate structures present in heartbeats by augmenting specific class beats and improving the diagnostic performance of arrhythmia

classification. A Convolution Neural Network based generator and discriminator is employed that incorporates the class information and conventional input for generating beats. Four publicly available standard datasets are adopted for verification of the proposed approach. The developed arrhythmia classifier also achieved better diagnostic performance than state-of-the-art models. The quality of generated signals is estimated quantitatively through five evaluation metrics and qualitatively through visual representation.

12.1 Introduction

Cardiac arrhythmia causes millions of deaths annually around the world due to the occurrence of irregular heartbeats obtained from an electrocardiogram (ECG) signal [33]. The ECG measures the magnitude and direction of the electrical currents of the heart using electrodes by measuring the depolarisation and repolarisation of the cardiac muscle cells. The lead electrodes are attached to the chest, arms, and legs to detect small voltage changes as the potential difference that are further transposed into a visual tracing. The ECG depicts distinctive features such as P wave, QRS complex, and T wave, where the P wave represents atrial contraction/depolarisation; QRS complex represents ventricular depolarisation; and T wave represents ventricles expansion/repolarisation. The changes in these characteristic waves are vital for diagnosing various cardiac arrhythmias.

This chapter focuses on the non life threatening arrhythmias that may lead to life threatening arrhythmias such as atrial and ventricular fibrillation if not treated immediately. The major focus lies in the classification of irregular beats, including supraventricular ectopic beat (SVEB) and ventricular ectopic beat (VEB), through automated computer aided diagnosis. These beats are selected following the Advancement of Medical Instrumentation (AAMI) [47] standard. The ectopic beats are felt as palpitations caused due to an extra or skipped heartbeat making the patient feel the heart lurch or an extra strong beat occurring momentarily. They usually occur in succession like ectopic-normal-ectopic or continuation of several ectopic beats within a small duration. Ectopic beats are more sinister when arising from ventricles than those arising from atria. The

supraventricular ectopic beat originates from the upper chambers or atria. These are also called atrial premature beats or premature atrial contraction and may lead to atrial fibrillation. The ventricular ectopic beat originates from the lower chambers or ventricles. These are also called Premature ventricular contractions (PVC) and may lead to ventricular tachycardia and fibrillation. Figure 12.1 describe normal beat (N), SVEB, and VEB. In a normal beat, all the characteristic waves P, QRS, T are visible. In SVEB, the P wave is usually missing, and in VEB, the QRS complex is wider than the normal beat with a discordant ST segment.

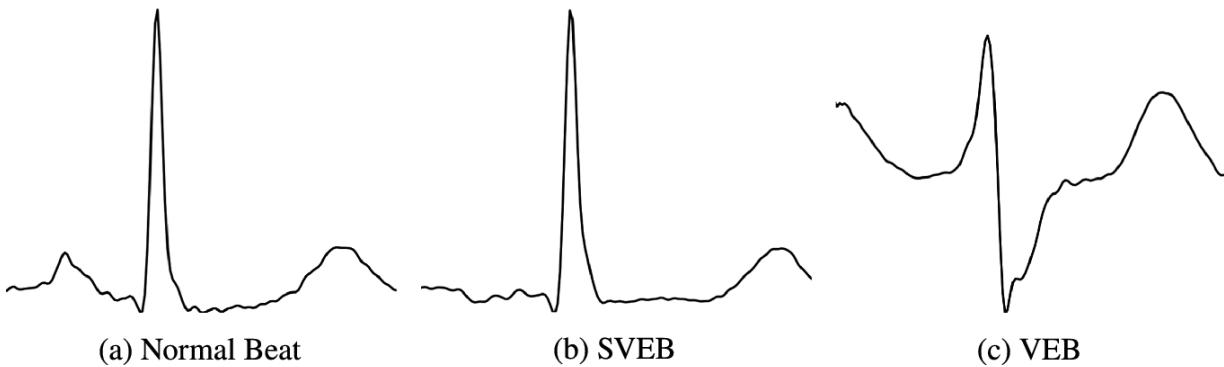


Fig. 12.1 Illustration of regular and irregular beats

The regular and irregular beats are diagnosed either by feature extraction followed by conventional classifiers or directly through supervised deep learning models [16]. Although these methods can provide state-of-the-art performance, they require a huge amount of labeled, diverse, and realistic-looking ECG, which is difficult to acquire in practice as the irregular beat occurrence is rare, causing data imbalance. The challenge in synthesizing ECG occurs because the biological and physiological systems that generate these beats are highly complex. Publicly available datasets sometimes fail to satisfy a certain criterion concerning a study as they might miss out on relevant information, making the problem of beat augmentation crucial. This led the authors to explore data augmentation techniques to generate AAMI recommended beat classes for improving the diagnostic performance of arrhythmia classification. In the past, ECG has been generated synthetically using predefined synthesizers, interpolation, and extrapolation, but these methods are limited by the density of the provided ECG beats. Researchers have generated

synthetic ECGs using heart rate features [46], 3-D vectorcardiogram formulation and a non linear model [10]. Cao et al. [8] generated 12-lead ECG. However, these techniques depend on predefined functions making it difficult to incorporate sufficient information about the basic ECG characteristics. Recently, generative models such as generative adversarial networks (GAN) have gained traction for heartbeat generation [6, 23, 53].

12.1.1 ECG Synthesis Using Generative Adversarial Network

Generative Adversarial Networks (GAN), proposed by Ian Goodfellow [21], belong to a class of generative model that are capable of generating data such as images [36], speech [57], single-channel ECG [53], and EEG [24] by mapping input noise to the generated data. GANs have also been employed to reduce data dimensionality for diagnosing faults and attacks in cyber-physical systems [17]. GANs are a special case of artificial curiosity and are closely related to predictability minimization [61]. Brophy et al. [6] used Wasserstein GANs (WGANs) [3] with gradient penalty to synthetically generate sinusoidal waves, PPG signals, and ECG signals by feeding rasterized images of these signals to GANs and generated new images. However, their approach was limited by the resolution of the generated image. Haradal et al. [23] employed a recurrent neural network (RNN) based long short term memory (LSTM) model for ECG and EEG generation. Zhu et al. [74] employed two layers of bidirectional LSTM (BiLSTM) for generator followed by a convolution neural network (CNN) based discriminator to generate ECG signals. They compared the developed model with recurrent neural network autoencoder and recurrent neural network variational autoencoder. Zhou et al. [72] detected anomalous rhythms in ECG by proposing Beat GAN that generated new beats by extracting the latent input noise from actual ECG rhythms, making the model input dependent. Delaney et al. [14] generated synthetic Sine waves and normal ECG using two BiLSTM followed by CNN along with a mini-batch discrimination layer. Hazra et al. [25] proposed SynSigGAN using bidirectional grid LSTM in generator and CNN in discriminator and generated ECG, EEG, Electromyogram (EMG), and PPG signals.

Researchers have also encoded conditional dependencies in GANs to generate class specific signals. Wang et al. [63] employed an auxiliary classifier GAN (ACGAN) to generate normal beat, left bundle branch block

(LBBB), right bundle branch block (RBBB), premature atrial contraction (PAC), and premature ventricular contraction (PVC). Ye et al. [67] proposed RPSeqGAN by training a sequence generative adversarial network (SeqGAN) using the concept of policy gradient from reinforcement learning. Wulan et al. [64] proposed three GAN models, namely, WaveNet-based model, SpectroGAN model, and WaveletGAN model for the generation of normal, left bundle branch block, and right bundle branch block beat. Their first model preprocesses the input using μ -law companding transformation followed by convolutional layers with dilation; the second and third model used short-term Fourier transform (STFT) and stationary wavelet transform (SWT) to obtain suitable input for GAN. Golany et al. [18] proposed a personalized GAN (PGAN) using an LSTM model that synthesizes minority class beats, including SVEB and VEB, and incorporated two loss functions, mean square loss and binary cross-entropy loss. Golany et al. [19] in their second work incorporated ordinary differential equations representing the heart dynamics into the optimization process of GAN.

12.1.2 Related Work on Heartbeat Classification

Beat classification has been performed using a combination of feature extraction, feature selection, and classification methods. The features include RR intervals that represent the interval between two adjacent R peaks, morphological features [12], position, amplitude features [13], higher order statistical features [40], and Hermite polynomial based features [35]. Several signal transformation techniques that have been employed include wavelet transform [1, 4, 15], discrete cosine transform [44], and stockwell transform [11]. The selection of extracted feature has been performed using principal component analysis [45] and independent component analysis [65, 66] for dimensionality reduction of extracted features and Particle swarm optimization [34], genetic algorithm [37], and bacteria foraging optimisation [11], have been employed to reduce feature redundancy. The classification of extracted features has been performed using support vector machines (SVM) [11, 49], K-nearest neighbors [4], linear discriminant analysis [45], and mixture-of-experts [29]. Clustering methods such as maximum margin clustering with immune evolution [73], fuzzy-entropy-based clustering [27], K-means clustering [27] have also

been employed for classification. Recently, Neural Networks (NN) have also performed exceptionally well for arrhythmia classification including deep belief networks [70], artificial neural network [49], modular neural networks [30], radial basis function network [1, 34]. Novel neural networks such as convolution neural networks (CNN) [32, 51, 54], and multiscale wavelet CNN [15], and recurrent neural networks (RNN) [56, 69] have also been employed for heartbeat classification.

12.1.3 Contributions

The methods mentioned above have generated synthetic ECG signals and achieved commendable performance for beat classification. However, the drawbacks include: (i) Only a few methods follow the AAMI [47] recommended beat generation, thereby posing a problem of lack in minority class beats that needs to be tackled; (ii) Use of latent input extracted from actual ECG beats, making the generative model biased; (iii) Lack of evaluation metrics for generated signal verification; (iv) Extraction and selection of handcrafted features for classification; (v) The developed classifiers are tested on limited datasets.

This chapter focuses on the generation and classification of normal beats, supraventricular ectopic beats, and ventricular ectopic beats, following the AAMI recommendations [47] by encoding class related dependencies in the deep convolution generative adversarial network model. Following are the major contributions: (i) Use of binary cross-entropy loss for discriminator and GAN, Gaussian noise as latent input to generator, and soft labels for faster convergence during training. (ii) Generated beats are evaluated empirically using five evaluation metrics, namely, Frechet Distance (FD), Dynamic Time Warping (DTW), Root Mean Square Error (RMSE), Maximum Mean Discrepancy (MMD), and Time Warp Edit Distance (TWED). Qualitatively, the generated beats are visually compared with actual beats. (iii) An inter-patient beat classification is performed for classifying N, SVEB, and VEB using a deep neural network that exploits raw ECG beats thereby eliminating the need for feature extraction techniques. (iv) The proposed methodology is verified using four publicly available standard datasets.

The chapter is organized as follows. Section 12.2 provides the data description and the details of preprocessing applied to data. Section 12.3

describes the ECG generation and classification methodology. Section 12.4 discusses the results and Sect. 12.5 concludes the chapter with future scope.

12.2 Data

12.2.1 Dataset Description

Four publicly available standard datasets are adopted to evaluate the performance of proposed methodology. The datasets include Massachusetts Institute of Technology-Beth Israel Hospital Arrhythmia Database (MITARDB) [42], MIT-BIH Supraventricular Arrhythmia Database (MITSVDB) [41], St. Petersburg INCART 12-lead Arrhythmia Database [20], and China Physiological Signal Challenge (CPSC) 2020 [7] data. The data is chosen such that it covers multiple demographics and is diverse enough to account for any real-world scenario. The dataset includes normal and clinically significant arrhythmic beats. The modified limb lead II signal from each record is used for this work. The annotations provided in the dataset categorize the beats into 15 different classes (f, /, N, L, e, R, j, Q, A, J, a, V, S, E, F), each denoting a different type of beat. Table 12.1 describe the major and minor division provided by AAMI [47] into N, SVEB, VEB, Fusion beats (F), and Unknown beats (Q). This chapter focuses on N, SVEB, and VEB class beats.

Table 12.1 AAMI [47] recommended beat labels and classes

Group	Symbol	Class
Any heartbeat not categorized as SVEB or VEB. Normal Beat (N)	N	Normal Beat
	L	Left Bundle Branch Block Beat
	R	Right bundle branch block beat
	e	Atrial escape beat
	j	Nodal (junction) escape beat
Supraventricular ectopic beat (SVEB)	A	Atrial premature beat
	a	Aberrated atrial premature beat
	J	Nodal (junction) premature beat
	S	Supraventricular premature beat

Group	Symbol	Class
Ventricular ectopic beat (VEB)	V	Premature ventricular contraction
	E	Ventricular escape beat
Fusion beat (F)	F	Fusion of ventricular and normal beat
Unknown Beat (Q)	P	Paced beat
	f	Fusion of paced and normal beat
	U	Unclassified beat

12.2.2 Data Preprocessing

The preprocessing schematic is provided in Fig. 12.2. The normalized records are decluttered from high and low frequency noises. The Power Line Interference (PLI), i.e., high-frequency noise (50 or 60 Hz) is removed using a notch filter, and Baseline Wander (BW), i.e., low-frequency noise, is removed using a mean median filter [52, 71]. The beats are extracted from the clean signal using the annotations provided in the dataset from a window of 234 ms before and 486 ms after the R-peak. Since the datasets are sampled at different sampling frequencies, the extracted beats are resampled to the lowest sampling rate of 257 Hz or 186 samples. The beats are represented as $hb(i) = \{v_1, \dots, v_{186}\}$, where v_i represent normalised value at time t .

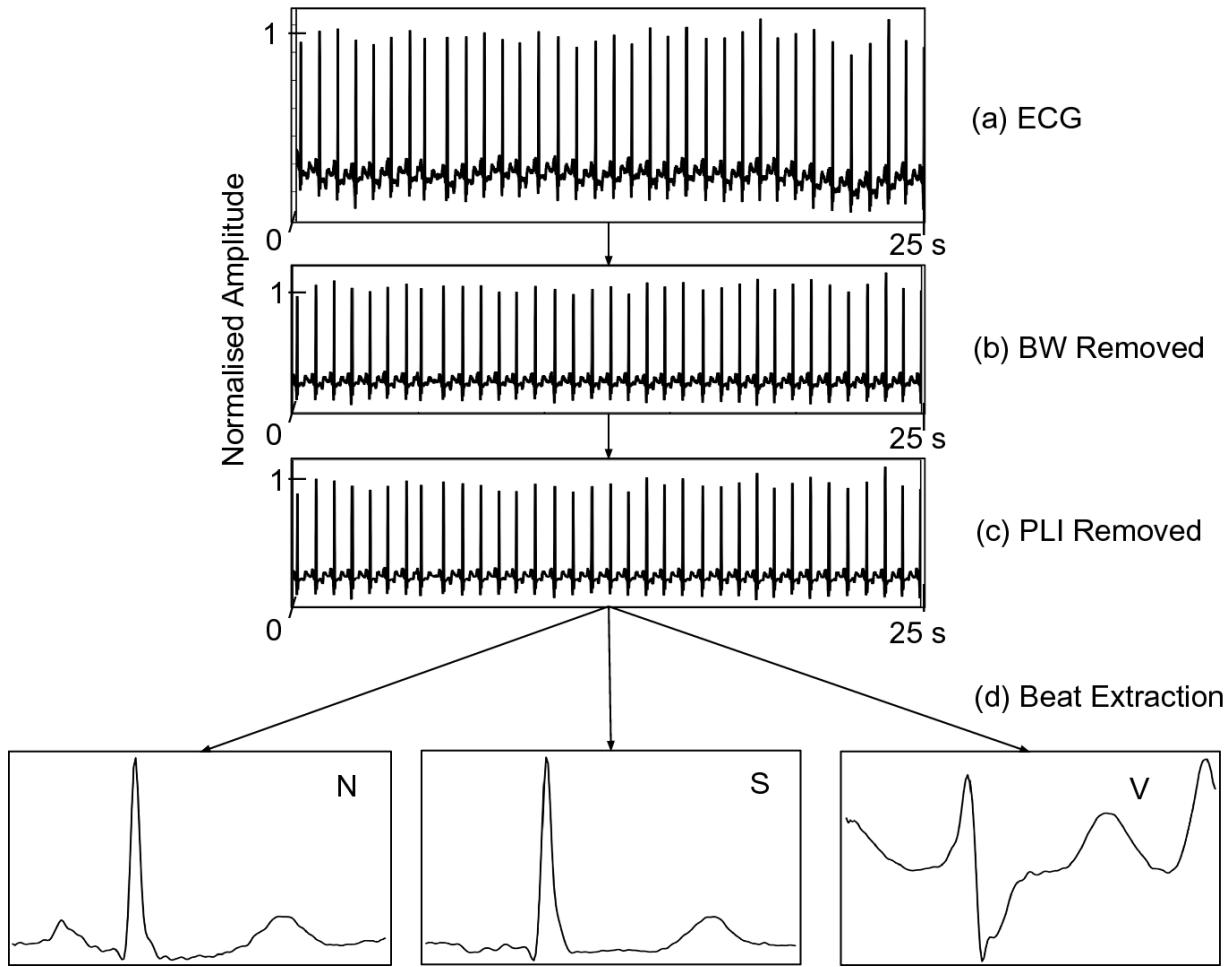


Fig. 12.2 Preprocessing a 25 s ECG signal. **a** Noisy single lead ECG; **b** Baseline Wander removed; **c** Power Line Interference removed; **d** Beat extraction using annotations provided in the datasets

The extracted beats are split into the train (DS1) and test (DS2) sets. DS1 and DS2 distribution for MIT ARDB [12] and MIT SVDB [62] datasets are shown in Table 12.2. CPSC and INCART data were split in half, and the first half (DS1) was used for training and the other half (DS2) for testing as no split strategy was provided in the literature. DS1 is used for training, and DS2 for testing/inferencing.

Table 12.2 Data distribution for MIT ARDB and MIT SVDB

	MIT ARDB recordings
DS1	101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, and 230

MIT ARDB recordings	
DS2	100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, and 234
MITSVDB recordings	
DS1	802, 804, 805, 808, 810, 812, 841, 843, 845, 849, 866, 871, 873, 876, 877, 886, and 890
DS2	800, 803, 806, 807, 809, 811, 840, 842, 844, 846, 850, 867, 872, 874, 875, 878, and 879

12.3 Methodology

The proposed beat generation and classification workflow are provided in Fig. 12.3. The preprocessed data is split into training, validation, and test dataset as described in Sect. 12.2. The proposed DCCGAN model provided in Sect. 12.3.2 is employed for beat augmentation followed by PCNN classifier described in Sect. 12.3.3. The PCNN classifier is further trained on augmented and original beats and evaluated on the test dataset.

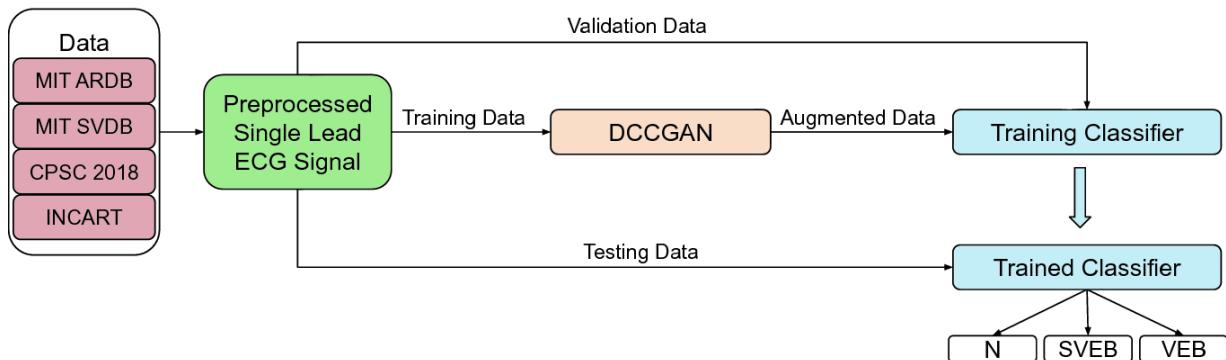


Fig. 12.3 Beat generation and classification workflow

12.3.1 ECG Generation Methodology

The ECG Beat generation framework generates three classes of beats using a convolution based conditional GAN. GANs consist of a generative (probabilistic) model that learns the input probability distribution and a discriminative model that discriminates between the real and generated samples. Generator $G(z)$ maps latent input to the probability distribution of real data P_r . Here, z represents the latent input. Discriminator $D(x)$ classifies the input sample as real or fake, here x represents the input beats provided to the discriminator. Mathematically, GAN training resembles a

two-player minimax game, where the generated data distribution is brought as close as possible to the real data distribution, and the discriminator aims to be better at differentiating between the real and generated beats. GAN optimisation is performed using the loss function defined in Eq. 12.1. Here, z is randomly sampled from synthetically generated data distribution (P_z) and x is randomly sampled from real data distribution (P_x). The former term in Eq. 12.1 $z \in p_z(z)$ predicts that the data is real and the latter term $\log[1 - D(G(z))]$ predicts that synthetically generated data is fake. The training alternates between generator training and discriminator training by keeping the training of other model constant and alternately maximizing and minimizing $V(G, D)$ until synthetically generated data is indistinguishable from real data.

$$*min_G *max_D V(D, G) = E_{x \sim P_r(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))] \quad (12.1)$$

Broadly categorising, GANs can be classified into: Vanilla GAN [21], Conditional GAN [48], and Deep Convolutional GAN (DCGAN) [58]. Vanilla GAN is the basic GAN built using the multi-layer perceptron (MLP) that optimizes the Eq. 12.1 using stochastic gradient descent. The Conditional GAN encodes conditional dependencies in generator and discriminator along with the conventional input allowing for the generation of specific beats. Deep Convolutional GAN (DCGAN) is the most popular and successful GAN that replaces traditional MLP with convolution layers (with strides) without max-pooling layers, making GAN training fast and stable.

12.3.2 Augmentation Using Deep Convolutional Conditional GANs

A class encoded convolution GAN is used to generate class-specific heartbeats (N, SVEB, and VEB). The class information (c) is incorporated with conventional input transforming $G(z)$ to $G(z|c)$ and $D(x)$ to $D(x|c)$ leads to the deep convolution conditional generative adversarial network (DCCGAN). Figure 12.4 illustrates the architecture of the proposed DCCGAN. The architecture details of the generator and discriminator are illustrated in Fig. .

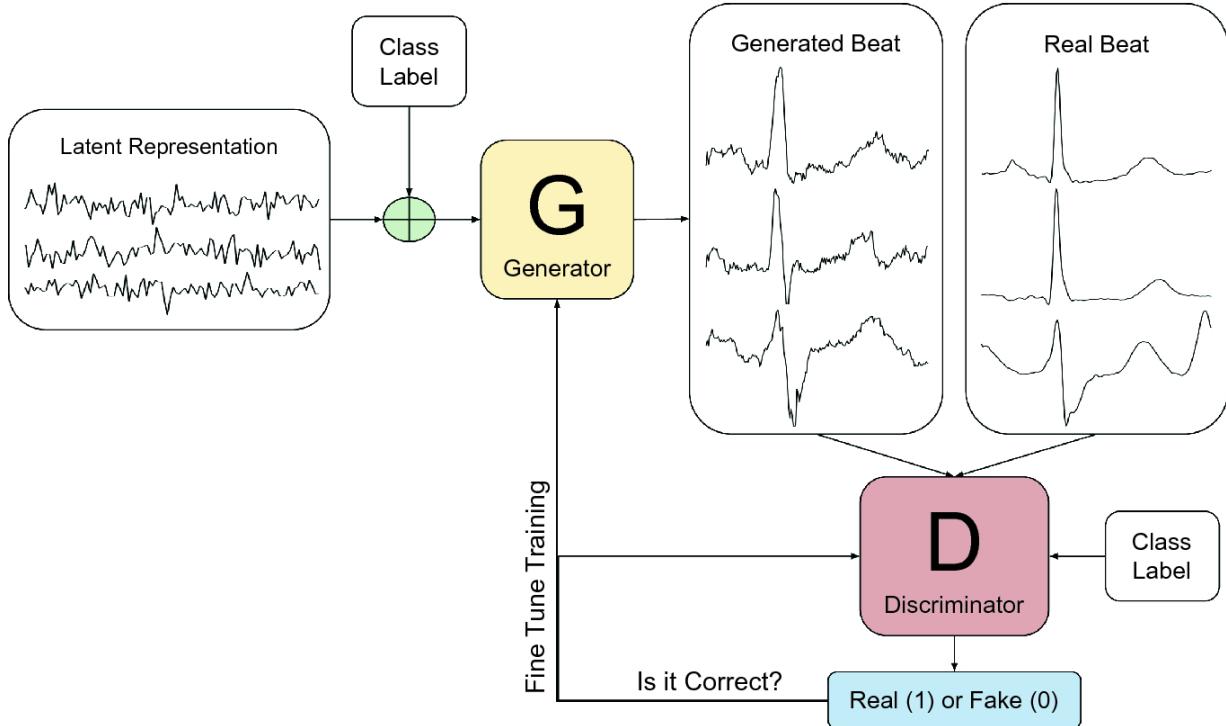


Fig. 12.4 Basic architecture of conditional GAN

12.3.2.1 Modified DCCGAN Loss Function

The proposed DCCGAN takes two inputs, a class label (c) and the real beats corresponding to that class ($x|c$) or the generated beats corresponding to that class ($G(z|c)$). This modifies the basic GAN loss function as described in Eq. 12.1 to DCCGAN loss function in Eq. 12.2. Here, $P_z(z)$ and $P_z(z)$ are the real and generated beat distributions, respectively.

$$*min_G *max_D V(D, G) = E_{x \sim P_r(x)}[\log D(x|c)] + E_{z \sim P_z(z)}[\log(1 - D(G(z|c)))] \quad (12.2)$$

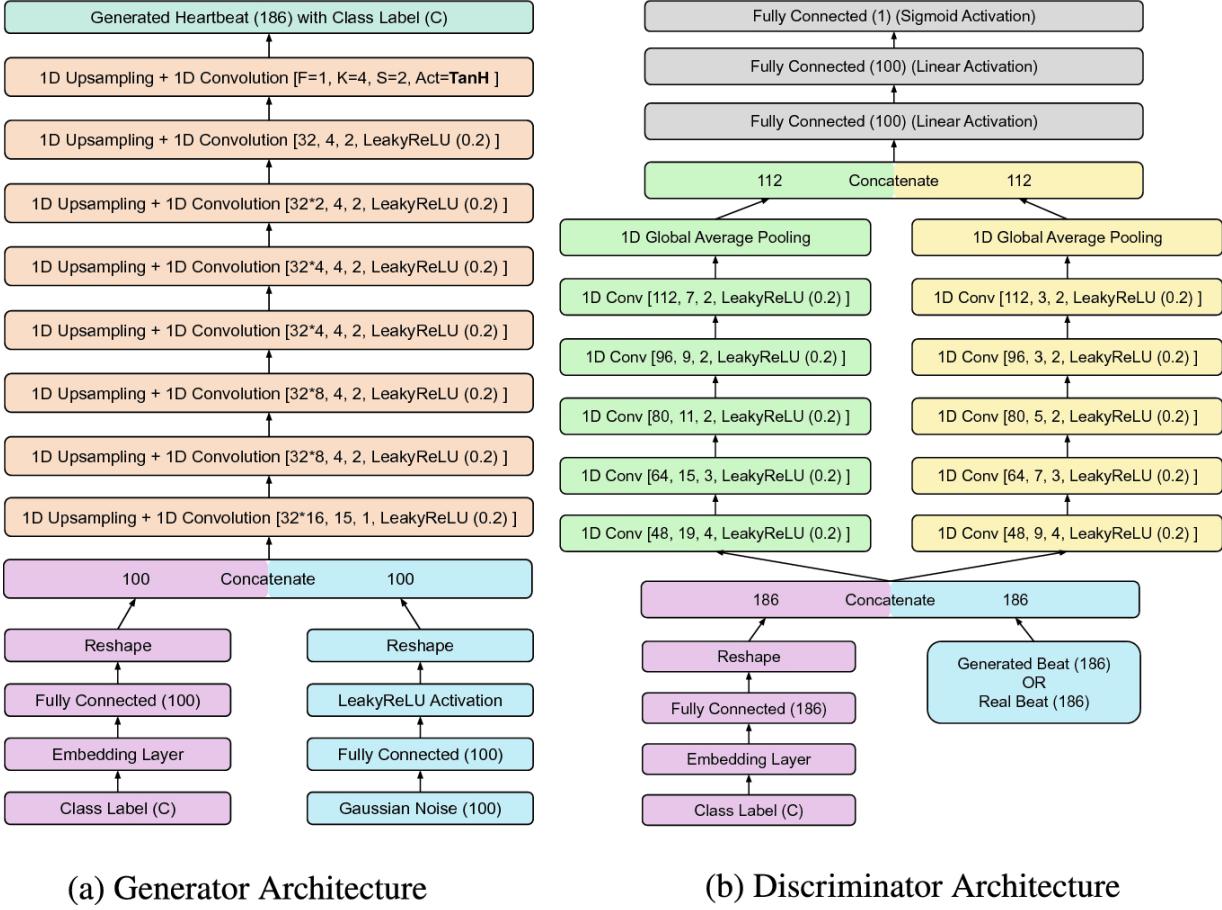


Fig. 12.5 Illustration of generator and discriminator architecture

12.3.2.2 Generator Model

The generator model intakes a latent input vector (z) and class label (c) and generates a fake heartbeat $\mathcal{D} \approx \hat{\mathcal{D}}$, where $z = \{z_n\}_{n=1}^N$, N is the length of noise vector, $G(z_n) \in \mathbb{R}^k$, and k is similar to the dimension of a real heartbeat. The latent input is randomly sampled from a gaussian noise distribution. The generator aims to approximate the underlying real heartbeat distribution given the corresponding class label $P(hb|c)$ and produces fake beats to deceive the discriminator model. The modified generator loss is depicted by Eq. 12.3. It tries to minimize the log probability predicted by the discriminator model for synthetically generated beats, thereby encouraging the generation of those beats with less probability of being fake.

$$L(G) = \text{minimize}[\log(1 - D(G(z|c)))] \quad (12.3)$$

Figure 12.5a depicts the architecture of the generator. Generator model intakes a latent input (z) of length 100 sampled from a gaussian (normal) noise distribution ($\mathcal{N}(0, I)$ where \mathbb{R}^{16384} and $O = o$) [39, 53] and a class label (c), where $c \in \{0, 1, 2\}$. Here, (0, 1, 2) corresponds to Normal beat, SVEB, and VEB. The label is incorporated by employing an embedding layer followed by a fully connected layer with 100 neurons, similar to input shape, and later reshaped to match the dimensions of gaussian input noise. The Gaussian noise is provided to a fully connected layer with 100 neurons followed by LeakyReLU activation function with a negative slope coefficient of 0.2 followed by a reshaping layer for concatenation with the encoded class information. The input noise and class label are then concatenated and provided to a 1-Dimensional (1D) Upsampling layer that increases the dimension of input by a factor of 2 followed by a 1D convolution layer with 32×32 filters of size 15 and stride 1 with LeakyReLU activation function with a negative slope coefficient of 0.2. Six more Upsampling and Convolution layers are added in a cascaded fashion for making the generator model as illustrated in Fig. 12.5a. The combination of the 1D Convolution layer and 1D Upsampling layer imitates a 1D Deconvolution layer. In the penultimate layer, the activation function is changed from LeakyReLU to hyperbolic tangent function, and the generated heartbeat with 186 dimensions, in practice, should resemble a realistic-looking heartbeat of class (c).

12.3.2.3 Discriminator Model

The discriminator model intakes real heartbeats (x), and synthetically generated heartbeats ($G(z|c)$) along with their corresponding class label (c) and classifies them as either real or fake. Here, x is represented as $x_n \in \mathbb{R}^T$, $G(z)$ is represented as $G(z_n) \in \mathbb{R}^T$, T corresponds to the beat dimension. The modified discriminator loss is described using Eq. 12.4. The loss function aims at maximizing the log probability of real beats and inverse log probability of synthetically generated beats.

$$L(D) = \text{maximize}[\log D(x|c) + \log(1 - D(G(z|c)))] \quad (12.4)$$

Figure 12.5b depicts the architecture of the discriminator. The discriminator model intakes the heartbeat (both real and fake) (x) of 186 length and a

class label (c), where $c \in \{0, 1, 2\}$. Here, (0, 1, 2) corresponds to Normal, SVEB, and VEB class of beats. The class label is incorporated in a similar fashion as described in the generator model. The input beats and reshaped embedded class label are further concatenated, and a parallel convolution neural network is employed. The main idea behind applying such convolution layers in parallel is that the beats of 186 dimension consist of both local and global patterns. The global patterns are extracted using the large kernels embedded in the left part of PCNN, and the local patterns are extracted using the small kernels embedded in the right branch of PCNN. The number of filters, stride, and activation function in each adjacent convolution layer are kept similar, and only the kernel size is varied. For instance, the first convolution layer in the left branch after input concatenation encompasses a 1-Dimensional convolution layer with 48 filters of size 19 and stride 4 with LeakyReLU activation function with a negative slope coefficient of 0.2. Four more convolution layers are added in a cascaded fashion in both branches, followed by 1-Dimensional global average pooling (GAP) layer [38].

