



Faculty of Electrical Engineering and Information Technology
Professorship of Digital Signal Processing and Circuit Technology

Master Thesis

Using Neural Networks for the Domain Adaptation of Synthetic Generated Document Images

Pawar, Giriraj Sukumar

Submitted in Fulfilment of the Requirements for the
Academic Degree M.Sc.

Chemnitz, September 3, 2021

Supervisor: Prof. Dr.-Ing. Gangolf Hirtz

Advisors: Dr.-Ing. Ana Cecilia Perez Grassi, M.Sc. Tobias Scheck

External Advisors: Dr.-Ing. Martin Voigt, M.Sc. Paul Fischer

Pawar, Giriraj Sukumar

Matriculation Number: 551205

Using Neural Networks for the Domain Adaptation of Synthetic Generated Document Images

Master Thesis, Faculty of Electrical Engineering and Information Technology,

Professorship of Digital Signal Processing and Circuit Technology,

Technische Universität Chemnitz, September 2021.

Abstract

Neural networks have improved significantly in past decades. They are competent to solve complex problems in the field of deep learning, and they are capable to manage a large amount of complex data like images, videos, and sound. However, the training of neural networks requires a significantly large amount of annotated data, which is not always possible. Machine learning engineers inevitably have to generate synthetic data. Although, the neural networks trained on synthetic data will not able to perform or generalize well on real data. In recent years, an effective technique named domain adaptation has evolved, to address the problem of scarcity of annotated data. The domain adaptation technique can transform data from the source domain to the target domain. For example, domain adaptation techniques like image-to-image translation can be used to transform images of zebras into images of horses and vice-versa. This thesis proposes an image-to-image translation application that aims to close the domain gap between synthetic data distribution and real data distribution using Cycle-Consistent Adversarial Networks (CycleGANs). The proposed application is used to transform synthetic document images into realistic document images, to overcome the scarcity of annotated real document images. In addition, these generated real document images are used to train a classifier to classify similar unlabeled real document images, thereby accelerating the process of labeling images in an unsupervised and automated manner. Experimental results show the generated realistic document images are qualitatively convincing and need improvement quantitatively. Such preliminary results show that CycleGAN can solve the problem of data scarcity by generating high-quality images in the target domain. The purpose of this thesis is limited to improving the classification of real document images. Once the rich data is generated in the target domain, the performance of the real document image classifier can eventually be improved. The work of this thesis is limited to the study of CycleGAN as an image-to-image conversion method without paired training data. The remaining methods and comparisons with them are left for future work. In the future, CycleGAN can be used to generate realistic images in many tasks, such as handwriting recognition, image classification, image segmentation, and object detection.

Keywords: CycleGAN, Generative Adversarial Network (GAN), Domain Gap, Domain Adaptation, Image-to-Image Translation, Data Distributions.

Acknowledgments

I would like to express my gratitude to my advisors, Dr.-Ing. Gangolf Hirtz, Dr.-Ing. Ana Cecilia Perez Grassi, Dr.-Ing. Martin Voigt, Tobias Scheck, Paul Fischer, Thaddäus Strobel, and Clemens Reinhardt, who helped me throughout this project. Also, I would like to extend a special thanks to my parents and my friends for their enormous support. I would like to dedicate this thesis to my teacher Subhash K. U. for teaching me how to program, may his soul rest in peace and god give strength to his family.

Contents

List of Figures	3
List of Tables	5
List of Algorithms	6
List of Abbreviations	7
List of Symbols	8
1 Introduction	9
1.1 Overview	9
1.2 Motivation	10
1.3 Problem Statement	13
1.4 Thesis Objectives	15
1.5 Thesis Limitations and Structure	15
1.6 Terminology	15
2 Related Works	16
2.1 Literature Survey	16
2.2 Discussion	21
2.3 Conclusion	21
3 Fundamentals	22
3.1 Generative Adversarial Networks (GANs)	22
3.1.1 GAN Training	24
3.2 Convolution Neural Networks	26
3.2.1 Convolution Layer	27
3.2.2 Pooling Layer	31
3.2.3 Fully connected layer	32
4 Methodology	34
4.1 Proposed Approach	34
4.2 Cycle-Consistent Adversarial Networks	34
4.2.1 Formulation	34
4.2.2 Least-Square Loss	36
4.2.3 Cycle-Consistency Loss	36
4.2.4 Identity Mapping Loss	37
4.2.5 Complete Objective Function	37
4.3 CycleGAN Training Algorithm	38
5 Implementation	39
5.1 Dataset Preparation	39
5.2 Network Architecture	41
5.2.1 CycleGAN	41
5.2.2 Classifier	44
5.3 Training Details	45
5.3.1 CycleGAN	45
5.3.2 Classifier	46

6 Experiments and Evaluation	48
6.1 Evaluation Metrics	48
6.2 Experiments	49
6.2.1 Experiment Steps	50
6.2.2 CycleGAN Training	50
6.2.3 Training a Classifier on Synthetic Document Images	51
6.2.4 Training a Classifier on CycleGAN Generated Document Images	52
6.2.5 Training a Classifier on Faxified Document Images	54
6.3 Results	56
6.3.1 Quantitative Results	56
6.3.2 Qualitative Results	58
6.3.3 Failure Cases	59
7 Conclusion and Future Work	61
7.1 Overview	61
7.2 Conclusion	61
7.3 Future Work	62
A Appendix	64
A.1 Modified National Institute of Standards and Technology database (MNIST) Handwritten Numbers Dataset	64
A.2 CycleGAN Models Training	64
A.3 GAN Training	65
A.4 Confusion Matrices	66
A.5 Examples of Document Images	69
A.6 Classifier Architecture Diagram	72
A.7 Generator Model Summary	73
A.8 Discriminator Model Summary	79

List of Figures

1.1	Relationship between Artificial Intelligence, Machine Learning, and Deep Learning.	9
1.2	Simple examples of transfer learning.	10
1.3	Simple example of domain adaptation from Street View House Numbers (SVHN) dataset to MNIST Dataset for the digit recognition task.	11
1.4	The paired training data set consists of training examples, in which there is a correspondence between the source domain and the target domain. But in the unpaired training data set, there is no correspondence between the source domain and the target domain.	12
1.5	Difference between traditional machine learning and domain adaptation.	12
1.6	An illustration of the problem to be solved in this thesis.	13
1.7	Illustration of the solution proposed to reduce the domain gap between the synthetic document images and the real document images.	14
2.1	Evolution of GANs Over the Years.	17
2.2	Illustration of training a Conditional Adversarial Network (cGAN) to map edges →photo transformation.	18
2.3	Illustration of CycleGAN transforming the noisy document images into clean document images. and vice versa.	19
2.4	Illustration of CycleGAN transforming an image from one into the other and vice versa.	20
3.1	Intuitive example of GAN training progress.	22
3.2	Overview of core GAN architecture.	23
3.3	Illustration of GANs converging to match generated data distribution p_g to real data distribution p_{data}	24
3.4	Illustration of the training of the discriminator D using backpropagation.	25
3.5	Illustration of the training of the generator G using backpropagation.	26
3.6	Overview of Convolutional Neural Network (CNN) architecture and its training process.	27
3.7	A Convolution Operation With Zero Padding.	28
3.8	An illustration of Convolution Operation.	29
3.9	Simple Neural Network.	30
3.10	Illustration of Artificial Neuron.	31
3.11	Most common nonlinear activation functions used while constructing Neural Networks.	31
3.12	Illustration of Max Pooling Operation.	32
4.1	Overview of the image-to-image translation application using CycleGAN - It consists of two generators, G and F which map synthetic document images to real document images and real document images to synthetic document images, respectively using cycle-consistency loss [1]. It also contains two discriminators D_X and D_Y which acts as the adversary and reject images generated by generators.	35
4.2	Illustration of CycleGAN model with its mapping functions, respective discriminators, and forward and backward consistency loss.	37
5.1	Examples of handwritting crops from the handwritting number dataset.	39
5.2	Inserting handwritten crops on empty form templates.	40
5.3	Templates with minor differences.	40
5.4	Templates with major differences.	41
5.5	Illustration of ResNet blocks in CycleGAN generator architecture.	42
5.6	Steps involved in preprocessing of training images of CycleGAN.	46
5.7	Steps involved in preprocessing of training images of Classifiers.	47
6.1	CycleGAN generator G training epochs vs loss plot.	50
6.2	CycleGAN discriminator D_Y training epochs vs loss plot.	50
6.3	CycleGAN generator F training epochs vs loss plot.	51
6.4	CycleGAN discriminator D_X training epochs vs loss plot.	51

6.5	Epochs vs accuracy plot while training a classifier on synthetic document images.	52
6.6	Epochs vs loss plot while training a classifier on synthetic document images.	52
6.7	Epochs vs accuracy plot while training a classifier on CycleGAN generated document images.	53
6.8	Epochs vs loss plot while training a classifier on CycleGAN generated document images.	53
6.9	Illustration of faxification process applied on synthetic document images.	54
6.10	Illustration of faxified document images to conclude that faxification process is a random process, the input images are faxified randomly to create distinct output.	54
6.11	Epoch vs Accuracy Plot.	55
6.12	Epoch vs Loss Plot.	55
6.13	Comparison of accuracies and f1-scores, when the classifiers trained on different data distributions and evaluated on annotated real document images.	57
6.14	Synthetic document images transformed into realistic document images by the image-to-image translation application.	58
6.15	The snippet from CycleGAN generated document image illustrates that the proposed image-to-image translation application did not transform or reconstruct handwritten crops in the target domain from the synthetic document image in the source domain.	59
6.16	The CycleGAN generated document image consists of noisy artifacts that are unrealistic.	59
6.17	The CycleGAN generated document image consists of dark border.	60
6.18	There are some artifacts appear in the generated image that are not present in the synthetic image.	60
A.1	Examples of Handwritten Numbers from the MNIST Dataset.	64
A.2	CycleGAN generators G and F training epochs vs loss plot.	64
A.3	CycleGAN discriminators D_X and D_Y training epochs vs loss plot.	64
A.4	Illustration of training of the GAN as per the algorithm.	65
A.5	Confusion matrix to analyze the performance of the classifier trained on synthetic document images and evaluated using real annotated document images (test dataset).	66
A.6	Confusion matrix to analyze the performance of the classifier trained on CycleGAN generated document images and evaluated using real annotated document images (test dataset).	67
A.7	Confusion matrix to analyze the performance of the classifier trained on faxified document images and evaluated using real annotated document images (test dataset).	68
A.8	Example of template.	69
A.9	Example of real document image.	70
A.10	Examples of faxified document image.	71
A.11	Classifier Model Summary.	72
A.12	Generator Model Summary. Continue to Next Page.	73
A.13	Generator Model Summary. Continue to Next Page.	74
A.14	Generator Model Summary. Continue to Next Page.	75
A.15	Generator Model Summary. Continue to Next Page.	76
A.16	Generator Model Summary. Continue to Next Page.	77
A.17	Generator Model Summary. Ends Here.	78
A.18	Discriminator Model Summary.	79

List of Tables

5.1	Size of datasets used for training CycleGAN and classifiers.	41
5.2	Number of images in each class of annotated real document images dataset.	41
5.3	Generator architecture	43
5.4	Discriminator architecture	44
5.5	Classifier architecture	45
6.1	Classification report generated after the classifier is trained on synthetic document images, its classification performance evaluated on the annotated real document images.	52
6.2	Classification report generated after the classifier is trained on CycleGAN generated document images, its classification performance evaluated on the annotated real document images.	53
6.3	Classification report generated after the classifier is trained on faxified document images, its classification performance evaluated on the annotated real document images.	55
6.4	The accuracies and f1-scores when the classifiers trained on different data distributions and evaluated on annotated real document images.	56

List of Algorithms

1	GAN Training Algorithm [2].	26
2	CycleGAN training algorithm.	38

List of Abbreviations

GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
cGAN	Conditional Adversarial Network
LSGAN	Least Squares Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
EBGAN	Energy-based Generative Adversarial Network
CycleGAN	Cycle-Consistent Adversarial Network
ACL-GAN	Adversarial Consistency Loss Generative Adversarial Network
DBN	Deep Belief Network
Stacked-CAE	Stacked Convolutional Autoencoder
Deep-GSN	Deep Generative Stochastic Networks
GT	Ground Truth
DIBCO	Document Image Binarization Competition
PSNR	Peak Signal-to-Noise Ratio
ResNet	Residual Network
AMT	Amazon Mechanical Turk
CUT	Contrastive Unpaired Translation
MUNIT	Multimodal Unsupervised Image-to-image Translation
DRIT	Diverse Image-to-Image Translation
VAE	Variational Autoencoder
GCGAN	Geometry-Consistent Generative Adversarial Networks
FastCUT	Fast Contrastive Unpaired Translation
FID	Fréchet Inception Distance
HTR	Handwritten Text Recognition
OCR	Optical Character Recognition
WGAN	Wasserstein GAN
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
AI	Artificial Intelligence
MNIST	Modified National Institute of Standards and Technology database
SVHN	Street View House Numbers
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
COCO	Common Objects in Context
1D	One-dimensional
2D	Two-dimensional

List of Symbols

G	Generator
F	Generator
D	Discriminator
z	Noise distribution
Z	Noise distribution
X	Source domain
Y	Target domain
x_i	Samples from source domain
y_i	Samples from target domain
x	Real samples
p_z	Data distribution over noise input z
p_g	Generator's data distribution over data x
p_{data}	Data distribution over real samples x
\log	Logarithm
\min	Minimize
\max	Maximize

1. Introduction

1.1 Overview

Artificial Intelligence (AI) has been a game-changer in the computer science domain and has evolved tremendously over the years [3]. AI has a presence in many sectors like Healthcare [4], Autonomous Vehicles [5], Robotics [6], Space Exploration [7], and Computer Vision [8]. This is largely due to the research in Machine Learning (ML) and Deep Learning (DL). Machine Learning is a subdomain of Artificial Intelligence. Machine learning is an art of programming machines, so they can learn from data without being explicitly programmed. Machine learning is used to create many AI applications, where it is difficult or unfeasible to develop traditional algorithms to perform the needed tasks. Although machine learning and deep learning domains fall under the category of Artificial Intelligence, there are some important differences between them (figure 1.1). First, deep learning is a subdomain of machine learning. Second, deep learning algorithms are powered by Artificial Neural Networks (ANNs), and third, they require less human intervention while extracting features from the data compared to machine learning.

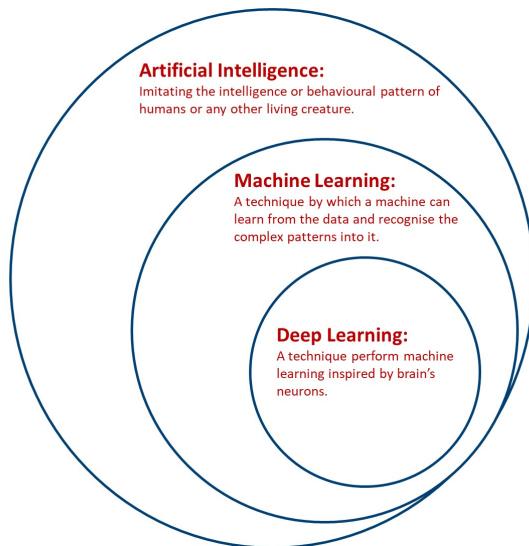


Figure 1.1: Relationship between Artificial Intelligence, Machine Learning, and Deep Learning.

The concept of deep learning was invented as early as the 1950s, although it was largely ignored until the 1980s and 1990s. However, with the efficient, fast computation hardware, and abundant data since the decade, it has become a popular research topic among many AI research institutions, organizations, and startups. Deep learning is inspired by the biological network of neurons present inside the brain. Deep learning algorithms learn to discover meaningful, complex patterns in the digital representation of data, like sounds and images. To achieve this, deep learning uses a multi-layered structure of algorithms called Artificial Neural Networks (ANNs). ANNs are the heart of deep learning. They are flexible, efficient, and scalable, and suitable for vast and highly complex deep learning tasks like classifying billions of images (e.g., Google Images, Instagram, and Facebook), object detection (e.g., Tesla's Self-driving Cars), improving speech recognition systems (e.g., Apple's Siri, Amazon's Alexa, and Google Assistant), defense systems (Israel's Iron Dome, U.S.A's Patriot Missile System), and recommending the best videos to watch to hundreds of millions of users every day (e.g., YouTube).

Neural Networks are capable enough to solve complex problems by extracting features, recognizing patterns in data efficiently. However, there are certain challenges while training the neural

networks. The two main things that can cause difficulties are bad algorithms and bad data. Let us start with some examples of bad data. Insufficient training data is one of the main problems that may arise when training deep learning models. A large amount of training data is required for most deep learning models to work properly. The second is the poor quality data. If the training data is wrong, noisy, and full of outliers, it will make it difficult for the model to detect patterns in the data. Hence the model will not perform well. The third is the irrelevant features. The model will only learn if the training data has more relevant features than irrelevant features.

Now, that after some examples of bad data, let's look at some of the examples of bad algorithms. Overfitting and underfitting are one of the main problems of deep learning. “Overfitting happens when the model is too complex relative to the amount and noisiness of the training data” [9]. In the overfitting, model learns training data including noise to the extent, it negatively impacts the performance of the model, leading to higher generalization error on unseen data. One of methods used to decrease the risk of overfitting is regularization [10]. Regularization is one of the solutions provided to generalize a model better to the new examples. “Underfitting is the opposite of overfitting, it occurs when a model is too simple to learn the underlying structure of the data” [9]. In the case of the underfitting model, it neither learns training data nor generalize to the unseen data. “By choosing a powerful model with more parameters and providing better features during the model learning process” [9] can resolve the problem of underfitting. The training of the deep learning model is highly influenced by the quality and quantity of the data that has been used for the training. But in many cases, data is scarce and it is very difficult to have data that is labeled and annotated.

1.2 Motivation

In recent years, deep learning methods have shown great effectiveness in many fields, including natural language processing and computer vision. However, such kinds of methods are still limited by a poor generalization due to the insufficient quantity of training data [11]. Annotated data are scarce when it comes to developing deep learning models for computer vision applications. The performance of such deep learning models can be improved with the introduction of a large amount of annotated data. However, due to the high cost of data annotation, it is difficult to obtain a large set of annotated training data, especially when there are many data variants. To overcome the obstacle of the scarcity of annotated data, there are some methods available to tackle this problem. The popular methods are Active Learning [12], Data Augmentation [13], Transfer Learning [14], and Domain Adaptation [15].

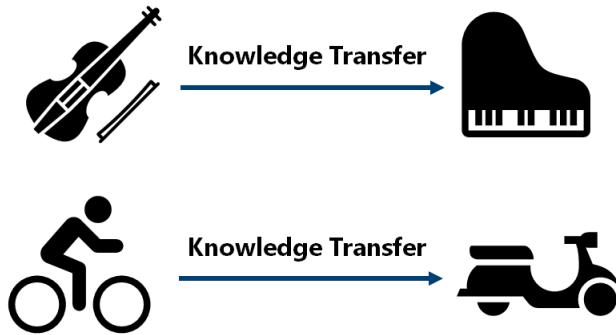


Figure 1.2: Simple examples of transfer learning.

This thesis aims to solve the data scarcity problem using the domain adaptation method. Domain adaptation is a subcategory of transfer learning in which the task is to transfer the knowledge from the source domain to the target domain. Before discussing domain adaptation, let us first look at transfer learning. Transfer learning is the ability of a system to recognize and apply the knowledge learned from one task to another task [14]. For example, if a person has mastered one musical instrument like the violin, can learn piano faster compared to others, as the knowledge of one musical instrument can be applied while learning another musical instrument. Figure 1.2 shows an

intuitive example of transfer learning. Transfer learning inspired by human behavior, humans beings are capable to transfer knowledge from one domain to another domain. The transfer learning grasps knowledge from the source domain to improve the learning performance in the target domain so the number of labeled samples required in the target domain can be reduced. It is important to mention, transfer learning is effective only if the source domain and target domain are related. For example, learning bicycles will not help to learn piano faster. Qiang Yang et al.[16] have performed survey on transfer learning, more information about the transfer learning can found in their research paper.

When the source and target domains are the same and learning tasks also the same, then such a learning problem becomes a traditional machine learning problem [16]. When the source and target domains are different but related, and learning tasks are the same then, such a learning problem becomes a domain adaptation problem [16]. In domain adaptation, source and target domains have the same feature space but different distributions in contrast to transfer learning, which includes cases where the target domain's feature space is different from the source domain's feature space [16]. Domain adaptation is distinguished depending upon the similarity or dissimilarity of feature space and availability of annotated data in the source domain and target domains. The domain adaptation has two categories, if the feature space is the same between the source domain and target domain is called homogeneous domain adaptation. If the feature space is different between the source and target domain is called heterogeneous domain adaptation. Further homogeneous and heterogeneous domain adaptation divided into three types of domain adaptation, supervised domain adaptation, semi-supervised domain adaptation, unsupervised domain adaptation. In the supervised domain adaptation, the samples in source and target domains are labeled. Semi-supervised domain adaptation has a small set of samples labeled in the target domain. And, in unsupervised domain adaptation, the samples in the source and target domain are not labeled [16]. A simple example of domain adaptation of SVHN transformed into Handwritten Digits shown in figure 1.3. The difference between traditional machine learning and domain adaptation is shown in figure 1.5.

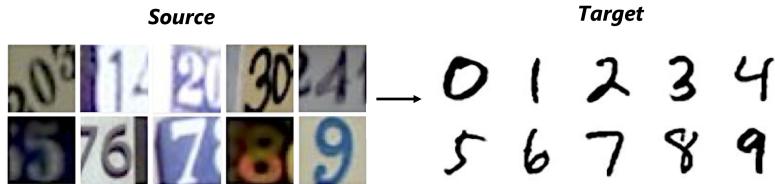


Figure 1.3: Simple example of domain adaptation from SVHN dataset [17] to MNIST Dataset¹ for the digit recognition task.²

To understand how domain adaptation works, let's have a simple example. Consider the source domain represented by the SVHN dataset. It is a collection of house number images, and the target domain represented by the MNIST dataset, which is a collection of handwritten digit images. When the CNN is trained and evaluated on the source domain SVHN dataset for the task of identifying numbers it will achieve good accuracy. However, the same classifier will perform worst when evaluated the MNIST dataset. This performance gap occurs due to differences between the domain data distribution. The images in the SVHN dataset consist of different fonts, blur, noise, and different backgrounds. But the images in the MNIST dataset contain a clean background and handwritten strokes. Now consider images are scarce in the target domain. Only a small amount of the target domain images are available, which are unlabeled. As we know training a classifier using a smaller amount of data leads to underfitting and eventually leads to the worst performance on unseen data. Hence, to create a sufficient amount of data, the domain adaptation model is trained. It learns to transfer the underlying knowledge from the source domain to the target domain. In this case, labeled data is available in the source domain, and unlabeled data available in the target domain. Such a setup is called unsupervised domain adaptation because the model learns to transform images from one domain to another in the absence of labeled data in the target domain, and without taking much help from the labeled data from the source domain during the learning process. Using this

¹<http://yann.lecun.com/exdb/mnist/> last access: September 3, 2021

²<https://machinelearning.apple.com/research/bridging-the-domain-gap-for-neural-models> last access: September 3, 2021

domain adaptation model, large amount of annotated data can be created in the target domain by transforming source domain images into target domain images. Once a sufficient amount of images present in the target domain, the task to identify numbers in the target domain can be improved significantly. Nowadays, the domain adaptation technique is widely used in the field of Handwritten Text Recognition (HTR) [18], Image Classification [16], Style Transfer [19], and Optical Character Recognition (OCR) [20] to solve the problem of scarcity of data. To make these applications robust and efficient comes under a very big scope. The scope of this thesis is limited to image classification tasks.

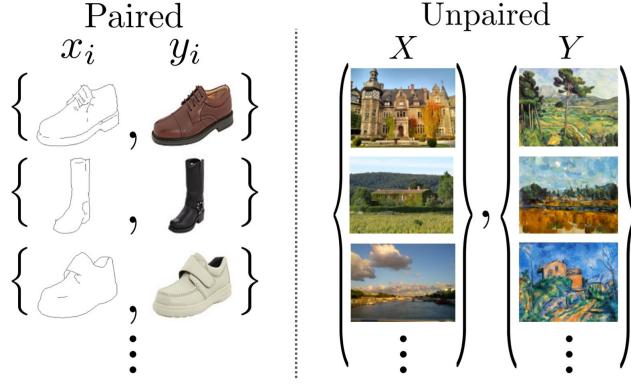


Figure 1.4: The paired training data set consists of training examples, in which there is a correspondence between the source domain and the target domain. But in the unpaired training data set, there is no correspondence between the source domain and the target domain [1].

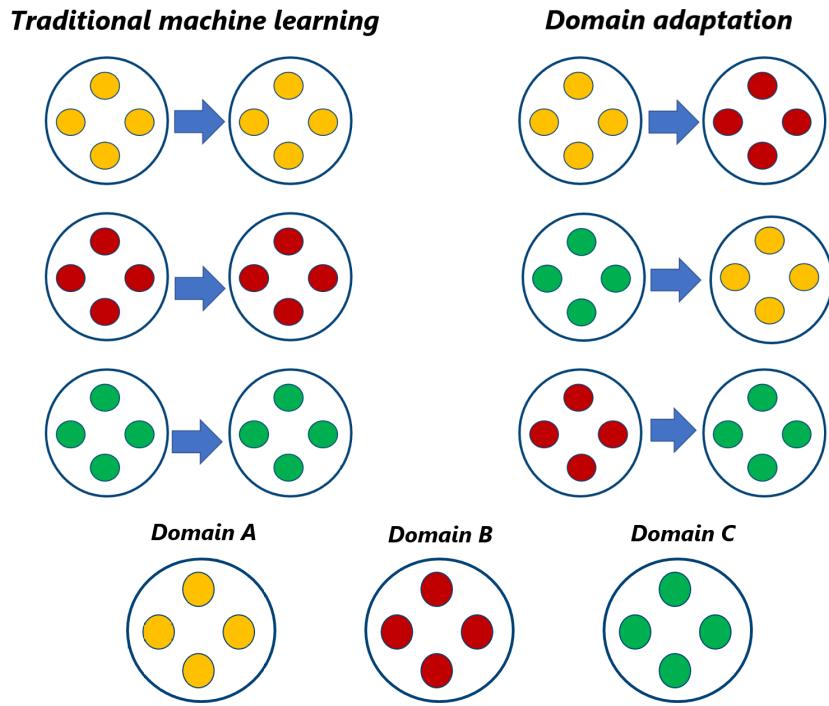


Figure 1.5: Difference between traditional machine learning and domain adaptation.

1.3 Problem Statement

As mentioned earlier, annotated data are scarce when it comes to the training of neural networks, and labeling a large set of data is a costly and tedious job. For example, real document images (figure A.9) with different types of handwriting. In such cases, machine learning engineers have to inevitably generate synthetic data. However, deep learning models trained using synthetic data will not generalize well on real data [11] (figure 1.6). The synthetic data lacks realism [11]. It does not possess a similar noise distribution as real data [11]. Hence, in the last two decades, numerous domain adaptation methodologies have been introduced [20]. Such methods are used to transform synthetic data into realistic data by reducing the divergence between the distribution of real data and the distribution of synthetic data. In this thesis, a domain adaptation application is developed using CycleGAN [1] to reduce the domain gap between synthetic data distribution and real data distribution. The CycleGAN is an extended variant of Generative Adversarial Network (GAN) [21].

This application is designed in consultation with, ML developers at Elevait Deutschland GmbH, a Germany-based company that develops AI applications for business use-cases. Elevait is widely contributing in the field of Cognitive Business Robotics [22] to automate document processing. Elevait has developed state-of-the-art HTR and OCR tools to process documents and extract information from documents. To make those systems robust and efficient a large number of document images are required. The idea is to create a large number of synthetic document images to have a large quantity of annotated data. However, the synthetic document images will not generalize well when they have to process real document images because the real document images have different noise distribution. Furthermore, artifacts such as salt-and-pepper, background noise, blur due to camera motion or shake, watermarks, stains, wrinkles, and fading text are often introduced during the scanning process. Hence this application is used to transform synthetic document images into realistic document images. This application is capable to capture the noise distribution of real documents and transform an image from the source domain to the target domain. Such a kind of transformation is called image-to-image translation.

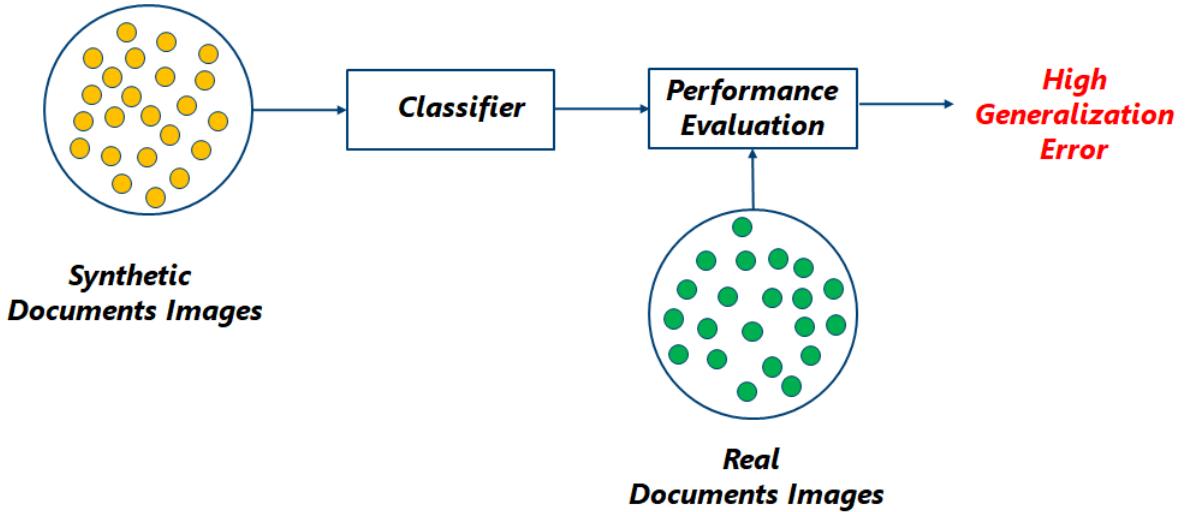


Figure 1.6: An illustration of the problem to be solved in this thesis.

The synthetic document images are created using templates (figure A.8) and handwritten crops (figure 5.1) retrieved from handwriting datasets like MNIST (figure A.1) or any other datasets. These templates contain fields like customer number, customer name, and other customer information. These fields are represented by using bounding boxes. The bounding boxes are annotated using Common Objects in Context (COCO) annotations [23]. In this thesis, the handwritten crops are inserted over the templates to generate numerous synthetic document images. As mentioned earlier deep learning models trained using synthetic document images will not generalize well on real document images. In this thesis, the handwritten crops are inserted over the templates to generate numerous synthetic document images. As mentioned earlier deep learning models trained

using synthetic document images will not generalize well on real document images. Further, the image-to-image translation application is developed using CycleGAN to transform synthetic document images into realistic document images. ultimately, to reduce the domain gap between synthetic data distribution and real data distribution.

A large number of realistic document images can be generated to tackle the problem of data scarcity in the target domain. As we already know that real document images are scarce and labeling images is a tedious and costly job. Our image-to-image translation application transforms synthetic document images to realistic document images to solve the following problems. First, We have labeled synthetic document images in the source domain, ultimately we get labeled, transformed realistic document images in the target domain. So the problem of labeling, annotations, and scarcity is solved. Second, Consider we have a huge chunk of real document images which are not labeled. If the simple classifier is trained on realistic document images. Later, these chunk of real document images can be automatically classified with ease. This will save data annotation and data collection efforts. Further, making image classification tools in the target domain robust and efficient even in the absence of real data.

The figure 1.6 describes if the classifier is trained using synthetic document images, it won't generalize well on the unseen real document images. Further, it leads to high generalization error, hence represented in dark red color for better understanding. In figure 1.7 Data distribution in light green color represents realistic data which is generated by the domain adaptation method. Dark green represents the real data. The proposed solution is to use domain adaptation methods like the image-to-image translation to reduce the divergence between data distributions. Further, the classifier is trained using realistic document images, it generalizes better on the unseen real document images compared to the classifier are trained using synthetic document images, hence low generalization error represented in light red color for better understanding compared to figure 1.6.

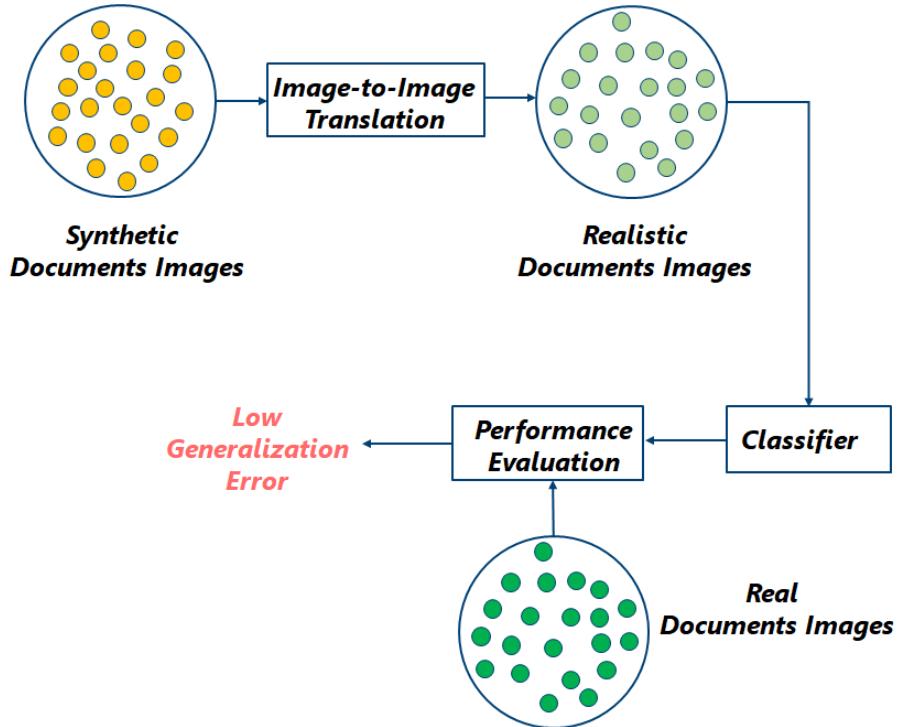


Figure 1.7: Illustration of the solution proposed to reduce the domain gap between the synthetic document images and the real document images.

1.4 Thesis Objectives

Every research work has its own objectives. The purpose of this thesis to close domain gap between synthetic data distribution and real data distribution by developing an image-to-image translation application using CycleGAN. The developed image-to-image translator is used to transform synthetic document images into realistic document images. In addition, in this thesis, experiments are conducted to understand the domain gap between data distributions. The objectives of this thesis are listed below:

- * Objective 1 is to conduct a literature survey, in which different types of image-to-image translation methods should be discussed and explained in related work. In addition, it is necessary to propose a theoretical comparison between existing methods, which ultimately leads to the decision to adopt a specific method to solve the reasons for the problem statement of the thesis.
- * Objective 2 is to create and collect source and target domain datasets to train the proposed image-to-image translation application. In addition, a test dataset of annotated real images is also collected for evaluation.
- * Objective 3 is to design, implement, and train image-to-image translation application.
- * Objective 4 is to perform experiments, to determine the quality of images generated by the image-to-image translation application. In addition, train classifiers on different data distribution and evaluate them on real data, to analyze the domain gap between those distributions and real data distribution.
- * Objective 5 is to document all the information regarding the thesis, for example, related works, selected methodology, fundamentals required to understand the work, experiments, results, limitations, conclusion, and future work.

1.5 Thesis Limitations and Structure

The domain adaptation field is promoting incremental findings, hence this thesis has its limitations due to time deadlines. The proposed image-to-image application is implemented only using CycleGANs. The implementation in and comparison with other methodologies are placed apart for future work. The scope of this thesis is limited to the improvement of real document images classification by increasing the quality and quantity of labeled data in the target domain, and reducing the domain gap between synthetic data distribution and real data distribution. The rest of this thesis is organized as follows. Chapter 2 briefly reviews literature of numerous GANs variants and existing methods. Chapter 3 discusses the about GANs and Convolution Neural Networks (CNNs) to develop fundamental knowledge to understand this thesis. The proposed approach is described in chapter 4 along with the objective function and algorithm. The implementation of the proposed solution, dataset details, and neural networks architectures, and training details are described in chapter 5. The experiments and results are presented in chapter 6. Finally, this work has been concluded in Chapter 7 along with the future work and the limitations.

1.6 Terminology

In this thesis for simplicity and readability, many terms are explained beforehand for better understanding and to avoid confusion. The following terms described in this thesis are consistently used throughout this thesis. The term “Artificial Neural Networks” will be used as “Neural Networks” interchangeably. The images generated by the CycleGAN generator in the target domain are called “Realistic Document Images” and they are also interchangeably called “CycleGAN Generated Document Images”. So, the distribution of “CycleGAN Generated Document Images” is called “CycleGAN Generated Data Distribution”.

2. Related Works

In the last decades, the field of computer vision has made a tremendous amount of progress, especially in the field of domain adaptation. Numerous image-to-image translation methods are invented and are improved significantly, GANs were among them [24]. Further, several variants of GANs are evolved over the years to resolve various problems. This chapter aims to discuss different image-to-image translation methods. Also, a brief theoretical comparison of existing methods is carried out in section 2.2. Finally, section 2.3 concludes the motivation behind choosing the proposed approach of this thesis.

2.1 Literature Survey

Ian J. Goodfellow et al.[21] proposed a framework of GANs in which two models are simultaneously trained. “A generative model is trained to learn the real data distribution, and a discriminative model that predicts the probability that a sample came from the training data rather than a generative model. The idea of training a generative model is to generate realistic data and maximize the probability of a discriminative model making a mistake. The generator learns to generate realistic data, and the discriminator learns to distinguish real data from the generator’s fake data. The generator is penalized by the discriminator for producing non-credible results. As training starts, the generator begins to create fake data, and the discriminator quickly learns to classify that it’s fake and the generator is penalized to improve and produce credible results. The generator gets closer to producing output samples that can fool the discriminator as training progresses. Finally, if generator training goes well, the discriminator becomes worse at distinguishing between real and fake samples. At the end of the training, ultimately, we have a generator model which produces credible results which are similar to real data” [21]. Authors have trained GAN on a range of datasets including MNIST [25], the Toronto Face Database (TFD) [26], and CIFAR-10 [27]. Also compared against already existing methods like Deep Belief Network (DBN) [28], Stacked Convolutional Autoencoder (Stacked-CAE) [28], and Deep Generative Stochastic Networks (Deep-GSN) [29]. The authors do not claim that “the samples generated by GANs are better than samples generated by already existed methods” [21]. Authors believe that “at least these samples are competitive when compared with the better generative models in the literature, which highlights the potential of the generative adversarial framework” [21]. “Primarily GANs have a computational advantage” [21]. “Adversarial models may have added statistical advantage from the generator network being updated by the gradients flowing through the discriminator using backpropagation, not being updated directly with data examples” [21]. Backpropagation is an algorithm to calculate gradients of the loss function to update weights of the neurons in the neural network [3]. To prevent the Helvetica Scenario [30], special care must be taken while training the GANs. For example, without updating the discriminator, the generator must not be trained too much. The discriminator must be updated well to provide maximum error to update the generator to produce high-quality images. In the Helvetica Scenario, the generator collapses, repeatedly producing the same output or a small set of outputs. Usually, GANs should produce a wide variety of outputs. The Helvetica Scenario is also called Mode Collapse [31].