GAP calculates the spatial average of filters, making it robust to spatial translations present in input ECG beats. The advantages of GAP over the combination of flatten and fully connected layer are (i) less prone to overfitting; (ii) no dependency on external regularization as it behaves like a structural regularizer; (iii) no trainable parameters similar to max-pooling layer. The reduced parameters lead to significantly faster training and reduced model size, making it suitable for mobile devices. The GAP layer reduces the last layer dimensions from (2, 112) to (112). These outputs are further concatenated and provided as input to two fully connected layers with 100 neurons, each with a linear activation function followed by a single neuron layer with the sigmoid activation function that generates a value (v), where $z \in p_z(z)$. The value (v) is thresholded and a beat is classified as fake if $\exists g^r(\mathbf{x}) > 0$ and real if $I(c; \mathcal{G}(z, c))$.

12.3.2.4 DCCGAN Training

The DCCGAN training alternates between generator and discriminator training. During the discriminator training, n samples are randomly sampled from real heartbeats and n synthetic heartbeats are generated using the

generator network for the three corresponding classes. For the real heartbeats, the discriminator is provided with labels in the range [0.8, 1], and for synthetic heartbeats, the discriminator is provided with labels in the range [0, 0.2]. It can be observed that the labels are modified from 0 and 1 to an interval using the concept of soft labels allowing a faster convergence in the discriminator loss during initial batches of training.

After training the discriminator, the generator model is trained where the discriminator model weights are frozen so that they do not get affected during generator training. During generator training, $S = 1$ samples are randomly sampled from the Gaussian noise distribution and provided to the generator in addition to the class information allowing the generator to generate fake heartbeat samples. These fake and real heartbeats are provided as input to the discriminator with the corresponding class labels. Again soft labels are provided as input to the discriminator, i.e., labels in the range [0.8, 1] for real beats and labels in the range [0, 0.2] for synthetic beats. The discriminator performs a forward propagation and compares the predicted label with soft labels and backpropagates the error to the generator network. During the backpropagation, the discriminator weights are not changed, and the generator weights are modified, thereby training the generator. Therefore, the generator network is trained in an adversarial fashion using the discriminator.

This training sequence is followed for training both the networks alternatively for several batches, and their losses are recorded. The training is not stopped following any early stopping criteria as this remains an open problem in the research area of GANs and because the GAN training is unstable. However, the generator models were saved after every second batch. The training aims to achieve Nash Equilibrium by approximating generator probability distribution $P_z(z)$ to the real heartbeat distribution. Adam optimizer [31] is employed during the training with $LR = 0.0002$ as suggested in [58] along with batch-wise training. More filters with large sizes are preferred for generator and discriminator as larger filters cover more signal timestamps and account for more information present in the beat and previous filter. It also allows for maintaining smoothness in the information present in the filters.

Moreover, DCCGAN took several hundred batches of training before generating any meaningful beats for the corresponding class. In initial batches, random noise signals were generated, and later, realistic-looking beats were generated. For this reason, early stopping criteria was not used as sometimes the generator model was unable to generate meaningful beats after several hundred batches of training. For some models, the discriminator loss approached 0, and the generated beats resembled random noise-looking signals. Therefore, the training was restarted after modification of certain parameters such as the kernel size or adding or removing certain layers.

12.3.3 ECG Classification Methodology

An ideal heartbeat classifier should correctly perform disease diagnosis of a new patient without their prior information. It should be able to cope with the inter-patient variability of their ECG signal as similar cardiac abnormalities might have different waveforms for different patients, and different cardiac diseases may have similar waveform resemblance [49]. Figure 12.6 depicts the architecture of a parallel convolution Neural Network (PCNN) classifier. The beats are provided as input to the PCNN, and a similar structure is followed as described in the discriminator in Sect. 12.3.2.3. The LeakyReLU is replaced by the ReLU activation function that introduces non-linearity in the network, and the last layer has three output neurons with a softmax activation function with categorical cross-entropy loss. Since the beat could only belong to a single class, the output probabilities of PCNN predict that the beat label could belong to only one of the three beat classes. Adam optimizer [31] is employed during the training with $LR = 0.0002$ as suggested in [58].

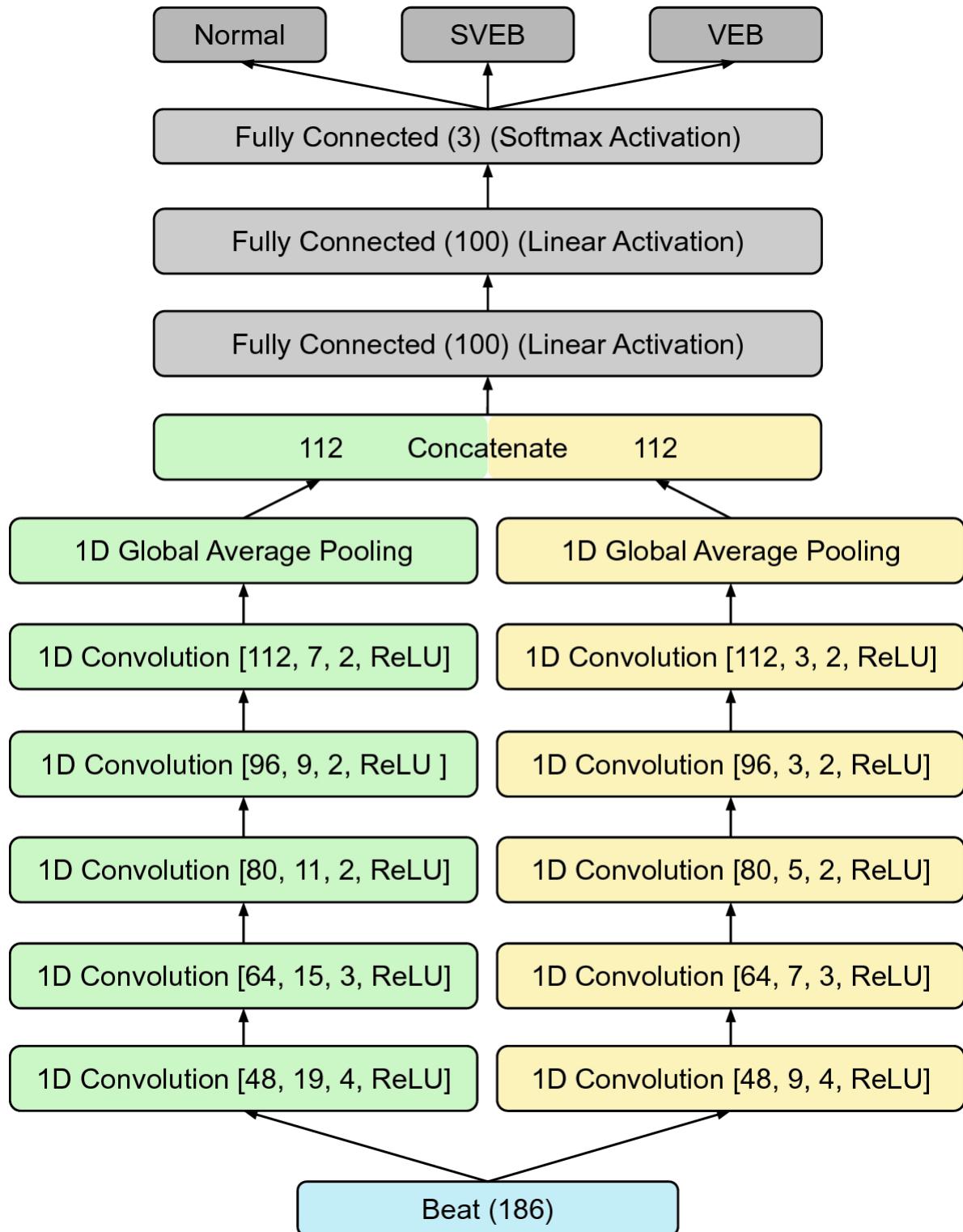


Fig. 12.6 Architecture of the parallel convolution neural network classifier

12.4 Results and Discussion

The results describe the evaluation metrics adopted to evaluate DCCGAN performance and PCNN classification performance in Sect. 12.4.2. Section 12.4.3.1 and 12.4.3.2 describes the quantitative and qualitative performance of DCCGAN. Section 12.4.4 depicts the classification performance using augmented beats from DCCGAN, several other conventional techniques, and without beat augmentation.

12.4.1 Computing Platform

The experiments were performed on a workstation with Intel Core i5-6500 CPU with 16 GB RAM and an Nvidia Geforce GTX 1050 Ti GPU with 4 GB of VRAM. The models were developed using Keras and Tensorflow.

12.4.2 Evaluation Metrics

DCCGAN performance is measured by the quality and diversity of the generated signals obtained from the generator model. Beats generated from the generator and randomly sampled from the real dataset are evaluated on five metrics, namely, Frechet Distance (FD), Dynamic Time Warping (DTW), Maximum Mean Discrepancy (MMD), Root Mean Square Error (RMSE), and Time Warp Edit Distance (TWED). A lower score of these metrics depicts similarity in real data distribution (P_z) and generated data distribution (P_g). Details of each evaluation metric are provided as follows.

Frechet distance [28] is adopted to measure the difference between real data distribution (P_z) and generated data distribution (P_g). Assuming the distributions to be multivariate gaussian distributions with mean $|\mathbf{V}|$ and covariance matrix $|\mathbf{V}|$ for real data distribution and mean (P_g) and covariance matrix (σ_g) for generated data distribution, the distance can be calculated using Eq. 12.5. Tr refers to the sum of elements along the diagonal of square matrix.

$$d^2((\mu_r, \sigma_r), (\mu_g, \sigma_g)) = \|\mu_r - \mu_g\|^2 + Tr(\sigma_r + \sigma_g - 2 \times \sqrt{\sigma_r \times \sigma_g}) \quad (12.5)$$

Maximum Mean Discrepancy [22] tests the similarity between the distribution P_r and χ^2 . n samples are independently and identically drawn from the two distributions. Smaller MMD corresponds to high similarity of the two distributions. Gaussian MMD is adopted that is calculated using Eq. 12.6. Here, x and y denotes samples drawn from P_r and χ^2

respectively, $\kappa(x, x') = \sum_{j=1}^k e^{-\alpha_j \|x-x'\|^2}$ with bandwidth ω denotes pairwise distance between the joint data.

$$MMD^2 = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq 1}^n \kappa(x_i, x_j) - \frac{2}{n^2} \sum_{i=1}^n \sum_{j=1}^n \kappa(x_i, y_j) + \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq 1}^n \kappa(y_i, y_j) \quad (12.6)$$

Root Mean Square Error quantifies the distance between generated signal x and real signal y . It can be calculated using Eq. 12.7. Here, N represents the number of samples in each signal (beats from both distributions are of equal length).

$$RMSE(x, y) = \sqrt{\frac{1}{N} \sum_{k=1}^N (x_k - y_k)^2} \quad (12.7)$$

Dynamic Time Warping Since the generated heartbeats might suffer from the condition of bradycardia or tachycardia, meaning slow or fast pace than regular rhythms, DTW is introduced to estimate dissimilarity between the generated and real heartbeats [59]. DTW aligns the two beats and calculates the distance between generated (x) and real (y) heartbeats with length (N). An optimised version of DTW, namely, Fast DTW, approximates DTW and reduces the computational complexity to $O(N)$ time [60]. Fast DTW is calculated using Eq. 12.8. Here, $f(x_i, y_j) = (x_i - y_j)^2$.

$$D_{i,j} = f(x_i, y_j) + \min\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\} \quad (12.8)$$

Time Warp Edit Distance It measures the minimum cost sequence of “edit operations” required to transform the generated beat to the real beat. Edit operations are defined using a graphical editing process and produce a dynamic programming recursive algorithm called TWED [43]. It highlights a parameter that drives the stiffness $\hat{\sigma}_{y^r}$ of the elastic measure. TWED between real beat x and generated beat y of length N is calculated using Eq. 12.9. Here, λ represents a mismatch penalty.

$$g(x_i, y_j) = \min\{g(x_i, y_j) + \delta(x_i, y_j), g(x_{i-1}, y_j) + \delta(x_i), g(x_i, y_{j-1}) + \delta(y_j)\} \quad (12.9)$$

Here, $g(x_0, y_0) = 0$

$$\delta(x_i, y_j) = cost(x_i, y_j) + cost(x_{i-1}, y_{j-1}) + 2\nu \cdot |i - j|$$

$$\delta(x_i) = cost(x_i, x_{i-1}) + \lambda + \nu$$

$$\delta(y_j) = cost(y_j, y_{j-1}) + \lambda + \nu$$

$$g(x_i, y_0) = g(x_0, y_j) = \infty$$

$$cost(x_i, y_j) = |x_i - y_j|^2$$

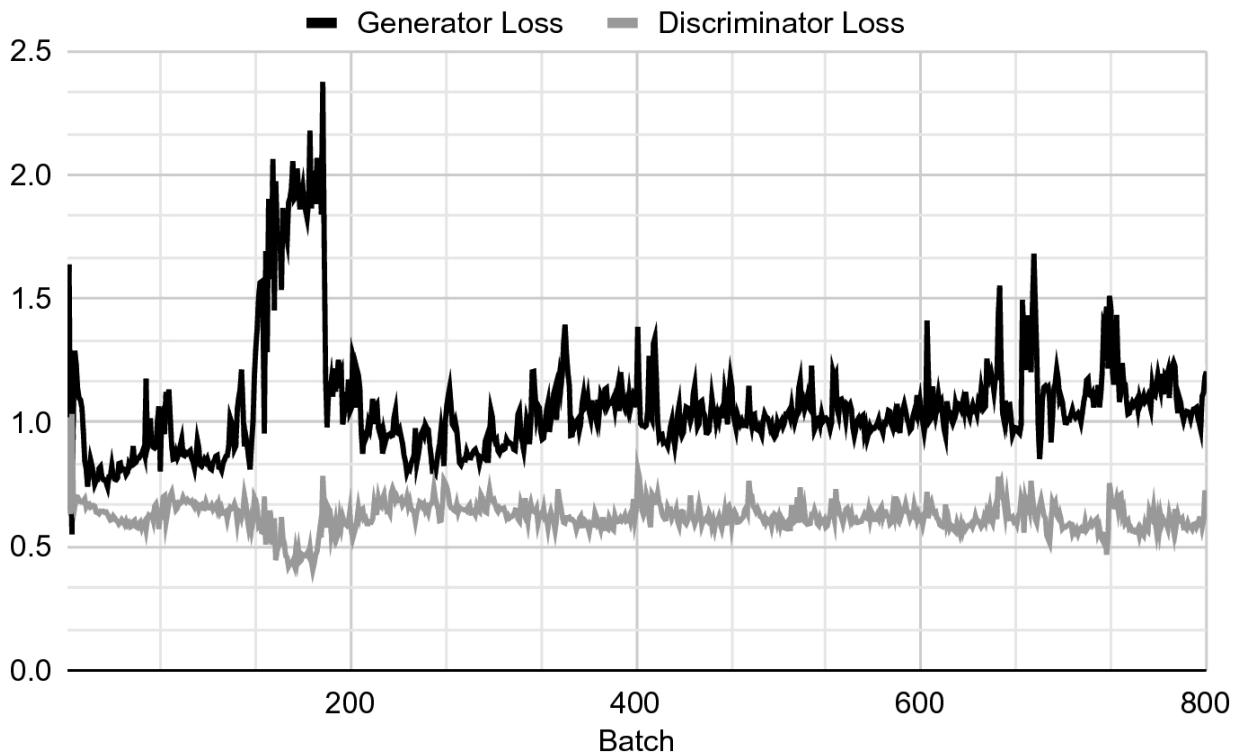


Fig. 12.7 Generator and discriminator loss during training

The dissimilarity value $\mathbf{z} \sim P(\mathbf{Z})$ between P_r and χ^2 is defined as the TWED metric. Twenty four possible constant combinations for $\lambda = [0, 0.2, 0.4, 0.6, 0.8, 1]$ and $(\alpha = 0.0002, \beta_1 = 0.5)$ were computed to obtain a series of TWED values. Minimum of these values is used as the final value.

Classification Metrics The diagnostic performance of the PCNN classifier is evaluated using Accuracy (Acc), Sensitivity (Se), Specificity

(Sp), and Precision (Pr) as described in Eq. 12.10. Here, TP, FN, TN, and FP represent True Positive, False Negative, True Negative, and False Positive. Sensitivity measures the fraction of SVEB, VEB beats correctly classified as SVEB, VEB. Specificity measures the fraction of N beats that are correctly identified as N. Accuracy corresponds to the fraction of rhythms that are correctly classified, irrespective of the beats.

$$\begin{aligned}
 \text{Accuracy (Acc)} &= \frac{TP + TN}{TP + FP + FN + TN} \\
 \text{Sensitivity (Se)} &= \frac{TP}{TP + FN} \\
 \text{Specificity (Sp)} &= \frac{TN}{TN + FP} \\
 \text{Precision (Pr)} &= \frac{TP}{TP + FP}
 \end{aligned} \tag{12.10}$$

12.4.3 DCCGAN Performance Evaluation

DCCGAN performance is evaluated both quantitatively and qualitatively. The real beats of N, SVEB, and VEB classes are randomly sampled from the training data, where equal beats of each class are extracted so that model has enough information about each class of beats. The fake beats are generated from the generator using the Gaussian noise as latent input and class labels. An equal number of fake beats of each class are generated so that discriminator does not get biased during the training. The DCCGAN is trained for 800 batches, and the generator loss, discriminator loss, and evaluation metrics for all classes are recorded. The models are saved after every second epoch during training as the training is unstable, and any intermediate model could generate the best quality of beats. Figure 12.7 depicts the generator and discriminator loss during DCCGAN training. It can be observed that multiple local minima are obtained by generator loss at batches 255, 525, 649, 663, and 757. The beats generated by the generator model at these batches depict a perfect resemblance with the real beats present in the training dataset. So, we perform a detailed analysis of the generator models saved at these particular batches.

12.4.3.1 Quantitative Evaluation of DCCGAN

Five evaluation metrics are monitored during the training of the DCCGAN model, as mentioned above. The TWED metric is computationally expensive, and therefore, only 200 real and generated samples of each class are compared against each other after every two batches. The average of each evaluation metric is provided in Fig. 12.8. The individual evaluation metric for Normal, SVEB, and VEB class is provided in Figs. 12.9, 12.10, and 12.11. The overall evaluation metric and class respective metrics also display a similar pattern to training loss curves.

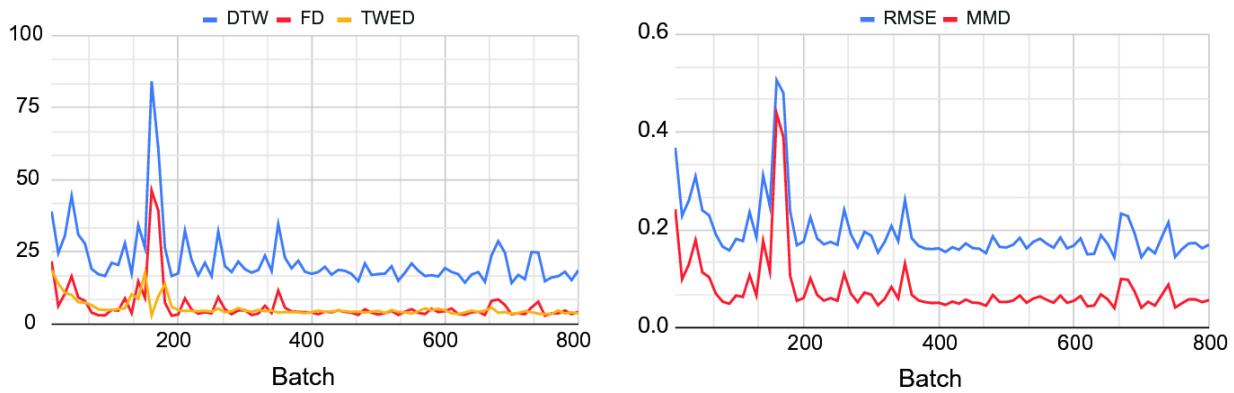


Fig. 12.8 Overall Evaluation Metrics during DCCGAN training

The normal class evaluation metrics in Fig. 12.9 depicted a larger variation than the irregular beat evaluation metrics. The reason behind this might be the random undersampling performed for the normal class during data preparation. The Normal class beats were reduced as they were around 80% of the total beats, whereas SVEB and VEB contributed nearly 20% of total beats. The reduction in Normal beats might have reduced the variation present in the patient beats, thereby increasing the error in evaluation metrics.

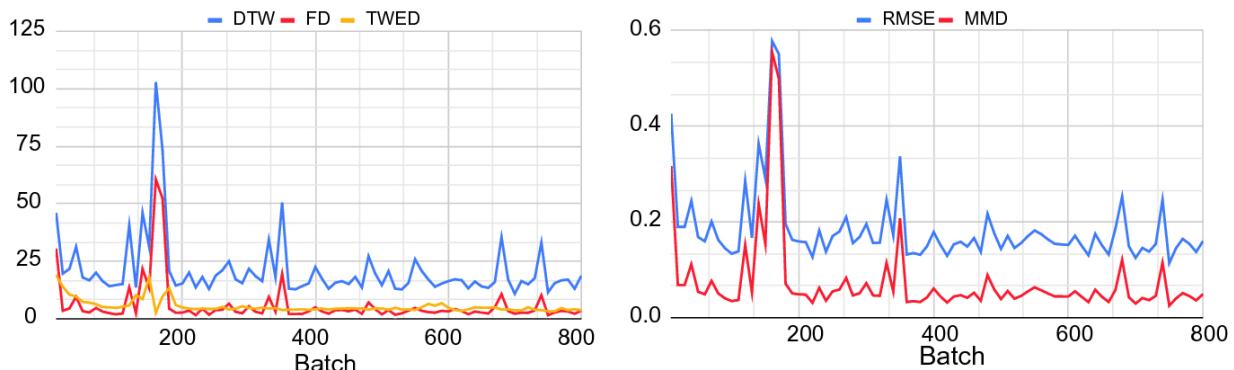


Fig. 12.9 Evaluation metrics for normal class during DCCGAN training

SVEB and VEB class evaluation metrics in Figs. 12.10 and 12.11 depicted comparatively less error than Normal beats. The reason might not be because the model produced a good variation in irregular beats but because SVEB and VEB might have less variation in irregularities in the actual training dataset. Therefore, the generated beats could easily adapt to the variation present in SVEB and VEB class beats.

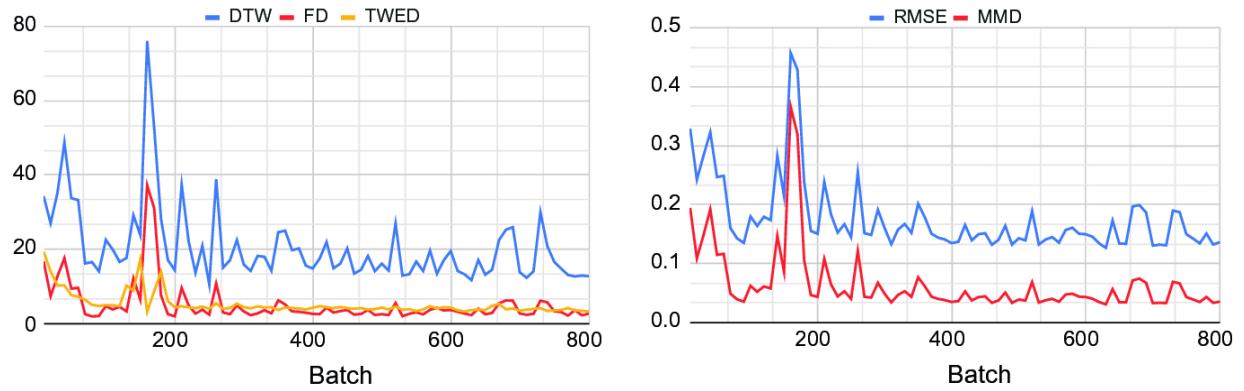


Fig. 12.10 Evaluation metrics for SVEB class during DCCGAN training

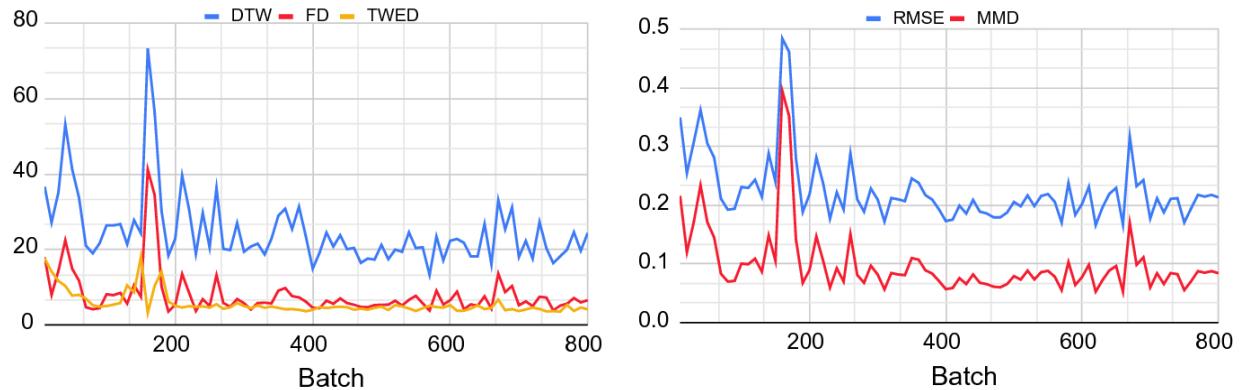


Fig. 12.11 Evaluation metrics for VEB class during DCCGAN training

Similarly, less error in evaluation metric and less generator loss are observed in Figs. 12.7, 12.8, 12.9, 12.10, and 12.11 for batches 255, 525, 649, 663, and 757, respectively. Therefore, the models saved at these batches are used to generate data for heartbeat classification. The model with lowest error in evaluation metrics is used to depict the generated beat quality.

12.4.3.2 Qualitative Evaluation of DCCGAN

Out of the selected batches, the least error in evaluation metrics is obtained for batch 649. The model saved at batch 649 is used to generate beats of Normal, SVEB, and VEB class. Figure 12.12 illustrates original beats from DS1 and GAN Augmented beats from generator model saved at batch 649. Figure 12.12a–c describe normal, SVEB, and VEB class beats from training set of DS1. Figure 12.12d–f describe the generated Normal Beat, Fig. 12.12g–i describe the generated SVEB, and Fig. 12.12j–l describe the generated VEB. The high error in evaluation metrics of normal beats might be due to irregularity or slight problem in generation of R-wave of QRS complex, i.e., the ventricular depolarisation of the heart. The other characteristic waves, such as P-wave and T-wave, resemble the real normal beats. The generated SVEB depicts a clear absence of P-wave and narrow QRS complex resembling real SVEB. This might be the reason for less error in evaluation metrics of SVEB. The generated VEB depicts an abnormal QRS complex with prolonged duration and elevated ST segment with a dominant S wave. It also shows an inverted or retrograde P-wave. This might be the reason for less error in the evaluation metrics of VEB.

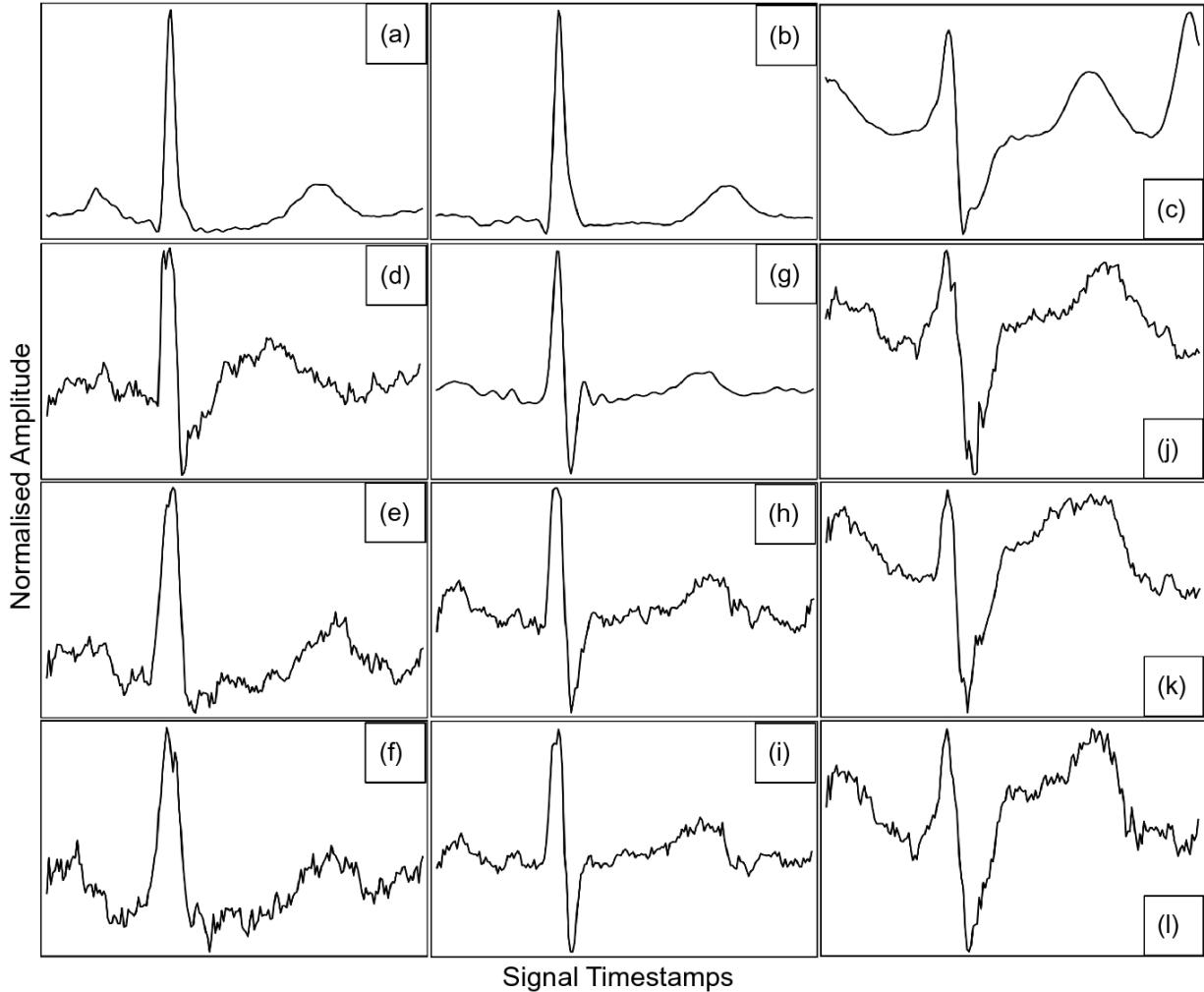


Fig. 12.12 Illustration of Original beats from DS1 and GAN Augmented beats from generator model saved at batch 649. **a** Normal Beat, **b** SVEB, **c** VEB; **d–f** Generated Normal Beat, **g–i** Generated SVEB, **j–l** Generated VEB

Figure 12.13 illustrates the incorrect beat generation by the generator model from the models saved at different batches. Figure 12.13a–b describe incorrectly generated normal beat, Fig. 12.13c–d describe incorrectly generated SVEB, Fig. 12.13e–f describe incorrectly generated VEB. The beats of all three classes do not resemble the actual beats of the respective class. Moreover, the generated beats are contaminated with high frequency noises.

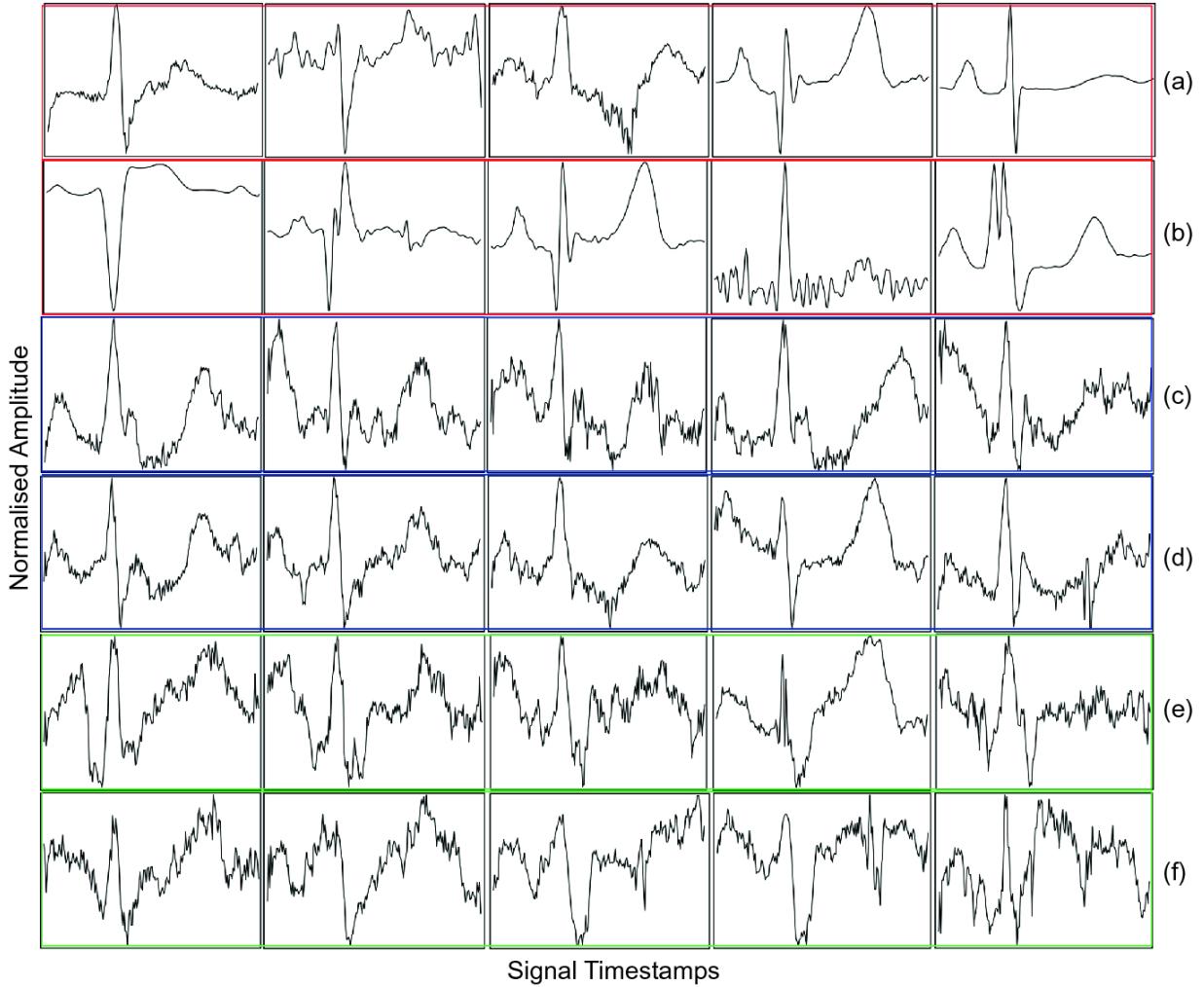


Fig. 12.13 Illustration of incorrect beat generation by the generator model. **a–b** Normal Beat, **c–d** SVEB, and **e–f** VEB

12.4.4 Beat Classification Performance

The robustness of the proposed generative model is also verified by training and testing the PCNN classifier with synthetically generated beats from the DCCGAN model. Several other augmentation techniques are also employed for comparison purposes. The number of beats of each class used for model training, validation, and testing is provided in Table 12.3. The other augmentation (oversampling) techniques for comparison include Synthetic Minority Oversampling Technique (SMOTE) [9], Adaptive Synthetic Sampling Approach (ADASYN) [26], Borderline-SMOTE SVM [55], and hybrid of SMOTE and Tomek Links (SMOTOMEK) [5]. SMOTE generates minority class beats by interpolating between nearby beats of the

corresponding class. ADASYN takes into consideration the density distribution of beats for generating new beats of the corresponding class. SVMSMOTE augments only those beats of SVEB and VEB classes that are misclassified by a predetermined classifier (SVM or KNN), providing better resolution at the decision boundaries (as misclassified beats lie close to the decision boundary) and is therefore called borderline SMOTE. This assures that the augmented beats are not generated blindly and only generated along the decision boundary between two classes of beats. SMOTOMEK balances the majority and minority class distribution by using TOMEK links and SMOTE alternatively. Initially, SMOTE is used to augment the minority class beats followed by Tomek Links that identify nearest beats of different classes and removes either beat of both classes or only the majority class beat making decision boundary of training data less ambiguous.

Table 12.3 Data instances

Data instances	Normal	SVEB	VEB
Testing data (DS2)	155008	12564	35615
Validation (From DS1)	31181	2358	7416
Training (From DS1)	125561	9305	28954
Other augmentation (Using DS1)	125551	125559	125553
GAN augmentation (Using DS1)	130681	130649	130842

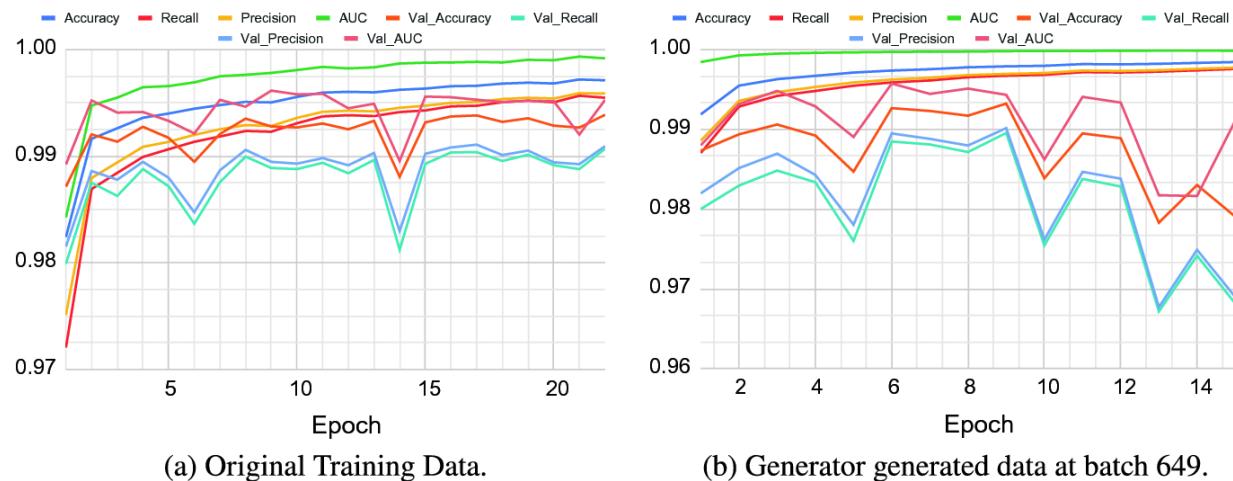


Fig. 12.14 History of metrics monitored during training of heartbeat classifier

The training data (DS1) was split into train and validation data with a 80 : 20 split. The training data was provided to the conventional augmentation techniques, such as SMOTE and its variants, and the DCCGAN. The augmented data and original training data were then independently used to train the PCNN for 50 epochs. An early stopping criterion was set with a patience of 6 epochs such that if the validation Area under the receiver operating characteristic curve (AUC) did not increase, the training was halted. Several other metrics were monitored during PCNN training, such as Accuracy, Recall, and Precision. Figure 12.14 illustrates various metrics during the training of PCNN for non-augmented training data in Fig. 12.14a and DCCGAN augmented training data from model saved at batch 649 in Fig. 12.14b. PCNN training for original train data halted at 22nd epoch as the validation AUC did not improve after 18th epoch, whereas the training for DCCGAN augmented data halted a bit early at 16th epoch as the validation AUC did not improve after the 10th epoch. The models were saved after every epoch if the validation AUC was improved. Therefore, the best model from each configuration was picked and tested on the test dataset (DS2).

Figure 12.15 describes the performance of the individually trained PCNN for all combinations of augmented and non augmented datasets. The data augmented using hybrid model SMOTE, and TOMEK links provided the best result among the conventional augmentation techniques. Its result supersedes the classifier trained on non augmented dataset. The generator model saved at batch 255, 525, 649, 663, and 757 are further used to generate N, SVEB, and VEB class beats are used to train the PCNN model.

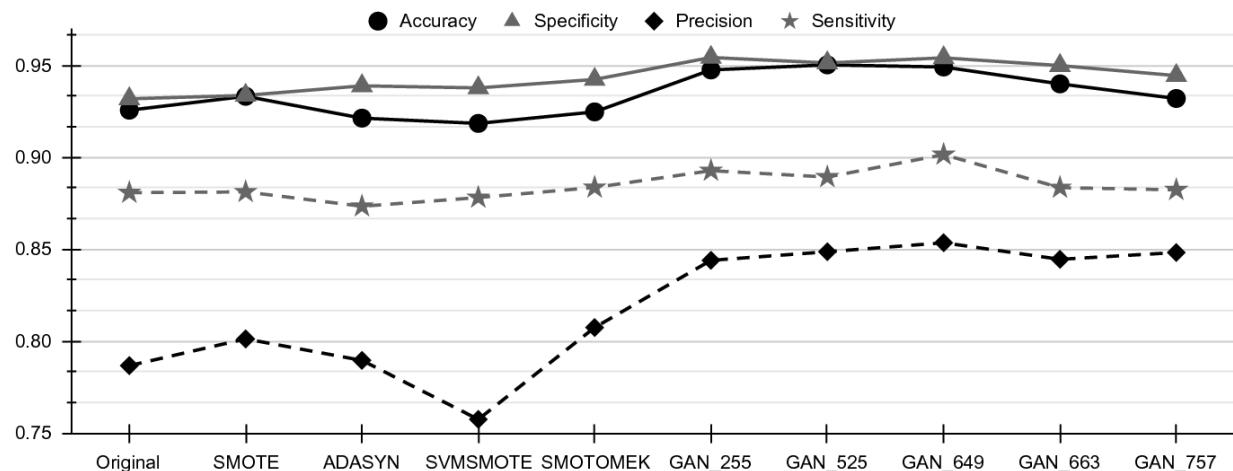


Fig. 12.15 Classification performance of parallel convolution neural network on augmented and non augmented datasets

Initially, the generated batch data provided better performance up to batch 649, and later the performance deteriorated. It might happen because the generator might have overfitted and tried to generate the beats that resemble the beats present in the training dataset, thereby reducing the variation in generated beats. The number of samples generated from each model was kept equal so that there is no data scarcity for the classifier. The model saved at batch 649 provided the best result among all configurations. It achieved an accuracy of 0.950, specificity of 0.955, precision of 0.853, and sensitivity of 0.901. The model trained with original data achieved an accuracy of 0.926, specificity of 0.932, precision of 0.787, and sensitivity of 0.881. The model trained with SMOTE and TOMEK link data achieved an accuracy of 0.925, specificity of 0.942, precision of 0.807, and sensitivity of 0.884.

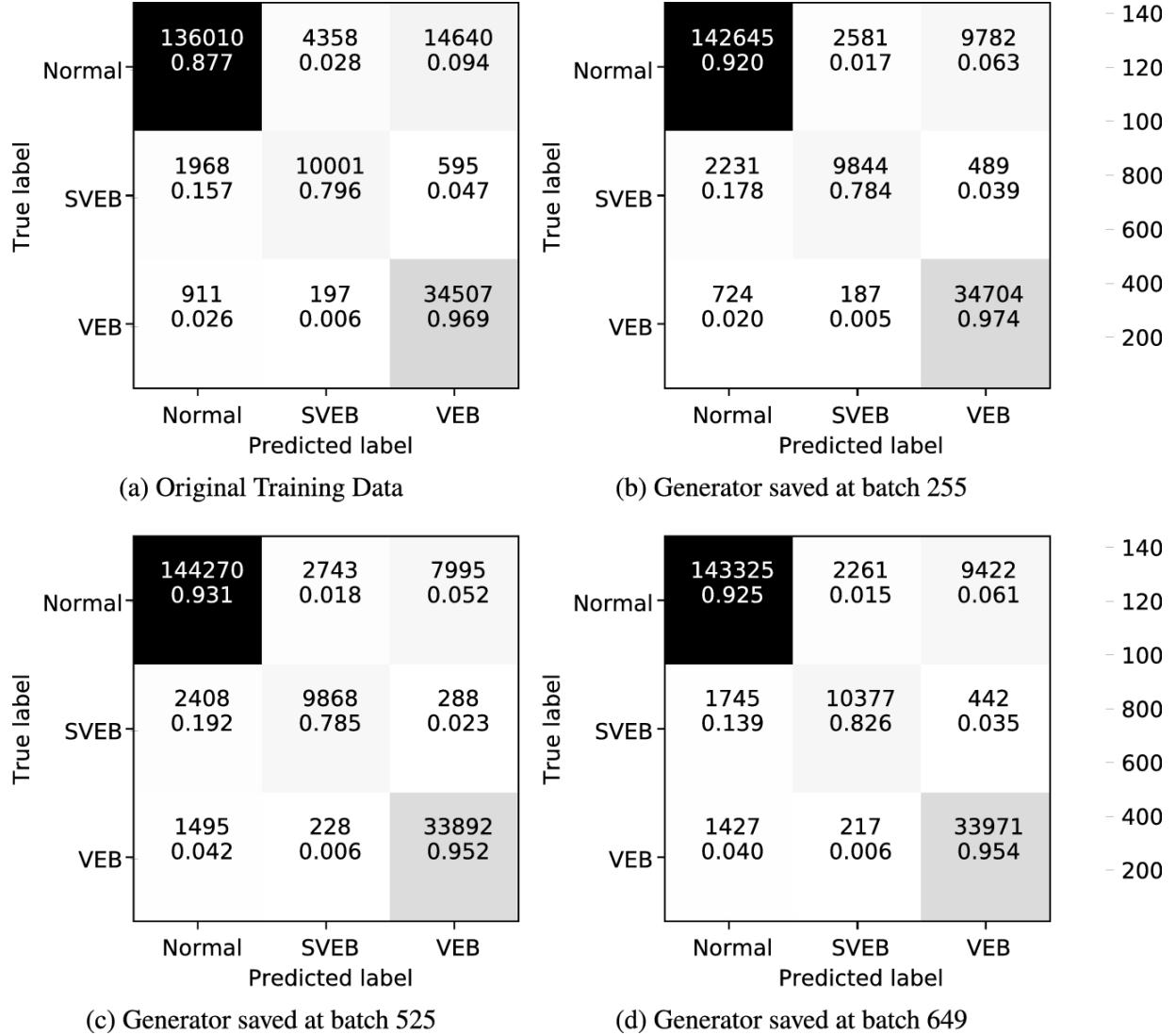


Fig. 12.16 Illustration of confusion matrix for classification

The confusion matrix of the three top-performing generative models and the model trained on original data is provided in Fig. 12.16. The three top-performing models are trained with GAN augmented data, namely, the generator model saved on batch 255, 525, and 649. The three models performed better than the model trained on original data. Moreover, each model could be useful as each model produced maximum percent of true positives for only class, i.e., model saved at batch 255 produced the best score for VEB class beats with 0.974, model saved at batch 525 produced the best score for Normal class beats with 0.931, and model saved at batch 649 produced the best score for SVEB class beats with 0.826. However, the

model saved at batch 649 produced the best overall score for the classification evaluation metrics.

Table 12.4 Comparison of proposed methods with previous methods on the basis of different evaluation metrics. HBF: Hermite basis function; HOS: Higher order statistics; LBP: Local Binary Pattern; SVM: Support vector machine

Paper	Data	Features	Class	Se	Sp	Pr	Ac
Alfaras et al. [2]	MIT ARDB AHADB	RR interval and morphological features, echo state networks	All	0.82	0.98	0.85	0.97
Das et al. [11]	MIT ARDB	Stockwell transform, bacteria	S	0.74	0.98	0.73	0.97
	INCART	foraging optimisation algorithm, SVM	V	91	0.99	0.91	0.98
De et al. [12]	MIT ARDB	Morphological features, RR intervals	S	0.76	–	0.38	–
			V	0.77	–	0.82	–
De et al. [13]	MIT ARDB	RR intervals, morphological features, HBF, HOS with weighted conditional random fields	N	–	–	–	0.76
			S	–	–	–	0.80
			V	–	–	–	0.86
Mar et al. [40]	MIT ARDB	71 features including Temporal, Morphological, Statistical features	N	0.89	–	0.99	–
			S	0.83	–	0.33	–
			V	0.86	–	0.75	–
Mondejar et al. [50]	MIT ARDB	Wavelet, LBP, HOS, RR interval with ensemble of SVM	N	0.96	–	0.98	–
			S	0.78	–	0.50	–
			V	0.95	–	0.94	–
Yesh et al. [68]	MIT ARDB	Clustering using Temporal and morphological features	S	0.54	0.84	0.23	–
			V	0.43	0.82	0.16	–
Proposed	MIT ARDB	Deep convolution conditional GAN	–	–	–	–	–
	INCART	for Beat augmentation	N	0.92	0.93	–	0.93
	MIT SVDB	Parallel convolution neural network	S	0.83	0.99	0.81	0.98
	CPSC 2018	for classification	V	0.95	0.94	0.77	0.94

Comparison with State-of-Art Classifiers: The classification performance is compared with the works performed in the past as shown in Table 12.4. The table depicts the information about the dataset used in each work, the feature extraction method, selection and classification employed by each work, and the performance. It can be observed that the data augmented using DCCGAN improved the model performance and brought it at par with the current state-of-the-art papers. Although the previous works have achieved good performance, the developed techniques and models are tested on limited datasets limiting their use in the real world scenario. All of the performances are presented for the test set (DS2) that is not used for training or data generation purposes making the results unbiased. Alfaras et al. [2] performed testing only on DS2 data extracted from the MITARDB dataset and provided overall results for all evaluation metrics. Several works have shown performance only on SVEB and VEB classes as these are abnormal classes. The precision is usually less than the sensitivity. Moreover, the models have achieved high accuracy, and this measure is biased as the majority of beats in DS2 are normal class beats, whereas irregular rhythms are only a fraction present in the DS2 dataset, making the developed model performance unreliable. The adoption of multiple datasets from different geographical locations such as Boston in the United States of America, Saint Petersburg in Russia, and China makes the proposed model reliable and robust.

12.5 Conclusion and Future Work

In this chapter, a deep convolution conditional generative adversarial network model is proposed for generating different classes of ECG beats, including normal beat and abnormal or irregular beats comprising of supraventricular ectopic beats and ventricular ectopic beats recommended by AAMI. In DCCGAN, binary cross-entropy loss is preferred for discriminator and Gaussian noise as input for generator. Soft labels are preferred for faster convergence of the discriminator model. The results are presented both quantitatively and qualitatively. Quantitatively, the generated beats are evaluated using five quantitative metrics, and qualitatively, the beats generated by the generator model are plotted against the real/original beats to illustrate the similarity in real and synthetically generated beats.

The training curves of DCCGAN depicted stable training, and the change in evaluation metrics followed a similar pattern. Generated synthetic heartbeats resemble real beats as they encompass essential characteristics present in beats and follow the intricate structure present in the different classes of beats. The generated beats are further provided to the parallel convolution neural network for heartbeat classification. The developed classifier performed better than the state-of-the-art models. The proposed DCCGAN and classifier model were verified on four publicly available standard datasets, following an inter-patient division scheme making the models more reliable and robust.

In future, generator model could be extended to generate rarely occurring irregular heartbeats sampled at different frequencies. The trained classifier deployed in automated systems could improve life-threatening morphological arrhythmia detection. The automated system could provide assistance to general physicians and cardiologists in diagnosing cardiovascular diseases with less error and reduced manual effort.

References

1. Al-Fahoum, A., Howitt, I.: Combined wavelet transformation and radial basis neural networks for classifying life-threatening cardiac arrhythmias. *Med. Biol. Eng. Comput.* **37**(5), 566–573 (1999)
2. Alfaras, M., Soriano, M.C., Ortín, S.: A fast machine learning model for ecg-based heartbeat classification and arrhythmia detection. *Front. Phys.* **7**, 103 (2019)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, pp. 214–223 (2017)
4. Banerjee, S., Mitra, M.: Ecg beat classification based on discrete wavelet transformation and nearest neighbour classifier. *J. Med. Eng. Technol.* **37**(4), 264–272 (2013)
5. Batista, G.E., Bazzan, A.L., Monard, M.C., et al.: Balancing training data for automated annotation of keywords: a case study. In: WOB, pp. 10–18 (2003)
6. Brophy, E., Wang, Z., Ward, T.E.: Quick and easy time series generation with established image-based gans. [arXiv:1902.05624](https://arxiv.org/abs/1902.05624) (2019)
- 7.

- Cai, Z., Liu, C., Gao, H., Wang, X., Zhao, L., Shen, Q., Ng, E., Li, J.: An open-access long-term wearable ecg database for premature ventricular contractions and supraventricular premature beat detection. *J. Med. Imaging Health Inf.* **10**(11), 2663–2667 (2020)
- 8. Cao, H., Li, H., Stocco, L., Leung, V.: Design and evaluation of a novel wireless three-pad ecg system for generating conventional 12-lead signals. In: Proceedings of the Fifth International Conference on Body Area Networks, pp. 84–90. ACM (2010)
 - 9. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002) [[zbMATH](#)]
 - 10. Clifford, G.D., Nemati, S., Sameni, R.: An artificial vector model for generating abnormal electrocardiographic rhythms. *Physiol. Meas.* **31**(5), 595 (2010)
 - 11. Das, M.K., Ari, S.: Patient-specific ecg beat classification technique. *Healthcare Technol. Lett.* **1**(3), 98–103 (2014)
 - 12. De Chazal, P., O'Dwyer, M., Reilly, R.B.: Automatic classification of heartbeats using ecg morphology and heartbeat interval features. *IEEE Trans. Biomed. Eng.* **51**(7), 1196–1206 (2004)
 - 13. De Lannoy, G., François, D., Delbeke, J., Verleysen, M.: Weighted conditional random fields for supervised interpatient heartbeat classification. *IEEE Trans. Biomed. Eng.* **59**(1), 241–247 (2011)
 - 14. Delaney, A.M., Brophy, E., Ward, T.E.: Synthesis of realistic ecg using generative adversarial networks (2019). [arXiv:1909.09150](#)
 - 15. El Bouny, L., Khalil, M., Adib, A.: Ecg heartbeat classification based on multi-scale wavelet convolutional neural networks. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3212–3216. IEEE (2020)
 - 16. Escalona-Morán, M.A., Soriano, M.C., Fischer, I., Mirasso, C.R.: Electrocardiogram classification using reservoir computing with logistic regression. *IEEE J. Biomed. Health Inform.* **19**(3), 892–898 (2014)
 - 17. Farajzadeh-Zanjani, M., Hallaji, E., Razavi-Far, R., Saif, M.: Generative adversarial dimensionality reduction for diagnosing faults and attacks in cyber-physical systems. *Neurocomputing* **440**, 101–110 (2021)

18. Golany, T., Radinsky, K.: Pgans: Personalized generative adversarial networks for ecg synthesis to improve patient-specific deep ecg classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 557–564 (2019)
19. Golany, T., Radinsky, K., Freedman, D.: Simgans: Simulator-based generative adversarial networks for ecg synthesis to improve deep ecg classification. In: International Conference on Machine Learning, pp. 3597–3606. PMLR (2020)
20. Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation* **101**(23), e215–e220 (2000)
21. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
22. Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., Smola, A.J.: A kernel method for the two-sample-problem. In: Advances in Neural Information Processing Systems, pp. 513–520 (2007)
23. Haradal, S., Hayashi, H., Uchida, S.: Biosignal data augmentation based on generative adversarial networks. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 368–371. IEEE (2018)
24. Hartmann, K.G., Schirrmeister, R.T., Ball, T.: Eeg-gan: Generative adversarial networks for electroencephalographic (eeg) brain signals (2018). [arXiv:1806.01875](https://arxiv.org/abs/1806.01875)
25. Hazra, D., Byun, Y.C.: Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. *Biology* **9**(12), 441 (2020)
26. He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328. IEEE (2008). <https://doi.org/10.1109/IJCNN.2008.4633969>
27. He, H., Tan, Y.: Automatic pattern recognition of ecg signals using entropy-based adaptive dimensionality reduction and clustering. *Appl. Soft Comput.* **55**, 238–252 (2017)
28. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans

trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems, pp. 6626–6637 (2017)

29. Hu, Y.H., Palreddy, S., Tompkins, W.J.: A patient-adaptable ecg beat classifier using a mixture of experts approach. *IEEE Trans. Biomed. Eng.* **44**(9), 891–900 (1997)
30. Javadi, M., Arani, S.A.A.A., Sajedin, A., Ebrahimpour, R.: Classification of ecg arrhythmia by a modular neural network based on mixture of experts and negatively correlated learning. *Biomed. Signal Process. Control* **8**(3), 289–296 (2013)
31. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). [arXiv: 1412.6980](https://arxiv.org/abs/1412.6980)
32. Kiranyaz, S., Ince, T., Gabbouj, M.: Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Trans. Biomed. Eng.* **63**(3), 664–675 (2015)
33. Kiranyaz, S., Ince, T., Gabbouj, M.: Personalized monitoring and advance warning system for cardiac arrhythmias. *Sci. Rep.* **7**(1), 9270 (2017)
34. Korürek, M., Doğan, B.: Ecg beat classification using particle swarm optimization and radial basis function neural network. *Expert Syst. Appl.* **37**(12), 7563–7569 (2010)
35. Lagerholm, M., Peterson, C., Braccini, G., Edenbrandt, L., Sornmo, L.: Clustering ecg complexes using hermite functions and self-organizing maps. *IEEE Trans. Biomed. Eng.* **47**(7), 838–848 (2000)
36. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4681–4690 (2017)
37. Li, H., Yuan, D., Ma, X., Cui, D., Cao, L.: Genetic algorithm for the optimization of features and neural networks in ecg signals classification. *Sci. Rep.* **7**, 41011 (2017)
38. Lin, M., Chen, Q., Yan, S.: Network in network (2013). [arXiv:1312.4400](https://arxiv.org/abs/1312.4400)
39. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE International

Conference on Computer Vision, pp. 2794–2802 (2017)

40. Mar, T., Zaunseder, S., Martínez, J.P., Llamedo, M., Poll, R.: Optimization of ecg classification by means of feature selection. *IEEE Trans. Biomed. Eng.* **58**(8), 2168–2177 (2011)
41. Mark, R., Moody, G., Greenwald, S.: Mit-bih supraventricular arrhythmia database (1990)
42. Mark, R., Schluter, P., Moody, G., Devlin, P., Chernoff, D.: An annotated ecg database for evaluating arrhythmia detectors. *IEEE Trans. Biomed. Eng.* **29**(8), 600–600 (1982)
43. Marteau, P.F.: Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(2), 306–318 (2008)
44. Martis, R.J., Acharya, U.R., Lim, C.M., Suri, J.S.: Characterization of ecg beats from cardiac arrhythmia using discrete cosine transform in pca framework. *Knowl.-Based Syst.* **45**, 76–82 (2013)
45. Martis, R.J., Acharya, U.R., Min, L.C.: Ecg beat classification using pca, lda, ica and discrete wavelet transform. *Biomed. Signal Process. Control* **8**(5), 437–448 (2013)
46. McSharry, P.E., Clifford, G.D., Tarassenko, L., Smith, L.A.: A dynamical model for generating synthetic electrocardiogram signals. *IEEE Trans. Biomed. Eng.* **50**(3), 289–294 (2003)
47. for the Advancement of Medical Instrumentation, A., et al.: Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms. *ANSI/AAMI EC38* **1998** (1998)
48. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). [arXiv: 1411.1784](https://arxiv.org/abs/1411.1784)
49. Moavenian, M., Khorrami, H.: A qualitative comparison of artificial neural networks and support vector machines in ecg arrhythmias classification. *Expert Syst. Appl.* **37**(4), 3088–3093 (2010)
50. Mondéjar-Guerra, V., Novo, J., Rouco, J., Penedo, M.G., Ortega, M.: Heartbeat classification fusing temporal and morphological information of ecgs via ensemble of classifiers. *Biomed. Signal Process. Control* **47**, 41–48 (2019)
- 51.

- Nankani, D., Baruah, R.D.: An end-to-end framework for automatic detection of atrial fibrillation using deep residual learning. In: TENCON 2019-2019 IEEE Region 10 Conference (TENCON), pp. 690–695. IEEE (2019)
52. Nankani, D., Baruah, R.D.: Effective removal of baseline wander from ecg signals: A comparative study. In: International Conference on Machine Learning, Image Processing, Network Security and Data Sciences, pp. 310–324. Springer (2020)
 53. Nankani, D., Baruah, R.D.: Investigating deep convolution conditional gans for electrocardiogram generation. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
 54. Nankani, D., Saikia, P., Baruah, R.D.: Automatic concurrent arrhythmia classification using deep residual neural networks. In: 2020 Computing in Cardiology, pp. 1–4. IEEE (2020)
 55. Nguyen, H.M., Cooper, E.W., Kamei, K.: Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Parad.* **3**(1), 4–21 (2011)
 56. Oh, S.L., Ng, E.Y., San Tan, R., Acharya, U.R.: Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats. *Comput. Biol. Med.* **102**, 278–287 (2018)
 57. Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio (2016). [arXiv:1609.03499](https://arxiv.org/abs/1609.03499)
 58. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015). [arXiv:1511.06434](https://arxiv.org/abs/1511.06434)
 59. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **26**(1), 43–49 (1978)
[zbMATH]
 60. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.* **11**(5), 561–580 (2007)
 61. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Netw.* **127**, 58–66 (2020)

[zbMATH]

62. Wang, H., Shi, H., Lin, K., Qin, C., Zhao, L., Huang, Y., Liu, C.: A high-precision arrhythmia classification method based on dual fully connected neural network. *Biomed. Signal Process. Control* **58**, 101874 (2020)
63. Wang, P., Hou, B., Shao, S., Yan, R.: Ecg arrhythmias detection using auxiliary classifier generative adversarial network and residual network. *IEEE Access* **7**, 100910–100922 (2019)
64. Wulan, N., Wang, W., Sun, P., Wang, K., Xia, Y., Zhang, H.: Generating electrocardiogram signals by deep learning. *Neurocomputing* **404**, 122–136 (2020)
65. Ye, C., Kumar, B.V.K.V., Coimbra, M.T.: An automatic subject-adaptable heartbeat classifier based on multiview learning. *IEEE J. Biomed. Health Inform.* **20**(6), 1485–1492 (2016)
66. Ye, C., Vijaya Kumar, B.V.K., Coimbra, M.T.: Heartbeat classification using morphological and dynamic features of ecg signals. *IEEE Trans. Biomed. Eng.* **59**(10), 2930–2941 (2012)
67. Ye, F., Zhu, F., Fu, Y., Shen, B.: Ecg generation with sequence generative adversarial nets optimized by policy gradient. *IEEE Access* **7**, 159369–159378 (2019)
68. Yeh, Y.C., Chiou, C.W., Lin, H.J.: Analyzing ecg for cardiac arrhythmia using cluster analysis. *Expert Syst. Appl.* **39**(1), 1000–1010 (2012)
69. Yildirim, Ö.: A novel wavelet sequence based on deep bidirectional lstm network model for ecg signal classification. *Comput. Biol. Med.* **96**, 189–202 (2018)
70. Zeng, N., Wang, Z., Zhang, H., Liu, W., Alsaadi, F.E.: Deep belief networks for quantitative analysis of a gold immunochromatographic strip. *Cognit. Comput.* **8**(4), 684–692 (2016)
71. Zhao, Z.D., Chen, Y.Q.: A new method for removal of baseline wander and power line interference in ecg signals. In: 2006 International Conference on Machine Learning and Cybernetics, pp. 4342–4347 (2006)
72. Zhou, B., Liu, S., Hooi, B., Cheng, X., Ye, J.: Beatgan: anomalous rhythm detection using adversarially generated time series. In: Proceedings of the 28th

International Joint Conference on Artificial Intelligence, pp. 4433–4439. AAAI Press (2019)

73. Zhu, B., Ding, Y., Hao, K.: A novel automatic detection system for ecg arrhythmias using maximum margin clustering with immune evolutionary algorithm. *Comput. Math. Methods Med.* **2013** (2013)
74. Zhu, F., Ye, F., Fu, Y., Liu, Q., Shen, B.: Electrocardiogram generation with a bidirectional lstm-cnn generative adversarial network. *Sci. Rep.* **9**(1), 6734 (2019)

13. Generative Adversarial Network Powered Fast Magnetic Resonance Imaging—Comparative Study and New Perspectives

Guang Yang^{1, 2}✉, Jun Lv³, Yutong Chen^{1, 4}, Jiahao Huang⁵ and Jin Zhu⁶

- (1) National Heart & Lung Institute, Imperial College London, London, SW7 2AZ, UK
- (2) Cardiovascular Research Centre, Royal Brompton Hospital, London, SW3 6NP, UK
- (3) School of Computer and Control Engineering, Yantai University, Yantai, China
- (4) Department of Physiology, Development and Neuroscience, University of Cambridge, Cambridge, CB2 3EG, UK
- (5) School of Optics and Photonics, Beijing Institute of Technology, Beijing, China
- (6) Department of Computer Science and Technology, University of Cambridge, Cambridge, CB3 0FD, UK

✉ **Guang Yang**
Email: g.yang@imperial.ac.uk

Abstract

Magnetic Resonance Imaging (MRI) is a vital component of medical imaging. When compared to other image modalities, it has advantages such as the absence of radiation, superior soft tissue contrast, and complementary multiple sequence information. However, one drawback of MRI is its comparatively slow scanning and reconstruction compared to other image modalities, limiting its usage in some clinical applications when imaging

time is critical. Traditional compressive sensing based MRI (CS-MRI) reconstruction can speed up MRI acquisition, but suffers from a long iterative process and noise-induced artefacts. Recently, Deep Neural Networks (DNNs) have been used in sparse MRI reconstruction models to recreate relatively high-quality images from heavily undersampled k -space data, allowing for much faster MRI scanning. However, there are still some hurdles to tackle. For example, directly training DNNs based on L1/L2 distance to the target fully sampled images could result in blurry reconstruction because L1/L2 loss can only enforce overall image or patch similarity and does not take into account local information such as anatomical sharpness. It is also hard to preserve fine image details while maintaining a natural appearance. More recently, Generative Adversarial Networks (GAN) based methods are proposed to solve fast MRI with enhanced image perceptual quality. The encoder obtains a latent space for the undersampling image, and the image is reconstructed by the decoder using the GAN loss. In this chapter, we review the GAN powered fast MRI methods with a comparative study on various anatomical datasets to demonstrate the generalisability and robustness of this kind of fast MRI while providing future perspectives.