Xudong Mao et al.[33] proposed another variant of GANs called Least Squares Generative Adversarial Networks (LSGANs). “The discriminator is hypothesized to be a classifier with the sigmoid cross-entropy loss function in Regular GANs. They realized, however, that the cross-entropy loss function can cause vanishing gradients during the learning process. The authors presented the LSGANs, which uses the least-squares loss function for the discriminator to solve this problem” [33]. “The fake samples are penalized by the least-squares loss function, which forces the generator to generate samples closer towards to the decision boundary” [33]. The authors evaluated LSGANs using two datasets LSUN [34] and HWDB1.0 (Handwritten Chinese Character Dataset) [35]. When trained on the LSUN dataset [34] they observed, the images generated by LSGANs are of better quality than the ones generated by the two baseline methods, Deep Convolutional Generative

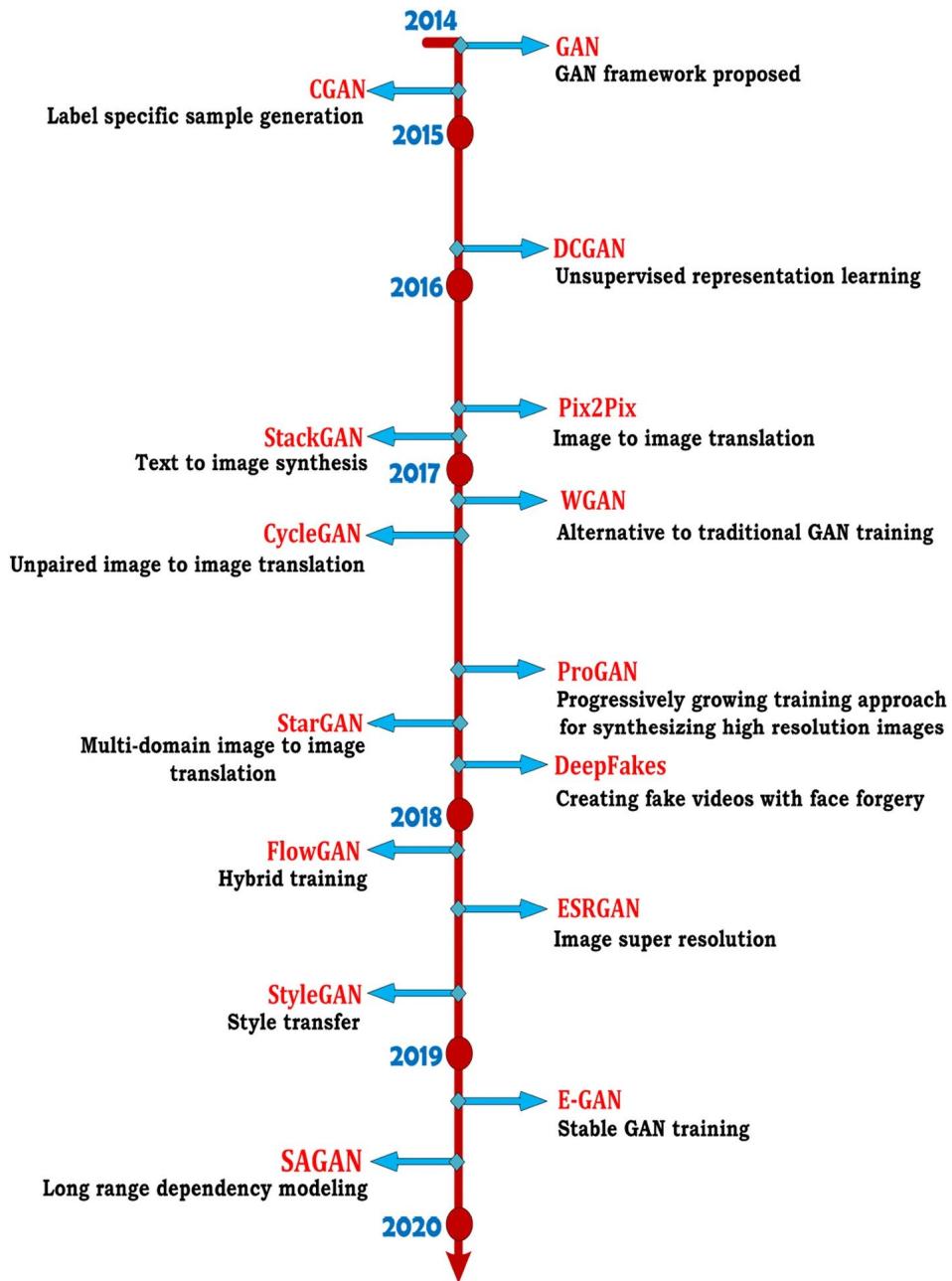


Figure 2.1: Evolution of GANs Over the Years [32].

Adversarial Networks (DCGANs) [36] and EBGANs [37]. Also, when trained on the Handwritten Chinese Character Dataset, the generated characters were readable and clear. Another experiment was conducted to evaluate the stability of LSGANs on a Gaussian mixture distribution dataset, which is designed in literature by Luke Metz et al.[38]. They train LSGANs and regular GAN on a 2D mixture of 8 Gaussian datasets using a simple architecture, where both the generator and the discriminator contain three fully-connected layers. “It is observed that regular GANs suffer from mode collapse” [33].“GANs generate samples around a single valid mode of the data distribution. But LSGANs learn the Gaussian mixture distribution successfully” [33]. Further, in this paper, for evaluating the stability numerous comparison experiments are conducted and the results demonstrate that LSGANs can generate higher quality images than regular GANs, DCGANs [36], and Energy-based Generative Adversarial Networks (EBGANs) [37] and the learning process is stable compared to the regular GANs.

Phillip Isola et al.[39] proposed cGANs as a generic solution to numerous image-to-image translation problems. The authors wanted to provide a single solution for multiple types of image-to-image translation problems. For example, synthesizing photos from label maps [40], reconstructing objects from edge maps [41] [42] , and colorizing images. “cGANs learns the mapping from the input image to output image along with a loss function to train this mapping. This allows using the same generic method to the distinct problems, that traditionally would require very different loss formulations” [39]. Authors implemented cGAN and released it as the pix2pix software to solve distinct image-to-image translation problems. “In cGAN the generator uses U-Net-based architecture [43], the U-Net is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks. The discriminator uses a convolutional PatchGAN classifier [44], which only penalizes structure at the scale of image patches. Unlike unconditional GANs, both the generator and discriminator observe the input images” [39]. Authors performed multiple experiments during ablation studies using evaluation metrics like, Amazon Mechanical Turk (AMT) perceptual study, FCN-Score, and semantic segmentation metrics. They found that “L1 distance loss encourages less blurring compared to the L2 distance loss” [39]. Also combining L1 Loss and cGAN (L1 + cGAN) generates better results compared to combining Unconditional GAN and L1 Loss ((L1 + GAN)). “The cGAN appears to be more effective on the problem where the output is highly detailed or photographic” [39]. The pix2pix software code is available at GitHub. In figure 2.2 the mapping from edges →photo transformation illustrated, and “both generator and discriminator are conditioned on auxiliary information like input edge map” [39].

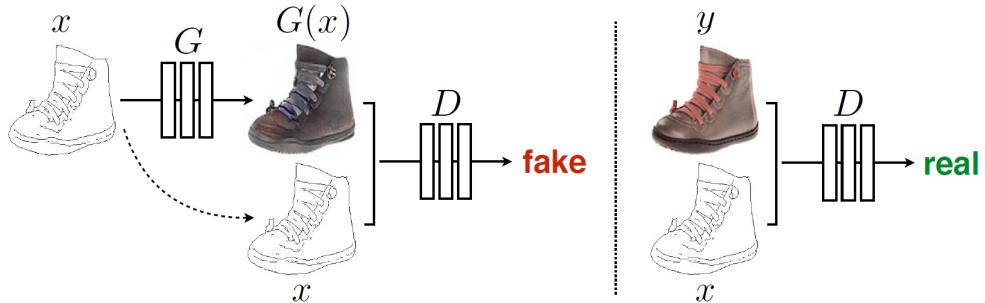


Figure 2.2: Illustration of the training a cGAN to map edges →photo transformation. Unlike unconditional, GAN’s both discriminator and generator observe the input edge map [39].

Taesung Park et al.[45] proposed a framework for encouraging content preservation in unpaired image-to-image translation problems by maximizing the mutual information between input and output with patchwise contrastive learning [46]. “In the patchwise contrastive learning for image-to-image translation, a generated output patch should appear closer to its corresponding input patch in comparison to other random patches present in the same input. To achieve patchwise contrastive learning, drawing patches internally from within the input image, rather than externally from other images in the dataset, forces the patches to better preserve the content of the input” [45]. This method requires neither a memory bank nor specialized architecture. The authors demonstrated that “the framework enables one-sided translation in the unpaired image-to-image translation setting while improving quality, consuming less memory, and reducing training time” [45]. They call

this approach Contrastive Unpaired Translation (CUT). Since contrastive representation is formulated within the same image, this method can even be trained on single images, where each domain is having only a single image. The several prior methods like CycleGAN [1], Multimodal Unsupervised Image-to-image Translation (MUNIT) [47], Diverse Image-to-Image Translation (DRIT) [48], and Geometry-Consistent Generative Adversarial Networks (GCGAN) [49] were unable to achieve significant results compared to the CUT method, on other hand, it often produced higher quality images and more accurate generations with light footprint in terms of training speed and GPU memory usage. Since CUT method is one-sided, it is memory efficient and faster compared to prior baselines. The evaluation metrics like Fréchet Inception Distance (FID) [50] and semantic segmentation scores are used to compare the quality of generated images using CUT method. Furthermore, the authors also introduced faster and lighter variant Fast Contrastive Unpaired Translation (FastCUT). FastCUT also produces competitive results with even lighter computation cost of training. The code and models for CUT are available at GitHub.

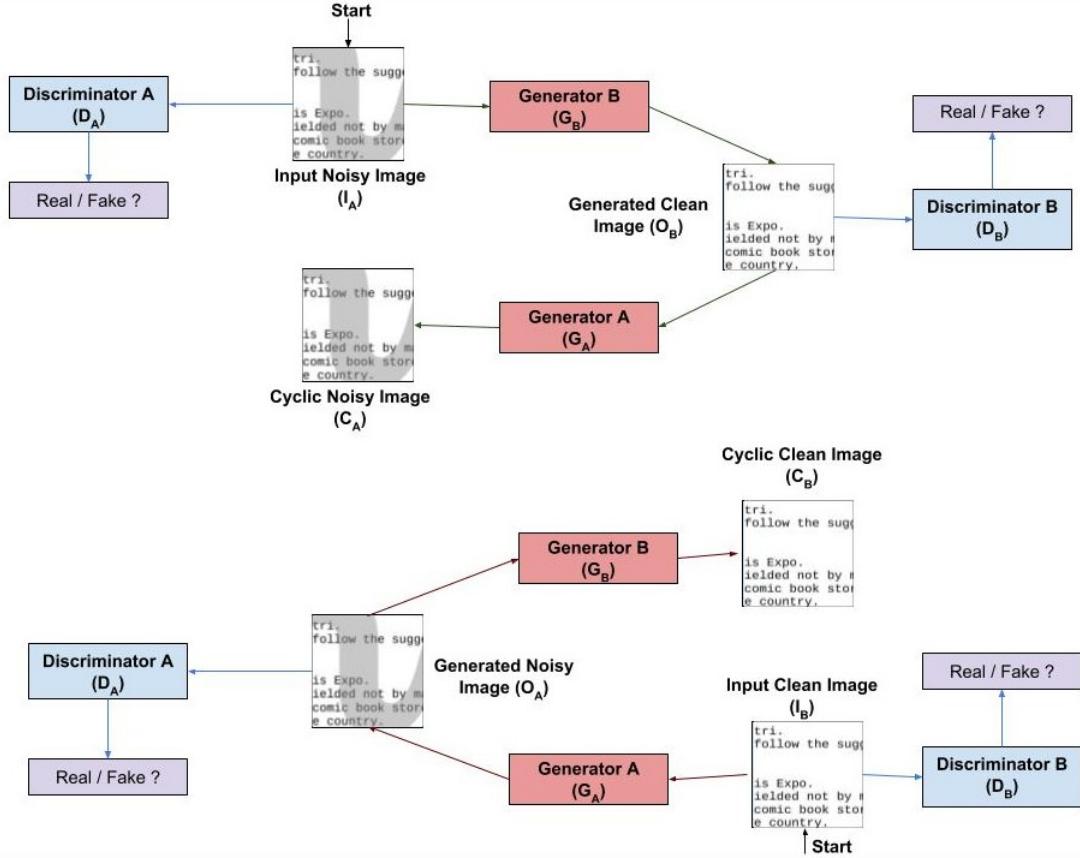


Figure 2.3: Illustration of CycleGAN transforming the noisy document images into clean document images. The generators G_A and G_B are responsible for the mapping of noisy images to clean images using cycle-consistency loss [1]. And the two discriminators D_A and D_B rejects samples generated by D_A and D_B acting like an adversary [51].

Monika Sharma et al.[51] is addressing “a problem in the scanning process that often results in the introduction of blur due to camera motion, salt and pepper noise, or water markings, shake, wrinkles, coffee stains, or faded data. These artifacts pose many readability challenges to current text recognition algorithms and significantly degrade their performance” [51]. So, “the existing denoising techniques require a paired training dataset consisting of noisy documents paired with cleaned versions of the same document. However, in the real world, to generate clean documents from noisy versions, such a paired training dataset is not available to train a model” [51]. Hence, the authors proposed to use CycleGAN because “it is state-of-the-art and known to learn a mapping between the distributions in the absence of paired training dataset. Using CycleGAN, noisy document images transformed into denoised and clean document images to achieve image-to-image translation” [51]. They have compared the performance of CycleGAN for document cleaning tasks with a cGAN by training them over the same dataset. The only difference was CycleGAN trained

using unpaired images and cGAN trained using the paired images. They have used Peak Signal-to-Noise Ratio (PSNR)¹ as a evaluation metric to evaluate the quality of transformed denoised images. Several experiments were performed on 4 separate public document datasets, one each for deblurring, watermark removal, background noise removal, and defading. Finally, they conclude that “CycleGAN learns a more robust mapping from the space of noisy to clean documents compared to cGAN” [51]. The complete architecture of CycleGAN for denoising documents illustrated in figure 2.3.

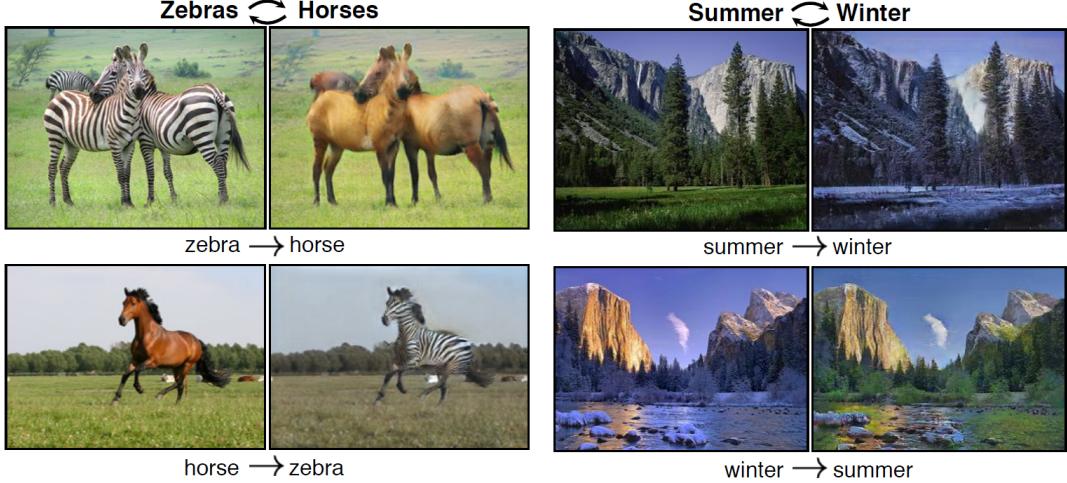


Figure 2.4: Illustration of CycleGAN transforming an image from one into the other and vice versa. For example, zebra image transformed into horse image, and vice versa. The view of Yosemite mountains in summer transformed into winter, and vice versa [1].

Chris Tensmeyer et al.[11] realized solving binarization tasks using deep learning models is very challenging. It is due to the scarcity of large quantities of labeled data available to train such models. They also mention there have been efforts to create synthetic data for binarization using image processing techniques but, they generally lack realism. In this paper, the authors proposed “a method to produce realistic synthetic data using an adversarially trained image translation model” [11]. They extended “the popular CycleGAN model to be conditioned on the ground truth binarization mask as it translates images from the domain of synthetic images to the domain of real images” [11]. They have found that modifying the discriminator to condition on the binarization Ground Truth (GT) leads to increased realism and better agreement between the GT and the produced image. They called the proposed model DGT-CycleGAN. Also shown “DGT-CycleGAN model produces more realistic synthetic data” [11]. They validated their approach by pretraining deep networks on realistic synthetic datasets generated by DGT-CycleGAN, CycleGAN, and image compositing. They evaluate both pretrained only and finetuned models on each of the Document Image Binarization Competition (DIBCO) datasets. They have concluded that pretraining deep neural networks on the more realistic synthetic data generated using DGT-CycleGAN lead to better predictive performance both before and after finetuning on real data.

Jun-Yan Zhu et al.[1] proposed a modified version of GAN which is the state-of-the-art method for the image-to-image translation that can transform the images from source domain X to target domain Y in the absence of paired training dataset. “This method can learn to capture special characteristics of one image collection and figuring out how these characteristics could be translated into the other image collection, all in the absence of any paired training examples. This modified version of GAN is called CycleGAN. In this method, the goal is to learn a mapping $G : X \rightarrow Y$ such that the distribution of images $G(X)$ is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained and coupled with an inverse mapping $F : Y \rightarrow X$ to introduce a cycle-consistency loss to enforce $F(G(X)) \approx X$ and vice versa” [1]. Along with an adversarial loss and cycle-consistency loss, this work also introduces identity mapping loss which helps to preserve the colour of input images. Authors considered evaluation

¹Peak Signal-to-Noise Ratio: <http://www.ni.com/white-paper/13306/en/> last access: September 3, 2021.

metrics like AMT perceptual studies, FCN-score [39], and semantic segmentation metrics to compare the quality of generated images against other baseline. The several prior methods like Bi-GAN/ALI [[52], [53]], CoGAN [54], SimGAN [55] were unable to achieve compelling results. The CycleGAN method, on other hand, can produce images that are often of similar quality to the fully supervised pix2pix [39]. Authors provide both PyTorch and Torch implementations. In figure 2.4 examples of CycleGAN transforming an image from one into the other and vice versa illustrated.

2.2 Discussion

Over the years, GANs have evolved to solve different kinds of problems. The evolution of the GANs is illustrated in the figure 2.1 using a timeline diagram. GANs suffer from unstable training, vanishing gradients problem, and mode collapse. Consequently, Martin Arjovsky et al.[56] proposed Wasserstein GAN (WGAN) to solve problems with GANs. WGANs improve the stability of learning, get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches. Although the implementation of WGAN is straightforward, the theory behind it is heavy and requires some hack, for example, Weight Clipping [57]. Hence, Xudong Mao et al.[33] proposed a simple and more intuitive method compared to WGAN, called LSGAN. First, LSGANs are able to generate higher quality images than regular GANs. Second, LSGANs perform more stable during the learning process. Moreover, Jun-Yan Zhu et al.[1] proposed CycleGAN. It is an unsupervised image-to-image translation approach. Compared to GANs, CycleGANs can deal more meticulously with the problems like unstable training, vanishing gradients problems, and mode collapse. Also, they described that CycleGAN has outperformed existing baselines as Bi-GAN/ALI [[52], [53]], CoGAN[54], and SimGAN [55]. Furthermore, Monika Sharma et al.[51] proclaimed CycleGAN can transform noisy document images into denoised and clean document images to achieve image-to-image translation. They have also compared the performance of the developed image-to-image application with cGAN and demonstrated CycleGAN had outperformed cGAN. Chris Tensmeyer et al. [11] proposed DGT-CycleGAN, a modified version of CycleGAN, which is adequate to translate images from the domain of synthetic images to the domain of real images to solve binarization tasks. Moreover, Taesung Park et al.[45] proposed CUT. They have demonstrated that CUT is a better, faster, and memory-efficient approach to perform unsupervised image-to-image translation. It has outperformed CycleGAN [1], MUNIT [47], DRIT [48], and GCGAN [49]. However, due to time constraints, multiple available references, and code repositories, CycleGAN has been a choice for this thesis and research to perform unsupervised image-to-image translation.

2.3 Conclusion

The image-to-image translation is a classic example of computer vision and computer graphics problems. In which the goal is to learn the mapping between a source image and target image using a training set of aligned image pairs. Although, for many tasks, aligned or paired training data will not be available. Also, collecting and annotating paired document images is tedious, difficult, time-consuming, and costly. This thesis is attempting to close a domain gap between synthetic document images and real document images in the absence of paired training data. The thesis proposes an image-to-image translation application to transform synthetic document images into realistic document images to perform domain adaptation, by closing the gap between synthetic data distribution and real data distribution. Ultimately, a large amount of realistic annotated set of document images can be generated using this application. Furthermore, they can be used to improve real document image classification. Also, in case, if a classifier is trained using such generated realistic document images, it can fasten the process of labeling the new annotated or unlabeled real document images. As per the above literature survey [[1], [51]] and problem statement, in this thesis CycleGAN is a decided method to transform synthetic document images into realistic document images in the absence of paired training data. Hence, the proposed image-to-image translation application is realized using CycleGAN.

3. Fundamentals

This chapter aims to develop a better understanding of fundamental concepts required to understand this thesis. It discusses the mathematics and working principle of the GANs. It also discusses CNNs and its layers. The formulation and architecture of GANs is explained in section 3.1. In section 3.2, layers like convolution layer, activation layer, pooling layer, and fully connected layer have briefly explained.

3.1 Generative Adversarial Networks (GANs)

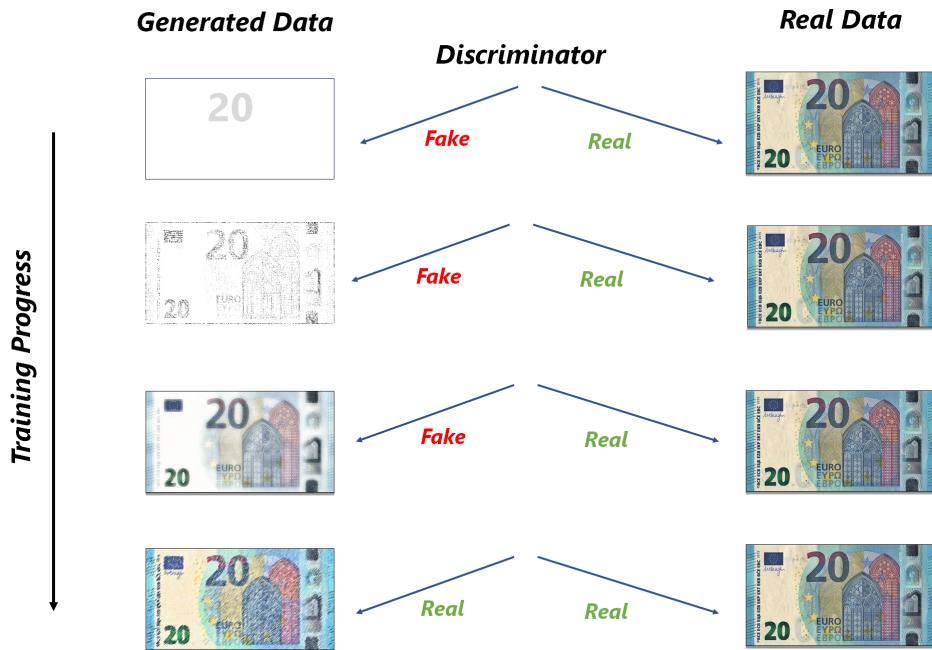


Figure 3.1: Intuitive example of GAN training progress.