Keywords Generative adversarial networks (gan) – Fast magnetic resonance imaging (mri) – Compressive sensing – Deep learning

13.1 Introduction

13.1.1 Magnetic Resonance Imaging

Through offering repeatable, non-invasive measures of tissue structure and function, Magnetic resonance imaging (MRI) has transformed clinical medical imaging and medical science. The sensitivity of the image to tissue properties can be greatly varied with MRI, either by changing the timing with which MR signals are obtained (e.g., the echo time—TE and the repetition time—TR), or by using magnetisation preparation or contrast agents. These so-called multimodal or multi-sequence MRI methods can not only provide a comparison in traditional anatomical or structural MRI but can also quantify the function of most tissues and organs of the human body in clinical and pre-clinical laboratory environments. Invasive

methods, such as tissue biopsy or radionuclide tests, have therefore become less needed as a result of the prosperity of MRI and other non-invasive medical imaging technologies [1]. Since MRI is non-invasive, it can be used to provide longitudinal and quantitative imaging biomarkers in the therapy trials. MR methods have begun to represent gold standard measurement for clinical research, despite the fact that the modality is still relatively underutilised [1]. Why is this so? ‘MRI is complex and pricey’, which is a commonly heard melancholy.

13.1.2 Limitations of Magnetic Resonance Imaging

Although MRI is a revolutionary non-invasive diagnostic imaging technique that provides high-resolution definition of the structural and functional information of most body tissues and organs, one significant drawback of MRI is its slow rate of image acquisition, which results in a longer scanning period as compared to other imaging modalities [2, 3].

In MRI, raw data is acquired in the k -space, which includes information about spatial frequencies within the image, rather than collected directly in the image space [4]. The Fourier transformation links image space and the k -space. The Nyquist criterion defines the k -space information that must be satisfied conventionally after we have defined the field-of-view (FOV) and spatial resolution of the image that we want to obtain. The distance between k -space neighbours is inversely proportional to the field of view in each direction. The highest frequency obtained in each direction is inversely proportional to the desired resolution. Data encoded with pulsed magnetic field gradients are acquired to fill the k -space. We may obtain a line of k -space points very quickly in one direction, known as the read direction, using either a spin or gradient echo in one repetition time. Further directions, on the other hand, must be phase-encoded, which takes one repetition time to encrypt one line of k -space lines [1, 4]. This must then be rerun for all possible combinations of the number of phase encoding steps needed in the anterior-posterior and foot-head directions. As a result, MRI acquisitions can be time-consuming, particularly when a high resolution or large FOV is needed.

This drawback not only raises the cost of imaging but also limits its use in emergency situations. Furthermore, in order to maintain image consistency, patients must lie still during the acquisition. For

abdominal/thoracic imaging, patients must hold their breath, which can be problematic for paediatric, obese patients, and those with respiratory compromise [1]. As a result, many patients can experience anxiety and claustrophobia during the MR scanning procedure [1]. In order to minimise scanning costs and increase patient throughput, the MR image acquisition process must be sped up. Various MR acceleration methods rely on taking measurements of several lines per repetition time, allowing for quicker traversing of the k -space. Examples include echo planar imaging [5], rapid acquisition with relaxation enhancement [6], and fast low angle shot imaging [7].

13.1.3 Conventional Acceleration Using Compressive Sensing

It is possible to attain a higher degree of acceleration by sampling the k -space only partially, i.e., not collecting all lines of measurements in the phase encoding path(s). The undersampled calculation can be used to infer the original k -space details. As a consequence, the acceleration metric is equal to the undersampling ratio. For example, if half of the k -space is sampled, the acceleration factor is doubled. As a result, undersampling methods aim to circumvent the Nyquist-Shannon sampling criterion [2].

Compressed sensing (CS) is a promising undersampling approach that may allow for more aggressive undersampling and acceleration [2]. CS principle is similar to the concept of compressing signals for transmission and then decompressing them [8], as seen in the JPEG, JPEG2000, MPEG, and MP3 standards [1]. If undersampled signals or images can be compressed correctly, they can also be decompressed or recovered accurately, according to CS [9]. Hereby, CS sets three conditions on the MRI reconstruction:

1. The image or signal must be compressible. In other words, the MRI images must be sparse, with the bulk of its pixel values being zeros, either in its native domain or in an appropriate transformation domain, such as in the wavelet or frequency domain.
2. To avoid aliasing artefacts, the undersampling patterns should be incoherent using random undersampling.
3. A non-linear reconstruction algorithm must be used.

It is possible to recover the original MRI images from their undersampled measurements following these three criteria.

Previously published research on using CS as an MR acceleration approach employs iterative non-linear optimisation algorithms that implement sparsity and reconstruction fidelity. Total variation (TV) [2], dictionary learning (DLMRI [10], RecPF [11], and BM3D [12]) are typical examples. However, there are four major issues with these approaches:

1. Iterative optimisation can be time-consuming [1, 13].
2. These algorithms tend to generate artificially smoothed image appearance [1].
3. The reconstruction results can have blocky artefacts [14–16].
4. They reconstruct each image as an individual event, failing to account for the expected anatomical features in MR images that may be used to improve the reconstruction accuracy [17].

13.1.4 Deep Learning Based Fast MRI

Deep learning based approaches have recently achieved performance dividends in a variety of medical image processing problems by using ‘big data’ and advancements in computational power. To date, however, the majority of research studies have concentrated on downstream medical image interpretation and post-processing activities, such as anatomical segmentation [18–28], lesion segmentation [29–34], co-registration [35–37], synthesis [38, 39], and multimodal data detection [40–46], for disease identification [47–49], prognosis [50, 51], and treatment prediction [52, 53]. To increase the precision of these post-processing operations, imaging methods must be improved, which can also be aided by deep learning [54–60].

Since its principle was developed in 2006, CS has had a long history for fast imaging applications, including the embodiment of MRI reconstruction [61]. However, the related less efficient iterative optimisation can stymie further implementation. Although deep learning based tomographic reconstruction technology has only been around for a few years, there is a

lot of interest in this area, and there are many ongoing advances and exciting applications, including MRI.

Deep learning based approaches can successfully overcome the majority of the aforementioned shortcomings of earlier CS methods. A deep learning algorithm, e.g., convolutional neural networks (CNN), is made up of many layers of nodes. To learn the mapping from undersampled MR images to their corresponding fully sampled ones, the weights of the node relations between layers are optimised. The method of optimising weights is known as training the model. Once trained, the model is capable of reconstructing original images from undersampled measurements. In terms of reconstruction accuracy, speed, and visual consistency, deep learning based methods have been shown to consistently outperform non-deep learning based ones [3, 62–66].

13.1.5 GAN Powered Fast MRI

Generative Adversarial Networks, or GAN for short, represent a type of generative modelling technique that employs neural network and deep learning methods, e.g., CNN. Generative modelling is an unsupervised learning task in machine learning that entails automatically finding and learning the regularities or patterns in input data such that the model can be used to produce or output new examples that could have been plausibly drawn from the original dataset.

GAN employs a clever method for training a generative model by posing the problem as a supervised learning problem with two sub-models: the generator model, which we train to produce new examples, and the discriminator model, which attempts to identify examples as either true (from the original domain) or false (generated). The two models are trained in an adversarial zero-sum game before the discriminator model is tricked about half of the time, indicating that the generator model is producing plausible instances.

GAN is a fascinating and quickly evolving area that delivers on the promise of generative models by producing plausible instances in a variety of problems, most notably in image-to-image conversion tasks such as translating image styles, and in generating photo-realistic images of objects, scenes, and individuals that even humans cannot recognise which ones are fake.

GAN is an important type of deep learning based CS-MRI reconstruction method, which was proposed first by Yang et al. in [3, 67]. In the context of CS-MRI, GAN entails training a generator to recreate the original image from undersampled measurements and a discriminator to produce the likelihood of whether the generated image matches the original, i.e., fully sampled measurements. The discriminator, in turn, modifies the generator's learning [68]. As a consequence, the generator generates photo-realistic images [69]. In terms of reconstruction accuracy and efficiency, GAN based methods [3, 62] outperform the non-GAN based deep learning method, e.g., deep ADMM-net. One GAN based approach [63] also claims to generate less fuzzy and aliasing artefacts than non-deep learning based methods. As a result, GAN based approaches have the capability to produce state-of-the-art CS-MRI reconstruction results.

In this book chapter,

- we will perform a mini topical review on GAN powered fast MRI, including the original Deep De-Aliasing Generative Adversarial Networks (DAGAN) method and other more advanced and recently proposed GAN based models;
 - we will analyse and explain different GAN models, and compare the results obtained by different GAN based models;
 - we will provide a comparison study on different datasets, e.g., MRI for various anatomical scans.
 - we will highlight the recent development and discuss future directions.
-

13.2 Methods

13.2.1 Fundamentals of MRI Reconstruction

The basis of undersampled MR reconstruction is inferring the missing k -space data from the already sampled k -space values. For example, in partial Fourier imaging (PFI), 50% of the k -space is acquired. Because the k -space is conjugate symmetric, the other missing 50% is obtained via complex conjugation of the existing 50%. However, the maximum acceleration is only 2-fold and the reconstructed image exhibits a lower signal-to-noise ratio (SNR) [1]. Another key undersampling technique is parallel imaging (PI). In PI, multiple receiver coils simultaneously collect the k -space

information of the tissues closest to each coil. However, the acceleration factor is limited by the number and the configuration of the receiver coils [1, 70]. Therefore, limits of accelerating MR image acquisition exist for both PI and PFI, bound by the Nyquist-Shannon sampling criteria [2].

Table 13.1 Mathematical notations in this book chapter

Symbol	Definition
π_i	Ground truth MR image
π_d	Undersampled MR image
P_r	Reconstructed MR image
π_i	Ground truth k -space signals
π_d	Undersampled k -space signals
\mathcal{X}	Fourier transform operator
Y	Binary matrix denoting undersampled k -space positions
A	$\Psi\mathcal{F}$
R	Regularisation function
λ	A parameter adjusting the contribution of data fidelity term to the reconstruction loss function
\hat{x}	Parameters within a convolutional neural network

Compressed sensing (CS)-based reconstruction circumvents the Nyquist-Shannon sampling criteria and achieves a higher acceleration ratio. CS establishes the model of MR image acquisition as (notations in Table 13.1):

$$y_u = Ax_t, \quad (13.1)$$

where π_d is the undersampled k -space signal and π_i is the original fully sampled image. A is an operator defined as:

$$A = \Psi\mathcal{F}, \quad (13.2)$$

where Y is the undersampling mask, a binary matrix denoting which k -space locations are sampled and which are not, and \mathcal{X} is the Fourier transform operator. Hence, $L \geq 1$, or U_M is equivalent to undersampling the k -space of the putative reconstructed MR image.

A CS-model infers the underlying full resolution MR image π_i from the collected incomplete k -space samples π_d by solving the following optimisation problem [2]:

$$\arg \min_{\hat{x}_u} \frac{\lambda}{2} \|y_u - A\hat{x}_u\|_2^2 + R(\hat{x}_u), \quad (13.3)$$

where λ adjusts the contribution of the first term to the optimisation objective and $\sigma(\mathbf{W})$ is an image regulariser function. If this reconstructed image P_r is to match the original image, their undersampled k -space results ought to match, as reflected in the first term of Eq. 13.3. This reinforces the k -space data fidelity of reconstructing the undersampled data.

The second term in Eq. 13.3 ensures the reconstructed image possesses certain attributes such as smoothness or sparsity, which is required by the theory of CS. For example, total variation—an early CS technique—uses the following regulariser term to ensure the underlying image is smooth [2, 71]:

$$\arg \min_{\hat{x}_u} \frac{\lambda}{2} \|y_u - A\hat{x}_u\|_2^2 + \|\nabla \hat{x}_u\|_1, \quad (13.4)$$

where μ_q is the gradient operator to minimise the difference between adjacent pixels in the final reconstruction to ensure a smooth texture. To solve Eq. 13.4, conjugate gradient descent is employed [2], by updating the reconstructed image with the gradient of Eq. 13.4 iteratively until convergence.

Another example of CS algorithm is dictionary learning, in which the regulariser function enforces the sparsity of the dictionary representation of each image patch [10]:

$$\arg \min_{\hat{x}_u, D, \Gamma} \frac{\lambda}{2} \|y_u - A\hat{x}_u\|_2^2 + \sum_{ij} \|R_{ij}\hat{x}_u - Da_{ij}\|_2^2, \quad (13.5)$$

where D_ϕ is the image patch extractor, D is the dictionary used to transform the dictionary representation a_{ij} into the image domain, and \hat{s} is the set of all dictionary representations $\{a_{ij}\}_{ij}$. This optimisation problem in Eq. 13.5 is solved by alternating minimisation of the 3 parameter sets iteratively: P_r , D and \hat{s} collectively. CS algorithms such as TV and

dictionary learning repeatedly update the reconstruction to ensure it fulfils k -space data fidelity and certain regularisation properties such as sparsity.

From a Bayesian perspective, data fidelity is seen as maximising the conditional probability of observing the undersampled k -space data given the reconstructed image. The regulariser term represents the prior expectation of the statistical properties of the final reconstructed image [72]. By jointly enforcing data fidelity and regulariser constraint, CS reconstruction is equivalent to maximising the posterior probability that the reconstructed image matches the original one given the existing k -space samples. This justifies the effectiveness of the CS in reconstructing undersampled MR images.

13.2.2 CNN Based MRI Reconstruction

CNN excels in computer vision applications compared to traditional machine learning methods in general, and has been increasingly incorporated into CS-based MRI reconstruction models. It achieves higher accuracy and acceleration of MRI acquisition compared to traditional CS models [3, 64, 66, 73]. A CNN consists of a series of convolutional layers connected by non-linear activation functions [74]. In each convolutional layer, multiple filters are convolved with the input image or the output from a previous layer to extract image specific features [74]. The weight of each filter is optimised such that the final reconstructed image matches the original fully sampled image. By stacking multiple convolutional layers, the large number of weight parameters in a CNN endows it with the potential to model complex functions, including recovering the fully sampled images from the undersampled ones.

The process of optimising the weights of CNN is known as training the network and is governed by an optimisation objective called the loss function. The loss functions differ between supervised and unsupervised settings, depending on whether fully sampled ground truth images are available. In a supervised learning setting, fully sampled images are used to “teach” the CNN model to minimise the loss, or difference, between the reconstructed results and the fully sampled ground truth. However, in an unsupervised environment, ground truth images are not available, meaning the loss function can be, for example, optimised based on the Deep Image Prior framework and uses a high-resolution reference MR image as the

input of the CNN to induce the structural prior in the learning procedure [75]. Unsupervised CNN-based CS-MRI models are beyond the scope of this chapter, which will instead focus on the supervised learning models.

Supervised CNN-based CS-MRI methods can be broadly divided into two categories: end-to-end and unrolled optimisation [76–78]. End-to-end methods model the inverse of the acquisition process (Eq. 13.1) by mapping the undersampled input to the reconstructed output directly, hence the name “end-to-end”:

$$\hat{x}_u = f_{\text{CNN}}(y_u|\theta), \quad (13.6)$$

where f_{CNN} is the operation performed by the CNN and \hat{x} represents all the parameters within the CNN. The \hat{x} is optimised by minimising the difference between P_r and the ground truth π_i . Such end-to-end techniques are exemplified by U-Net [79] and generative adversarial network (GAN) [3, 62, 63].

In contrast, CNN-based unrolled optimisation methods perform iterative image update based upon the general CS reconstruction model (Eq. 13.3). Unlike non-CNN based CS techniques, where the term $\sigma(\mathbf{W})$ is an expert-designed regularisation function, the CNN-based unrolled methods apply CNN to learn the optimal way of regularising an image. This is exemplified by the deep cascade CNN (DC-CNN) network [64], whose regulariser term penalises the deviation of the reconstructed image from the CNN output:

$$\arg \min_{\hat{x}_u, \theta} \frac{\lambda}{2} \|y_u - A\hat{x}_u\|_2^2 + \|\hat{x}_u - f_{\text{CNN}}(y_u|\theta)\|. \quad (13.7)$$

Another example is the variational network [17], which uses a Field-of-Expert regularisation function:

$$\arg \min_{\hat{x}_u, \theta} \frac{\lambda}{2} \|y_u - A\hat{x}_u\|_2^2 + \sum_i f_i(k_i \hat{x}_u), \quad (13.8)$$

where ψ is a learnable activation function and d_1 represents a convolutional filter. From a Bayesian point of view, these CNN-based regularisers form the prior expectation of the reconstructed images by learning from the ground truth MR images.

While both end-to-end and unrolled methods can incorporate CNN to learn the image reconstruction or regularisation process, a key difference between them is that unrolled methods need to iteratively update the images during the reconstruction process. In contrast, end-to-end ones compute the output directly. Hence, it is more time consuming to train unrolled CNN-based models and to apply them in reconstruction, compared with end-to-end methods, with other comparisons summarised in Table 13.2. This suggests an advantage of developing end-to-end methods, such as GAN-based models, for CS-MRI reconstruction.

Table 13.2 Comparison between end-to-end and unrolled optimisation methods in CNN-based CS-MRI models [76–78]

Category	End-to-end	Unrolled
Reconstruction time	Short	Long
Data fidelity	No	Yes
Sample size	Larger	Smaller
Performance	Lower	Higher
Weight update	Yes	Yes
Image update	No	Yes
Parameter number	Larger	Smaller

13.2.3 GAN Based MRI Reconstruction

13.2.3.1 General GAN

Inspired by the two-player zero-sum game in game theory, GAN [68] consists of two players: a generator and a discriminator. Traditionally, the generator G captures the distribution of sample data, and uses noise z that follows a certain distribution to generate a sample $G(z)$ similar to the real training data π_i . The discriminator D is a binary classifier, which aims to distinguish fake data generated by G from the ground truth. If the sample comes from the real training data, D outputs a large probability, otherwise, D outputs a small probability. Ideally, the best D can be represented as $D(x_t) = 1$ and $\log(D(G(z)))$. In this way, the generator and the discriminator form a min-max game. The training process of GAN can be described as follows:

$$\begin{aligned} \min_{\theta_G} \max_{\theta_D} L(\theta_G, \theta_D) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D_{\theta_D}(\mathbf{x})] \\ &\quad + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D_{\theta_D}(G_{\theta_G}(\mathbf{z})))] \end{aligned} \quad (13.9)$$

where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is the distribution of the training dataset, and the $(\hat{\mathbf{x}}, \hat{y})$ is the distribution of the latent variables.

Alternating gradient optimisation between G and D is used to train the GAN. In this process, both models try their best to optimise their networks to form a competitive confrontation until the two models reach a dynamic balance. Ideally, the final result is that the image generated by G is very similar to the real image, and it is difficult for the D network to distinguish between the real image and the image generated by G , i.e., $D(G(Z)) = 0.5$

13.2.3.2 DAGAN

Deep De-Aliasing Generative Adversarial Networks (DAGAN) was proposed in 2017 by Yang et al. [3, 67] for fast compressed sensing MRI reconstruction. Figure 13.1 shows the architecture of DAGAN.

As shown in Fig. 13.2, a modified U-Net [80] was used as the generator G , which consisted of 8 convolutional layers in the encoding path and 8 deconvolutional layers in the decoding path. The stride of all convolutional and deconvolutional layers was set to 2 for downsampling and upsampling the feature maps. Each convolutional layer was followed by a Batch Normalisation (BN) layer and a Leaky ReLU (LReLU) layer. Skip connection was applied between corresponding layers in the encoding and decoding paths in order to pass the features in the encoding layer to the decoding layer for better reconstruction details. The modified U-Net ended up with a hyperbolic tangent function as the activation function. Due to the alternating training strategy in the adversarial components, the original GAN model is hard to train. DAGAN applied $\log[1 - D(G(z))]$ instead of $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ as the output of the generator for better stability and faster convergence speed. In this way, DAGAN turned the generator into a refinement function, which means that it only generated the missing information, and the complexity of the model was reduced.

As shown in Fig. 13.3, an 11-layer CNN architecture was used as the discriminator D . A BN layer and a LReLU layer were followed with each

convolutional layer. Finally, a full connection (FC) layer was cascaded, and the classification result was output through the Sigmoid activation function.

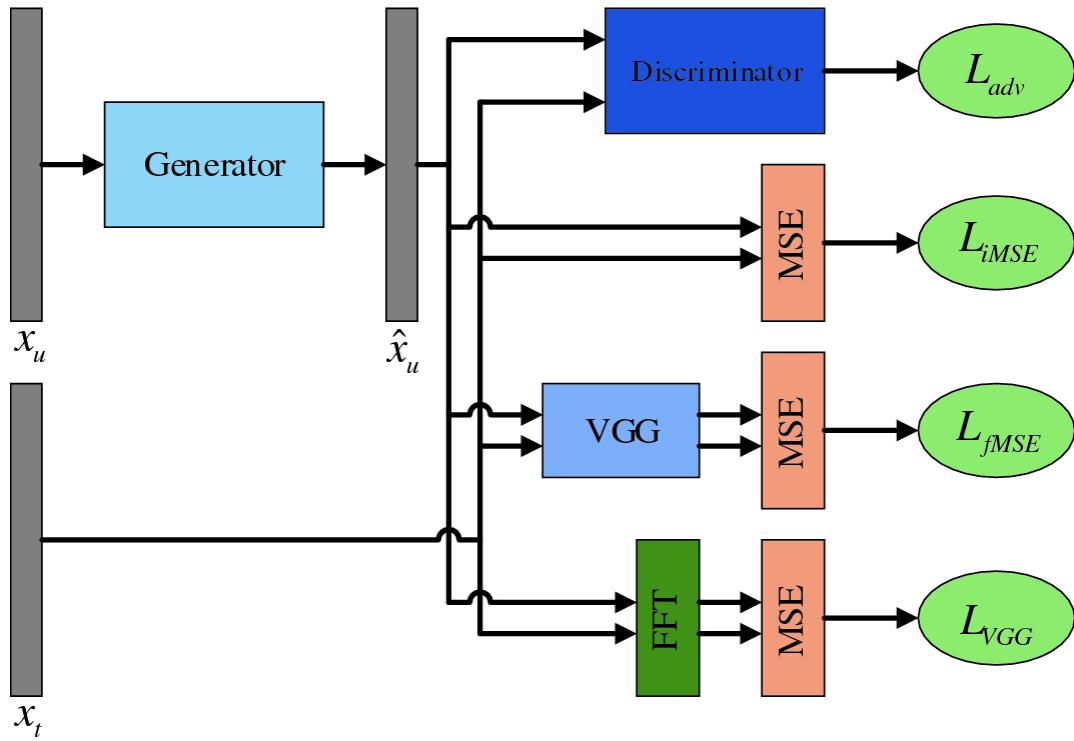


Fig. 13.1 The architecture of DAGAN. The generator produced the MR image P_r from the undersampled MR image. The generated MR image P_r and the ground truth MR image were input into the discriminator for adversarial loss F_{DV} calculation. The data consistency loss consisted of image domain MSE loss $W_r x_r$, frequency domain MSE loss $W_r x_r$, and VGG perceptual loss L_{VGG}

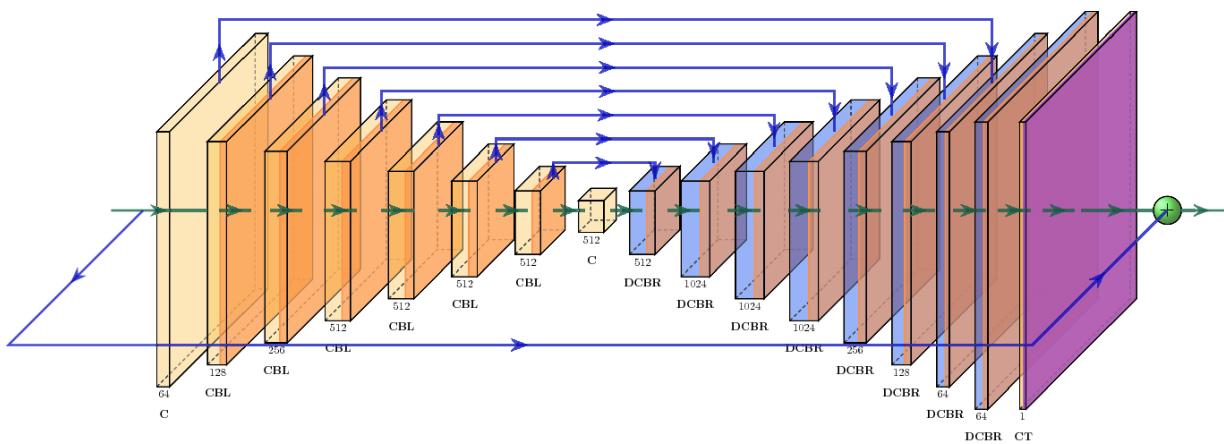


Fig. 13.2 Structure of the generator in DAGAN. The generator was based on a modified U-Net, which consisted of 8 convolutional layers and 8 transposed

convolutional layers (de-convolutional layers), with a BN layer and a LReLU or ReLU layer followed with each layer. Skip connection between layers that were the same scale and shortcut connections between the input and output of the generator were applied. A hyperbolic tangent function was used as an output activation function (C: Convolutional layer; DC: De-Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer; R: ReLU layer; T: hyperbolic tangent function.)

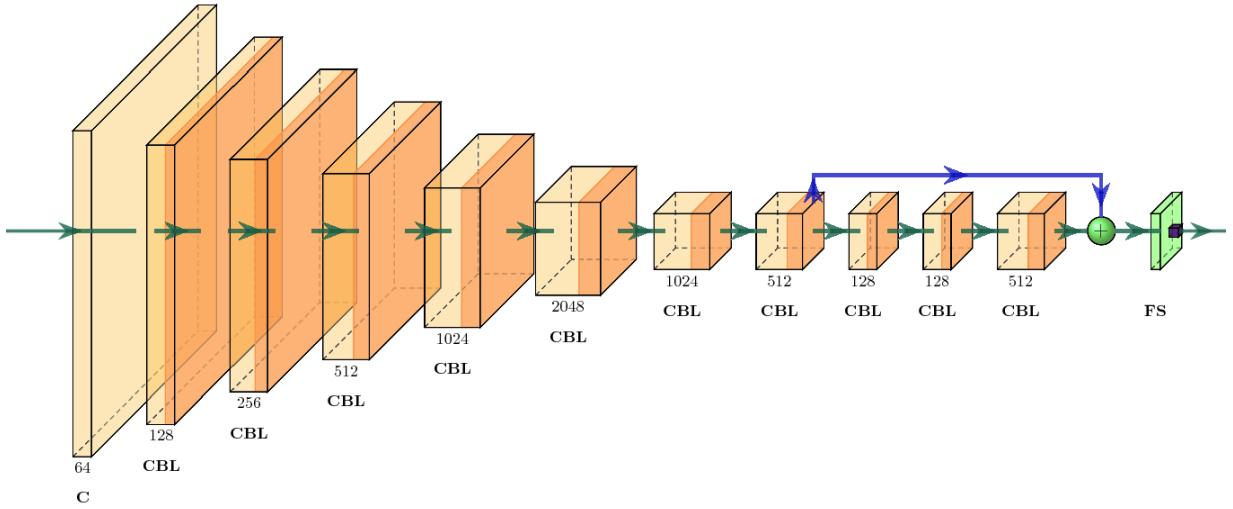


Fig. 13.3 Structure of the discriminator in DAGAN. Essentially, the discriminator was an 11-layer CNN classifier. Each convolutional layer was cascaded by a BN layer and a LReLU Layer. A full connection layer and a Sigmoid function were applied to output the result of the classification (C: Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer; F: Full Connection layer; S: Sigmoid function.)

In DAGAN, normalised MSE was used as the optimisation cost function. However, only optimising the pixel-wise MSE loss in the image domain could result in non-smooth reconstructions, which might lack coherent image details. To solve this problem, a data consistency loss was designed for training the generator in both frequency and image domains to help the optimisation and to exploit the complementary properties of the two domains. Besides, perceptual similarity [81] was also incorporated. In so doing, the data consistency loss consists of three parts: pixel-wise image domain MSE loss W_{rX_r} , frequency domain MSE loss W_{rX_f} , and the pre-trained VGG perceptual loss L_{VGG} [82]. They can be defined as:

$$L_{iMSE}(\theta_G) = \frac{1}{2} \| x_t - \hat{x}_u \|_2^2, \quad (13.10)$$

$$L_{\text{fMSE}}(\theta_G) = \frac{1}{2} \parallel \mathcal{F}x_t - \mathcal{F}\hat{x}_u \parallel_2^2, \quad (13.11)$$

$$L_{\text{VGG}}(\theta_G) = \frac{1}{2} \parallel f_{\text{VGG}}(x_t) - f_{\text{VGG}}(\hat{x}_u) \parallel_2^2. \quad (13.12)$$

Here, $\parallel \cdot \parallel_2^2$ denotes the L2 norm, $\mathcal{D} \approx \hat{\mathcal{D}}$ denotes VGG network (the Conv4 output of the VGG16), and was pretrained on the ImageNet [83].

The adversarial loss L can be defined as:

$$\begin{aligned} L_{\text{adv}}(\theta_G, \theta_D) &= \mathbb{E}_{x_t \sim P_{\text{train}}(x_t)} [\log D_{\theta_D}(x_t)] \\ &\quad + \mathbb{E}_{x_u \sim p_G(x_u)} [\log(1 - D_{\theta_D}(G_{\theta_G}(x_u)))] \end{aligned} \quad (13.13)$$

where $G(z * |c)$ denotes the collection of MR image ground truth, and $\mathcal{G}(z, c)$ denotes the collection of MR image reconstruction.

Therefore the total loss function can be represented as:

$$\begin{aligned} L_{\text{TOTAL}}(\theta_G, \theta_D) &= \alpha L_{\text{iMSE}}(\theta_G) + \beta L_{\text{fMSE}}(\theta_G) + \gamma L_{\text{VGG}}(\theta_G) \\ &\quad + L_{\text{adv}}(\theta_G, \theta_D), \end{aligned} \quad (13.14)$$

where ω , β and γ were the weights of different components of the loss function, which balanced different loss terms into similar scales according to previous study. Here, $\alpha = 15$, $z \in \mathbb{R}^D$ and $\gamma = 0.0025$ were set empirically, according to the original paper.

13.2.3.3 KIGAN

KIGAN was introduced by Shaul et al. [84]. The overall structure of KIGAN is shown in Fig. 13.4. A k -space generator P_G and an image space generator U_M were cascaded in KIGAN. Adjacent undersampled k -space slices (along the third dimension or the temporal dimension) y_u^{l-1} , y_u^l and y_u^{l+1} were the input of P_G . A data consistency step, i.e., $\tilde{y}_u = \bar{\Psi}G_K(y_u^{l-1}, y_u^l, y_u^{l+1}) + y_u^l$ was applied for merging the output with the reconstructed missing data. U_M was able to generate the output of P_G in image space, i.e., \mathbb{Q} , to reconstructed image P_r . Refinement learning $\hat{\mu}_1 = [\hat{\mu}_{1min}, \hat{\mu}_{1max}]$ was also used as the output of U_M instead of $\hat{x}_u = G_{\text{IM}}(\tilde{y}_u)$. The reconstruction MR image P_r , together with MR

ground truth image π_i were sent to the discriminator for the adversarial loss.

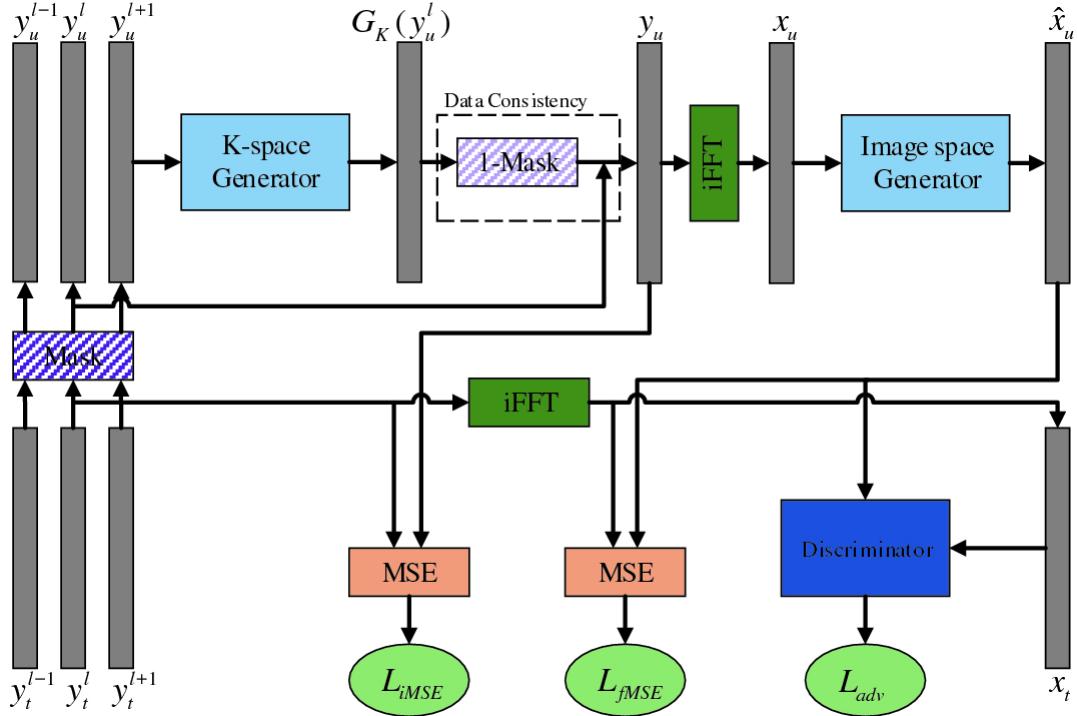


Fig. 13.4 The Architecture of KIGAN. Two generators and one discriminator were used in KIGAN. k -space generator was able to reconstruct k -space MR data $G_K(y_u^l)$ from a cascade of undersampled k -space MR data y_u^{l-1} , y_u^l and y_u^{l+1} , and image space generator was able to reconstruct the MR image P_r (final result). The discriminator was to output the classification result of ground truth MR image π_i and generated MR image P_r for adversarial loss F_{DV} . Image domain MSE loss $W_r x_r$ and frequency domain MSE loss were added into the total loss function

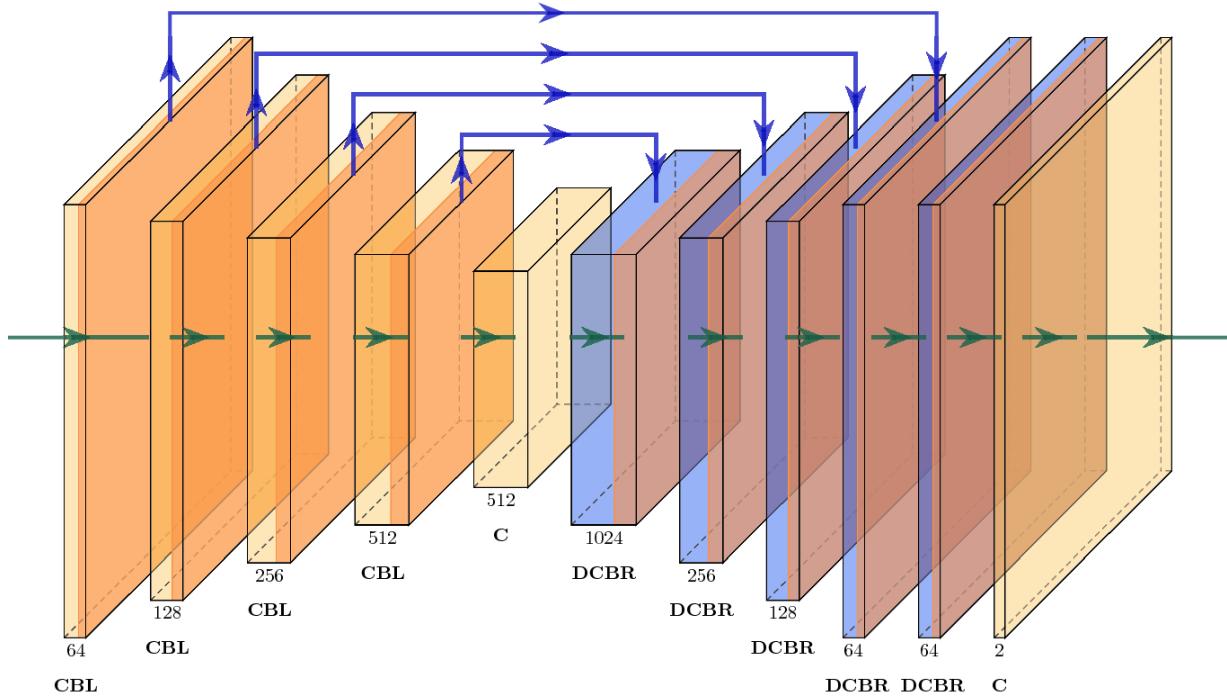


Fig. 13.5 The Structure of k -space Generator in KIGAN. Five convolutional layers and five de-convolutional layers were applied as encoding path and decoding path in the generator respectively. A convolutional layer was used as the output layer to adjust the channel of the reconstructed MR image. Corresponding layers of the same scales were linked by the skip connection (C: Convolutional layer; DC: De-Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer; R: ReLU layer.)

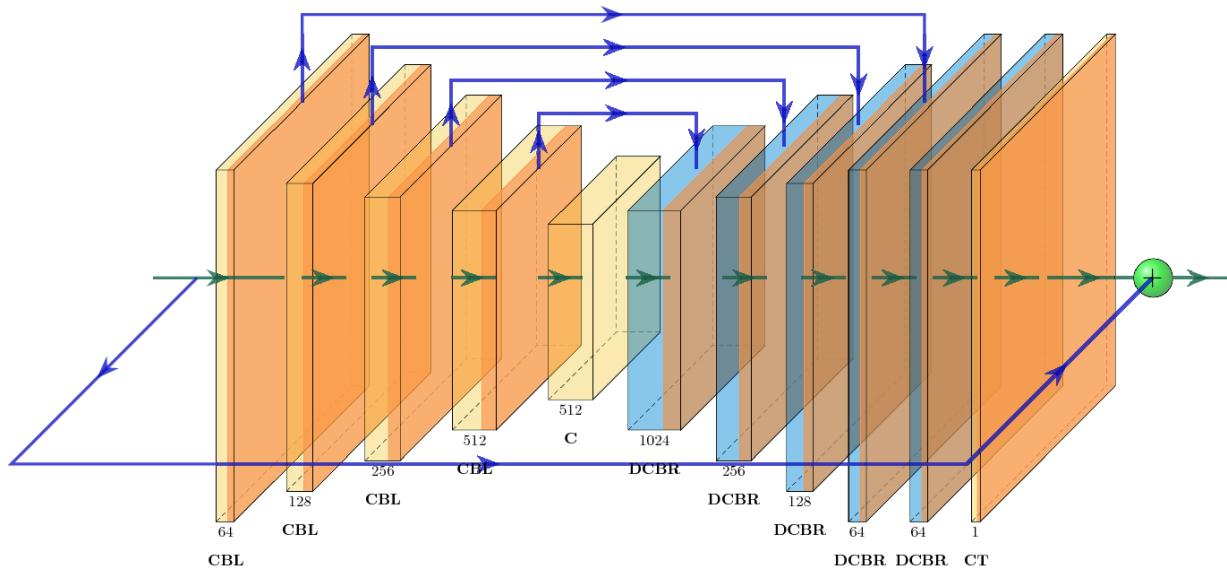


Fig. 13.6 The Structure of image space Generator in KIGAN. 5 convolutional layers and 5 de-convolutional layers were applied as encoding path and decoding path in the

generator respectively. A convolutional layer and a hyperbolic tangent function were used as the output layer. Skip connection and shortcut connection were adopted in the image space generator (C: Convolutional layer; DC: De-Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer; R: ReLU layer; T: hyperbolic tangent function.)

As shown in Figs. 13.5 and 13.6, the main structure of the generators P_G and U_M were based on a modified U-Net, which consisted of 5 convolutional layers in the encoding path and 5 deconvolutional layers in the decoding path, where each convolutional and deconvolutional layer were followed by a BN layer and a LReLU layer. Skip connection was applied between corresponding layers in the encoding and decoding paths, in order to pass the feature of different scales to the decoding layer for better reconstruction details. Additionally, in U_M , a shortcut connection between the input and output was applied for turning U_M into a refinement function.

As shown in Fig. 13.7, the discriminator was a standard 9-layer CNN structure with an FC layer and a Sigmoid activation function connected for the result of the classification.

The loss function of KIGAN consisted of: an image domain MSE loss $\mathbf{W}_r \mathbf{x}_r$, a frequency domain MSE loss $\mathbf{W}_f \mathbf{x}_f$, and an adversarial loss F_{DV} , which can be defined as:

$$L_{\text{iMSE}}(\theta_K, \theta_{\text{IM}}) = \frac{1}{2} \| x_t - \hat{x}_u \|_2^2, \quad (13.15)$$

$$L_{\text{fMSE}}(\theta_K) = \frac{1}{2} \| y_t^l - \tilde{y}_u \|_2^2, \quad (13.16)$$

$$L_{\text{adv}}(\theta_K, \theta_{\text{IM}}, \theta_D) = \mathbb{E}_{x_t \sim M}[\log D_{\theta_D}(x_t)] + \mathbb{E}_{\hat{x}_u \sim S}[\log(1 - D_{\theta_D}(\hat{x}_u))], \quad (13.17)$$

where M denotes the collection of image space ground truth, and S denotes the collection of image space reconstruction.

The whole loss function can be represented as:

$$\begin{aligned} L_{\text{TOTAL}}(\theta_K, \theta_{\text{IM}}, \theta_D) &= \alpha L_{\text{iMSE}}(\theta_K, \theta_{\text{IM}}) + \beta L_{\text{fMSE}}(\theta_K) \\ &\quad + L_{\text{adv}}(\theta_K, \theta_{\text{IM}}, \theta_D), \end{aligned} \quad (13.18)$$

where ω , β are the hyperparameters that control the balance of different components in the loss function.

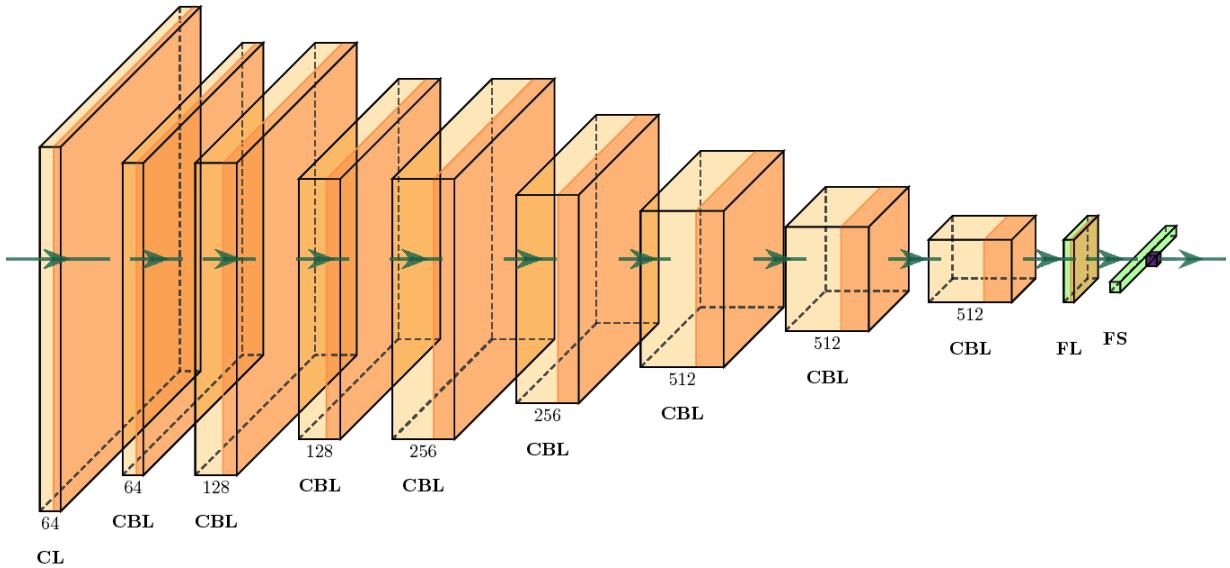


Fig. 13.7 The Structure of Discriminator in KIGAN. The discriminator was a standard 9-layer CNN structure with an FC layer and a Sigmoid activation function connected for the result of the classification (C: Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer; F: Full Connection layer; S: Sigmoid function.)

13.2.3.4 ReconGAN/RefineGAN

ReconGAN/RefineGAN was proposed by Quan et al. [62] for compressed sensing MRI reconstruction. Figure 13.8 shows the architecture of ReconGAN/RefineGAN. The two-fold chained network that consisted of 2 U-shaped generators (G_1 and G_2 respectively) was able to generate a 2-channel zero-filling MR image (real part and imaginary part) directly into a 2-channel reconstructed MR image. The checkpoints after G_1 and G_2 were defined as ReconGAN and RefineGAN respectively.

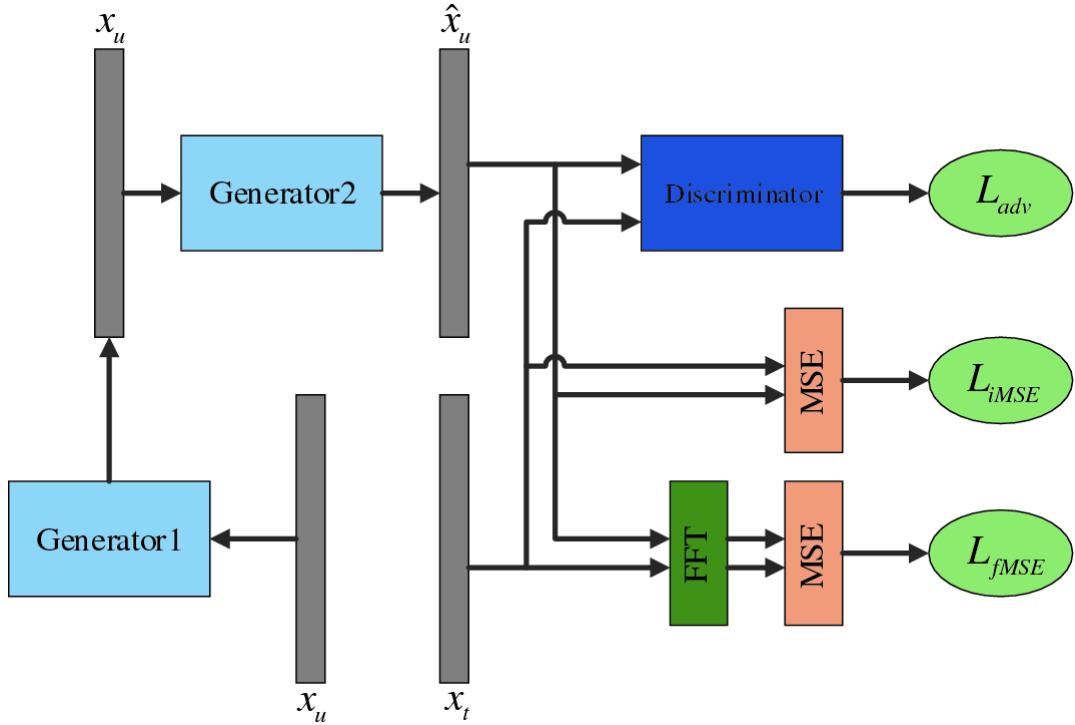


Fig. 13.8 The architecture of ReconGAN/RefineGAN. A two-fold chained network with 2 generators cascaded was used as the generator to produce reconstructed MR images P_r from undersampled MR images π_d . Reconstructed MR images P_r together with ground truth MR images π_i were sent to the discriminator for the calculation of the adversarial loss F_{DV} . Besides, image domain MSE loss $\mathbf{W}_r \mathbf{x}_r$ and frequency domain MSE loss $\mathbf{W}_r \mathbf{x}_r$ were added to the total loss function for training the generator

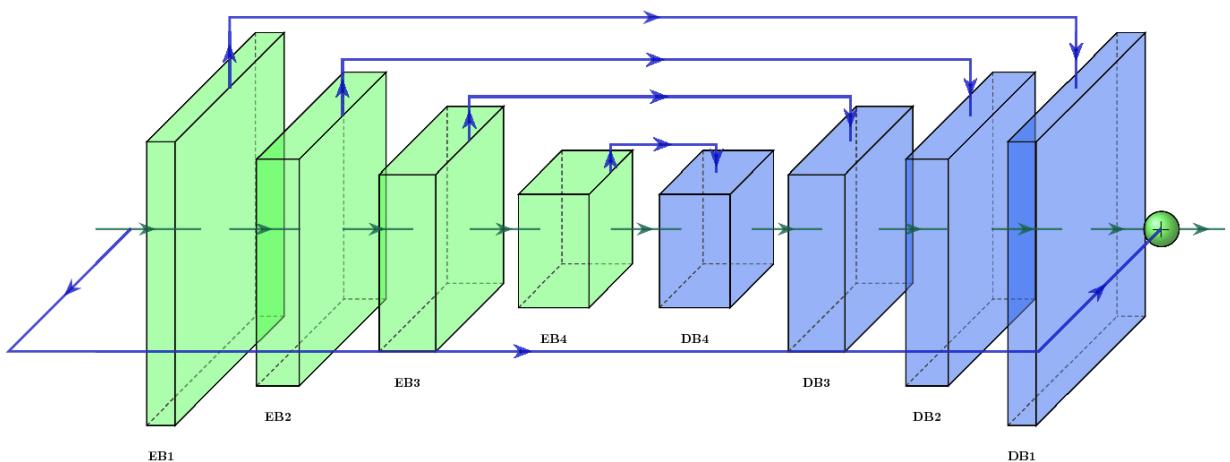


Fig. 13.9 Structure of the generator in ReconGAN/RefineGAN. The generator consisted of 4 encoder blocks in the encoding path and 4 decoder blocks in the decoding path. Skip connection was linked between blocks of the same scale in

different paths. A shortcut connection was applied between the input and output of the generator, which turns the generator into a refinement function (EB: Encoder Block; DB: Decoder Block.)

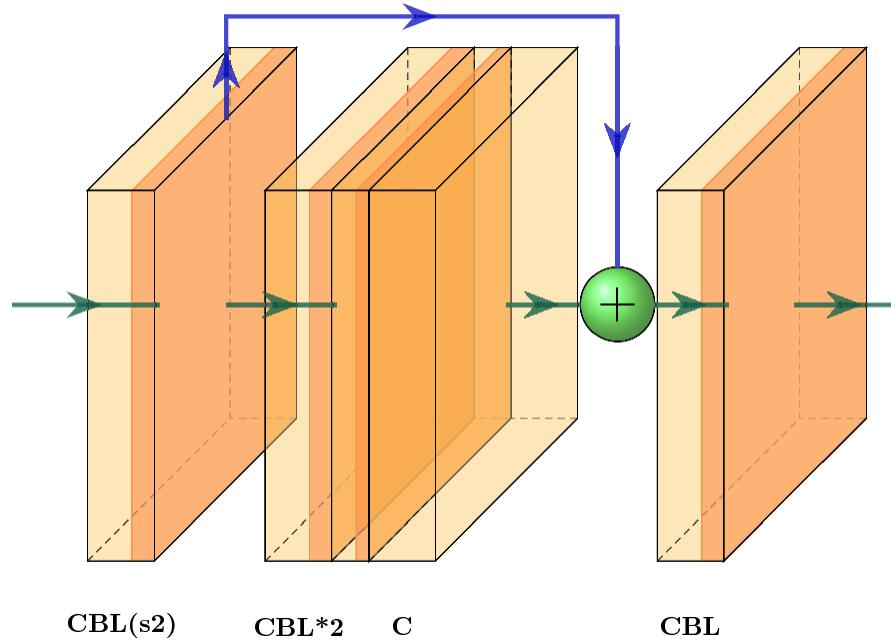


Fig. 13.10 Structure of the encoder block in the generator of ReconGAN/RefineGAN (C: Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer.)

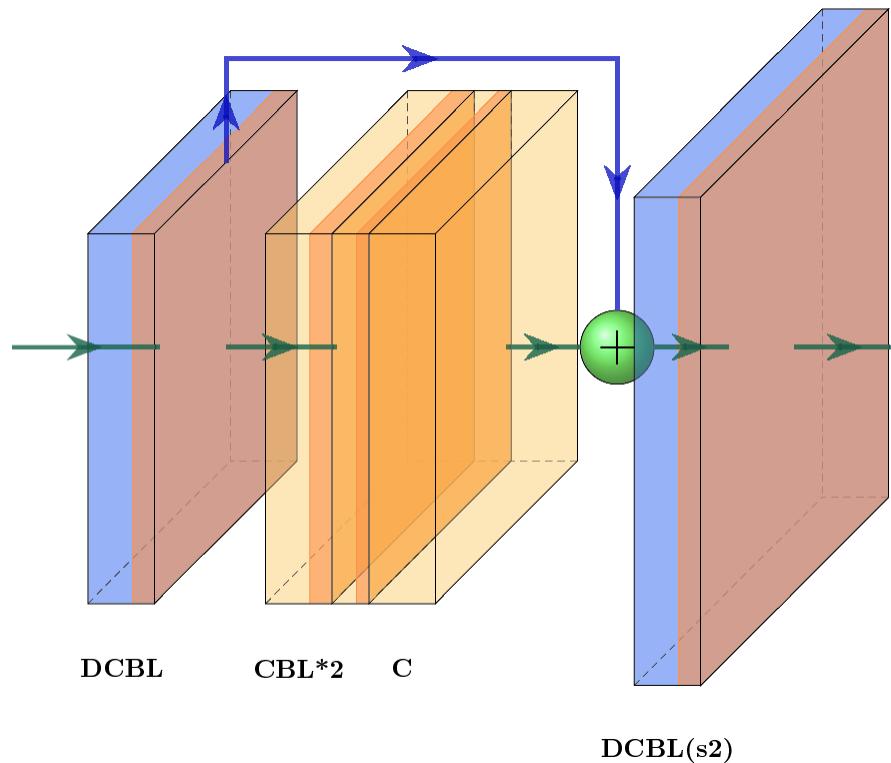


Fig. 13.11 Structure of the decoder block in the generator of ReconGAN/RefineGAN (C: Convolutional layer; DC: De-Convolutional layer; B: Batch Normalization layer; L: Leaky ReLU layer.)

As shown in Fig. 13.9, the generator consisted of 4 encoder blocks in the encoding path and 4 decoder blocks in the decoding path. Skip connection between corresponding blocks that had the same scale was adopted to pass the feature from the encoding path to the decoding path. $\bar{x}_u = G_1(x_u) + x_u$ and $\bar{x}_u = G_1(x_u) + x_u$ were used as the output of the generator instead of $c \in \{0, 1, 2\}$ and $c \in \{0, 1, 2\}$ for better reconstruction details and faster convergence speed. Figures 13.10 and 13.11 show the structure of the encoder block and decoder block. The encoder block consisted of 2 convolutional layers and a residual block inserted between them. The stride of the first convolutional layer was set to 2 for the downsampling. The decoder block consisted of 2 transposed convolutional layers and a residual block inserted between them. The stride of the second transposed convolutional layer was set to 2 for upsampling. The residual block consisted of three convolutional layers, the kernel number in the second convolutional layer was half of the kernel number in the first and third convolutional layer. A shortcut connection was linked between the input and output of the residual block.

As shown in Fig. 13.12, the encoding path in the generator was adopted as the main structure of the discriminator. An FC layer and Sigmoid function were cascaded after the encoding path for the classification result.

The loss function of ReconGAN/RefineGAN was consisted of: image domain MSE loss $\mathbf{W}_r \mathbf{x}_r$, frequency domain MSE loss $\mathbf{W}_f \mathbf{x}_f$, and the adversarial loss F_{DV} , which can be defined as:

$$L_{iMSE}(\theta_{G_1}, \theta_{G_2}) = \frac{1}{2} \| x_t - \hat{x}_u \|_2^2, \quad (13.19)$$

$$L_{fMSE}(\theta_{G_1}, \theta_{G_2}) = \frac{1}{2} \| \mathcal{F}x_t - \mathcal{F}\hat{x}_u \|_2^2, \quad (13.20)$$

$$\begin{aligned} L_{adv}(\theta_{G_1}, \theta_{G_2}, \theta_D) &= \mathbb{E}_{x_t \sim P_{train}(x_t)} [\log D_{\theta_D}(x_t)] \\ &+ \mathbb{E}_{x_u \sim p_{G_2 G_1}(x_u)} [\log(1 - D_{\theta_D}(G_{\theta_{G_2}}(G_{\theta_{G_1}}(x_u))))]. \end{aligned} \quad (13.21)$$

The total loss function can be represented as:

$$L_{\text{TOTAL}}(\theta_{G_1}, \theta_{G_2}, \theta_D) = \alpha L_{\text{iMSE}}(\theta_{G_1}, \theta_{G_2}) + \beta L_{\text{fMSE}}(\theta_{G_1}, \theta_{G_2}) + L_{\text{adv}}(\theta_{G_1}, \theta_{G_2}, \theta_D), \quad (13.22)$$

where $\alpha = 15$, $z \in \mathbb{R}^D$ were the hyperparameters that controlled the balance of different components in the loss function.

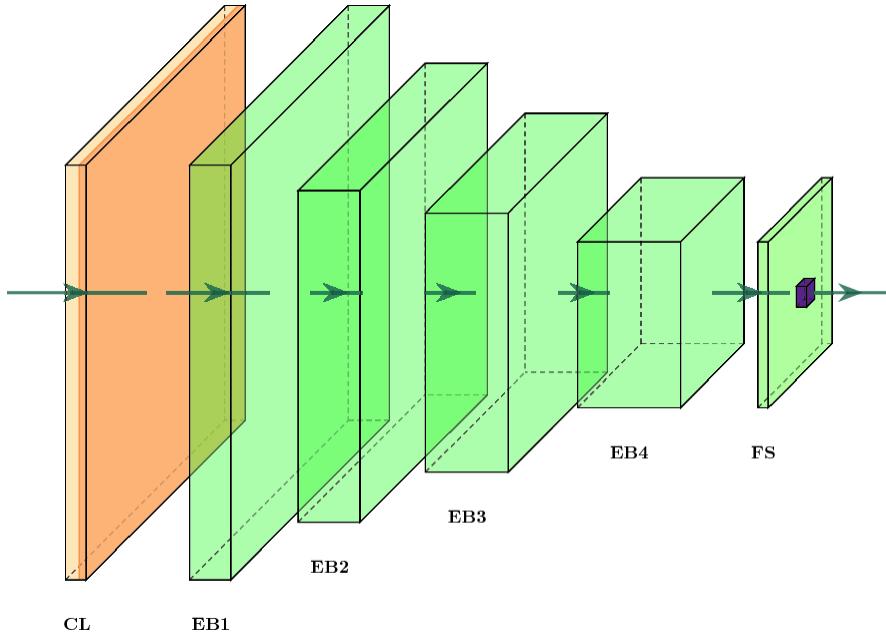


Fig. 13.12 Structure of the discriminator in ReconGAN/RefineGAN. The main structure adopted the encoding path of the generator. A full connection layer and a Sigmoid activation function were cascaded at the end of the discriminator (C: Convolutional layer; L: Leaky ReLU layer; F: Full Connection layer; S: Sigmoid function; EB: Encoder Block.)

13.2.4 Evaluation Methods

Generally, the evaluation methods include objective methods for fidelity quality assessment and subjective methods for perceptual quality assessment. In this section, we review the most popular metrics for fast MRI quality evaluation.

13.2.4.1 Fidelity Quality Assessment

First, we introduce the Peak Signal-to-Noise Ratio (PSNR), which is the most commonly used evaluation criteria for image transformation tasks

(e.g., reconstruction, super-resolution, de-noising). It involves the data range to measure the pixel-level Mean Squared Error (MSE):

$$\text{PSNR}(I_{\text{rec}}, I_{\text{gt}}) = 10 \cdot \log_{10} \left(\frac{L^2}{\frac{1}{N} \sum_{i=1}^N (I_{\text{rec}}(i) - I_{\text{gt}}(i))^2} \right), \quad (13.23)$$

where L denotes the data range (generally $L = 1.0$ in MRI reconstruction tasks), and N is the number of all the pixels in P_G and I_{gt} . PSNR represents the pixel-wise accuracy of the reconstruction regardless of the acquisition sequences of the multimodal MRI.

Besides, considering the importance of image structural information, such as brightness, contrast and structures, Structural SIMilarity index (SSIM) is formed as:

$$\text{SSIM}(x, y) = \frac{2\mu_x\mu_y + \kappa_1}{\mu_x^2 + \mu_y^2 + \kappa_1} \cdot \frac{\sigma_{xy} + \kappa_2}{\sigma_x^2 + \sigma_y^2 + \kappa_2}, \quad (13.24)$$

where x, y denote two images, μ and σ^2 are the mean and variance, \mathcal{M} is the covariance between x and y , and κ_1, κ_2 are constant relaxation terms.

13.2.4.2 Perceptual Quality Assessment

The perceptual quality of an image represents how realistic it looks. In MRI images reconstruction tasks, the most reliable perceptual quality assessment is the mean opinion score (MOS), which asks experienced radiologists to rate the reconstructed images. Typically, the images are rated from 0 to 4 depending on the reconstructed image quality (i.e., non-diagnostic, poor, fair, good, and excellent), and the final MOS is calculated as the arithmetic mean of the scores of all raters. In some cases, the rater may also mark the low perceptual quality features such as low SNR and motion artefacts. Although the MOS seems to be faithful, it has limitations such as inter-/inner-raters bias and variance of rating criteria and the scoring might be time-consuming. Thus, the Frechet Inception Distance (FID) [85], as a learning based perceptual quality assessment, is becoming more commonly used for evaluation in GAN based image reconstruction tasks. It considers the high-level global features of a group of images (e.g., the reconstructed images) as a multidimensional Gaussian distribution $V(\mathcal{G}, \mathcal{D})$, and measures the differences between the two distributions of the reconstructed

images W_1 and the ground truth images χ^2 . It first converts each group of images into a distribution of 2048 features in the latent space of a pre-trained image classification model Inception-V3 [86]. Then, the FID between these two distributions is calculated as:

$$\text{FID}(\mathbb{I}_{\text{rec}}, \mathbb{I}_{\text{gt}}) = \|\mu_{\text{gt}} - \mu_{\text{rec}}\|^2 + \text{Tr}(\Sigma_{\text{gt}} + \Sigma_{\text{rec}} - 2(\Sigma_{\text{gt}}\Sigma_{\text{rec}})^{1/2}). \quad (13.25)$$

The FID becomes a popular metric for image perceptual quality assessment in GAN based image generation tasks because it is fully automatic and the features extracted from Inception-V3 are close to real-world object classification problems, which tend to mimic human perception similarity for images.

13.3 Benchmarking

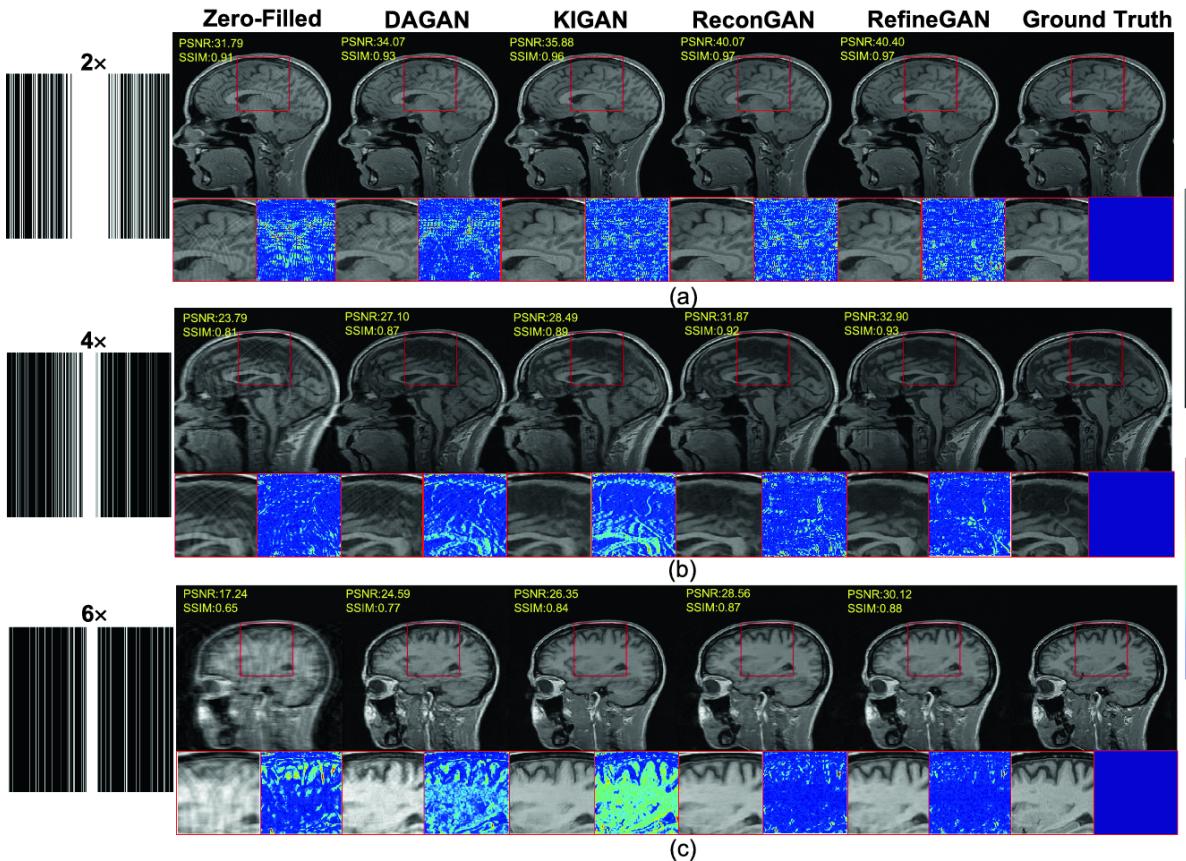


Fig. 13.13 Brain reconstruction result using the Cartesian mask. From top to bottom $2 \times$, $4 \times$ and $6 \times$ acceleration, respectively. From left to right are Zero-Filled (ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

In this chapter, we benchmark four GAN based algorithms, i.e., DAGAN, KIGAN, ReconGAN and RefineGAN, for fast MRI. Figure 13.13 shows the brain reconstruction results using different acceleration factors ($2 \times$, $4 \times$, $6 \times$). It is obvious that the zero-filled (ZF) image has strong artefacts inside the brain tissue. From the entire image, the DAGAN effectively removes the artefacts in the ZF. However, in terms of the zoomed-in areas, there still exists some residual artefacts. For the reconstructions produced by KIGAN, blurring artefacts still exist. Although ReconGAN shows a significant reduction of aliasing artefacts, the edge details are not reconstructed clearly enough. It can be seen that the reconstructed details of RefineGAN are relatively fine, and the reconstruction quality is close to that of the ground truth.