Ian J. Goodfellow et al.[21] proposed GANs. It has two models in its architecture, one is generator, and another is discriminator. Authors mentioned both can be implemented using multilayer networks. Normally, the term multilayer networks indicated for CNNs or ANNs. Let's try to understand the function of GANs intuitively. Consider generator is a forger who creates fake images of currency, with the intention of making them realistic as much as possible. Furthermore, the discriminator is an expert, which receives both forged and real images, and distinguishes between real and forged images. The discriminator has access to images generated by the generator and real images present in the dataset. The generator generates fake images without having access to the real images using random noise distribution. Both generator and discriminator are trained simultaneously. The discriminator learns by the error signal provided by the ground truth of distinguishing whether the image came from the dataset of real images or from the generator. The generator learns using the same error signal given by the discriminator to produce a better quality of forged or fake images. It is a setup where both generator and discriminator are competing with each other.

In figure 3.1 an intuitive example of GAN training process is displayed. During the initial stage of the training, images produced by the generator are easily distinguishable by the discriminator as fake images. After a certain number of iterations, the quality images generated by the generator

increases by the feedback error signal given by the discriminator. Also, the discriminator gets smarter and smarter to distinguish fake images and real images, like mentioned earlier both models are trained simultaneously. But at a certain point generator starts to produce realistic images which the discriminator classifies as real. In figure 3.2, we can see GAN in action. Let us have a look into the GAN's core architecture. As described earlier, the task of the GAN is to generate fake samples. Hence, the training dataset has a set of real data samples, from which the generator learns to create fake samples that are similar to the real samples. This set of real data samples serves input X to the discriminator D . The random noise vector Z retrieved from the random noise distribution serves as an input to the generator to produce fake samples. The generator G takes Z as an input and outputs a fake sample X' . Its goal is to create fake samples that are indistinguishable from the real samples from the training dataset. The discriminator D takes input from real data sample X that are present in the training dataset and fake samples X' generated from the generator G . For each sample, it determines and outputs the probability of the sample that is real. For every discriminator's prediction, The classification error is backpropagated to update both generator and discriminator during iterative training. The discriminator's weights are updated to maximize classification accuracy which means, maximizing the probability of correct prediction X as real and X' as fake. The generator's weights are updated to maximize the probability that the discriminator misclassifies X' as real.

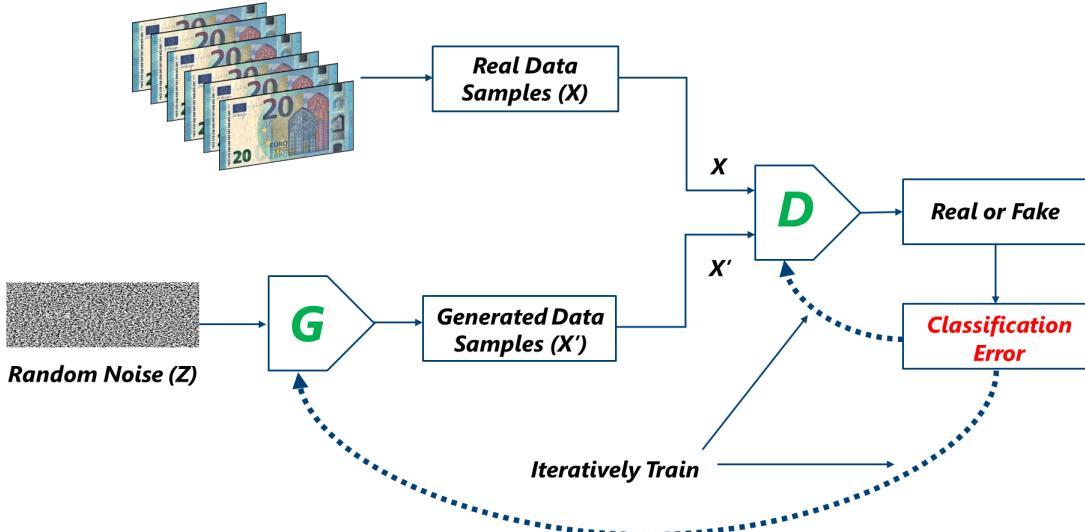


Figure 3.2: Overview of core GAN architecture along with the generator's and discriminator's inputs, outputs, and their interactions.

Now let's try to understand the mathematics behind the GANs. For learning the generator's distribution p_g over data x , the input noise variable defined as $p_z(z)$. The generator G and discriminator D are the differentiable functions represented by multilayer networks with parameters θ_g and θ_d respectively. The mapping function between some representation space, called, the latent space to the space of data, represented by $G(z; \theta_g)$. The $D(x; \theta_d)$ is a mapping function that maps data x to the probability that it came from the real data distribution rather than generator distribution p_g . Basically, $D(x)$ represents the likelihood of x came from the data rather than p_g . $D(x; \theta_d)$ outputs a single scalar value between $[0, 1]$. D is trained to maximize the probability of correctly labeling both training examples and data generated by the G . Usually, when the generator is training, the discriminator does not train and vice versa. This means for a fixed generator G , the discriminator is trained to classify data as either being from the real data distribution (real, probability close to one) or from the fixed generator(fake, probability close to zero). After certain iterators of the training, when the discriminator is optimal, it can be frozen. Following, the generator G will be continued to be trained to lower the accuracy of the discriminator. After training, if the generator can match the real data distribution, then the discriminator maximally confused, will predict 0.5 probability for its inputs. In practice, the discriminator is may not be trained till it is optimal [58]. Furthermore, simultaneously G is being trained to minimize $\log(1 - D(G(z)))$. Simply

put, D and G play the two-player minimax game with the objective function $\mathcal{L}_{GAN}(G, D)$:

$$\min_G \max_D \mathcal{L}_{GAN}(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (3.1)$$

Authors mentioned, in practice, the equation 3.1 may not give enough gradient for G to learn well. During Early in learning phase, when G is poor, D can reject samples with high confidence because they are clearly distinct from the training data. In this case, term $\log(1 - D(G(z)))$ in equation 3.1 saturates very quickly. Hence, we can train G to maximize $\log D(G(z))$ rather than training G to minimize $\log(1 - D(G(z)))$. This objective function leadsto the same fixed point of the dynamics of G and D by providing much stronger gradients early in learning [21].

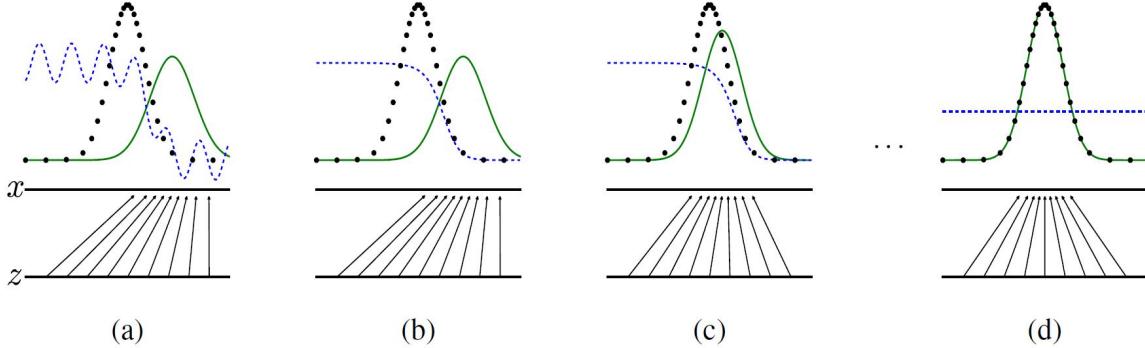


Figure 3.3: Illustration of GANs converging to match generated data distribution p_g to real data distribution p_{data} [21].

In figure 3.3, GANs converging to match generated data distribution p_g to real data distribution p_{data} . While training GANs, discriminative distribution (D , blue, dashed line) is simultaneously updated, so it will discriminate samples from the real data distribution (black, dotted line) p_x from those of the generated data distribution p_g (G , green, solid line). The lower horizontal line represents the domain noise distribution p_z from which z is sampled uniformly. The horizontal line above represents the part of the domain of real data x . The upward arrows depict the mapping of $x = G(z)$, which creates the non-uniform generated data distribution p_g on transformed samples. (a) Consider, generator G and discriminator D are on the verge of convergence, p_g is almost similar to p_{data} and D is a partially accurate classifier. (b) Slowly, discriminator D trained further to distinguish real data samples from generated data samples. For a fixed generator, there is an optimal discriminator, $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$. (c) The generator G is updated using gradients that are backpropagated from discriminator D and makes $G(z)$ to flow to regions that are more likely to be classified as real data. The parameters of the generator are updated, while the parameters of the discriminator are fixed and vice versa. (d) After several steps of training, the generator, G , is optimal where $p_g = p_{data}$, which is equivalent to the optimal discriminator D predicting 0.5 for all samples drawn from x , at this stage discriminator D is maximally confused and will not able to distinguish real data samples from generated data samples, i.e. $D(x) = \frac{1}{2}$ [21].

3.1.1 GAN Training

This section presents the algorithm of the GAN and in the figure A.4, we can see the pictorial representation of the algorithm. The GAN training algorithm is divided into two sections. These two sections are discriminator training and generator training. In figure A.4 shows the same GAN network in different stages of the training process at distinct time points. For example, while training generator, the discriminator is not training and vice versa. Hence, the training dataset of real data samples is grayed out or disabled in the section when the generator is getting trained. While training discriminator both generated samples from the generator and real samples are used as an input. In the following sections describe generator's and discriminator's architecture and their

training process. Further, the GAN training algorithm has been described in the form of pseudo code.

Discriminator Architecture

The discriminator in a GAN is a binary classifier. It attempts to classify real data samples from fake data samples generated by the generator. It outputs the probability of input being a real data sample. The discriminator can be implemented using a multilayer neural network which is suitable to the kind of data it's classifying. The discriminator's task is to classify images, hence, it is implemented using CNN [36]. The discriminator's training data come from the two resources, training dataset or collection of real data samples and fake data samples generated by the generator. During GAN training, the discriminator is trained by both real data samples and fake data samples generated by the generator. The discriminator is penalized for misclassifying a real data sample as fake or a fake data sample as real. Such classification error is called discriminator loss. The discriminator loss is backpropagated to update the weights of the discriminator network. The training process of the discriminator D using backpropagation illustrated in figure 3.4.

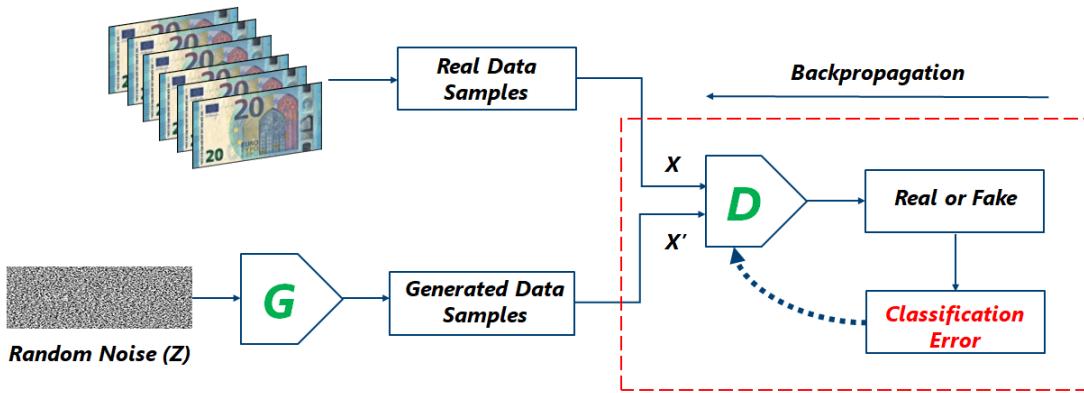


Figure 3.4: Illustration of the training of the discriminator D using backpropagation.

Generator Architecture

The generator produces fake samples using random noise vectors sampled from the uniform random noise distribution. As described earlier, the discriminator's weights are frozen while training the generator. This means the discriminator is not training while the generator is generating fake samples and vice versa. The generator can be implemented using a multilayer neural network which is suitable for the kind of data it's generating. The generator's task is to generate images using the random noise distribution, so it is implemented using CNN [36]. The generator aims to produce fake samples that are as realistic as possible. But when the generator fails to fool the discriminator or when the discriminator classifies the generated sample as fake, then it is penalized with classification error. Such classification error is called generator loss. The generator loss is backpropagated to update the weights of the generator. The training process of the generator G using backpropagation illustrated in figure 3.5.

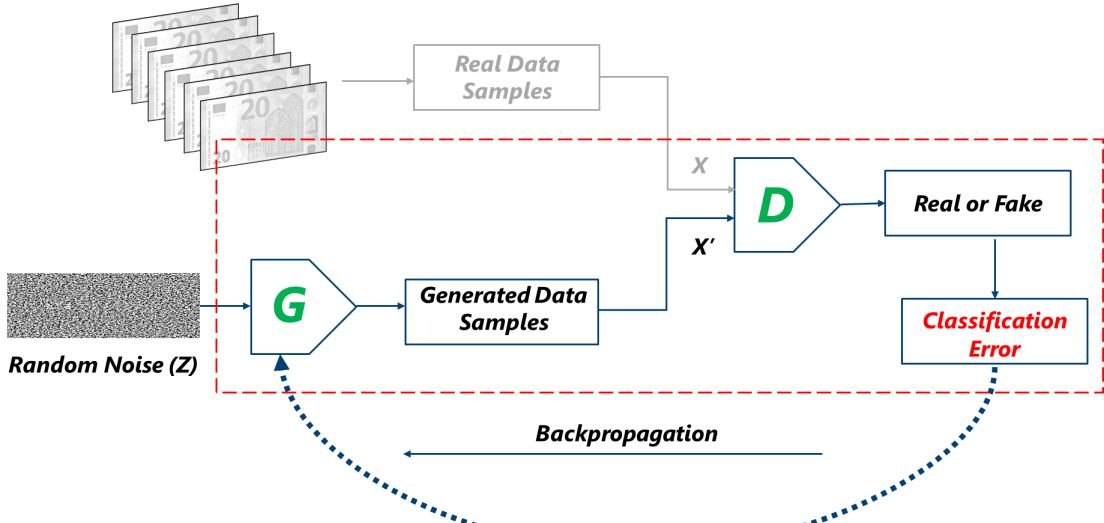


Figure 3.5: Illustration of the training of the generator G using backpropagation.

GAN Training Algorithm

```

1 for each training iteration do
2   1. Train the Discriminator:
3     a) Get a random real data sample  $X$  from the training dataset.
4     b) Get a random noise vector  $Z$  from the random noise distribution, pass it thorough
        the Generator and create a fake sample  $X'$ .
5     c) Classify  $X$  and  $X'$  using the Discriminator.
6     d) Backpropagate the calculated classification error to update the Discriminator's
        weights to minimize classification error.
7   2. Train the Generator:
8     a) Get a random noise vector  $Z$  from the random noise distribution, pass it thorough
        the Generator and create a fake sample  $X'$ .
9     b) Classify  $X'$  using the Discriminator.
10    c) Backpropagate the calculated classification error to update the Generator's
        weights to maximize the Discriminator's error.
11 end

```

Algorithm 1: GAN Training Algorithm [2].

3.2 Convolution Neural Networks

In recent years in the field of machine learning, drastic improvements have occurred to increase the performance of machine learning models. Also, numerous deep learning techniques like ANNs are evolved significantly. These biologically inspired neural networks were able to exceed the performance compared to the traditional machine learning techniques to solve complex problems [59]. The most successful image-driven pattern recognition technique among ANNs is CNNs [59]. Nowadays, CNNs are used to solve difficult image recognition, image classification, and object detection tasks. The CNNs have been a popular method because of its extraordinary results at object recognition

competition known as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 for solving computer vision tasks. CNN is a kind of deep learning technique for processing data that has a grid pattern, for example, images. CNNs are inspired by the early findings in the study of biological vision, especially, organization of the animal visual cortex [[60], [61], [9]] and designed to automatically and adaptively learn spatial hierarchies of features, from low-level to high-level patterns [9]. It is composed of various building blocks, such as convolution layers, pooling layers, and fully connected layers. As shown in figure 3.6, the first two layers are convolution and pooling layers. They perform feature extraction. whereas the third, a fully connected layer, maps the extracted features into the final output, such as classification [62]. Let us have a look at each layer in the following sections.

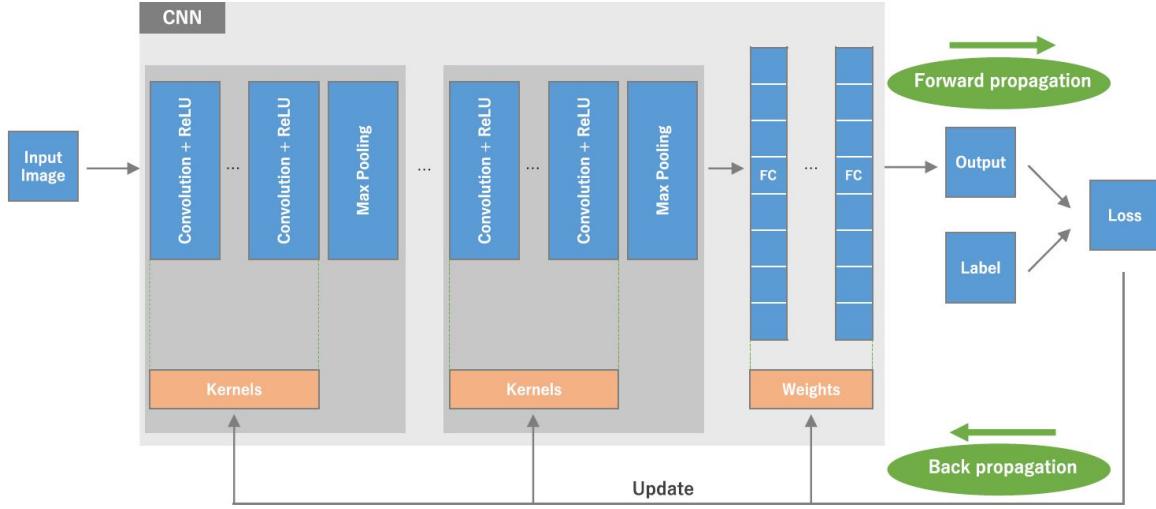


Figure 3.6: Overview of CNN architecture and its training process. CNNs is a combination of several building blocks like convolution layers, pooling layers, and fully connected layers. These building blocks are stacked upon each other. The CNNs is trained using the training dataset, the input data is fed in the forward direction through the network. The process of feeding data in the forward direction to the CNN is called forward propagation. Each layer accepts the input data, processes it as per the activation function like ReLU (Rectified Linear Unit) (figure 3.11), and passes it to the successive layer. Using a loss function through forward propagation on a training dataset, a model's performance under particular kernels and weights is calculated. The learnable parameters, for example, weights and kernels, are updated as per the loss value through backpropagation using a gradient descent optimization algorithm [63].

3.2.1 Convolution Layer

The convolution layer is a basic component of the CNN architecture that performs feature extraction. It is typically a combination of linear and nonlinear operations like convolution operation and activation function. It is one of the core building blocks of CNNs. Also, it is responsible for most of the heavy computations. A convolution layer plays an important role in CNN, it performs mathematical operations like convolution, a specialized type of linear operation. The digital images store pixel values in a Two-dimensional (2D) grid, i.e., an array of numbers as shown in figure 3.8. The small grid of parameters called the kernel, an optimizable feature extractor, is applied at each image position [62], which makes CNNs highly efficient for image processing, feature extraction, since a feature could occur at any location in the image. The layers are connected, one layer feeds its output to the next layer. The extracted features can hierarchically and progressively become more complex [62]. Training is the process of optimizing parameters such as kernels which minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation [64] and gradient descent [63], among others.