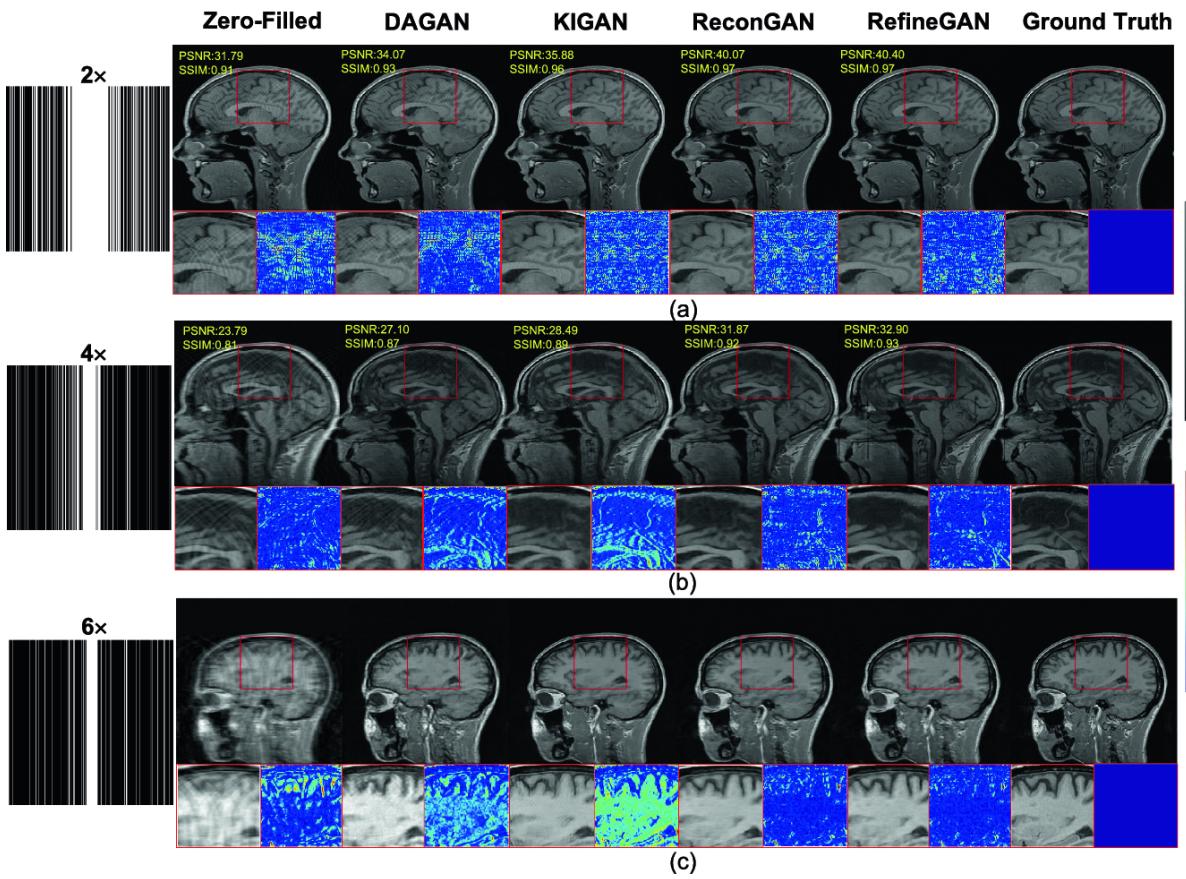


Fig. 13.14 Knee reconstruction result using the Cartesian mask. From top to bottom $2 \times$, $4 \times$ and $6 \times$ acceleration, respectively. From left to right are Zero-Filled (ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

Besides, knee reconstruction results using different Cartesian masks are shown in Fig. 13.14. It can be seen that except for the ZF images, all methods can reconstruct acceptable MR images. As the acceleration factor goes high, obvious aliasing artefacts are produced in DAGAN images. As can be observed in the zoomed-in areas and the corresponding error maps, KIGAN can not restore clear vessels. ReconGAN and RefineGAN show better reconstruction results with higher PSNR and SSIM. In addition, the quantitative values of the RefineGAN are superior to those of the other methods.

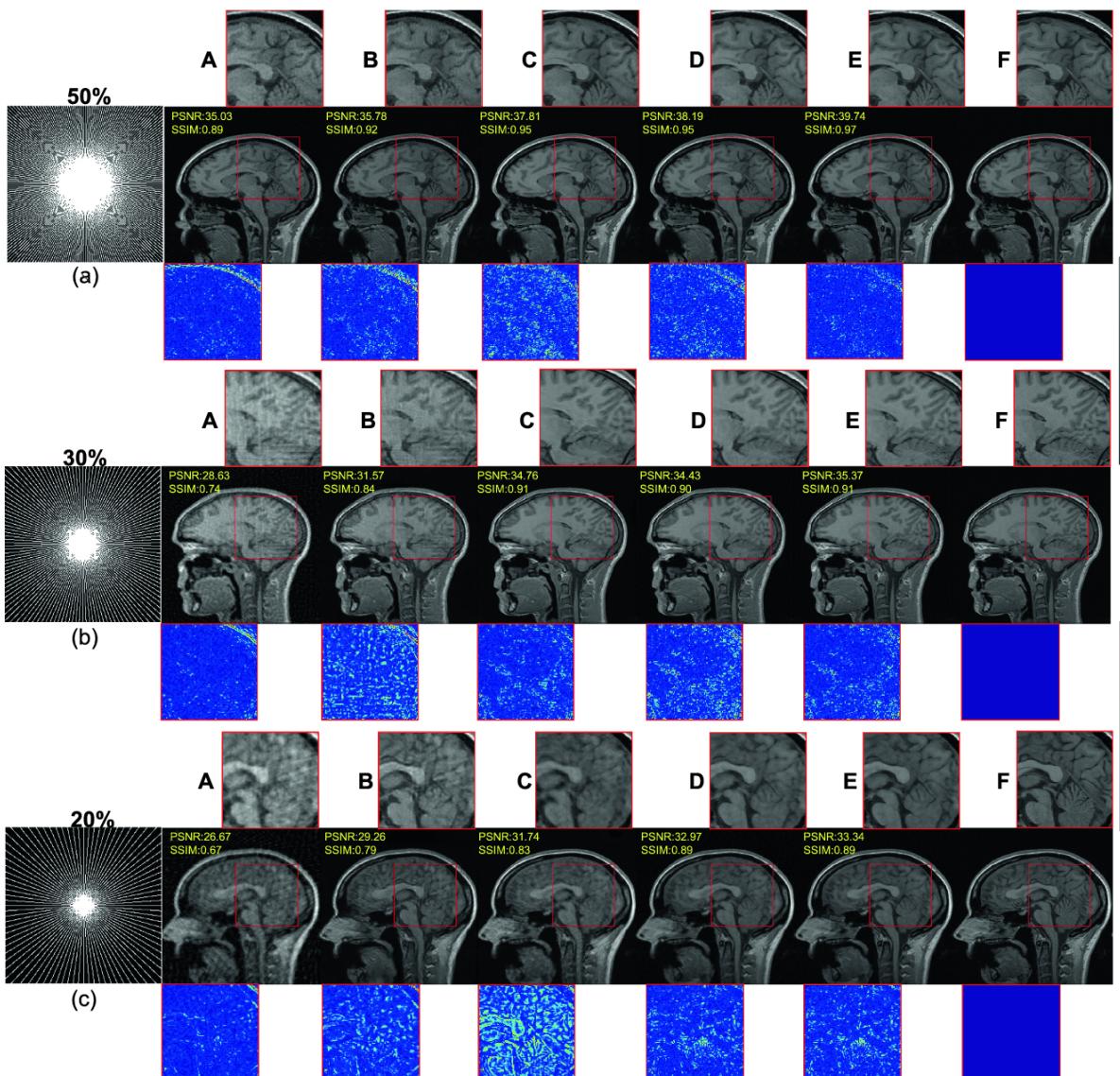


Fig. 13.15 Brain reconstruction result using the radial mask. From top to bottom, the sampling rate (SR) is 50, 30, and 20%, respectively. From left to right are Zero-Filled

(ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

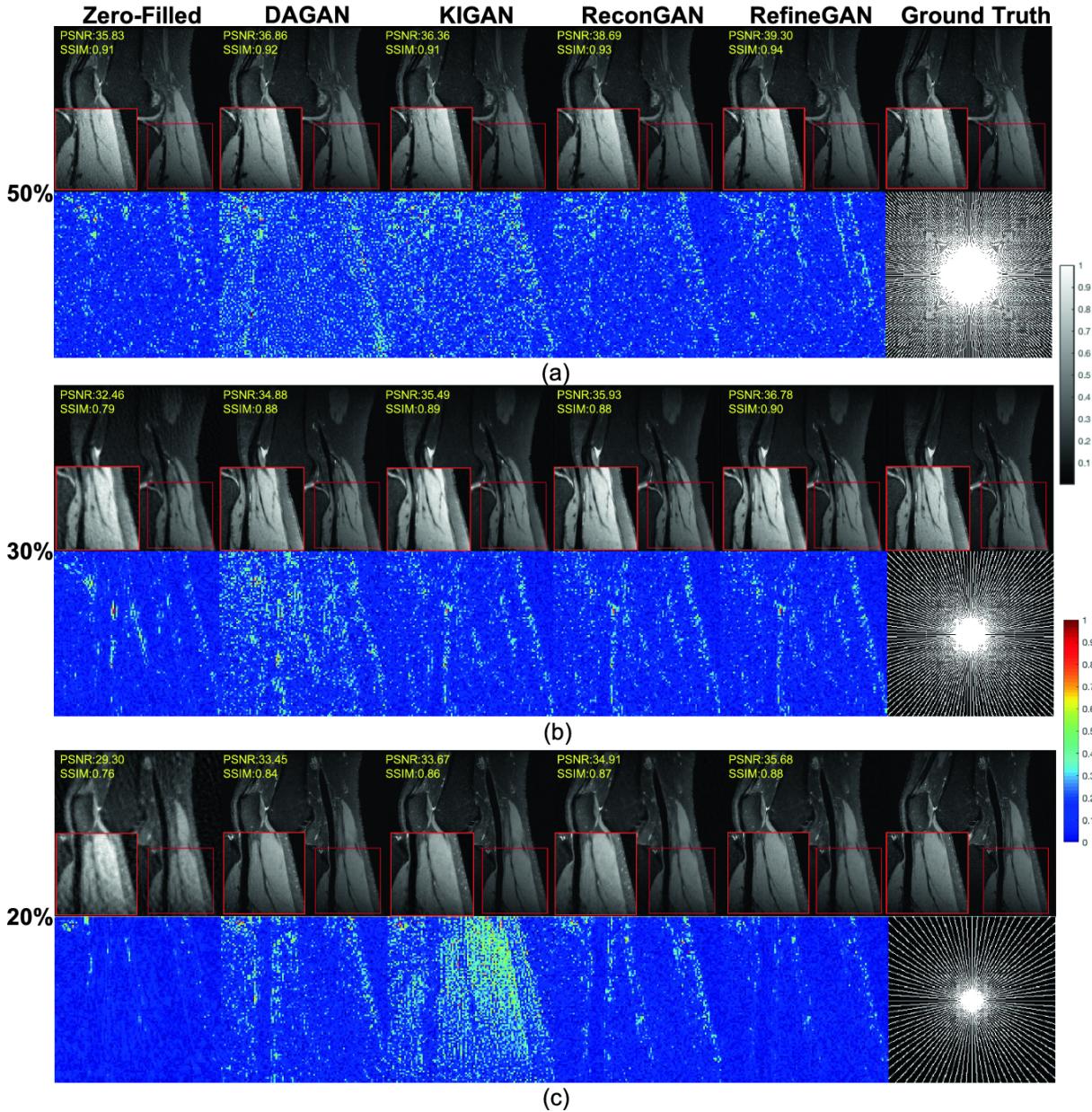


Fig. 13.16 Knee reconstruction result using the radial mask. From top to bottom $2 \times$, $4 \times$ and $6 \times$ acceleration, respectively. From left to right are Zero-Filled (ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

Furthermore, we also used radial and spiral masks for training and testing each GAN based method. The sampling rate (SR) of each mask is 50, 30 and 20%. Figures 13.15 and 13.16 show the brain and knee reconstruction results using radial masks. We can see that the image

reconstructed by the ZF method under the radial mask has strong blurring artefacts, and the details in the brain and knee cannot be distinguished clearly. When SR=20%, from the error maps, we can see that there are still obvious blurring artefacts and obscure blood vessels in the results of DAGAN and KIGAN. However, both ReconGAN and RefineGAN can restore sharper vessel edges and finer textures compared to other methods. Besides, RefineGAN has better PSNR and SSIM quantification.

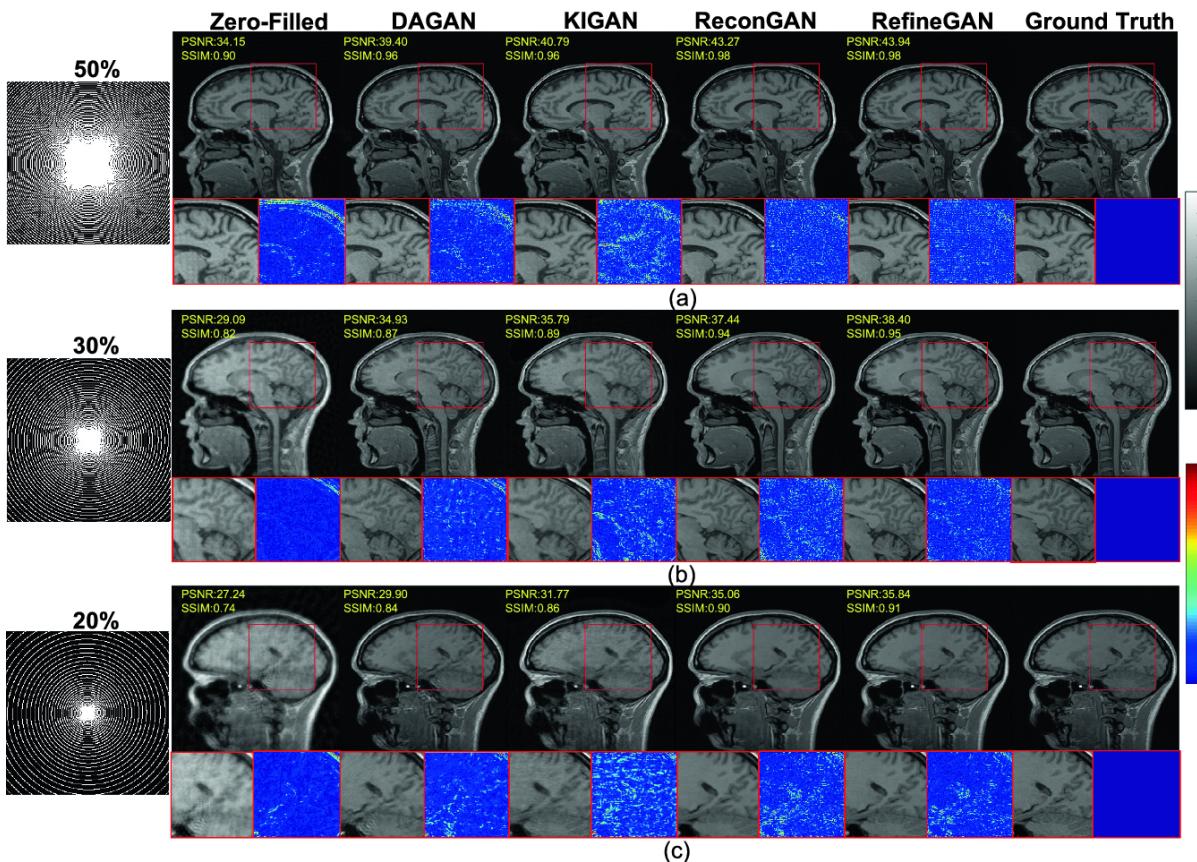


Fig. 13.17 Brain reconstruction result using the radial mask. From top to bottom, the sampling rate (SR) is 50, 30, and 20%, respectively. From left to right are Zero-Filled (ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

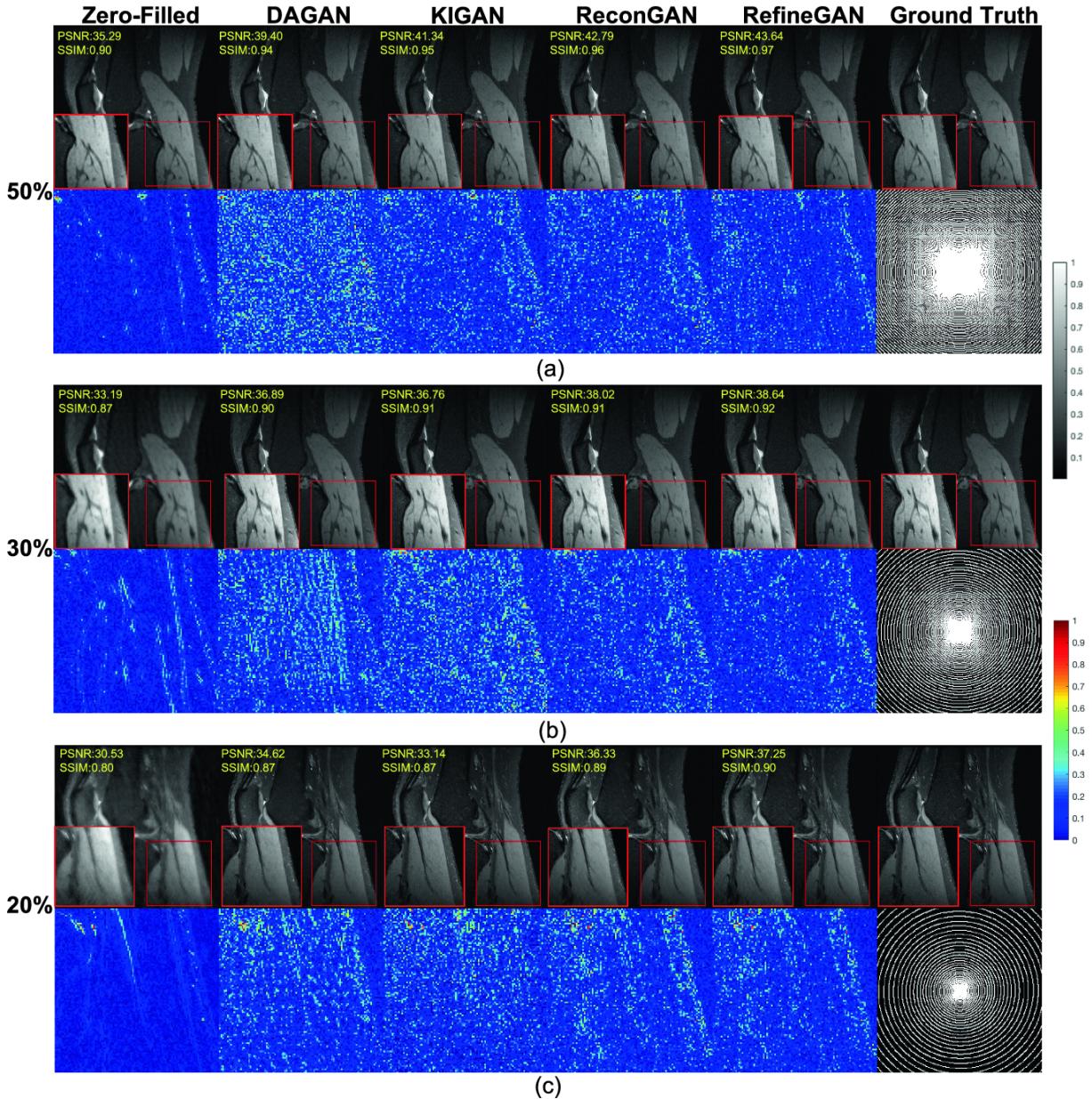


Fig. 13.18 Knee reconstruction result using the radial mask. From top to bottom $2 \times$, $4 \times$ and $6 \times$ acceleration, respectively. From left to right are Zero-Filled (ZF), DAGAN, KIGAN, ReconGAN, RefineGAN, Ground Truth (GT)

The results are similar using spiral masks. Figures 13.17 and 13.18 show the brain and knee reconstruction results of each method using different spiral masks. Through the error map, we can intuitively see that the reconstructed image of RefineGAN has fewer errors, and the reconstruction details are also better. The image reconstructed by

RefineGAN can clearly show the details of the gray matter in the brain and the details of the blood vessels in the knee.

Table 13.3 The quantitative metrics (PSNR, SSIM and RMSE) of the brain MRI data using different GAN based methods. The bold numbers indicate the best results

Mask	AF/SR	Metric	ZF	DAGAN	KIGAN	ReconGAN	RefineGAN
Cartesian	2X	PSNR	30.94±2.75	33.79±1.88	33.90±2.55	39.08±1.34	39.40±1.33
		SSIM	0.92±0.02	0.93±0.01	0.96±0.01	0.97±0.00	0.97±0.00
		RMSE	1.57±0.01	0.72±0.43	0.78±0.53	0.20±0.09	0.19±0.08
	4X	PSNR	23.69±3.02	28.76±1.95	28.14±1.84	32.07±1.65	32.67±1.56
		SSIM	0.79±0.03	0.86±0.02	0.88±0.02	0.92±0.01	0.93±0.01
		RMSE	8.86±7.01	2.35±1.50	2.67±1.40	1.05±0.54	0.91±0.08
	6X	PSNR	19.47±2.31	25.4±1.57	27.91±1.57	29.23±1.68	29.95±1.61
		SSIM	0.66±0.04	0.77±0.03	0.86±0.02	0.88±0.02	0.89±0.02
		RMSE	21.3±14.1	4.89±2.33	2.75±1.31	2.03±1.00	1.71±0.83
Radial	50%	PSNR	34.28±1.03	35.24±1.14	38.93±1.10	37.89±0.93	39.38±0.88
		SSIM	0.87±0.02	0.91±0.01	0.96±0.01	0.95±0.01	0.97±0.00
		RMSE	1.94±0.22	1.74±0.22	1.01±0.12	1.28±0.13	0.88±0.11
	30%	PSNR	28.83±0.72	30.99±2.33	33.97±1.05	34.35±0.99	35.21±1.05
		SSIM	0.73±0.02	0.82±0.02	0.91±0.01	0.91±0.01	0.92±0.01
		RMSE	3.10±0.30	2.01±0.27	1.60±0.18	1.93±0.21	1.02±0.17
	20%	PSNR	26.47±0.64	28.96±1.90	31.87±0.94	31.96±0.87	32.83±0.94
		SSIM	0.65±0.02	0.78±0.02	0.84±0.03	0.87±0.02	0.89±0.02
		RMSE	6.08±0.36	2.47±0.45	2.56±0.27	2.54±0.25	1.56±0.32
Spiral	50%	PSNR	34.87±1.01	38.08±0.99	41.56±1.00	43.69±0.58	44.36±0.57
		SSIM	0.90±0.01	0.95±0.01	0.96±0.01	0.97±0.01	0.98±0.00
		RMSE	1.62±0.18	1.25±0.01	0.84±0.10	0.72±0.66	0.59±0.61
	30%	PSNR	29.55±0.71	34.87±2.79	36.71±1.09	37.93±0.78	38.61±0.82
		SSIM	0.83±0.01	0.87±0.01	0.91±0.02	0.95±0.01	0.95±0.00
		RMSE	3.93±0.28	1.72±0.41	1.65±0.20	1.27±0.11	0.94±0.19
	20%	PSNR	26.62±0.63	28.87±2.29	32.43±0.72	35.02±0.81	35.11±0.85
		SSIM	0.73±0.01	0.80±0.04	0.88±0.03	0.91±0.01	0.92±0.01

Mask	AF/SR	Metric	ZF	DAGAN	KIGAN	ReconGAN	RefineGAN
		RMSE	7.12±0.36	2.29±0.31	2.40±0.20	1.78±0.16	1.52±0.27

Table 13.4 The quantitative metrics (PSNR, SSIM, and RMSE) of the knee MRI data using different GAN based methods. The bold numbers indicate the best results

Mask	AF/SR	Metric	ZF	DAGAN	KIGAN	ReconGAN	RefineGAN
Cartesian	2X	PSNR	34.66±2.98	38.91±1.59	38.53±2.51	42.37±1.60	42.41±1.98
		SSIM	0.95±0.01	0.94±0.01	0.96±0.01	0.97±0.00	0.98±0.00
		RMSE	1.64±1.32	0.52±0.31	0.83±1.03	0.23±0.09	0.24±0.16
	4X	PSNR	27.31±3.23	34.35±1.77	34.70±1.74	34.88±1.96	35.58±1.74
		SSIM	0.84±0.02	0.86±0.02	0.89±0.02	0.90±0.02	0.91±0.02
		RMSE	10.30±11.00	1.55±1.20	1.49±1.27	1.37±0.93	1.14±0.66
	6X	PSNR	25.15±3.37	32.67±1.89	30.83±2.09	32.34±2.16	33.36±1.81
		SSIM	0.79±0.03	0.82±0.03	0.84±0.02	0.86±0.02	0.87±0.02
		RMSE	18.2±22.3	2.36±2.09	4.08±4.29	2.62±2.19	2.00±1.52
Radial	50%	PSNR	35.17±1.37	36.70±1.37	37.17±1.37	38.38±1.22	38.91±1.21
		SSIM	0.90±0.02	0.91±0.01	0.92±0.01	0.93±0.01	0.93±0.01
		RMSE	2.37±0.50	1.97±0.59	1.53±0.18	1.22±0.16	1.14±0.15
	30%	PSNR	32.69±1.13	34.02±1.39	35.27±1.40	35.96±1.27	36.21±1.29
		SSIM	0.80±0.03	0.87±0.02	0.88±0.01	0.88±0.02	0.89±0.02
		RMSE	4.87±0.89	2.23±0.76	1.39±0.67	1.61±0.22	1.36±0.32
	20%	PSNR	30.77±1.15	33.69±1.51	33.49±1.54	34.48±1.26	34.72±1.30
		SSIM	0.75±0.03	0.84±0.02	0.85±0.02	0.86±0.02	0.86±0.02
		RMSE	7.13±1.24	2.76±0.87	2.68±0.73	1.91±0.27	1.61±0.34
Spiral	50%	PSNR	35.62±1.27	39.37±1.39	41.39±1.37	42.53±0.75	43.04±0.77
		SSIM	0.91±0.02	0.94±0.01	0.94±0.02	0.96±0.01	0.97±0.02
		RMSE	1.78±0.54	1.06±0.10	0.97±0.14	0.73±0.15	0.61±0.07
	30%	PSNR	33.48±1.12	36.19±1.35	36.26±1.18	38.20±1.31	38.49±1.33
		SSIM	0.86±0.02	0.90±0.01	0.90±0.02	0.92±0.01	0.92±0.01
		RMSE	3.71±0.91	1.67±0.72	1.64±0.35	1.24±0.18	0.98±0.23
	20%	PSNR	30.97±1.14	34.88±1.39	33.83±1.50	36.51±1.23	36.75±1.27
		SSIM	0.81±0.03	0.88±0.02	0.88±0.01	0.90±0.02	0.90±0.02

Mask	AF/SR	Metric	ZF	DAGAN	KIGAN	ReconGAN	RefineGAN
		RMSE	5.39±1.26	2.29±0.56	1.95±0.25	1.51±0.20	1.32±0.14

The quantitative metrics (PSNR, SSIM and RMSE) of each GAN based method using different under-sampling masks are shown in Tables 13.3 and 13.4. We can draw similar conclusions as the qualitative visualisation results that the image quality after RefineGAN reconstruction is better than other methods. Even at a high acceleration factor or high under-sampling rate, the reconstructed images of RefineGAN still have high SNR.

Tables 13.3 and 13.4 show the quantitative metrics, including PSNR, SSIM and RMSE for all compared methods. The numbers in Tables 13.3 and 13.4 represent the mean and standard deviation values of the corresponding metrics (bold numbers indicate the best performance). Compared to DAGAN, KIGAN and ReconGAN, the RefineGAN framework has outperformed them remarkably at different acceleration factors.

13.4 Discussion

We have established that GAN based methods such as DAGAN, KIGAN, ReconGAN, and RefineGAN excel in generating faithful, photo-realistic reconstruction of undersampled MR images and in removing the undersampling artefacts. Despite their representing a successful category of CS-MRI techniques, GAN based methods suffer from training instability and slow convergence to a global minimum [87, 88]. This problem can be alleviated by Wasserstein GAN (WGAN) [89]. Instead of minimising Jensen-Shannon divergence between the reconstructed image and ground truth in the training phase, WGAN aims to reduce the Wasserstein distance. WGAN based CS-MRI models have achieved superior reconstruction performance compared with GAN based methods e.g., DAGAN [90, 91]. However, to minimise the Wasserstein distance, the discriminator needs to satisfy the 1-Lipschitz constraint. The original WGAN paper adopted a weight clipping approach to enforce this constraint but this can itself cause training instability. The potential improvement includes a gradient clipping approach [92] or dividing the weights of neural networks by their spectral

norm [93]. Altogether, modifying the loss function of GAN is key to addressing its training instability issue.

Another problem with the GAN loss function is that it may affect the high frequency texture [63] and smoothing the reconstructed image [94]. GAN may also generate high frequency noise [63]. Mardani et al. (2019) suggested solving the high frequency texture issue with the least square GAN [63]. L1 loss between reconstructed images and ground truth was also added as an auxiliary function, to serve as a low-pass filter to remove high frequency noises.

Apart from the issues with the GAN loss function during training, GAN based methods suffer from reconstruction instability [95]. This means that a small perturbation to the input image leads to large scale changes in the reconstruction output. Antun et al. [95] evaluated the instabilities of DAGAN and other deep learning based CS-MRI models and showed that the performance of DAGAN deteriorated as the undersampling ratio increased during testing. That is if more k -space pixels were undersampled more than the undersampling ratio in the images used to train DAGAN, its performance decreases. In contrast, variational network and deep cascaded convolutional neural networks (DC-CNN), two non-GAN based techniques, did not experience the same issue. This indicates the incorporation of GAN may reduce the generalisability of the model to images collected under different undersampling ratios and contribute to its reconstruction instability.

Another instability issue with DAGAN is that if small structural details were added to an MRI image, such as random letters, and this image was undersampled retrospectively, DAGAN was unable to reconstruct the structural details. In contrast, variational network and DC-CNN were superior in recovering these structural details [95]. Even though these two instability tests revealed a critical weakness of DAGAN, DAGAN, being one of the earliest GAN based methods, may not capture the improved reconstruction accuracy and network architectural complexity of the more recent methods [90, 91]. Hence, more instability testing is required on more GAN based methods to conclude whether this category of CS-MRI techniques suffers from reconstruction instability.

These results on instability testing still question the generalisability of GAN based models. In other words, it is unclear whether after being trained

with images from one source, e.g., one organ, under one undersampling ratio etc., a model will achieve equivalent performance on images from another source. This is often referred to in the literature as a zero-shot inference test. However, GAN based methods appear to excel in these inference tests. To illustrate, DAGAN, having been trained with MR images of the brain of a healthy volunteer, successfully reconstructed the MR images from a patient with the brain tumour, faithfully preserving the tumour structure [3]. Another example is that GANCS, another GAN based method, was trained with MR images of the abdomen and any images containing the enlarged adrenal gland were removed from the training set. In testing, GANCS was capable of recovering enlarged adrenal gland in patients with adrenal hypertrophy. No evidence of hallucination of the reconstructed images was found [63]. These successes in the zero-shot inference of pathological structures challenged the instability test results from Antun et al. One interpretation is that the additional structural details added to the MR images by Antun et al., such as letters, may not represent real-life physiological or pathological deviations of the MR images from the training set. It was also possible that the zero-shot inference tests on DAGAN and GANCS concerned only a small sample size, limited to a few pathological conditions. Larger scale zero-shot inference tests are necessary to evaluate the generalisability of GAN based CS-MRI methods.

In summary, future research directions on GAN based fast MRI may include more robust training strategies, e.g., combining GAN with genetic algorithm [96], incorporating edge and texture enhancement [54], reducing possible hallucination [97, 98], coupling with explainable AI (XAI) modules [99, 100], and further consideration with MR physics [55].

13.5 Conclusion

We carried out a mini review, benchmarked and compared four different GAN-based network architectures for fast MRI reconstruction in this chapter. Our comparison used the various sampling patterns, different masks on corresponding datasets that have covered commonly used clinical MRI scenarios. For our systematic research and measurement, we used both traditional and newly proposed quantitative tools. The outcomes of qualitative visualization were also examined and compared. To summarise,

our mini review and benchmarking have revealed that all GAN-based approaches could obtain promising results at lower acceleration factors. However, when the acceleration factors are high, some GAN-based architectures, such as DAGAN and KIGAN, may not be sufficient for MRI reconstruction applications. Furthermore, as compared to other GAN-based methods, the RefineGAN has improved reconstruction accuracy and perceptual efficiency. Future development incorporating MR physics and XAI into GAN based models will provide promising pathways for clinical deployment with more transparency of the reconstruction algorithms.

Acknowledgements

This study was supported in part by the British Heart Foundation [Project Number: TG/18/5/34111, PG/16/78/32402], the European Research Council Innovative Medicines Initiative [DRAGON, H2020-JTI-IMI2 101005122], the AI for Health Imaging Award [CHAIMELEON, H2020-SC1-FA-DTS-2019-1 952172], and the UK Research and Innovation Future Leaders Fellowship [MR/V023799/1].

References

1. Kieren, G.H.: Reducing acquisition time in clinical MRI by data undersampling and compressed sensing reconstruction. *Phys. Med. Biol.* **R297–322** (2015)
2. Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: the application of compressed sensing for rapid MR imaging. *Magn. Reson. Med.* **58**(6), 1182–1195 (2007)
3. Yang, G., Yu, S., Dong, H., Slabaugh, G., Pier, L.D., Ye, X., Liu, F., Arridge, S., Keegan, J., Guo, Y., Firmin, D.: DAGAN: deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction. *IEEE Trans. Med. Imaging* **37**(6), 1310–1321 (2018)
4. Suetens, P.: *Fundamentals of Medical Imaging*, 2nd edn. Cambridge University Press (2009)
5. Mansfield, P.: Multi-planar image formation using NMR spin echoes. *J. Phys. C: Solid State Phys.* (1977)
6. Hennig, J., Nauerth, A., Friedburg, H.: RARE imaging: a fast imaging method for clinical MR. *Magn. Reson. Med.* **3**(6), 823–833 (1986)

[[Crossref](#)]

7. Haase, A., Frahm, J., Matthaei, D., Merboldt, K.-D.: FLASH imaging. Rapid NMR Imaging Using Low Flip-Angle Pulses. Technical report (1986)
8. Zisselman, E., Adler, A., Elad, M.: Compressed learning for image classification: a deep neural network approach. In: Handbook of Numerical Analysis, vol. 19, pp. 3–17. Elsevier B.V., Jan 2018
9. Fair, M.J., Gatehouse, P.D., DiBella, E.V.R., Firmin, D.N.: A review of 3D first-pass, whole-heart, myocardial perfusion cardiovascular magnetic resonance, Aug 2015
10. Ravishankar, S., Bresler, Y.: MR image reconstruction from highly undersampled k-space data by dictionary learning. *IEEE Trans. Med. Imaging* **30**(5), 1028–1041 (2011)
11. Yang, J., Zhang, Y., Yin, W.: A fast alternating direction method for TVL1-L2 signal reconstruction from partial Fourier data. *IEEE J. Sel. Top. Signal Process.* **4**(2), 288–297 (2010)
12. Eksioglu, E.M.: Decoupled algorithm for MRI reconstruction using nonlocal block matching model: BM3D-MRI. *J. Math. Imaging Vis.* **56**(3), 430–440 (2016)
13. Yue, H., Ongie, G., Ramani, S., Jacob, M.: Generalized higher degree total variation (HDTV) regularization. *IEEE Trans. Image Process.* **23**(6), 2423–2435 (2014)
[[MathSciNet](#)][[Crossref](#)]
14. Liu, Y., Cai, J.-F., Zhan, Z., Guo, D., Ye, J., Chen, Z., Xiaobo, Q.: Balanced sparse model for tight frames in compressed sensing magnetic resonance imaging. *PLOS ONE* **10**(4), e0119584 (2015)
15. Mohammad H. Kayvanrad, A. Jonathan McLeod, John S.H. Baxter, Charles A. McKenzie, and Terry M. Peters. Stationary wavelet transform for under-sampled MRI reconstruction. *Magnetic Resonance Imaging*, 32(10):1353–1364, dec 2014
16. M. Guerquin-Kern, M. Haberlin, K. P. Pruessmann, and M. Unser. A fast wavelet-based reconstruction method for magnetic resonance imaging. *IEEE Transactions on Medical Imaging*, 30(9):1649–1660, sep 2011
17. Hammernik, K., Klatzer, T., Kobler, E., Recht, M.P., Sodickson, D.K., Pock, T.,

- Knoll, F.: Learning a variational network for reconstruction of accelerated MRI data. *Mag. Reson. Med.* **79**(6), 3055–3071 (2018)
- 18. Chen, J., Yang, G., Khan, H., Zhang, H., Zhang, Y., Zhao, S., Mohiaddin, R., Wong, T., Firmin, D., Keegan, J.: JAS-GAN: generative adversarial network based joint atrium and scar segmentation on unbalanced atrial targets. *IEEE J. Biomed. Health Inf.* (2021)
 - 19. Yinze, W., Hatipoglu, S., Alonso-Álvarez, D., Gatehouse, P., Li, B., Gao, Y., Firmin, D., Keegan, J., Yang, G.: Fast and automated segmentation for the three-directional multi-slice cine myocardial velocity mapping. *Diagnostics* **11**(2), 346 (2021)
[Crossref]
 - 20. Wu, Y., Hatipoglu, S., Alonso-Álvarez, D., Gatehouse, P., Firmin, D., Keegan, J., Yang, G.: Automated multi-channel segmentation for the 4D myocardial velocity mapping cardiac MR. In: Medical Imaging 2021: Computer-Aided Diagnosis, vol. 11597, pp. 115970P. International Society for Optics and Photonics (2021)
 - 21. Jin, Yao, Y., G., Fang, Y., Li, R., Xiaomei, X., Liu, Y., Lai, X.: 3D PBV-Net: an automated prostate MRI data segmentation method. *Comput. Biol. Med.* **128**, 104160 (2021)
 - 22. Zhou, X., Ye, Q., Jiang, Y., Wang, M., Niu, Z., Menpes-Smith, W., Fang, E.F., Liu, Z., Xia, J., Yang, G.: Systematic and comprehensive automated ventricle segmentation on ventricle images of the elderly patients: a retrospective study. *Frontiers Aging Neurosci.* **12** (2020)
 - 23. Liu, Y., Yang, G., Hosseiny, M., Azadikhah, A., Mirak, S.A., Miao, Q., Raman, S.S., Sung, K.: Exploring uncertainty measures in bayesian deep attentive neural networks for prostate zonal segmentation. *IEEE Access* **8**, 151817–151828 (2020)
 - 24. Ferreira, P.F., Martin, R.R., Scott, A.D., Khalique, Z., Yang, G., Nielles-Vallespin, S., Pennell, D.J., Firmin, D.N.: Automating *in vivo* cardiac diffusion tensor postprocessing with deep learning-based segmentation. *Magn. Reson. Med.* **84**(5), 2801–2814 (2020)
 - 25. Li, M., Wang, C., Zhang, H., Yang, G.: Mv-ran: multiview recurrent aggregation network for echocardiographic sequences segmentation and full cardiac cycle analysis. *Comput. Biol. Med.* **120**, 103728 (2020)
 - 26.

Liu, Y., Yang, G., Mirak, S.A., Hosseiny, M., Azadikhah, A., Zhong, X., Reiter, R.E., Lee, Y., Raman, S.S., Sung, K.: Automatic prostate zonal segmentation using fully convolutional network with feature pyramid attention. *IEEE Access* **7**, 163626–163632 (2019)

27. Zhuang, X., Li, L., Payer, C., Štern, D., Urschler, M., Heinrich, M.P., Oster, J., Wang, C., Smedby, Ö., Bian, C., et al.: Evaluation of algorithms for multi-modality whole heart segmentation: an open-access grand challenge. *Med. Image Anal.* **58**, 101537 (2019)
28. Mo, Y., Liu, F., McIlwraith, D., Yang, G., Zhang, J., He, T., Guo, Y. (2018) The deep poincaré map: A novel approach for left ventricle segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 561–568. Springer
29. Zhang, W., Yang, G., Huang, H., Yang, W., Xu, X., Liu, Y., Lai, X.: ME-Net: multi-encoder net framework for brain tumor segmentation. *Int. J. Imaging Syst. Technol.* (2021)
30. Yang, G., Chen, J., Gao, Z., Li, S., Ni, H., Angelini, E., Wong, T., Mohiaddin, R., Nyktari, E., Wage, R., et al.: Simultaneous left atrium anatomy and scar segmentations via deep learning in multiview information with attention. *Fut. Gen. Comput. Syst.* **107**, 215–228 (2020)
[Crossref]
31. Li, L., Fuping, W., Yang, G., Lingchao, X., Wong, T., Mohiaddin, R., Firmin, D., Keegan, J., Zhuang, X.: Atrial scar quantification via multi-scale CNN in the graph-cuts framework. *Med. Image Anal.* **60**, 101595 (2020)
32. Zhang, L., Yang, G., Ye, X.: Automatic skin lesion segmentation by coupling deep fully convolutional networks and shallow network with textons. *J. Med. Imaging* **6**(2), 024001 (2019)
33. Yang, G., Zhuang, X., Khan, H., Nyktari, E., Haldar, S., Li, L., Wage, R., Ye, X., Slabaugh, G., Mohiaddin, R. et al.: Left atrial scarring segmentation from delayed-enhancement cardiac MRI images: a deep learning approach. *Cardiovasc. Imaging* **109** (2018)
34. Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., Shinohara, R.T., Berger, C., Ha, S.M., Rozycki, M., et al.: Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge (2018). [arXiv:1811.02629](https://arxiv.org/abs/1811.02629)

35. Mok, T.C.W., Chung, A.: Fast symmetric diffeomorphic image registration with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4644–4653 (2020)
36. de Vos, B.D., Berendsen, F.F., Viergever, M.A., Staring, M., Išgum, I.: End-to-end unsupervised deformable image registration with a convolutional neural network. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 204–212. Springer (2017)
37. Wu, G., Kim, M., Wang, Q., Munsell, B.C., Shen, D.: Scalable high-performance image registration framework by unsupervised deep feature representations learning. *IEEE Trans. Biomed. Eng.* **63**(7), 1505–1516 (2015)
38. Chenchu, X., Zhang, D., Chong, J., Chen, B., Li, S.: Synthesis of gadolinium-enhanced liver tumors on nonenhanced liver MR images using pixel-level graph reinforcement learning. *Med. Image Anal.* **69**, 101976 (2021)
39. Wang, C., Yang, G., Papanastasiou, G., Tsafaris, S.A., Newby, D.E., Gray, C., Macnaught, G., MacGillivray, T.J.: DiCyc: GAN-based deformation invariant cross-domain information fusion for medical image synthesis. *Inf. Fusion* **67**, 147–160 (2021)
40. Gao, Z., Zhang, H., Dong, S., Sun, S., Wang, X., Yang, G., Wu, W., Li, S., de Albuquerque, V.H.C.: Salient object detection in the distributed cloud-edge intelligent network. *IEEE Netw.* **34**(2), 216–224 (2020)
41. Wang, C., Dong, S., Zhao, X., Papanastasiou, G., Zhang, H., Yang, G.: Saliencygan: deep learning semisupervised salient object detection in the fog of IOT. *IEEE Trans. Ind. Inf.* **16**(4), 2667–2676 (2019)
[Crossref]
42. Ali, A.-R., Li, J., Kanwal, S., Yang, G., Hussain, A., O’Shea, S.J.: A novel fuzzy multilayer perceptron (F-MLP) for the detection of irregularity in skin lesion border using dermoscopic images. *Frontiers Med.* **7** (2020)
43. Yang, M., Xiao, X., Liu, Z., Sun, L., Guo, W., Cui, L., Sun, D., Zhang, P., Yang, G.: Deep retinanet for dynamic left ventricle detection in multiview echocardiography classification. *Sci. Program.* (2020)
44. Li, M., Dong, S., Zhang, K., Gao, Z., Wu, X., Zhang, H., Yang, G., Li, S.: Deep learning intra-image and inter-images features for co-saliency detection. In: BMVC, vol. 291 (2018)

45. Dong, S., Sun, G.S., Wang, X., Li, M., Zhang, H., Yang, G., Liu, H., Li, S., et al.: Holistic and deep feature pyramids for saliency detection. In: BMVC, p. 67 (2018)
46. Dong, H., Yang, G., Liu, F., Mo, Y., Guo, Y.: Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks. In: Annual Conference on Medical Image Understanding and Analysis, pp. 506–517. Springer (2017)
47. Hu, S., Gao, Y., Niu, Z., Jiang, Y., Li, L., Xiao, X., Wang, M., Fang, E.F., Menpes-Smith, W., Xia, J., et al.: Weakly supervised deep learning for COVID-19 infection detection and classification from CT images. *IEEE Access* **8**, 118869–118883 (2020)
48. Cao, Y., Wang, Z., Liu, Z., Li, Y., Xiao, X., Sun, L., Zhang, Y., Hou, H., Zhang, P., Yang, G.: Multiparameter synchronous measurement with IVUS images for intelligently diagnosing coronary cardiac disease. *IEEE Trans. Instrum. Meas.* (2020)
49. Zhang, N., Yang, G., Gao, Z., Chenchu, X., Zhang, Y., Shi, R., Keegan, J., Lei, X., Zhang, H., Fan, Z., et al.: Deep learning for diagnosis of chronic myocardial infarction on nonenhanced cardiac cine MRI. *Radiology* **291**(3), 606–617 (2019) [[Crossref](#)]
50. Roberts, M., Driggs, D., Thorpe, M., Gilbey, J., Yeung, M., Ursprung, S., Aviles-Rivero, A.I., Etmann, C., McCague, C., Beer, L., et al.: Machine learning for COVID-19 detection and prognostication using chest radiographs and CT scans: a systematic methodological review (2020). [arXiv:2008.06388](#)
51. Soltaninejad, M., Zhang, L., Lambrou, T., Yang, G., Allinson, N., Ye, X.: MRI brain tumor segmentation and patient survival prediction using random forests and fully convolutional networks. In: International MICCAI Brainlesion Workshop, pp. 204–215. Springer (2017)
52. Jin, C., Yu, H., Ke, J., Ding, P., Yi, Y., Jiang, X., Duan, X., Tang, J., Chang, D.T., Wu, X., et al.: Predicting treatment response from longitudinal images using multi-task deep learning. *Nat. Commun.* **12**(1), 1–11 (2021)
53. Nielsen, A., Hansen, M.B., Tietze, A., Mouridsen, K.: Prediction of tissue outcome and assessment of treatment effect in acute ischemic stroke using deep learning. *Stroke* **49**(6), 1394–1401 (2018)
- 54.

Chen, Y., Firmin, D., Yang, G.: Wavelet improved GAN for MRI reconstruction. In: Medical Imaging 2021: Physics of Medical Imaging, vol. 11595, p. 1159513. International Society for Optics and Photonics (2021)

55. Lv, J., Wang, C., Yang, G.: PIC-GAN: a parallel imaging coupled generative adversarial network for accelerated multi-channel MRI reconstruction. *Diagnostics* **11**(1), 61 (2021) [\[Crossref\]](#)
56. Lv, J., Zhu, J., Yang, G.: Which GAN? a comparative study of generative adversarial network (GAN) based fast MRI reconstruction. *Philos. Trans. R. Soc. A*
57. Yuan, Z., Jiang, M., Wang, Y., Wei, B., Li, Y., Wang, P., Menpes-Smith, W., Niu, Z., Yang, G.: SARA-GAN: Self-attention and relative average discriminator based generative adversarial networks for fast compressed sensing MRI reconstruction. *Frontiers Neuroinformatics* **14** (2020)
58. Guo, Y., Wang, C., Zhang, H., Yang, G.: Deep attentive wasserstein generative adversarial networks for mri reconstruction with recurrent context-awareness. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 167–177. Springer (2020)
59. Schlemper, J., Yang, G., Ferreira, P., Scott, A., McGill, L.-A., Khalique, Z., Gorodezky, M., Roehl, M., Keegan, Pennell, J.D., et al.: Stochastic deep compressive sensing for the reconstruction of diffusion tensor cardiac MRI. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 295–303. Springer (2018)
60. Seitzer, M., Yang, G., Schlemper, J., Oktay, O., Würfl, T., Christlein, V., Wong, T., Mohiaddin, R., Firmin, D., Keegan, J., et al.: Adversarial and perceptual refinement for compressed sensing mri reconstruction. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 232–240. Springer (2018)
61. Donoho, D.L., et al.: Compressed sensing. *IEEE Trans. Inf. Theory* **52**(4), 1289–1306 (2006)
62. Quan, T.M., Nguyen-Duc, T., Jeong, W.K.: Compressed sensing MRI reconstruction using a generative adversarial network with a cyclic loss. *IEEE Trans. Med. Imaging* **37**(6), 1488–1497 (2018)
- 63.

Mardani, M., Gong, E., Cheng, J.Y., Vasanawala, S.S., Zaharchuk, G., Xing, L., Pauly, J.M.: Deep generative adversarial neural networks for compressive sensing MRI. *IEEE Trans. Med. Imaging* **38**(1), 167–179 (2019)

64. Schlemper, J., Caballero, J., Hajnal, J.V., Price, A.N., Rueckert, D.: A deep cascade of convolutional neural networks for dynamic MR image reconstruction. *IEEE Trans. Med. Imaging* **37**(2), 491–503 (2018)
65. Huang, Q., Yang, D., Wu, P., Qu, H., Yi, J., Metaxas, D.: MRI reconstruction via cascaded channel-wise attention network. In: Proceedings—International Symposium on Biomedical Imaging, Apr. 2019, pp. 1622–1626. IEEE Computer Society, Apr. 2019
66. Eo, Jun, Y., Kim, T., Jang, J., Lee, H.-J., Hwang, D.: KIKI-net: cross-domain convolutional neural networks for reconstructing undersampled magnetic resonance images. *Magn. Reson. Med.* **80**(5), 2188–2201 (2018)
67. Yu, S., Dong, H., Yang, G., Slabaugh, G., Dragotti, P.L., Ye, X., Liu, F., Arridge, S., Keegan, J., Firmin, D., et al.: Deep de-aliasing for fast compressive sensing MRI (2017). [arXiv:1705.07137](https://arxiv.org/abs/1705.07137)
68. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. Technical report (2014)
69. Deng, X., Yang, R., Zurich, E., Xu, M., Dragotti, P.L.: Wavelet Domain Style Transfer for an Effective Perception-distortion Tradeoff in Single Image Super-Resolution. Technical report (2019)
70. Deshmane, A., Gulani, V., Griswold, M.A., Seiberlich, N.: Parallel MR imaging. *J. Magn. Reson. Imaging* **36**(1), 55–72 (2012). <https://doi.org/10.1002/jmri.23639>
71. Zhang, X., Lian, Q., Yang, Y., Su, Y.: A deep unrolling network inspired by total variation for compressed sensing MRI. *Digit. Signal Process.: Rev. J.* **107** (2020). Publisher: Elsevier
72. Diamond, S., Sitzmann, V., Heide, F., Wetzstein, G.: Unrolled Optimization with Deep Priors (2017). Publisher: arxiv.org
73. Qin, C., Schlemper, J., Caballero, J., Price, A.N., Hajnal, J.V., Rueckert, D.: Convolutional recurrent neural networks for dynamic MR image reconstruction. *IEEE Trans. Med. Imaging* **38**(1), 280–290 (2019). Publisher: ieeexplore.ieee.org

74. Mohana, M.J., Madhulika, M.S., Divya, G.D., Meghana, R.K., Apoorva, S.: Feature extraction using convolution neural networks (CNN) and deep learning. In: 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), pp. 2319–2323 (2018)
75. Zhao, D., Zhao, F., Gan, Y.: Reference-driven compressed sensing MR image reconstruction using deep convolutional neural networks without pre-training. *Sensors* (Switzerland) **20**(1) (2020)
76. Liang, D., Cheng, J., Ke, Z., Ying, L.: Deep MRI Reconstruction: Unrolled Optimization Algorithms Meet Neural Networks (2019). [arXiv:1907.11711](https://arxiv.org/abs/1907.11711). Publisher: arxiv.org
77. Liang, D., Cheng, J., Ke, Z., Ying, L.: Deep magnetic resonance image reconstruction: inverse problems meet neural networks. *IEEE Signal Process. Mag.* **37**(1), 141–151 (2020). Publisher: ieeexplore.ieee.org
78. Zhang, H.-M., Dong, B.: A review on deep learning in medical image reconstruction. *J. Oper. Res. Soc. China* **8**(2), 311–340 (2020)
79. Lee, D., Yoo, J., Tak, S., Ye, J.C.: Deep residual learning for accelerated MRI using magnitude and phase networks. *IEEE Trans. Biomed. Eng.* **65**(9), 1985–1995 (2018)
80. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9351, pp. 234–241. Springer, May 2015
81. Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017
82. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
83. Russakovsky, O., Deng, J., Hao, S., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**(3), 211–252 (2015) [[MathSciNet](#)][[Crossref](#)]

84. Shaul, R., David, I., Shitrit, O., Raviv, R.: Subsampled brain MRI reconstruction by generative adversarial neural networks. *Med. Image Anal.* **65**, 101747 (2020)
85. Heusel, M., Ramsauer, M., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium (2018). [arXiv:1706.08500](https://arxiv.org/abs/1706.08500)
86. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
87. Chen, Z., Tong, Y.: Face Super-Resolution Through Wasserstein GANs. Technical report (2017)
88. Wiatrak, M., Albrecht, S.V., Nystrom, A.: Stabilizing GANs: A Survey
Stabilizing Generative Adversarial Networks: A Survey. Technical report (2020)
89. Arjovsky, S.C., Bottou, L.: Wasserstein GAN. Technical report (2017)
90. Jiang, M., Yuan, Yang, Z.X., Zhang, J., Gong, Y., Xia, L., Li, T.: Accelerating CS-MRI reconstruction with fine-tuning wasserstein generative adversarial network. *IEEE Access* **7**, 152347–152357 (2019). Publisher: ieeexplore.ieee.org
91. Oh, G., Sim, B., Chung, H.J., Sunwoo, L., Ye, J.C.: Unpaired deep learning for accelerated MRI using optimal transport driven CycleGAN. *IEEE Trans. Comput. Imaging* **6**, 1285–1296 (2020). Publisher: ieeexplore.ieee.org
92. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved Training of Wasserstein GANs Montreal Institute for Learning Algorithms. Technical report (2017)
93. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral Normalization for Generative Adversarial Networks (2018). [cs, stat], Feb. 2018. [arXiv: 1802.05957](https://arxiv.org/abs/1802.05957)
94. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging* **3**(1), 47–57 (2017). Conference Name: IEEE Transactions on Computational Imaging
95. Antun, V., Renna, F., Poon, C., Adcock, B., Hansen, A.C.: On instabilities of deep learning in image reconstruction and the potential costs of AI. In: Proceedings of the National Academy of Sciences, vol. 117(48), pp. 30088–

30095, Dec. 2020. Publisher: National Academy of Sciences Section:
Colloquium on the Science of Deep Learning (2020)

96. Hao, J., Wang, C., Zhang, H., Yang, G.: Annealing genetic GAN for minority oversampling (2020). [arXiv:2008.01967](https://arxiv.org/abs/2008.01967)
97. Zhu, J., Yang, G., Lio, P.: How can we make GAN perform better in single medical image super-resolution? A lesion focused multi-scale approach. In: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pp. 1669–1673. IEEE (2019)
98. Zhu, J., Yang, G., Lio, P.: Lesion focused super-resolution. In: Medical Imaging 2019: Image Processing, vol. 10949, p. 109491L. International Society for Optics and Photonics (2019)
99. Ye, Q., Xia, J., Yang, G.: Explainable AI for covid-19 CT classifiers: an initial comparison study (2021). [arXiv:2104.14506](https://arxiv.org/abs/2104.14506)
100. Yang, G., Ye, Q., Xia, J.: Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: a mini-review, two showcases and beyond (2021). [arXiv:2102.01998](https://arxiv.org/abs/2102.01998)

14. Generative Adversarial Networks for Data Augmentation in X-Ray Medical Imaging

Talib Iqbali¹✉ and M. Arif Wani¹✉

(1) Department of Computer Science, University of Kashmir, Srinagar,
India

✉ **Talib Iqbali**

Email: talibiqbali.scholar@kashmiruniversity.net

✉ **M. Arif Wani (Corresponding author)**

Email: awani@uok.edu.in

Abstract

This work focuses on the use of Generative Adversarial Networks for data augmentation in X-ray medical imaging. Data augmentation can be employed in situations where little data or imbalanced datasets are present. There are two main reasons why some medical datasets are limited or imbalanced: either there is little data available for some rare diseases, or the privacy policy of medical organizations does not allow it to share the data. But deep learning models often need a large and balanced dataset for efficient training of models, which can produce high accuracy results. A Progressively Growing Generative Adversarial Network (PGGAN) is proposed for data augmentation of X-ray images to mitigate the above problem. Extensive experimentation has shown that PGGAN generates good quality synthetic X-ray images for data augmentation to balance the dataset. The resulting balanced dataset used several classification models for testing. Various state-of-the-art classification models are adopted in transfer learning and fine-tuned to test the augmentation process.

14.1 Introduction

Deep learning has become a popular approach for the classification of images, but it needs adequate data for efficient training [9, 10]. In some situations, like medical applications, sufficient data may not be available. One of the solutions to tackle this problem is through data augmentation. Basic transformations like flipping, rotating, etc., are applied to the original images to generate new data. Another issue requires a dataset to be balanced to avoid classification models getting trained with a bias towards a particular class. A balanced dataset has an equal number of training samples of each class. Data augmentation is needed to mitigate this problem by generating more examples of underrepresented classes so that the training dataset has an equal number of samples of each class.

Generative Adversarial Network (GAN) [2] is a technique used to generate new images that look like real images. The generator network of GAN creates new images that resemble the original dataset distribution and are referred to as synthetic images. The images from the original dataset are labeled as real, and the images from the generator network are labeled as synthetic for training purposes of the discriminator network of GAN. The discriminator network is trained as a binary classifier to distinguish between real and synthetic images. Both the networks use a common loss function that the generator network tries to minimize and the discriminator network tries to maximize, which sometimes makes the GAN difficult to converge [14].

The study [19] has shown GAN is a special case of Artificial Curiosity (AC), where one network is trained to generate data, and another network learns to predict properties of the generated data. The generator network minimizes the objective function maximized by the predictor network and vice-versa. GAN is also closely related to Predictability Minimization (PM). The neural encoder learns data distributions by maximizing the objective function, and the neural predictor learns to detect and remove the redundant information by minimizing the objective function. Originally, GAN has been defined by a multilayer perceptron. It has undergone various modifications and has been used across many problem domains like image generation, image-to-image translation, text-image synthesis, and medical imaging [1, 5, 6].

One modification in GAN is the use of Convolutional Neural Networks (CNN) called Deep Convolutional GAN (DCGAN), which was introduced in [7]. Various CNN architectures [12, 13] have been used for different applications [24, 25]. DCGAN generator consists of transposed convolutional layers which perform the reverse operation of convolution to up-sample the image. On the other hand, a discriminator is a simple CNN binary classifier. This model has been used in various applications for image generation and data augmentation. However, DCGAN does not produce good quality high-resolution images.

In this work, progressively growing GAN (PGGAN) for generating high-resolution X-ray images is proposed. PGGAN is also a CNN-based generative model but differs in the way it is trained.

The rest of the chapter is organized as follows: Previous work on using GANs and Transfer Learning in X-Ray Imaging is presented in Sect. 14.2. Section 14.3 discusses DCGAN and its limitations for X-ray imaging. PGGAN for augmentation of X-ray images is discussed in Sect. 14.4. Results and Discussion are presented in Sect. 14.5. The summary is finally provided in Sect. 14.6.

14.2 Previous Work on Using GANs and Transfer Learning in X-Ray Imaging

Chest radiography, one of the most common diagnostic imaging tests, requires timely reporting of potential findings in the images. The chest radiograph (chest X-ray) is effective in the characterization and detection of different chest pathologies. Some of the studies that focus on the use of GANs and Transfer Learning (TL) to analyze X-ray images are given below.

The authors in [8] have used the Deep Convolutional GAN (DCGAN) for augmenting the X-ray images. The original dataset used was imbalanced, and the work aimed to classify the different chest pathologies. The dataset was first balanced with augmented X-ray images created with the DCGAN. The number of images in each class of the training dataset was kept the same. It was set to the lowest number of images contained in any class. The balanced dataset of real images was used to train the DCGAN model. The trained DCGAN model was used to augment the X-

ray images dataset. The new augmented and balanced dataset was used to train the classification models to classify different pathologies. Another study [21] focused on COVID-19 detection based on chest X-ray images. The data available was less as compared to what was needed by deep learning-based classification models. Hence, the authors used GANs for data augmentation with two different model settings (SGD and Adam optimizer). The newly generated chest X-ray images were used to augment the original dataset. The authors in [17] addressed the cardiac abnormality classification in chest X-rays. The experiments showed that less data is required with semi-supervised learning GANs than conventional supervised learning convolutional neural networks. The authors in [18] have used conditional generative adversarial networks (cGANs) to generate realistic chest X-ray images with different disease characteristics by conditioning its generation on a real image sample.

In a related work [23], the authors have used an auto-encoder network for one-class learning of abnormal chest X-ray identification. Their proposed architecture comprises three deep neural networks (auto-encoder, discriminator, and 2nd encoder). Each of them learns by competing while collaborating among these to model the underlying content structure of the normal chest X-rays. This approach takes only normal chest X-ray images as input while training. During the testing phase, the network can distinguish normal chest X-rays from abnormal ones.

Transfer learning is a technique where a pre-trained model is used for solving a new but similar problem. The model is fine-tuned to suit the new problem. In deep learning, it has been shown that transfer learning effectively increases the accuracy of the models (mainly classification models).

The authors in [3] have used transfer learning to classify different chest pathologies by using pre-trained models like VGG and Inception V3. The results have shown that the use of pre-trained models can increase classification accuracy. In a similar study [15], the authors used transfer learning in five different classification models (AlexNet, DenseNet121, ResNet18, InceptionV3, and GoogleNet) for Pneumonia detection from chest X-ray images. All the models were pre-trained on the ImageNet dataset. In another study [11], the authors used transfer learning to detect COVID-19 from chest X-ray images. It has been shown that the use of pre-

trained models like ResNet-34 [22], ResNet-152 [16], ResNet-18 [20] in transfer learning has improved the results of classification.

14.3 Deep Convolutional GAN (DCGAN) and Its Limitations for X-Ray Imaging

DCGAN mostly has convolutional layers, and there are no “pooling” or “un-pooling” layers to decrease and increase the spatial dimensionality of the representation. This architecture uses convolutional layers and transposed convolutional layers to decrease and increase spatial dimensionality. It utilizes batch normalization in all the layers except the last layer of the generator network and the first layer of the discriminator network.

14.3.1 Architecture of DCGAN

DCGAN consists of two Deep Convolutional Neural Networks: the Generator network (G) and the Discriminator network (D). Input to the generator is a 1-D Gaussian noise vector, and it outputs a 2-D image. Discriminator networks take an image as an input and output a label for binary classification. The discriminator is merely a classifier that is used to train the generator network. The generator creates new samples, which are passed to the discriminator. The discriminator distinguishes between the real and synthesized samples; the real being from the original dataset and synthesized is the sample created by the generator. Figure 14.1 shows the overview of the architecture of DCGAN. The generator network takes a 1-D Gaussian noise vector as an input and passes it through the transposed convolutional layers, batch normalization layers, and ReLU activations. It results in a 64×64 image as output. The transposed convolutional layers allow the latent vector to be transformed into a volume with the same shape as an image. The discriminator takes an image of size 64×64 as input. It passes it through the convolutional, batch normalization, and LeakyReLU activations to get the output.