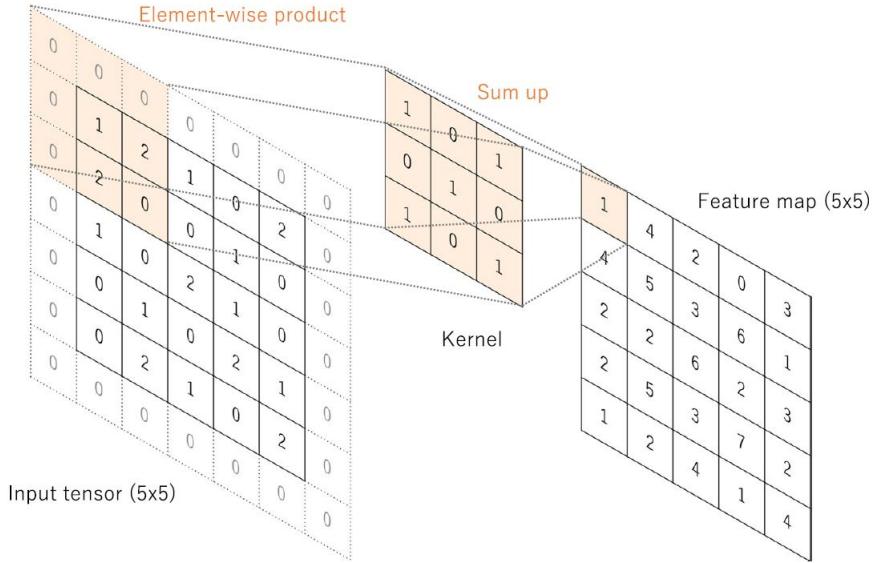


Figure 3.7: Illustration of a convolution operation with zero padding to retain spatial dimensions. Note that an input dimension of 5×5 is retained in the generated output feature map. In this example, kernel size is set to 3×3 , and stride is set to 1 [62].

For feature extraction linear operation, convolution is used, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between kernel's each element and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (figure 3.8a-c) [62]. This procedure is repeated by applying multiple kernels to create an arbitrary number of feature maps, which describe different characteristics of the input tensors [62]. The different kernels can be considered as different feature extractors. Two important hyperparameters that represent the convolution operation are the size and number of kernels. The size is typically 3×3 , but sometimes 5×5 or 7×7 , depends on the requirement. The number of kernels is arbitrary, and determines the depth of output feature maps. The above-mentioned convolution operation prevents the center of each kernel from overlapping the input tensor's outermost element. And the output feature map's height and width are reduced compared to the input tensor. To address this issue, the padding, predominantly zero-padding technique is used where rows and columns of zeros are added on each side of the input tensor, to fit the center of a kernel on the outermost element and keep the same spatial dimension through the convolution operation (figure 3.7). Modern CNN architectures normally employ zero padding to retain spatial dimensions to apply more further layers. Each successive feature map would get smaller after the successive convolution operation without zero padding. A stride is the distance between two successive kernel positions, and it also defines the convolution operation. A stride of 1 is the most common choice; however, a stride greater than 1 is sometimes used to achieve feature map downsampling. A pooling operation, as defined in section 3.2.2, is an alternative technique for downsampling.

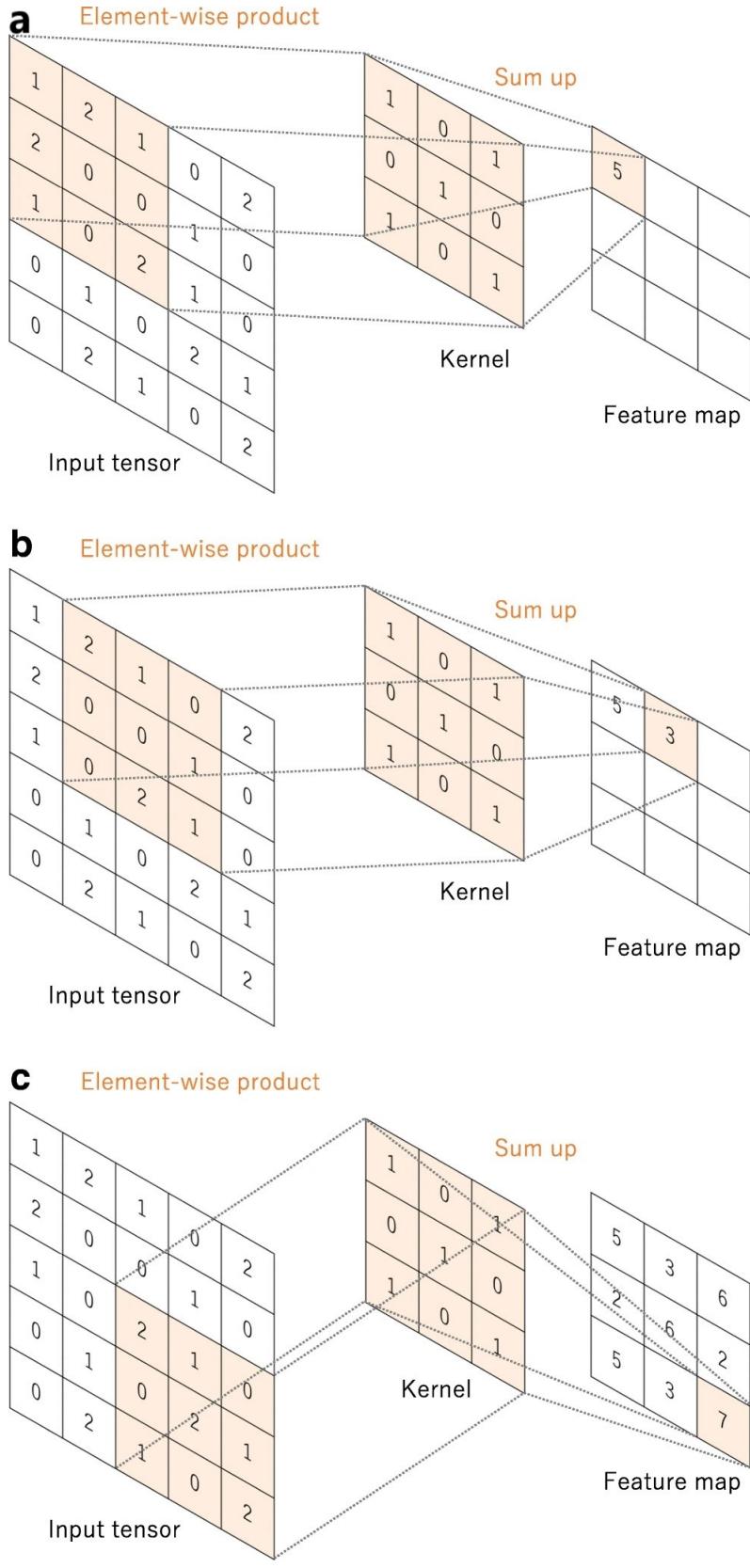


Figure 3.8: a–c An illustration of convolution operation with no padding. In this example, kernel size is set to 3×3 , and stride is set to 1. A kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor, called a feature map [62].

Activation Functions

ANNs are inspired by biological neural networks present in the animal brain. Before constructing any ANN, it is important to understand the artificial neuron model. In the figure 3.10 schematic diagram of an artificial neuron is illustrated. A biological neuron gets excited when other neurons with different weights send electrical signals to it. The value of the electrical signal should be big enough to excite the neuron. Otherwise, it will be in an inactive state. In the figure 3.3 schematic diagram of an artificial neuron is illustrated. Where $\{X_1, X_2, X_3, \dots, X_n\}$ are the inputs to the artificial neuron, $\{W_1, W_2, W_3, \dots, W_n\}$ are the weights corresponding to the inputs. b is the bias. The summation symbol represents the addition unit. The addition unit gets the linear weight sum Z of the inputs and bias. Dot product between inputs and weights performed before addition. If $X = [X_1, X_2, X_3, \dots, X_n] \in R^n$ and $W = [W_1, W_2, W_3, \dots, W_n] \in R^n$, then output Z is represented as

$$Z = XW^\top + b. \quad (3.2)$$

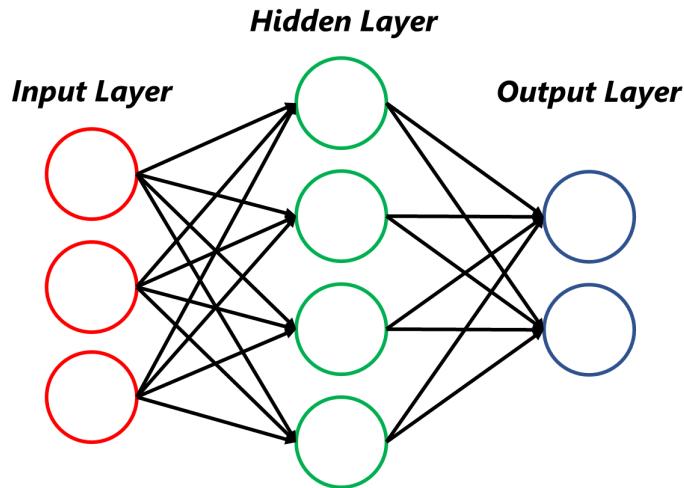


Figure 3.9: Simple Neural Network.

The function f is activation function. It is used to excite the response state of biological neurons and obtain output Y

$$Y = f(Z). \quad (3.3)$$

The data is fed to the input layer of the neural network, the input undergoes the linear operation. Later, activation functions are applied to it in the hidden layers, and output is produced. In neural networks the hidden layer lies between input layer and output layer. In figure 3.9 simple neural network is illustrated. The activation function is applied to the outputs of a linear operation like convolution. The activation functions important while constructing any neural network. They are mathematical functions attached to the neurons. Also, they tell which of the neuron is excited or triggered in each layer of neural networks. The purpose of the activation function is to introduce non-linearity in the neural network.

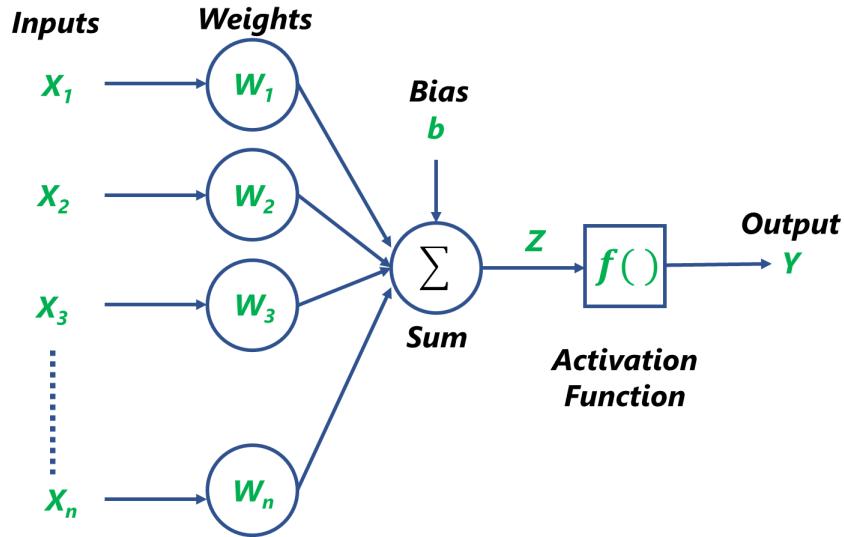


Figure 3.10: Illustration of Artificial Neuron.

The equation 3.2 performs a linear operation. This linear function gets repeated every hidden layer of the neural network. A combination of such linear functions also a linear function, equivalent to all the hidden layers collapsed into a single linear function performing linear regression. Hence, all the hidden layers become useless. Even if linear functions are easy to use, but they fail to learn complex patterns present in the data like images, speech, and videos. Hence, nonlinear activation functions are used, while constructing neural networks. The nonlinear activation functions are differentiable, and they make gradient descent optimization (backpropagation) possible. Backpropagation minimizes the error and enhances the accuracy and performance of the neural network. The nonlinear functions like the sigmoid and hyperbolic tangent (\tanh) functions were used previously because they are mathematical representations of biological neuron behavior. The rectified linear unit is now the most commonly used nonlinear activation function (ReLU), which simply computes the function: $f(x) = \max(0, x)$ (figure 3.11) [[65], [66], [[67]], [[68]], [69]].

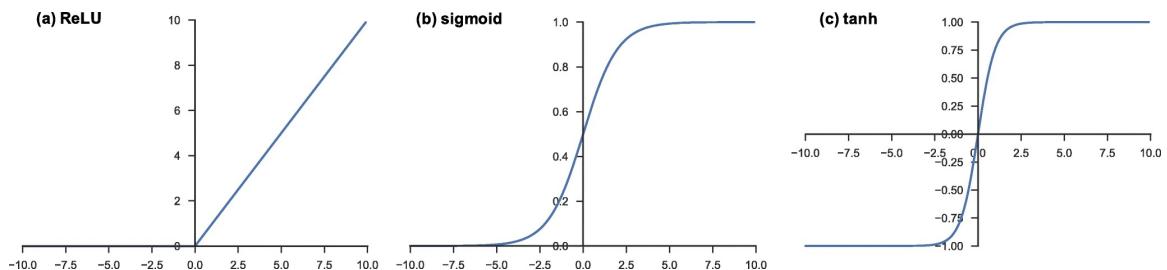


Figure 3.11: Most common nonlinear activation functions used while constructing Neural Networks: a) rectified linear unit (ReLU), b) sigmoid, and c) hyperbolic tangent (\tanh) [62].

3.2.2 Pooling Layer

A pooling layer performs downsampling on feature maps to gradually reducing the spatial dimensionality, which leads to the reduction of the number of parameters and computational complexity of the neural network and controlling overfitting. Downsampling introduces translation invariance to minor shifts and distortions [3]. Downsampling of feature maps can be accomplished by using convolution layers by increasing the stride of the convolution operation across the image. But the robust and common approach of downsampling is to use a pooling layer [3]. Similar to convolution operations, it used hyperparameters like filter size, stride, and padding. Also, it is common to periodically insert a pooling layer in between successive convolution layers while constructing neural

networks. There are two common types of pooling methods one is max pooling the other is average pooling. The max-pooling considers the most activated (maximum value) value in each input patch of the feature map. The average pooling averages values in each input patch of the feature map.

Max Pooling

Max pooling is the most common and popular type of pooling operation. The max-pooling operation is independently operated at every depth slice of the input and resized spatially. Commonly, in max-pooling filters of size 2×2 applied with a stride of 2. The max-pooling operation extracts small patches of given filter size from the input feature maps and outputs the maximum value in every patch discarding others (figure 3.12) [3]. The spatial dimension of feature maps is reduced by a factor of two by discarding 75% of its activations [70]. The height and width dimension of the features maps is reduced but the depth dimension remains unchanged. Pooling operations with larger filter sizes are too destructive. It's worth noting that max-pooling layers do not have learnable parameters.

Global Average Pooling

There is another type of pooling operation called global average pooling. It performs an extreme type of downsampling. It downsamples a feature map of size $\text{height} \times \text{width} \times \text{depth}$ into a $1 \times 1 \times \text{depth}$ array by averaging all the elements present in the feature map by keeping depth dimension unchanged. This operation is applied only once before fully connected layers. There are some benefits of using global average pooling: a) It reduces the number of learnable parameters. b) allows the CNN to consider inputs of variable size [62] [71].

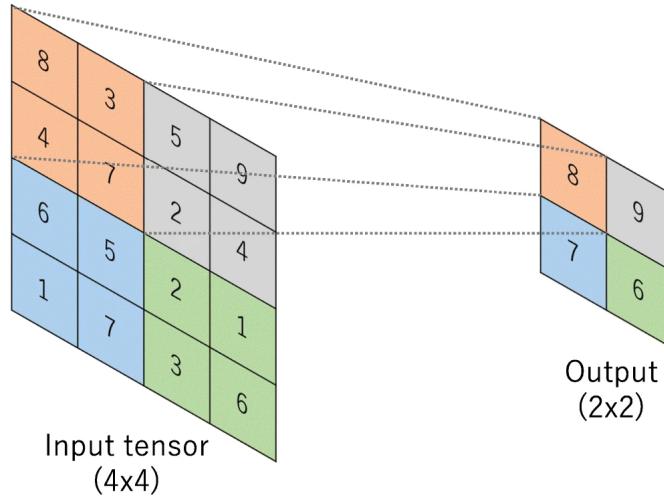


Figure 3.12: Illustration of max pooling operation with a filter size of 2×2 . No padding, and a stride of 2. Max pooling operation extracts 2×2 patches from the input tensors, outputs the maximum value in each patch, and discards the rest of the other values, which results in the spatial dimension of an input tensor downsampled by a factor of 2 [62].

3.2.3 Fully connected layer

In neural networks, fully connected layers are those layers where all the inputs from one layer are connected to every neuron of the next layer. The final output feature map of convolution layers or pooling layers is flattened and the output is transformed into a One-dimensional (1D) vector or array of numbers. The flattened output is connected to one or more fully connected layers. The fully connected layers are also called dense layers, where every input is connected to every output by

a learnable weight. A fully connected layer performs multiplication between the input and weight matrix and then adds a bias vector. Each fully connected layer is followed by a nonlinear activation function, for example, ReLU. As already described, the nonlinear activation function helps neurons learn complex patterns. The equation 3.3 represents the function of a fully connected layer. The features extracted using convolution layers and downsampled by pooling layers are mapped by a subset of fully connected layers to the final output of the neural network, for example, probabilities for each class in the classification tasks. Most of the time number of output nodes in the final fully connected layer equals the number of classes. The final fully connected layer's activation function is usually different than others. Each task chooses a suitable activation function. The sigmoid activation function is used in binary classification tasks. For example, logistic regression models the probabilities for binary classification tasks using the sigmoid activation function. The softmax activation function is widely used in multiclass classification tasks. It normalizes output real values from the last fully connected layer to target class probabilities, ranges between 0 and 1, and all values sum to 1 [62].

4. Methodology

In this chapter, the methodology of the proposed image-to-image translation application explained in depth. In section 4.1, the proposed approach is pictorially described and theoretically explained. In section 4.2, the mathematics behind CycleGAN is discussed thoroughly along with loss functions and objective functions. Also, the algorithm of the CycleGAN described in section 4.3.

4.1 Proposed Approach

In the below figure 4.1, the architecture of the proposed image-to-image translation application is illustrated. The figure top depicts the flow of transformation of source domain images into target domain images. The bottom figure depicts the flow of transformation of target domain images into source domain images. It is visible that architecture has two generators and two discriminators. The discriminator D_Y takes input $Input_Y$ from the target domain Y and fake image $Generated_Y$ generated by the generator G . The duty of the discriminator D_Y learns to distinguish target domain images from the fake images generated by the G . When the discriminator D_Y determines the image generated by the generator G is fake. It gives feedback or error to generator G through backpropagation to update its weights to produce good quality images. The discriminator D_X takes input $Input_X$ from the source domain X and fake image $Generated_X$ generated by the generator F . The duty of the discriminator D_X learns to distinguish source domain images from fake images generated by the generator F . When the discriminator D_X determines the image generated by generator F is fake. It gives feedback or error to generator F through backpropagation to update its weights to produce good quality images. The transformation flow of synthetic document images into realistic document images and vice versa (marked in green dotted arrows) by minimizing the cycle-consistency loss (marked in green dotted arrows) between the input image and cyclically generated image is illustrated in figure 4.1.

4.2 Cycle-Consistent Adversarial Networks

4.2.1 Formulation

The aim is to learn mapping functions between two domains X and Y . X is the source domain and Y is the target domain. The domain X represents synthetic data distribution and domain Y represents real data distribution. The synthetic data distribution consists of synthetic document images created using empty form templates and handwritten crops. The real data distribution consists of the real document images. For a given training samples $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_j\}_{j=1}^M$ where $y_j \in Y$. The synthetic data distribution represented as $x \sim p_{data}(x)$ and real data distribution represented as $y \sim p_{data}(y)$. The model includes two mappings functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$ as illustrated in figure 4.2, they are the generators. Along with generators, two adversarial discriminators D_X and D_Y are introduced. D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$. In the same way, D_Y aims to discriminate between $\{y\}$ and $\{G(x)\}$. The final objective function contains three loss functions. first, least-square loss[33] is used for matching the distribution of generated images to the data distribution in the target domain. The general GANs uses sigmoid cross-entropy loss function to optimize generator and discriminator. In this thesis generators and discriminators used in CycleGAN are optimized using least-square loss[33] which is opted from LSGANs. Second, The cycle-consistency loss to prevent the learned mappings functions G and F from contradicting each other[1]. The third is identity mapping loss to preserve the color of the input images[1].

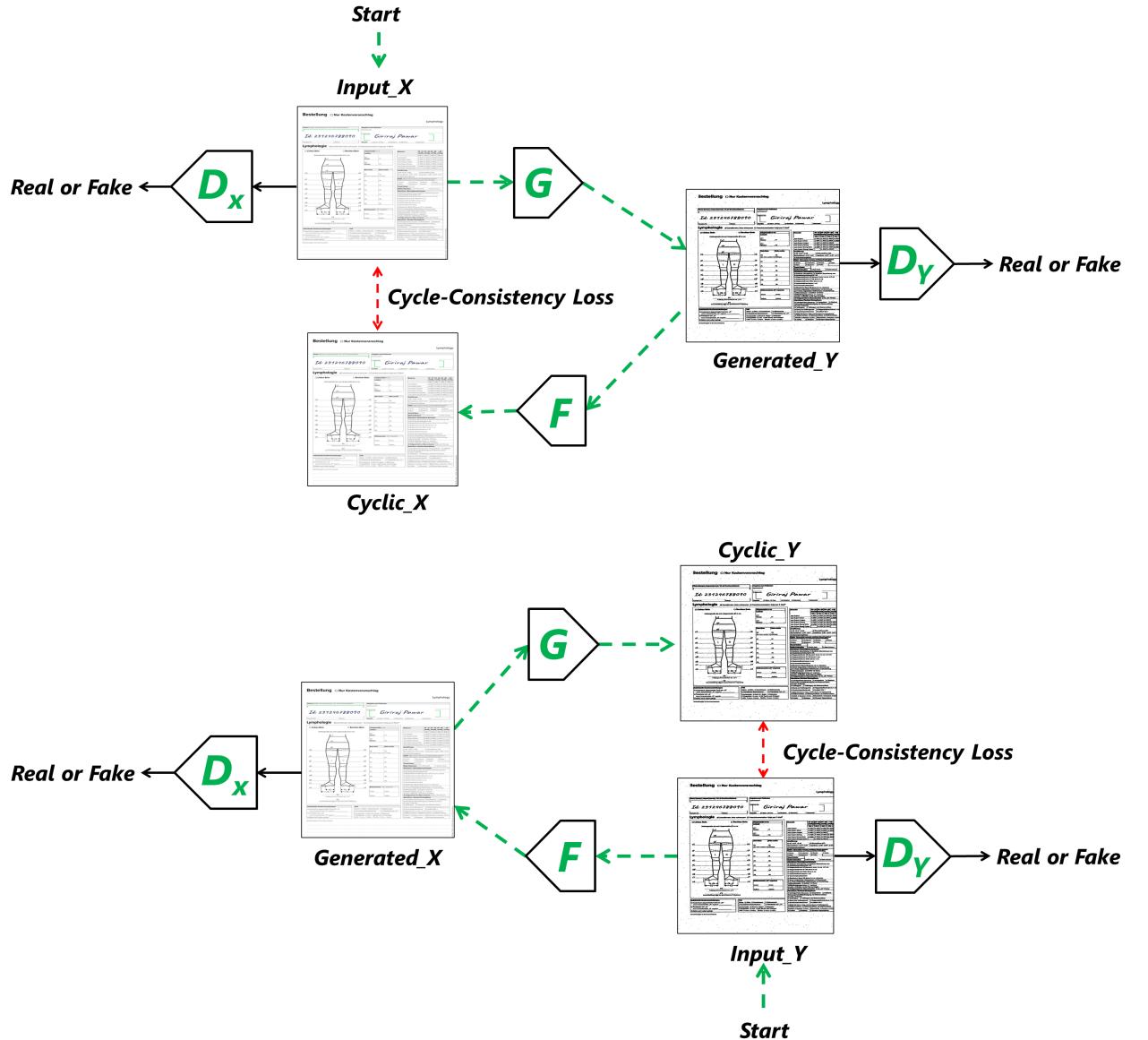


Figure 4.1: Overview of the image-to-image translation application using CycleGAN - It consists of two generators, G and F which map synthetic document images to real document images and real document images to synthetic document images, respectively using cycle-consistency loss[1]. It also contains two discriminators D_X and D_Y which act as the adversary and reject images generated by generators.

4.2.2 Least-Square Loss

In GANs the discriminator is a binary classifier that adopts the sigmoid cross-entropy loss function. As stated in section 2.2, while updating the generator, the sigmoid cross-entropy loss function causes the vanishing gradients problem for the samples that are on the correct side of the decision boundary but are still far from the real data. Also, the sigmoid cross-entropy loss function causes difficulty to stabilize the model training procedure[33]. First, \mathcal{L}_{GAN} (equation 3.1), the negative log-likelihood objective replaced by a least-squares loss functions[33]. The least-squares loss is more stable during training and generates higher quality results[33]. It is used to optimize the generator and discriminator adversarially. In the below equations, we have used the a, b , and c for the coding scheme for the equations of the discriminator. Where a and b are the labels for fake data and real data, respectively. c denotes the value that G wants D to believe for fake data. Basically, $a = 0$, $b = 1$, and $c = 1$. That means fake data is represented by 0, and real data is represented by 1. Then the modified objective functions using least-squares loss can be defined as follows:

$$\min_D \mathcal{L}_{ls}(D_Y) = \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_Y(y) - b)^2 + \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)) - a)^2 \quad (4.1)$$

$$\min_G \mathcal{L}_{ls}(G) = \mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)) - c)^2 \quad (4.2)$$

$$\mathcal{L}_{ls}(G, D_Y, X, Y) = \min_D \mathcal{L}_{ls}(D_Y) + \min_G \mathcal{L}_{ls}(G) \quad (4.3)$$

$$\min_D \mathcal{L}_{ls}(D_X) = \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_X(x) - b)^2 + \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - a)^2 \quad (4.4)$$

$$\min_F \mathcal{L}_{ls}(F) = \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - c)^2 \quad (4.5)$$

$$\mathcal{L}_{ls}(F, D_X, Y, X) = \min_D \mathcal{L}_{ls}(D_X) + \min_F \mathcal{L}_{ls}(F) \quad (4.6)$$

4.2.3 Cycle-Consistency Loss

Theoretically, adversarial training can learn mappings that produce outputs identically distributed as target domains Y and X , respectively. However, with a large amount of capacity, a neural network maps the same set of input images to any random permutations of images in the target domain, where any of the learned mappings can produce an output distribution that matches the target distribution. Hence, depending just on adversarial losses alone cannot promise that the learned mapping function can map an individual input from the source domain x_i to a desired output y_j in the target domain. To further narrow down and reduce the space of possible outcomes using mapping functions, we can attempt to argue that learned mapping functions should be cycle-consistent. The idea of being cycle-consistent can be understood by a simple example for translation a sentence in languages. For example, if an English sentence is translated into German and later from German to English, the outcome should be an original English sentence from where we have started. The figure 4.2, (a) CycleGAN model with two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and respective adversarial discriminators D_Y and D_X is illustrated. (b) for each image x from source domain X , the image translation cycle should be able to bring x back to the original image, i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. This is called forward cycle-consistency. (c) for each image y from

target domain Y , the image translation cycle should be able to bring y back to the original image, i.e., $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. This is called backward cycle-consistency. G and F both should satisfy cycle-consistency. We describe this behavior using a cycle-consistency loss in the form of a mathematical equation 4.7.

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}(F(G(x)) - x) + \mathbb{E}_{y \sim p_{data}(y)}(G(F(y)) - y) \quad (4.7)$$

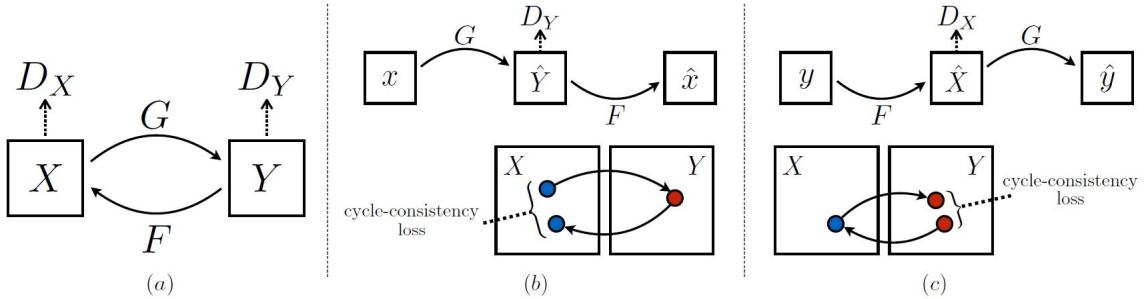


Figure 4.2: Illustration of CycleGAN model with its mapping functions, respective discriminators, and forward and backward consistency loss[1].

4.2.4 Identity Mapping Loss

It's beneficial to introduce an additional loss called identity mapping loss, to encourage the mapping to preserve color composition between the input and output[72]. The identity mapping loss regularizes the generator G to be near an identity mapping when real samples of the target domain are provided as the input to the generator G . Also, the identity mapping loss regularizes the generator F to be near an identity mapping when synthetic samples of the source domain are provided as the input to the generator F . It means when we provide an input to the generator from the same domain in which it is transforming into, the output should be identical to the input. We describe identity mapping loss in the form of a mathematical equation 4.8.

$$\mathcal{L}_{ide}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}(F(y) - y) + \mathbb{E}_{x \sim p_{data}(x)}(G(x) - x) \quad (4.8)$$

4.2.5 Complete Objective Function

Let's combine 4.6, 4.7, and 4.8 to form final objective function. This objective function contains terms of least-squares loss, cycle-consistency loss, and identity mapping loss. The objective that we have to achieve is given in the equation 4.9. In equation 4.9, the terms λ_{ide} and λ_{cyc} control the relative importance of the term in the final objective function.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{ls}(G, D_Y, X, Y) + \mathcal{L}_{ls}(F, D_X, Y, X) + \lambda_{ide} \mathcal{L}_{ide}(G, F) + \lambda_{cyc} \mathcal{L}_{cyc}(G, F) \quad (4.9)$$

4.3 CycleGAN Training Algorithm

```

1 for each training iteration do
2   Draw a random sample  $x_i$  from source domain  $X$  where  $\{x_i\}_{i=1}^N$  and  $x_i \in X$ 
3   Draw a random sample  $y_j$  from target domain  $Y$  where  $\{y_j\}_{j=1}^M$  and  $y_j \in Y$ 
4   1. Train the Discriminators:
5     a) Compute the discriminator loss on real and fake images:
6       
$$\mathcal{L}_{real,fake}^{(D_X)} = \frac{1}{2} (D_X(x_i) - 1)^2 + \frac{1}{2} (D_X(F(y_j)))^2$$

7       
$$\mathcal{L}_{real,fake}^{(D_Y)} = \frac{1}{2} (D_Y(y_j) - 1)^2 + \frac{1}{2} (D_Y(G(x_i)))^2$$

8     c) Update the discriminators
9   2. Train the Generators:
10    a) Compute the cycle-consistency losses:
11      
$$\mathcal{L}_{cyc}^{(X \rightarrow Y \rightarrow X)} = F(G(x_i)) - x_i$$

12      
$$\mathcal{L}_{cyc}^{(Y \rightarrow X \rightarrow Y)} = G(F(y_j)) - y_j$$

13    b) Compute the identity mapping losses:
14      
$$\mathcal{L}_{ide}^{(X \rightarrow X)} = G(x_i) - x_i$$

15      
$$\mathcal{L}_{ide}^{(Y \rightarrow Y)} = F(y_j) - y_j$$

16    c) Compute the Generator  $G$  loss:
17      
$$\mathcal{L}_G = (D_Y(G(x_i)) - 1)^2 + \mathcal{L}_{cyc}^{(Y \rightarrow X \rightarrow Y)} + \mathcal{L}_{ide}^{(Y \rightarrow Y)}$$

18      Compute the Generator  $F$  loss:
19      
$$\mathcal{L}_F = (D_X(F(y_j)) - 1)^2 + \mathcal{L}_{cyc}^{(X \rightarrow Y \rightarrow X)} + \mathcal{L}_{ide}^{(X \rightarrow X)}$$

20    d) Update the generators
21 end

```

Algorithm 2: CycleGAN training algorithm.

5. Implementation

This chapter discusses the implementation of the CycleGAN and Classifiers. In this thesis, classifiers are used to determine the domain gap between the distributions. Also, they are used to evaluate the quality of images generated by the CycleGAN. How the classifiers are being used to analyze the domain gap between distributions will be discussed in-depth in chapter Evaluation 6. In section 5.1 dataset preparation is described. The architecture of CycleGAN and Classifiers discussed in section 5.2. The training details of CycleGAN and Classifiers described in 5.3. The experiments are visualized using Tensorboard¹. TensorBoard is a tool that provides the visualization needed for machine learning research and experiments. The neural networks implemented in this thesis using Python, Keras APIs, and TensorFlow library[73]. The reference code for the CycleGAN is available here. All of the neural networks are trained upon GPUs (Graphics Processing Units) like Nvidia Tesla T4 and Tesla V100-SXM2.

5.1 Dataset Preparation

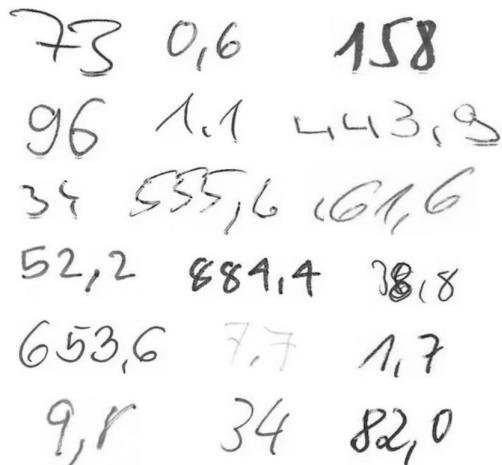


Figure 5.1: Examples of handwritting crops from the handwritting number dataset (figure reproduced from elevait GmbH & Co. KG with permission).

The dataset preparation is one of the vital aspects of training any neural network. Bad quality data leads to a poor generalization of neural networks. There are ten types of documents that were considered to work with this image-to-image translation application. Hence, the CycleGAN is trained using a stash of synthetic document images and real document images. Around 100,000 synthetic document images in the source domain and the same number of real document images in the target domain. The synthetic document images are generated using templates (figure A.8) and handwritten crops (figure 5.1). The process of inserting handwritten crops on empty templates can be visualized in figure 5.2. Each template is filled with the help of provided bounding box annotations [74]. For each class of template 10,000, synthetic document images are created. As mentioned earlier, 100,000 synthetic document images are created in total. The same created 100,000 synthetic document images are used while training CycleGAN. Just they are stashed at the same location collectively. The created 100,000 synthetic document images were also faxified and a faxified dataset of 100,000 images is created. It has the same structure as synthetic document images, 10 classes, and each class has 10,000 images. The faxification process uses several image transformations to make a clean

¹<https://www.tensorflow.org/tensorboard> last access: September 3, 2021

gray-scale image look like it was sent via fax. A sample faxified image can be seen in figure A.10. The faxification process is described briefly in section 6.2.5. Also, the faxification can be visualized in figure 6.9. In the table 5.1 the number of samples in each dataset is mentioned. For testing, around 1162 annotated real document images are used. This testing dataset is used to evaluate the performance of the classifiers trained upon different data distributions like synthetic document images, faxified document images, and CycleGAN generated document images. Basically, testing dataset is significant to understand the domain gap between real data distribution and remaining data distributions. In the table 5.2 the number of samples in each class in the test dataset is mentioned. The testing dataset is unbalanced. The datasets used in this thesis can not be cited or published because they are not open for public use.

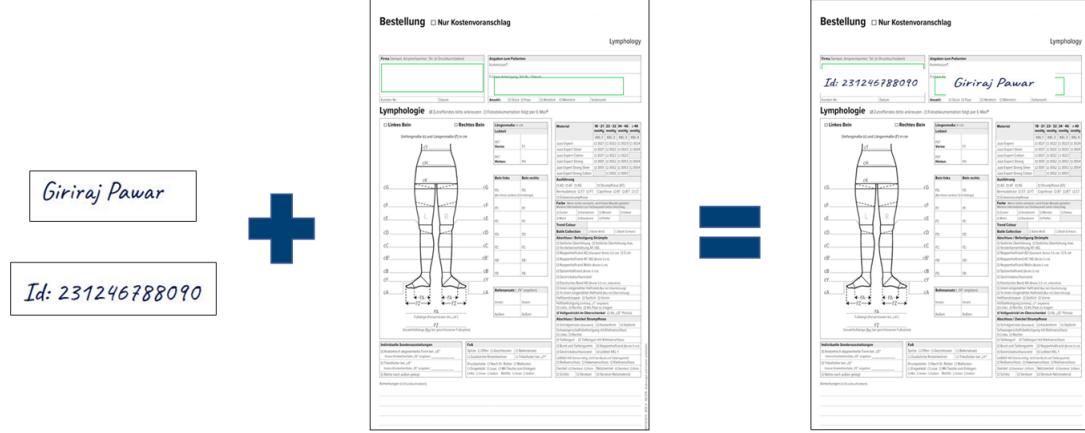


Figure 5.2: Inserting handwritten crops on empty form templates (figure reproduced from elevait GmbH & Co. KG with permission).

The templates are unfilled form images. As described before, in this thesis, ten classes of templates are chosen for creating document images datasets like synthetic and faxified document images datasets. The 8 out of 10 templates are a type of “Bein”. These templates are very similar to each other apart from minor differences. Examples of “Bein” templates are shown in figure 5.3. The other two classes of templates are of type “Hand” and “Arm”. In figure 5.4, (b) is the template of “Bein”, (a) is the template of “Arm”, and (c) is the template of “Hand”. The list of classes or templates is listed in the table 5.2.

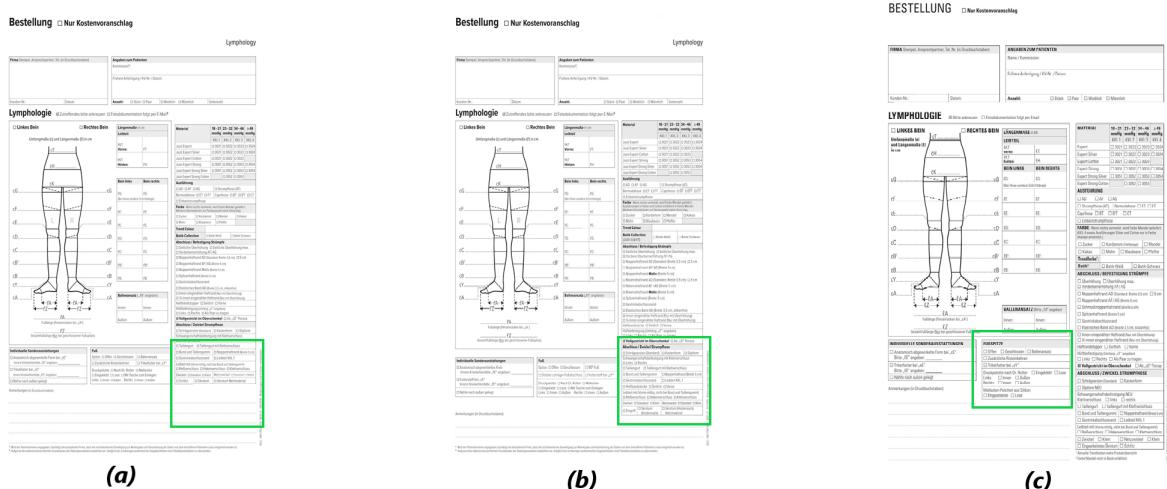


Figure 5.3: Templates with minor differences (figure reproduced from elevait GmbH & Co. KG with permission).

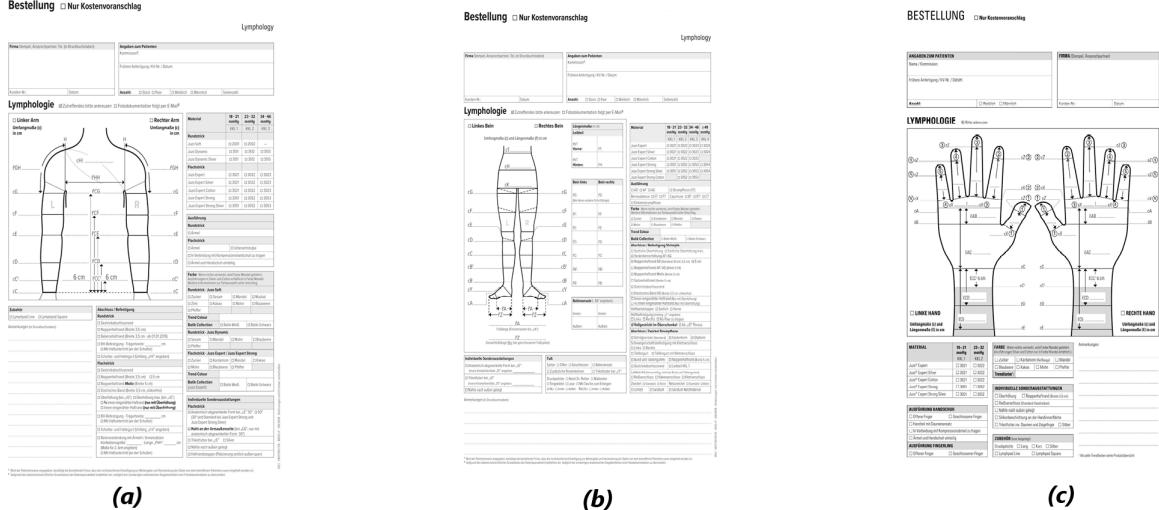


Figure 5.4: Templates with major differences (figure reproduced from elevait GmbH & Co. KG with permission).