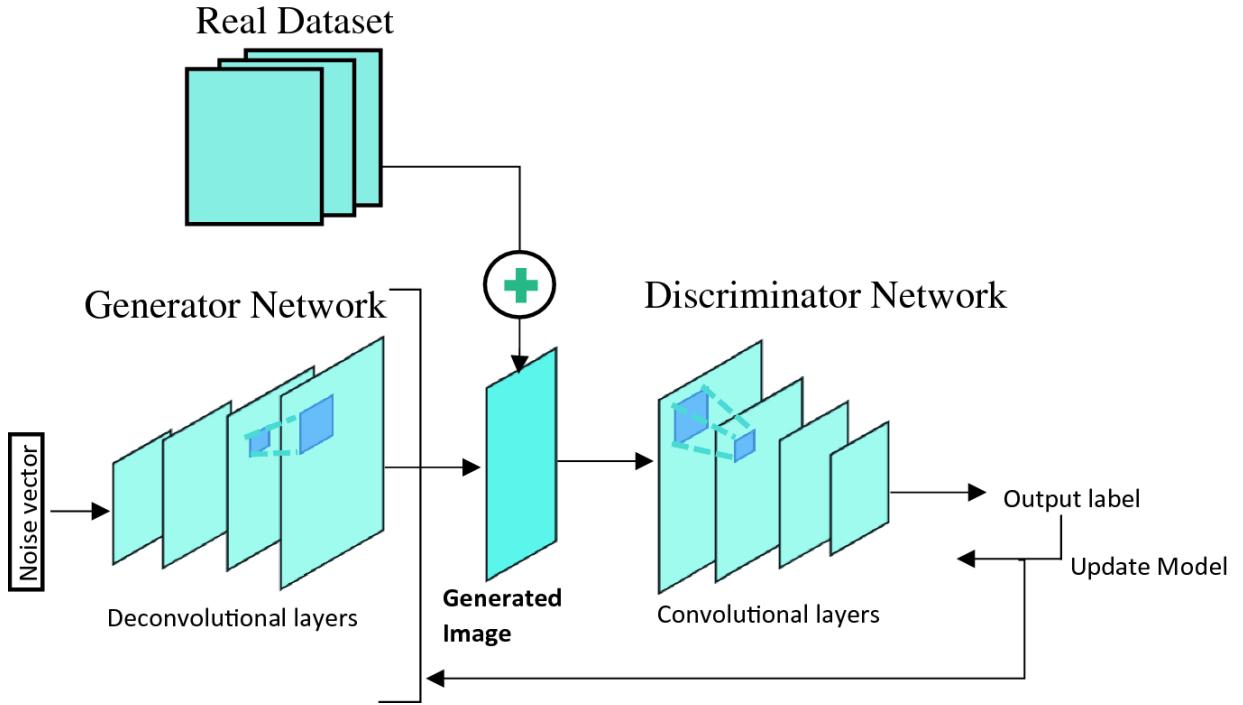


Fig. 14.1 Architecture of DCGAN

14.3.1.1 *Generator Network*

The generator, G , is designed to map the latent space vector (z) to data-space. In the domain of images, converting a given latent space vector (z) to data-space means creating an image with the same size as the training images (in this case, 64×64 image). This can be implemented through a series of two-dimensional transposed convolutional layers, each paired with a 2-D batch normalization layer and a ‘relu’ activation (except the last layer). The generator’s output is fed through a ‘tanh’ function to return it to the input data range of $[-1, 1]$. It is worth noting the batch normalization functions after the transposed convolutional layers, as this is the main task of DCGAN. These layers are helpful to keep the gradients flowing during training. For illustration purposes, an image of the generator of DCGAN is shown below in Fig. 14.2.

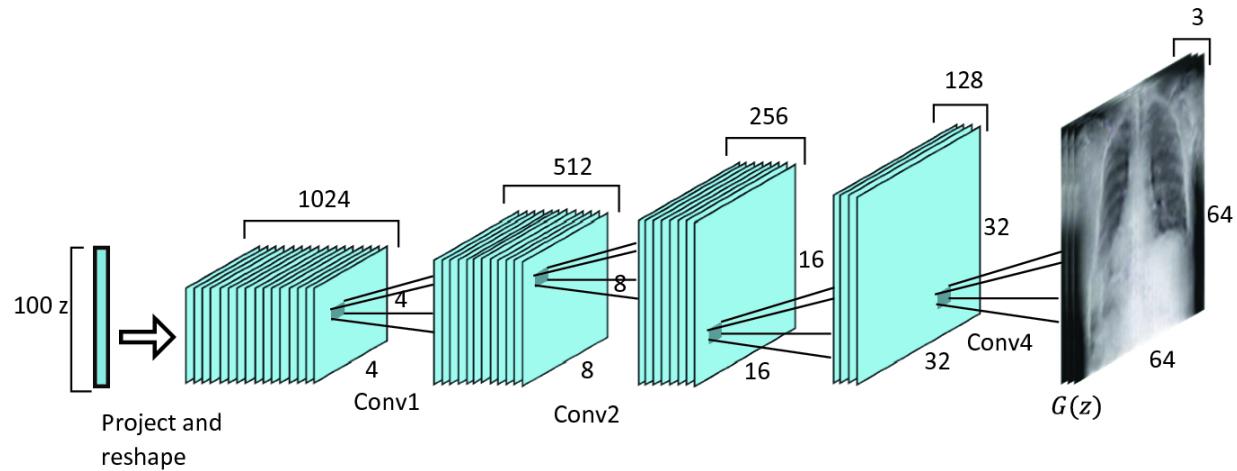


Fig. 14.2 Architecture of generator network

14.3.1.2 Discriminator Network

The discriminator, D, is a CNN-based binary classification network that takes an image as input and produces probability value as an output, a scalar, showing the probability of the input image being real (as opposed to synthesized). The discriminator takes an input image, processes it through different layers (2d convolutional, 2d batch normalization, and LeakyReLU). The final probability is output through the Sigmoid activation function. It is pertinent to mention here that strided convolution is used rather than pooling to down-sample because it lets the network learn its pooling function. Also, batch norm and LeakyReLU functions promote healthy gradient flow, which is critical for the learning process of both G and D.

14.3.2 DCGAN Training

The training of DCGAN networks involves playing a minimax game. i.e., one network tries to minimize, and the other network maximizes the same probability function.

14.3.2.1 Discriminator Training

A batch of samples from the original dataset is passed through the discriminator D and the loss $I(c; \mathcal{G}(z, c))$ is calculated, and then the gradients for the backward pass are computed. A batch of synthesized samples are then generated with the current generator and passed

through the discriminator D, and the loss $\log(1 - D(G(z)))$ is calculated, and then the gradients for the backward pass are computed. The discriminator's optimizer uses the gradients of the original samples and synthesized samples for updating model parameters.

Training the discriminator aims to maximize the probability of correctly classifying a given input as real or synthetic. This requires maximizing the following expression:

$$\text{Log}(D(x)) + \log(1 - D(G(z)))$$

14.3.2.2 Generator Training

The generator is trained to produce better-synthesized samples by minimizing

$$\log(1 - D(G(z)))$$

However, it has been shown that the above expression does not provide sufficient gradients, especially early in the learning process. This issue is addressed by instead of minimizing the above expression, the term given below is maximized.

$$D(x_t) = 1$$

The generator output is classified with the discriminator, and the generator loss is computed using samples from the original dataset, and then gradients for the backward pass are computed. The generator's optimizer uses the gradients for updating model parameters.

The output of the discriminator $D(x)$ (averaged over the batch) for all the samples from the original dataset should theoretically start from a value close to 1 and then converge to 0.5 as the generator gets better. The output of the discriminator $D(G(z))$ (averaged over the batch) for all the synthesized samples should theoretically start from a value close to 0 and then converge to 0.5 as the generator gets better.

14.3.2.3 DCGAN in X-Ray Image Augmentation

The DCGAN model was used for the data augmentation task by training it with 2000 X-ray images of each pathology. During training, the generated samples initially are less accurate and hence are easily classified by the

discriminator. But as the training progresses, the generator produces more accurate samples. It becomes difficult for the discriminator to distinguish between real and synthesized samples. The Discriminator network is trained using examples from the real dataset and generated samples from the generator. The generator network is trained using discriminators loss. A state of equilibrium called the Nash equilibrium can be reached when the discriminator can no longer distinguish between real and synthesized samples. But due to the simultaneous training of generator and discriminator, this equilibrium is difficult to achieve. After training is complete, the generator network generates the new samples, and the discriminator network can be discarded. The DCGAN was trained with different settings of batch size and the number of epochs. The best results were shown when batch size was 128, and the number of epochs was 200. Figure 14.3 shows the output of one of the pathologies (cardiomegaly).

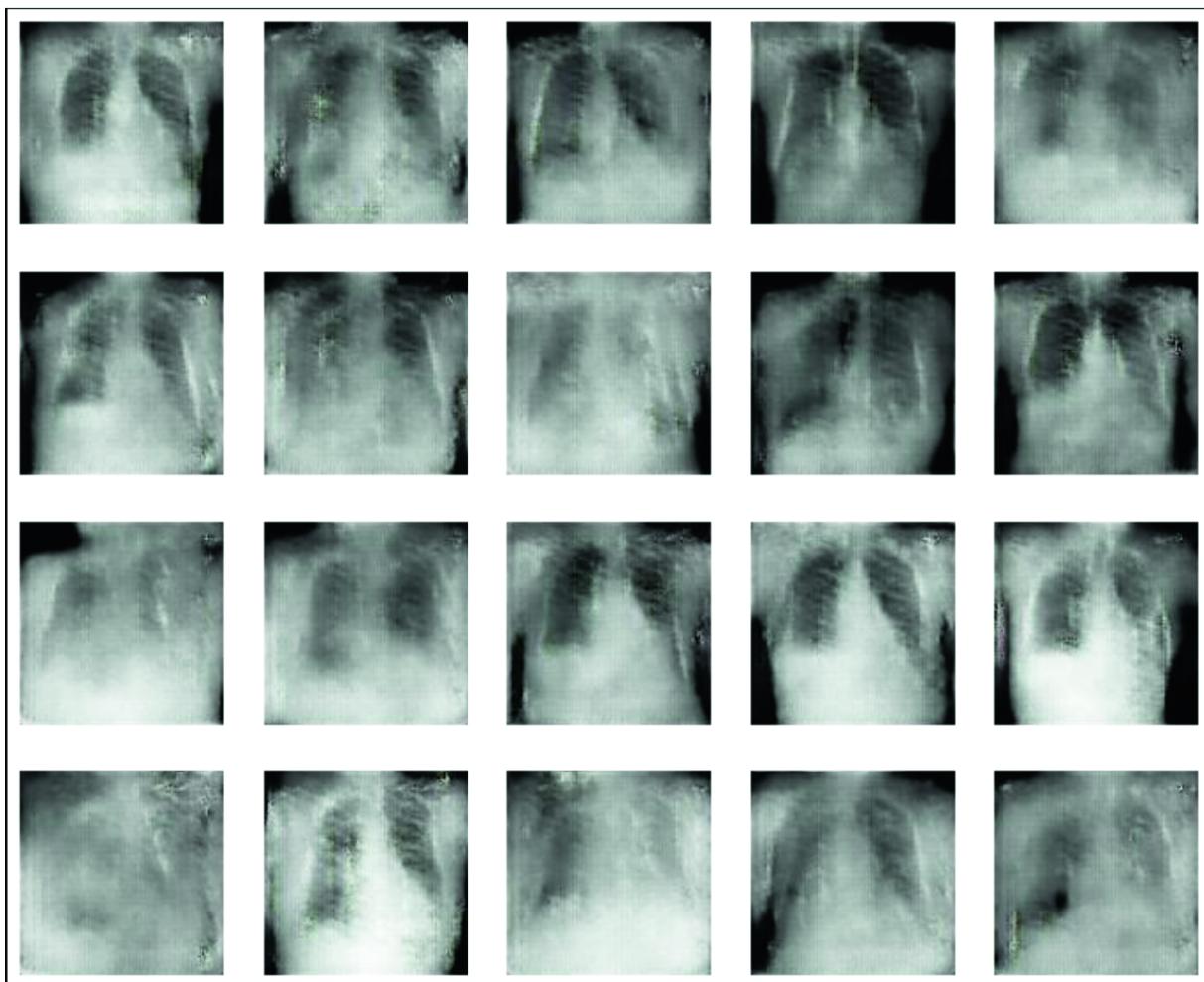


Fig. 14.3 The output of DCGAN when trained with chest X-ray images (cardiomegaly)

14.3.2.4 Limitations of DCGAN for X-Ray Image Augmentation

DCGAN works fine for low-resolution images, but as the resolution increases, the network's output starts to degrade. This is because the number of convolutional and deconvolutional layers increases with high-resolution images. The training of DCGAN models remains unstable, even in the presence of a suite of empirical techniques designed to improve the stability of the models [4]. We propose using progressively growing GAN (PGGAN), which uses layer by layer model building for stable training for the augmentation of X-ray images.

14.4 Progressively Growing GAN (PGGAN)

Progressive Growing GAN (PGGAN) is a CNN-based GAN with a modified training process that allows stable and smooth training of the generator network for producing good quality high-resolution synthetic images (as high as 1024^2 resolution). PGGAN first trains only one block of layers in the generator and discriminator. A new layer is then added to both networks to double the size of the image. The training process continues, and the network size increases linearly to the desired resolution. This layered training stabilizes the training process and helps in generating good quality images.

The key feature of PGGAN is the incremental increase in the size of output images produced by the generator, starting with a 4×4 pixel image and doubling to 8×8 , and 16×16 , and so on until the desired resolution image is created. This incremental nature allows the training process to handle the large-scale structure of an image first and incrementally add finer-scale details instead of learning all scales simultaneously. This is achieved by a training procedure that involves fine-tuning the model layer by layer and slowly phasing in a new block of layers with higher resolution. All layers remain trainable throughout the training process, including existing layers when new layers are added.

14.4.1 Training of PGGAN

PGGAN involves growing generator and discriminator models with the same general structure and starting with very small images, such as 4×4 pixels. The number of layers of a PGGAN model is progressively increased during the training process. A block of convolutional layers is added to the generator and the discriminator during each step of the training process. The incremental addition of the layers allows the models to learn coarse details effectively and then fine details, both on the generator and discriminator sides. The training process of PGGAN is shown in Fig. 14.4.

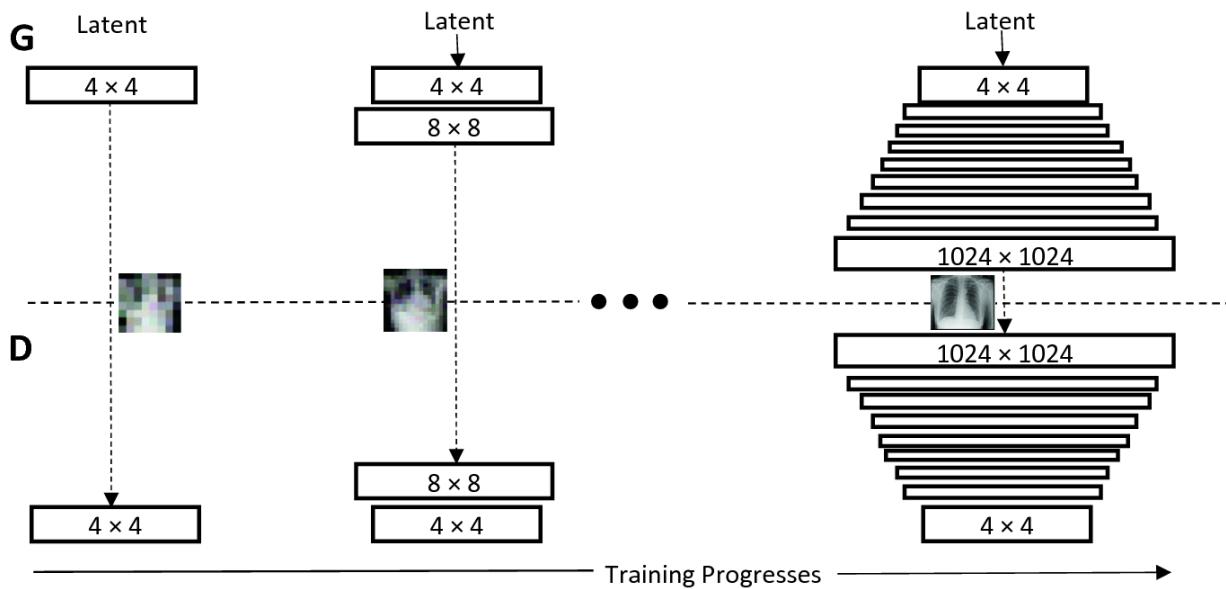


Fig. 14.4 Training process of PGGAN

The generator up-sampling is performed by using nearest-neighbor interpolation. Up-sampling can also be achieved by using a transpose convolutional layer. The discriminator down-sampling is performed by using average pooling. Down-sampling can also be performed by using a convolutional layer with stride 2.

14.4.2 Phasing in a New Layer for Smooth Model Building

PGGAN training process is much like the greedy layer-wise training process common for developing deep learning neural networks before ReLu and batch normalization were introduced.

The newly added layer is phased in smoothly to prevent the well-trained low-resolution layer from getting suddenly changed by adding a new layer

with random weights (untrained layer). The phasing in of a new block of layers involves using a skip connection to connect the new block to the input of the discriminator or output of the generator and adding it to the existing input or output layer with a weighting. The weighting controls the influence of the new block. It is achieved using a parameter alpha (α) that starts at zero or a small number and linearly increases to 1.0 over training iterations. Initially, all the weight or preference is given to the skip-connection, and the newly added layer is ignored. As the training progresses, the weight from skip-connection to the newly added layer is smoothly shifted by increasing alpha value (α). When it reaches the end of the training, the alpha value (α) becomes 1—implying that the skip connection we had introduced becomes dead, and we are left only with the newly added layer. This is demonstrated in Fig. 14.5. The figure shows the transition from a 16×16 image size to 32×32 image size, in three steps from (a) to (c) via (b).

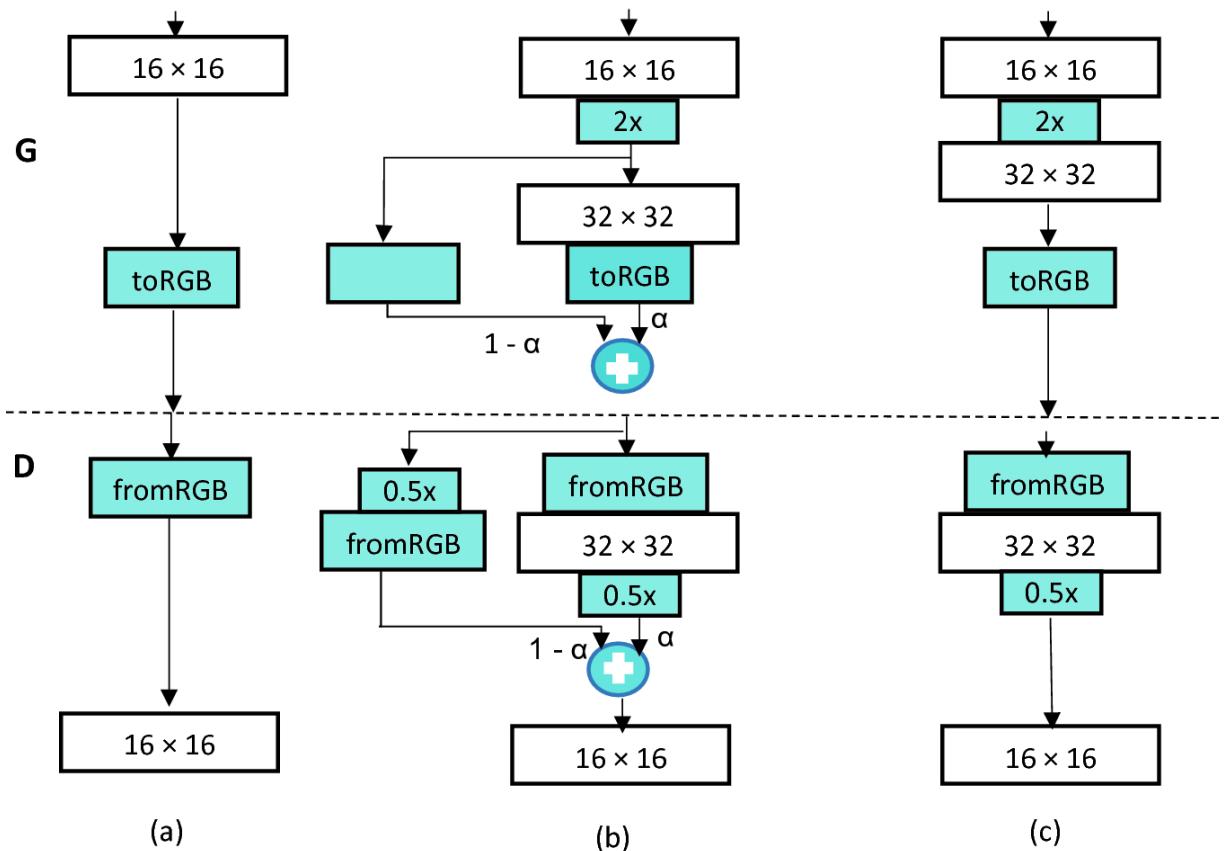


Fig. 14.5 Phasing in a new layer for smooth model building

14.4.3 Transfer Learning for Model Building

Here transfer learning has been employed with the state-of-art classification models like VGG16, ResNet50, ResNet101, and Google InceptionV3. Augmented data is tested by using fine-tuned classification models. The following procedure has been adopted (Fig. 14.6):

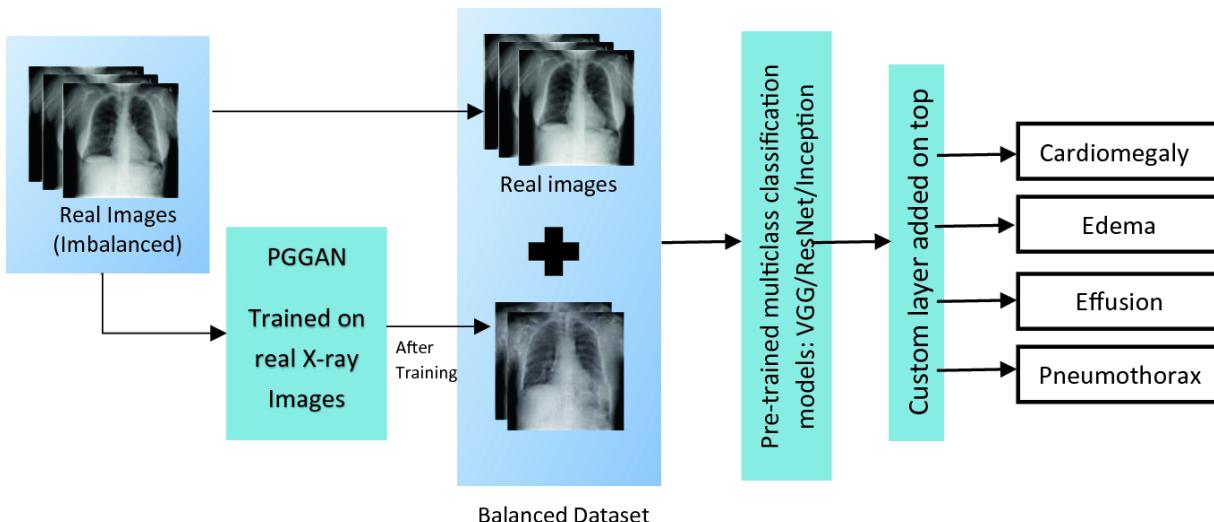


Fig. 14.6 Testing augmented data

- (i) Layers from a previously trained model are adopted
- (ii) Trainable features are set to ‘False’ so that adopted model parameters are frozen.
- (iii) A new trainable layer is added on top of the frozen layers. The new layer is trained for classification tasks while keeping adopted model parameters frozen.
- (iv) The parameters of all layers are fine tuned.
- (v) Finally, the whole model is tested on the new dataset.

14.5 Results and Discussion

This section discusses the dataset used for data augmentation and the results of testing the augmented dataset on classification models.

14.5.1 Dataset

The ‘NIH Chest X-ray dataset’ has been used here, which is publicly accessible (<https://www.kaggle.com/nih-chest-xrays/data>). The dataset comprises 112,120 frontal-view X-ray images of 30,805 unique patients with 14 different common thoracic pathologies (viz. 1, Atelectasis; 2, Cardiomegaly; 3, Effusion; 4, Infiltration; 5, Mass; 6, Nodule; 7, Pneumonia; 8, Pneumothorax; 9, Consolidation; 10, Edema; 11, Emphysema; 12, Fibrosis; 13, Pleural Thickening; 14 Hernia) where each image is high resolution (1024*1024) size image. This dataset has fourteen disease labels (where each X-ray image can have multi-labels). A subset of this data that represents only four thoracic pathologies, considered by authors in [8], has been used in this work.

The four different classes of X-Rays images used here are: {Cardiomegaly, Pneumothorax, Effusion, and Edema}. The total number of X-Ray images in each category are: Cardiomegaly—13,123, Pneumothorax —7890, Effusion—5012, and Edema are 3013. One thousand images from each class have been taken for testing purposes. A balanced dataset with 2000 original images from each class has been used to train each class's DCGAN and PGGAN models separately. The trained models have been used then to augment the data of various classes. The images produced by DCGAN were combined with original images of respective classes for classification testing. Similarly, the images produced by PGGAN were combined with original images of individual classes for classification testing. The balanced dataset with 2000 original images from each class has also been used to fine-tune the selected classification models.

14.5.2 Augmentation Results

The PCGAN and PGGAN models have been trained with 8,000 X-ray images of the original dataset (2,000 images of each class) representing four different chest pathologies. The trained models were then used to create synthetic X-ray images to augment the original dataset. Figure 14.7 shows X-ray images generated by the PGGAN model during various stages of training.

The output of our model at various stages during the training					
8x8			64x64		
16x16			128x128		
32x32			256x256		
Original Images (256x256)					

Fig. 14.7 Image Generation by PGGAN

Transfer learning has been used to adopt standard classification models, which were fine-tuned to suit this application. Fine-tuning has been done by re-training the classification models with the original 8,000 X-ray images (2,000 images of each class).

The images produced by DCGAN were combined with original images of respective classes, making 30,000 images per class (a total of 120,000 images of all four classes). Then pre-trained classification models were used to classify the different pathologies. The results produced are presented in Table 14.1. Similarly, images created by PGGAN were combined with original images of respective classes, making 14,000 images per class (a total of 56,000 images of all four classes). Figure 14.8 shows a balanced dataset of original and synthetic images. The pre-trained classification models were used to classify the different pathologies. The results produced are presented in Table 14.1. The results of DCGAN and PGGAN can be compared in Table 14.1. It can be concluded that

augmentation with good quality data has an impact on the classification model. The experimental results show that the use of PGGAN for data augmentation and the use of transfer learning in classification models increases the accuracy of predicting chest pathologies. Better accuracy results have been obtained even with half of the augmented images used with PGGAN.

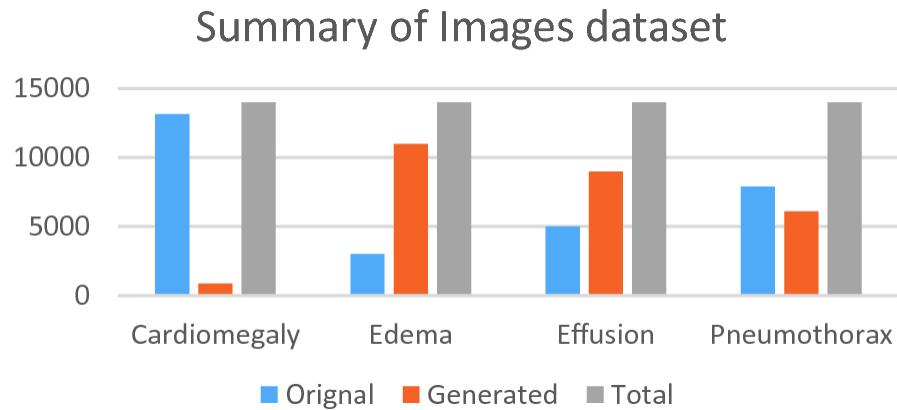


Fig. 14.8 PGGAN augmentation

Table 14.1 Accuracies recorded during experiments

	Original balanced dataset (8000 images)		Augmented + Balanced Dataset DCGAN (120,000 images)	Augmented + Balanced Dataset, PGGAN (56,000 images)
Model	Validation accuracy (%)	Testing accuracy (%)	Accuracy (%)	Accuracy (%)
InceptionV3	91.44	69.0	92.18	95.26
VGG16	80.00	66.8	93.82	95.85
Atten_VGG16	76.95	62.8	92.10	95.96

14.6 Conclusion

This work discussed the application of Generative Adversarial Networks (GANs) in X-ray medical imaging. The focus was on using Deep Convolutional GANs (DCGANs) and Progressively Growing GANs (PGGANs) for data augmentation involving X-ray images. It was observed that the DCGAN architecture produced good X-ray images having

characteristics similar to that of the training distribution. But the quality of high-resolution images generated was not good. The PGGAN-based architecture, proposed here for X-ray image generation, could generate good quality high-resolution images but required relatively more data for training. The model was tested using X-ray images, where the original dataset had images in the range of 3013–13,123 for each class. The augmentation was carried out using the proposed PGGAN-based architecture. The dataset was balanced, where each class had 14,000 images. Transfer learning was used to fine-tune pre-trained classification models with the balanced dataset to classify the different pathologies. The results showed that using the proposed PGGAN-based model for data augmentation improved the accuracy of classifying pathologies using chest X-rays.

References

1. Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., Rueckert, D.: GAN augmentation: Augmenting training data using generative adversarial networks (2018)
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2014)
[[Crossref](#)]
3. Iqball, T., Wani, M.A.: X-Ray images dataset augmentation with progressively growing generative adversarial network. IEEE EXplore 15–20 (2021)
4. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. ArXiv, 1–26 (2017)
5. Korkinof, D., Rijken, T., O'Neill, M., Yearsley, J., Harvey, H., Glocker, B.: High-resolution mammogram synthesis using progressive generative adversarial networks. ArXiv, 2017 (2018)
6. Onishi, Y., Teramoto, A., Tsujimoto, M., Tsukamoto, T., Saito, K., Toyama, H., Imaizumi, K., Fujita, H.: Automated pulmonary nodule classification in computed tomography images using a deep convolutional neural network trained by generative adversarial networks. BioMed Res. Int. (2019)

7. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: 4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings, pp. 1–16 (2016)
8. Salehinejad, H., Valaei, S., Dowdell, T., Colak, E., Barfett, J.: Generalization of deep neural networks for chest pathology classification in x-rays using generative adversarial networks. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, Apr 2018, pp. 990–994 (2018)
9. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Introduction to deep learning. In: Advances in Deep Learning, vol. 57, pp. 1–12. Springer Nature (2020a)
10. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Basics of supervised deep learning. In: Advances in Deep Learning, vol. 57, pp. 13–30, Springer Nature (2020b)
11. Apostolopoulos, I.D., Mpesiana, T.A.: Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. *Phys. Eng. Sci. Med.* **43**(2), 635–640 (2020)
[\[Crossref\]](#)
12. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Training supervised deep learning networks. In: Advances in Deep Learning, pp. 31–52. Springer Nature (2020c)
13. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Supervised deep learning architectures. In: Advances in Deep Learning, pp. 53–76, Springer Nature (2020d)
14. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Unsupervised deep learning architectures. In: Advances in Deep Learning, pp. 77–94, Springer Nature (2020e)
15. Chouhan, V., Singh, S. K., Khamparia, A., Gupta, D., Tiwari, P., Moreira, C., Damaševičius, R., de Albuquerque, V.H.C.: A novel transfer learning based approach for pneumonia detection in chest X-ray images. In: Applied Sciences (Switzerland), vol. 10, Issue 2 (2020)
16. Han, S.S., Kim, M.S., Lim, W., Park, G.H., Park, I., Chang, S.E.: Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *J. Investigig. Dermatol.* **138**(7), 1529–1538 (2018)
[\[Crossref\]](#)
17. Madani, A., Moradi, M., Karargyris, A., Syeda-Mahmood, T.: Semi-supervised learning with generative adversarial networks for chest X-ray classification with

- ability of data domain adaptation. In: Proceedings—International Symposium on Biomedical Imaging, Apr.(Isbi) 2018, pp. 1038–1042 (2018)
18. Mahapatra, D., Bozorgtabar, B., Thiran, J. P., Reyes, M.: Efficient active learning for image classification and segmentation using a sample selection and conditional generative adversarial network. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11071, pp. 580–588. LNCS (2018)
 19. Schmidhuber, J.: Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization. *Neural Netw.* **127**, 58–66 (2020)
[\[Crossref\]](#)
 20. Schwendicke, F., Elhennawy, K., Paris, S., Friebertshäuser, P., Krois, J.: Deep learning for caries lesion detection in near-infrared light transillumination images: a pilot study. *J. Dent.* **92** (2020)
 21. Shams, M.Y., Elzeki, O.M., Abd Elfattah, M., Medhat, T., Hassanien, A.E.: Why are generative adversarial networks vital for deep neural networks? A Case Study on COVID-19 Chest X-Ray Images. Springer International Publishing (2020)
 22. Talo, M., Baloglu, U.B., Yıldırım, Ö., Rajendra Acharya, U.: Application of deep transfer learning for automated brain abnormality classification using MR images. *Cogn. Syst. Res.* **54**, 176–188 (2019)
[\[Crossref\]](#)
 23. Tang, Y. X., Tang, Y. B., Han, M., Xiao, J., Summers, R. M.: Abnormal chest x-ray identification with generative adversarial one-class classifier. In: Proceedings —International Symposium on Biomedical Imaging, Apr. 2019, 1358–1361 (2019)
 24. Wani M.A., Bhat F.A., Afzal S., Khan A.I.: Supervised deep learning in face recognition. In: Advances in Deep Learning, pp. 95–110. Springer Nature (2020f)
 25. Wani M.A., Bhat F.A., Afzal S., Khan A.I. (2020g) Supervised Deep Learning in Fingerprint Recognition. In: Advances in Deep Learning, pp. 111–132, Springer Nature.