Datasets	Size (Number of Images)
Synthetic Document Images	100,000
Real Document Images	100,000
Faxified Document Images	100,000
Annotated Real Document Images (Used for testing)	1162

Table 5.1: Size of datasets used for training CycleGAN and classifiers.

Classes	Size (Number of Images)
DE_LY_Arm_2020-01	44
DE_LY_Bein_2018-08	47
DE_LY_Bein_2019-01	50
DE_LY_Bein_2019-07	60
DE_LY_Bein_2020-01	624
DE_LY_Bein_2020-03	128
DE_LY_Hand_2020-01	16
DE_PH_Bein_2018-09	22
DE_PH_Bein_2019-02	28
DE_PH_Bein_2020-01	143

Table 5.2: Number of images in each class of annotated real document images dataset (testing dataset).

5.2 Network Architecture

5.2.1 CycleGAN

Johnson et al.[19] proposed the architecture of CycleGAN. In which the generator has three sequences of blocks one is downsampling, transformation, and upsampling. The sequence of 2 down-sampling convolutional blocks encode the $256 \times 256 \times 1$ grayscale input image, 9 Residual Network (ResNet) convolutional blocks to transform the image, and 2 upsampling convolutional blocks to generate the output image of the same dimension as the input image. The reason behind using residual blocks is it resolves the vanishing gradient problem in deep neural networks. The discriminator classifier network is designed using PatchGAN architecture [39] [44]. The PatchGAN

discriminator is simply a CNN. The major difference between the PatchGAN discriminator and General GAN discriminator is, GAN discriminator maps input image to the scalar output, which represents image being real to fake. But, the PatchGAN discriminator maps the input image to $N \times N$ array of outputs, where each element in an output array represent a patch in an input image being real or fake. Basically, the PatchGAN discriminator penalizes structure at the scale of local image patches and attempts to classify if each $M \times M$ patch in an image is real or fake.

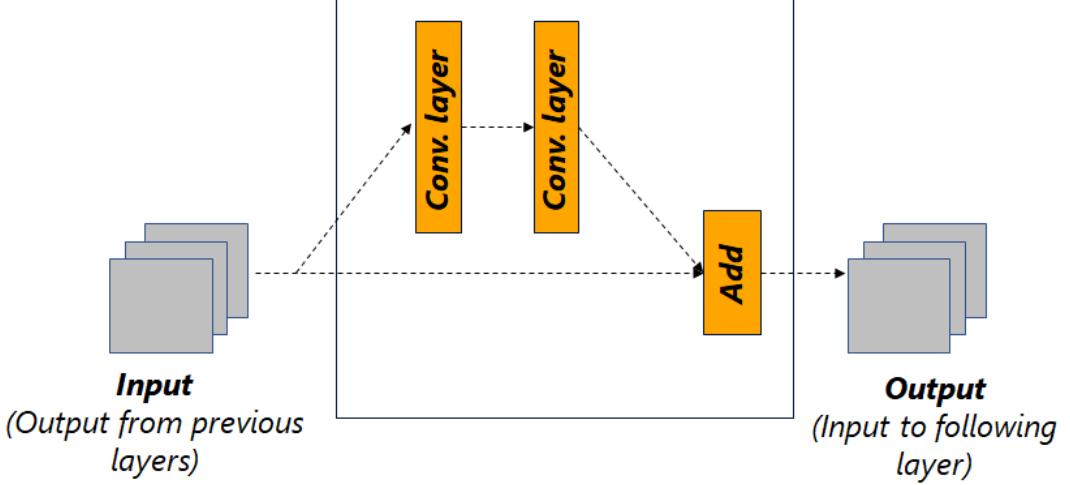


Figure 5.5: Illustration of ResNet blocks in CycleGAN generator architecture.

Johnson et al. [19] have provided naming conventions to define the architecture of generator and discriminator used in CycleGAN. $c7s1-k$ denotes a 7×7 Convolution-InstanceNormlization-ReLU layer with k filters and stride 1. The downsampling block d_k is denoted by a 3×3 Convolution-InstanceNormlization-ReLU layer with k filters and stride 2. To reduce artifacts reflection padding is used. R_k denotes a single residual block that has two 3×3 convolution layers with the same number of filters k on both layers and stride 1. The upsampling block u_k denoted a 3×3 TransposedConvolution-InstanceNormlization-ReLU layer with k filters and stride 2. The complete generator network with 9 residual blocks can be described as: $c7s1-64$, $d128$, $d256$, $R256$, $u128$, $u64$, $c7s1-1$. The last layer $c7s1-1$ denotes a 7×7 Convolution layer with 1 filter and stride 1. Next, the final output is followed by a tanh activation function (figure 3.11). All of the layers in the generator can be seen in figure A.12. The architecture of the generator is illustrated in table 5.3. Also, in the figure 5.5 ResNet blocks in the generator network illustrated. In which, at every ResNet block, the output from the previous layer is passed through ResNet convolution layers. Further, it concatenated with the ResNet convolution layer's output and forwarded to the following layers.

Operation Layer	Number of Filters/Units	Size of Each Filter	Stride Value
Input Image ($256 \times 256 \times 1$)	-	-	-
Convolution Layer Instance Normalization ReLU	64	7×7	1×1
Convolution Layer Instance Normalization ReLU	128	3×3	2×2
Convolution Layer Instance Normalization ReLU	256	3×3	2×2
9 Residual Blocks			
2 Convolution Layers <i>(each with)</i> Instance Normalization ReLU	256	3×3	1×1
Transposed Convolution Layer Instance Normalization ReLU	128	3×3	2×2
Transposed Convolution Layer Instance Normalization ReLU	64	3×3	2×2
Convolution Layer tanh	1	7×7	1×1
Output ($256 \times 256 \times 1$)	-	-	-

Table 5.3: Generator architecture

The discriminator uses 70×70 PatchGAN classifier architecture [39]. It is also called a Markovian discriminator [44]. The L1 and L2 loss functions produce blurry results while solving image generation problems [75]. These losses fail to encourage high-frequency crispness. To model high frequencies, more attention is given to the structure in local image patches [39]. Therefore Markovian discriminator is called the PatchGAN discriminator. C_k denotes a 4×4 Convolution-InstanceNormalization-LeakyReLU layer with k filters and stride 2. Leaky ReLUs with a slope of 0.2 are used. Instance Normalization is not used for the first C_{64} layer. After the last layer C_{512} , the convolution operation is applied with filter 1 to produce an output of depth 1 using 4×4 kernel and stride 1. The discriminator network can be described as: $C_{64}-C_{128}-C_{256}-C_{512}-C_{512}-C_1$. All of the layers in the discriminator can be seen in figure A.18. The architecture of the classifier is illustrated in table 5.4.

Operation Layer	Number of Filters/Units	Size of Each Filter	Stride Value
Input Image ($256 \times 256 \times 1$)	-	-	-
Convolution Layer Instance Normalization LeakyReLU (0.2)	64	4×4	2×2
Convolution Layer Instance Normalization LeakyReLU (0.2)	128	4×4	2×2
Convolution Layer Instance Normalization LeakyReLU (0.2)	256	4×4	2×2
Convolution Layer Instance Normalization LeakyReLU (0.2)	512	4×4	1×1
Convolution Layer Instance Normalization LeakyReLU (0.2)	512	4×4	1×1
Output ($16 \times 16 \times 1$)	-	-	-

Table 5.4: Discriminator architecture

For more information on the architectures of generators and discriminators, Github repository² can be referred to.

5.2.2 Classifier

In this thesis, the classifiers are used to analyze the domain gap between real data distribution and other data distributions like synthetic data distribution, faxified data distribution, and CycleGAN generated data distribution. Three separate classifiers are trained upon synthetic data distribution falsified data distribution, and CycleGAN generated data distribution respectively. Their classification performance was evaluated on annotated real document images using metrics like weighted f1 score and accuracy to investigate the domain gap between real data distribution and mentioned three data distributions. In the chapter, Evaluation 6 we will discuss this more. The classifier architecture is simplistic and easy to implement. It has just two convolution layers, one max-pooling layer, and one dropout layer. The architecture of the classifier is illustrated in table 5.5. Also, all the layers in the classifier can be seen in figure A.11.

²<https://github.com/junyanz/CycleGAN> last access: September 3, 2021

Operation Layer	Number of Filters/Units	Size of Each Filter	Stride Value
Input Image ($256 \times 256 \times 1$)	-	-	-
Convolution Layer ReLU	32	3×3	1×1
Convolution Layer ReLU	64	3×3	1×1
Max Pooling Layer	-	2×2	2×2
Dropout Layer (0.25)	-	-	-
Flatten Layer	-	-	-
Dense Layer ReLU	128	-	-
Dropout Layer (0.5)	-	-	-
Dense Layer Softmax	10	-	-
Output (1×10)	-	-	-

Table 5.5: Classifier architecture

5.3 Training Details

5.3.1 CycleGAN

One of the major challenges of the implementation part was designing an efficient input pipeline. The developed image-to-image translation application and classifiers are trained using 100,000 images. Loading such a large dataset is a tedious and time-consuming job but TensorFlow has provided wonderful APIs like `tf.data` to load large datasets spontaneously. To learn more about how to load large datasets efficiently in TensorFlow refer to this Tutorial. The CycleGAN model has two generators and two discriminators. As already described in Chapter Methodology 4, the CycleGAN model has two generators and two discriminators. If the source domain is X and the target domain is Y . The first generator G is called the forward generator, which transforms images from $X \rightarrow Y$, and D_Y acts as a discriminator for the generator G . The second generator F is called the backward generator, which transforms images from $Y \rightarrow X$, and D_X acts as a discriminator for the generator F . Now to stabilize the CycleGAN model training procedure, which means training both generators G and F with the help of discriminators D_Y and D_X by reduced oscillations. The general GAN equation 3.1 has to modified. In which, the negative log-likelihood objective is replaced by least-squares loss [33]. Hence for the GAN loss $\mathcal{L}_{ls}(G, D_Y, X, Y)$, forward generator G is trained to minimize $\mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)) - 1)^2$ and discriminator D_Y is trained to minimize $\frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_Y(y) - 1)^2 + \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)))^2$. Similarly, for the GAN loss $\mathcal{L}_{ls}(F, D_X, Y, X)$, backward generator F is trained to minimize $\mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - 1)^2$ and discriminator D_X is trained to minimize $\frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_X(x) - 1)^2 + \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)))^2$.

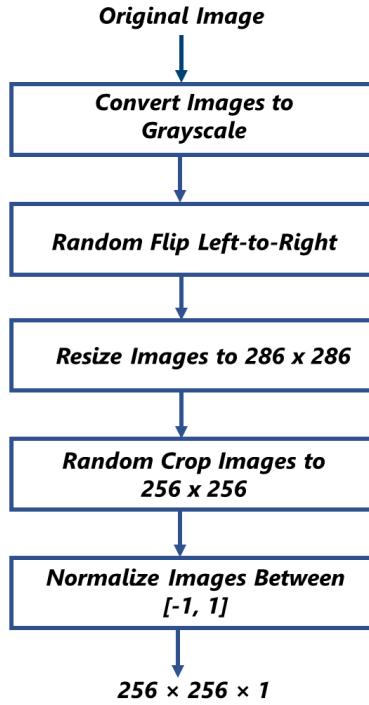


Figure 5.6: Steps involved in preprocessing of training images of CycleGAN.

The CycleGAN trained with a learning rate of 0.0002. The weights of all the neural networks(generators and discriminators) in the CycleGAN model are initialized by a Gaussian distribution with mean (μ) 0 and standard deviation (σ) 0.02. For all experiments in equation 4.9, λ_{cyc} is set to 10 and $\lambda_{identity}$ is set to 0.5 which mean importance of cycle-consistency loss is important in the final objective function. The weights of the neural networks of CycleGAN model are optimized by ADAM solver, which is a method for stochastic optimization[76]. Also while constructing generators and discriminators we have used instance normalization layers [77], the initializer for the gamma weight is initialized by Gaussian distribution with mean (μ) 0 and standard deviation (σ) 0.02. The CycleGAN is trained for 20 epochs and all the individual models like Forward Generator G , Backward Generator F , Discriminator D_Y , and Discriminator D_X . The training checkpoints³ are saved at the end of every epoch. The stack of 100,000 synthetic document images (Domain X) and 100,000 real document images (Domain Y) are used to train the complete CycleGAN model. The training images are preprocessed. The preprocessing process is illustrated in figure 5.6. Initially, images are converted into grayscale. Next, random mirroring is applied, in which the image is randomly flipped horizontally from left to right. Next random mirroring is applied, in which the image is resized to 286×286 and then randomly cropped to 256×256 . Random jittering and mirroring are image augmentation techniques that avoid overfitting [1]. Lastly, the images are normalized in the range of $[-1, 1]$. All the classifiers are trained for 10 epochs.

5.3.2 Classifier

As described earlier, in this thesis three separate classifiers are trained, first on synthetic document images, second on faxified document images, and third on CycleGAN generated document images. The synthetic and faxified document images are preprocessed. The preprocessing process of these images is illustrated in figure 5.7. Initially, images are converted into grayscale. Next, resized to 256×256 and later normalized between $[-1, 1]$. The CycleGAN generated document images are need not be preprocessed, as the images generated by the generator G ($G : X \rightarrow Y$) are of 256×256 dimension and already normalized between $[-1, 1]$. Because at the final layer of the generator, tanh activation function is used, this can be seen in the architecture of the generator in table 5.3.

³<https://www.tensorflow.org/guide/checkpoint> last access: September 3, 2021

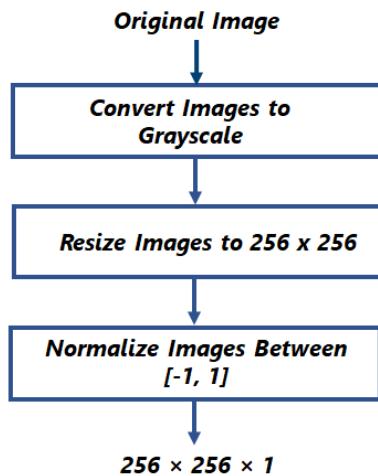


Figure 5.7: Steps involved in preprocessing of training images of Classifiers.

6. Experiments and Evaluation

In this chapter, the experiments conducted in this thesis are described along with evaluation metrics and results. In section 6.1 the evaluation metrics like accuracy, precision, recall, confusion matrix, and f1-score are explained. These metrics are used to evaluate the domain gap between the distributions and quality of CycleGAN generated document images. In section 6.2 all the experiments that are conducted in this thesis are explained along with training plots, confusion matrices, and classification reports. Finally, in section 6.3, quantitative and qualitative results are discussed thoroughly.

6.1 Evaluation Metrics

In machine learning research, numerous performance metrics have been introduced for neural networks. Each of which evaluates different aspects of neural network's performance. Hence, we need a specific set of performance metrics for a particular problem solved using neural networks. It's important to evaluate the performance of the neural network after training, using testing data, to determine its actual performance or generalization error on unseen data. Hence, the performance of classifiers trained upon different data distributions determined using annotated real document images (test dataset). The popular classifier performance evaluation metrics are accuracy, precision, recall, confusion matrix, and f1-score [78]. The testing dataset used to evaluate the classifiers is unbalanced, hence metrics like weighted average and macro average f1-scores are essential for the performance comparison of the classifiers trained on different data distributions.

The accuracy is the most common metric used to evaluate classifiers. It is the ratio of accurately classified data items to the total number of observations (equation 6.3). The precision is "how many selected items are relevant". "To put it another way, out of the observations that an algorithm has predicted to be positive, how many of them are actually positive" [79]. The precision is the ratio of the number of true positives divided by the sum of the true positive and false positives (equations 6.2). The recall is "how many relevant items are selected". "In fact, out of the observations that are actually positive, how many of them have been predicted by the algorithm" [79]. The recall is the ratio of the number of true positives divided by the sum of true positives and false negatives. In below equations 6.3, 6.2, and 6.1 the TP means true positives, TN means true negatives, FP means false positives, and FN means false negatives.

$$Recall = \frac{TP}{TP + FN} \quad (6.1)$$

$$Precision = \frac{TP}{TP + FP} \quad (6.2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6.3)$$

f1-score is the harmonic mean of the precision and recall (equation 6.4). The weighted average f1-score is determined by first calculating the f1-score of each class separately and each multiplied by the weight (the number of true instances for each class) and finally added together, hence favoring the majority class. The equation 6.6 represents the weighted average f1-score. The macro average

f1-score computes the unweighted mean of separate f1-score of each class. The macro average f1-score does not take label imbalance into account. This leads to bigger penalization when the classifier does not perform well on minority classes. The equation 6.5 represents the macro average f1-score. In equations 6.6 and 6.5, N represents number of classes.

More information about accuracy, precision, recall, and f1-score can be found here¹.

$$f1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (6.4)$$

$$\text{Macro average } f1\text{-score} = \frac{f1_{\text{class1}} + f1_{\text{class2}} + \dots + f1_{\text{classN}}}{N} \quad (6.5)$$

$$\text{Weighted average } f1\text{-score} = \frac{f1_{\text{class1}} \times W_1 + f1_{\text{class2}} \times W_2 + \dots + f1_{\text{classN}} \times W_N}{N} \quad (6.6)$$

The confusion matrix is the most intuitive metric to determine the accuracy of the classifiers. It is useful when the classifier has to classify more than two classes. A confusion matrix is a table that describes how well a classifier performs on a test dataset that is labeled or annotated. The instances of the true class are represented at each row of the confusion matrix whereas the instances of predicted class probabilities are represented by each column or vice versa. Here, the confusion matrix is extensively used to analyze the performance of the classifiers trained on different data distributions. More information about the confusion matrix can be found here².

6.2 Experiments

In this thesis, several experiments were performed to understand the domain gap between data distributions. Three data distributions are considered for the experiments. Each data distribution represents a different domain. The synthetic document images represent the synthetic data distribution. The faxified document images represent faxified data distribution. The CycleGAN generated images represent CycleGAN generated data distribution. The goal of the experiments is to illustrate and analyze the domain gap between the real data distribution and above mentioned three data distributions. Also, one of the experiments was performed to determine the quality of CycleGAN generated document images compared to the real document images. Now let's begin with the description of the experiments. First, the classifier(table 5.5) is trained on synthetic document images and its performance is evaluated on annotated real document images. Second, the new classifier with the same architecture is trained on faxified document images and its performance is evaluated on annotated real document images. Third, the CycleGAN is trained using synthetic document images and real document images and at the end of every epoch, the checkpoint is saved. As mentioned in training details 5.3.1, CycleGAN is trained for 20 epochs.

Once training is finished, the latest saved checkpoint can be loaded to generate or transform images. The CycleGAN trained considering synthetic data distribution as a source domain and real data distribution as a target domain. This means the synthetic document images represent the source domain and real document images represent the target domain. Next, the latest checkpoint is loaded, as we only need a generator G to transform images from the source domain to the target domain. Hence, generator G is retrieved to transform 100,000 synthetic document images into 100,000 realistic document images. We call these realistic document images as CycleGAN generated document images which represents CycleGAN generated data distribution. Further, 100,000 CycleGAN generated document images are used to train another new classifier with the same architecture as the previous, and its performance is evaluated on annotated real document images. The experiment aims to

¹https://en.wikipedia.org/wiki/Precision_and_recall last access: September 3, 2021

²https://en.wikipedia.org/wiki/Confusion_matrix last access: September 3, 2021

understand the quality of the images generated by CycleGAN. Especially, how efficiently CycleGAN was able to close the domain gap between synthetic data distribution and real data distribution by generating quality images using generator G in CycleGAN. The evaluation metrics like accuracy, weighted average f1-score, and macro average f1-score [80] are used to understand the contrast between the performance of the classifiers using annotated real document images.

6.2.1 Experiment Steps

1. Train a classifier on synthetic document images and evaluate its classification performance on the annotated real document images.
2. Train a classifier on faxified document images and evaluate its classification performance on the annotated real document images.
3. Train a CycleGAN using synthetic document images and real document images.
4. Generate realistic document images using generator G from the trained CycleGAN model. As mentioned above realistic document images are also called CycleGAN generated document images.
5. Train a classifier on CycleGAN generated document images and evaluate its classification performance on the annotated real document images.
6. Compare the classification performance of the above three classifiers on the annotated real document images, to illustrate the domain gap between real data distribution and other three distributions like synthetic data distribution, faxified data distribution, and CycleGAN generated data distribution. The performance evaluation of a classifier trained upon CycleGAN generated data distribution using annotated real document images describes, how close is the CycleGAN generated data distribution to the real data distribution. Simply this approach determines the quality of the CycleGAN generated document images compared to real document images.

6.2.2 CycleGAN Training

The image-to-image translation application, which is implemented using CycleGAN is trained for 20 epochs and with batch size 1. For more CycleGAN training details refer section 5.3.1. The CycleGAN consists of two generators G and F . Along with two discriminators D_X and D_Y . Domain X represents the source domain and domain Y represents the target domain. The developed image-to-image translation application aims to transform synthetic document images into realistic document images. Hence, the synthetic document images represent the source domain and real document images represent the target domain. The complete architecture of the image-to-image translation application using CycleGAN is illustrated in figure 4.1.

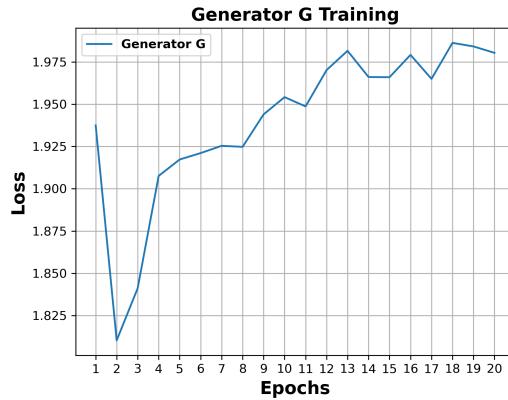


Figure 6.1: CycleGAN generator G training epochs vs loss plot.

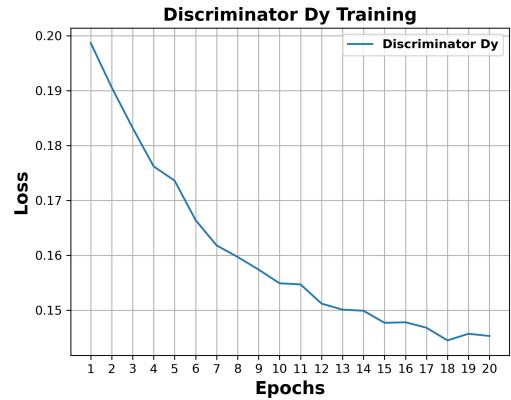


Figure 6.2: CycleGAN discriminator D_Y training epochs vs loss plot.

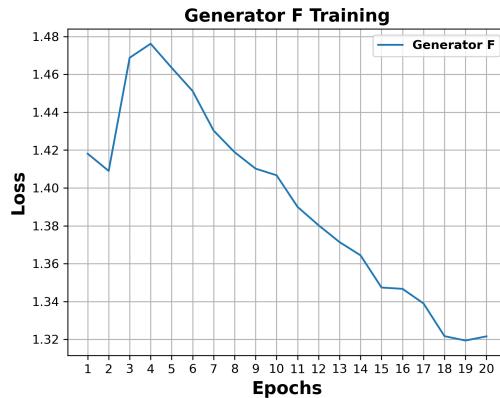


Figure 6.3: CycleGAN generator F training epochs vs loss plot.

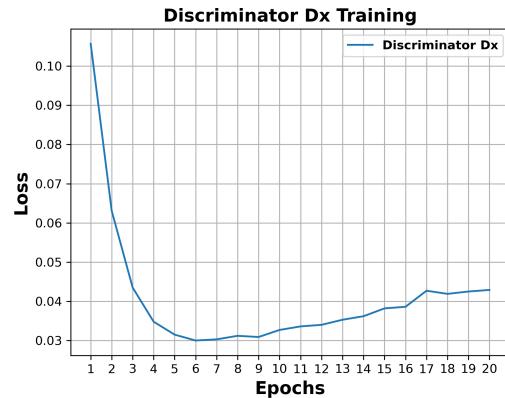


Figure 6.4: CycleGAN discriminator D_X training epochs vs loss plot.

The CycleGAN training happens by calculating loss and updating the weights of discriminators D_X and D_Y , and generators G and F respectively. First, random input images x_i and y_i are retrieved from source domain X and target domain Y . The fake image generated using generator G is *Generated_Y* and the fake image generated using generator by F is *Generated_X*. As we described earlier, G is a mapping function to transform an image from source domain X to target domain Y , mathematically it is $G : X \rightarrow Y$. And F is vice versa $F : Y \rightarrow X$. The generator G is called the forward generator, and F is called the backward generator. First, the discriminator loss for D_X and D_Y on real and fake images is computed. The computed discriminator loss is used to update the weights of both discriminators. Next, cycle-consistency loss is calculated by transforming the source domain image into the target domain, back again into the source domain. It is called forward cycle-consistency loss. It is calculated by determining the difference between the original image in the source domain and cycled image back in the source domain. If you refer to figure 4.1, the forward cycle-consistency loss is the difference between *Input_X* and *Cyclic_X*. The backward cycle-consistency loss is calculated by transforming the target domain image into the source domain, back again into the target domain. In figure 4.1, the backward cycle-consistency loss is the difference between *Input_Y* and *Cyclic_Y*. The cycle-consistency loss forces generating an image that is context instead of generating random images.

Further, identity mapping loss is calculated to preserve the color of images after domain translation using generators G and F . The generator G transforms the image from domain X to the domain Y . When the input y_i is given to generator G from the target domain Y , it should produce an identical image. The identity mapping loss for generator G is the difference between the original image y_i , and the transformed image *Same_Y* when y_i is used as an input for G . And, the identity mapping loss for generator F is the difference between input image x_i and the transformed image *Same_X* when x_i is used as an input for F . The generator loss for generators G and F is calculated by the feedback given by their respective discriminators D_Y and D_X . Combining generator loss, cycle-consistency loss, and identity mapping loss, the total generator loss is calculated separately for generators G and F , and their weights are updated to produce quality images. The epochs against loss plot while training generators G and F are shown in figure 6.1 and 6.3 respectively. Also, the epochs against loss plot while training discriminators D_X and D_Y are shown in figure 6.4 and 6.2 respectively. To understand the algorithm of CycleGAN in-depth, refer to the pseudo-code from 2.

6.2.3 Training a Classifier on Synthetic Document Images

Before this experiment, the dataset of synthetic document images is created using templates (figure A.8) and handwritten crops (figure 5.1) using the process mentioned in figure 5.2. In this experiment, a simple classifier (table 5.5), is trained on synthetic document images dataset. The performance of this classifier is evaluated on annotated real document images (test dataset) to understand the domain gap between real data distribution and synthetic data distribution. The classification report in table 6.1 states that the accuracy of this classifier on real document images is 25%, macro average f1-score is 27%, and weighted average f1-score is 31%. As mentioned earlier, the test dataset is

unbalanced, the weighted average f1-score is a suitable metric in this scenario. The weighted average f1-score concludes that there is a huge gap between real data distribution and synthetic data distribution. The confusion matrix is illustrated in figure A.6. Where it is completely visible most of the test images were classified wrongly. The epochs against accuracy and epochs against loss plots while training classifier on synthetic document images is illustrated in figure 6.5 and 6.6 respectively.

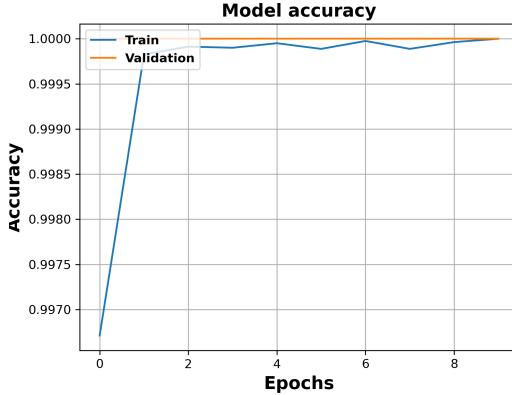


Figure 6.5: Epochs vs accuracy plot while training a classifier on synthetic document images.

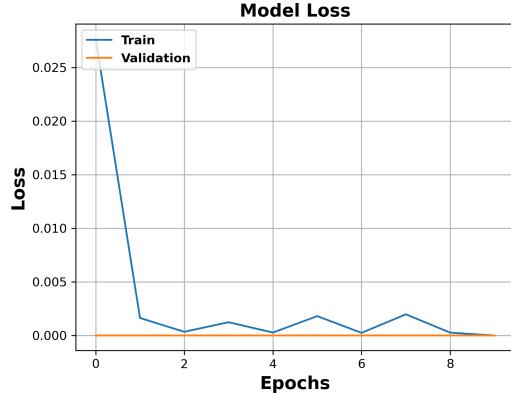


Figure 6.6: Epochs vs loss plot while training a classifier on synthetic document images.

	Precision	Recall	f1-score	Support
DE_LY_Arm_2020-01	0.65	0.50	0.56	44
DE_LY_Bein_2018-08	0.50	0.02	0.04	47
DE_LY_Bein_2019-01	0.06	0.90	0.11	50
DE_LY_Bein_2019-07	0.36	0.20	0.26	60
DE_LY_Bein_2020-01	0.96	0.21	0.34	624
DE_LY_Bein_2020-03	0.24	0.25	0.24	128
DE_LY_Hand_2020-01	0.70	0.44	0.54	16
DE_PH_Bein_2018-09	0.10	0.14	0.12	22
DE_PH_Bein_2019-02	0.12	0.04	0.06	28
DE_PH_Bein_2020-01	0.95	0.51	0.26	143
Accuracy			0.25	1162
Macro average	0.46	0.29	0.27	1162
Weighted average	0.74	0.25	0.31	1162

Table 6.1: Classification report generated after the classifier is trained on synthetic document images, its classification performance evaluated on the annotated real document images.

6.2.4 Training a Classifier on CycleGAN Generated Document Images

This is the most important experiment of this thesis. After the CycleGAN is trained, the generator G is loaded from the saved checkpoint, to perform image-to-image translation, simply to say, transforming synthetic document images to realistic document images. 100,000 synthetic document images are transformed into realistic document images. The transformed 100,000 realistic document images are used to train the same classifier (table 5.5). The performance of this classifier is evaluated on annotated real document images (test dataset) to understand the domain gap between real data distribution and CycleGAN generated data distribution and quality of generated images compared to real images. The classification report in table 6.2 states that the accuracy of this classifier on annotated real document images is 27%, macro average f1-score is 34%, and weighted average f1-score is 25%. The majority of documents are of the type “Bein”, and they are similar

to each other. It has been found, a small artifact transformation failure has caused the wrong classification because learning such minor differences and features present in training datasets by CycleGAN was very challenging. The confusion matrix is illustrated in figure A.7. In the confusion matrix, the document images DE_LY_Arm_2020-01, DE_LY_Bein_2019-07, DE_LY_Hand_2020-01, and DE_PH_Bein_2020-01 are classified satisfactorily. Document images DE_LY_Bein_2020-01 and DE_LY_Bein_2020-03 are wrongly classified as DE_LY_Bein_2019-07. Document images DE_PH_Bein_2018-09 and DE_PH_Bein_2019-02 are wrongly classified as DE_PH_Bein_2019-02. In section 6.3.2, qualitative results of the image-to-image translation are illustrated along with failure cases. The epochs against accuracy and epochs against loss plots while training classifier on CycleGAN generated document images is illustrated in figure 6.7 and 6.8 respectively.

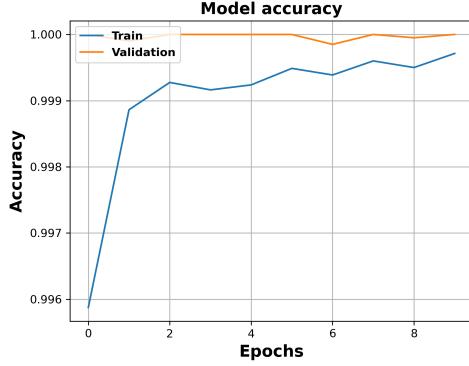


Figure 6.7: Epochs vs accuracy plot while training a classifier on CycleGAN generated document images.

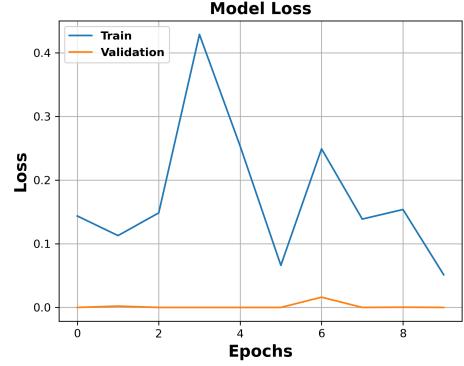


Figure 6.8: Epochs vs loss plot while training a classifier on CycleGAN generated document images.

	Precision	Recall	f1-score	Support
DE_LY_Arm_2020-01	0.53	0.75	0.62	44
DE_LY_Bein_2018-08	0.21	0.28	0.24	47
DE_LY_Bein_2019-01	0.16	0.14	0.15	50
DE_LY_Bein_2019-07	0.11	0.83	0.19	60
DE_LY_Bein_2020-01	0.72	0.07	0.12	624
DE_LY_Bein_2020-03	0.16	0.34	0.22	128
DE_LY_Hand_2020-01	0.80	0.75	0.77	16
DE_PH_Bein_2018-09	0.07	0.05	0.06	22
DE_PH_Bein_2019-02	0.33	0.46	0.39	28
DE_PH_Bein_2020-01	0.70	0.67	0.68	143
Accuracy			0.27	1162
Macro average	0.38	0.43	0.34	1162
Weighted average	0.55	0.27	0.25	1162

Table 6.2: Classification report generated after the classifier is trained on CycleGAN generated document images, its classification performance evaluated on the annotated real document images.

6.2.5 Training a Classifier on Faxified Document Images

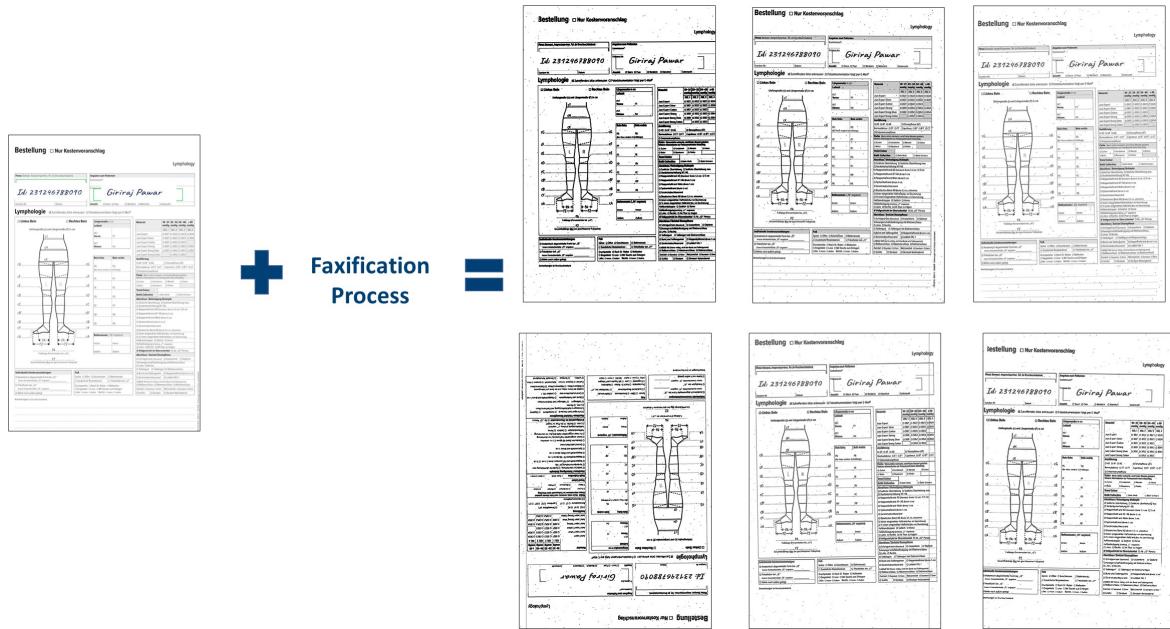


Figure 6.9: Illustration of faxification process applied on synthetic document images (figure reproduced from elevait GmbH & Co. KG with permission).

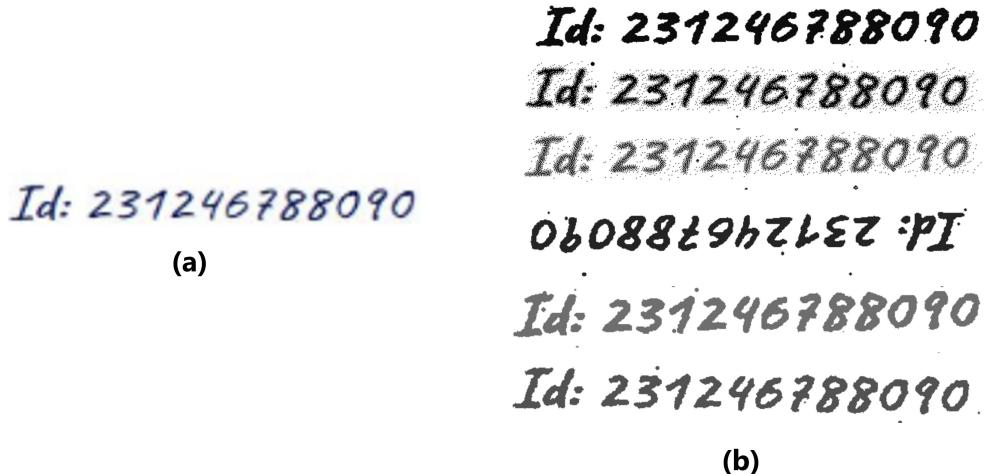


Figure 6.10: Illustration of faxified document images to conclude that faxification process is a random process, the input images are faxified randomly to create distinct and random output. For example, for a snippet in a synthetic document image (a), snippets of faxified document images shown in (b) are created distinct and random.

The faxification process mimics the way the fax machine works. Usually, fax machines transmit only black-and-white images, which might be dirty and are generally also not aligned perfectly. This leads to several common artifacts being introduced into transferred images, so faxification process attempts to mimic those introductions of artifacts into images. The faxification process transforms clean gray-scale synthetic document images in such a way like it was sent via fax. The faxification process is not deterministic, involves randomness during the process of faxification of the images. It uses several image transformations like gamma transformation, brightness transformation, 180-degree rotations, resizing, rescaling, binarization, adding noise, adding vertical lines, and conversion to a grayscale image. The faxification process can be visualized in figure 6.9. In figure 6.10, it is

visible a snippet from the synthetic document image that has transformed randomly into different image transformations when it has been through the faxification process. The faxification process is implemented in Python using a traditional programming approach.

Before this experiment faxified document images dataset is created using synthetic document images and faxification process (figure 6.9). In this experiment, the same classifier (table 5.5), is trained on faxified document images dataset. The performance of this classifier is evaluated on annotated real document images (test dataset) to understand the domain gap between real data distribution and faxified data distribution. The classification report in table 6.3 states that the accuracy of this classifier on annotated real document images is 43%, macro average f1-score is 58%, and weighted average f1-score is 43%. The weighted average f1-score concludes that the faxified data distribution matched more than 40% to the real data distribution. Also, compared to other data distributions like synthetic and CycleGAN generated, the faxified data distribution is more similar to the real data distribution. The confusion matrix is illustrated in figure A.7. In the confusion matrix, it's visible that document images of class DE_LY_Bein_2020-01 are wrongly classified as DE_LY_Bein_2020-03. The rest of the document images are classified satisfactorily. The epochs against accuracy and epochs against loss plots while training classifier on synthetic document images is illustrated in figure 6.11 and 6.12 respectively.

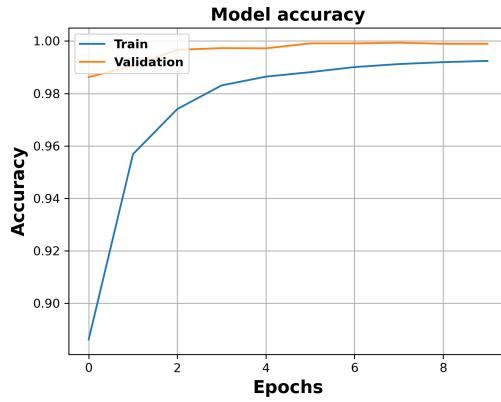


Figure 6.11: Epoch vs Accuracy Plot.

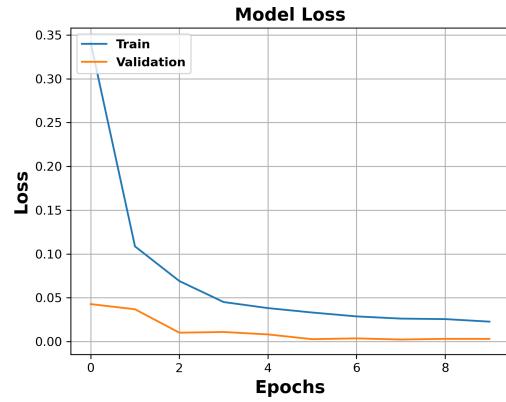


Figure 6.12: Epoch vs Loss Plot.

	Precision	Recall	f1-score	Support
DE_LY_Arm_2020-01	0.97	0.75	0.85	44
DE_LY_Bein_2018-08	1.00	0.53	0.69	47
DE_LY_Bein_2019-01	0.74	0.34	0.47	50
DE_LY_Bein_2019-07	0.53	1.00	0.69	60
DE_LY_Bein_2020-01	1.00	0.17	0.29	624
DE_LY_Bein_2020-03	0.19	0.98	0.32	128
DE_LY_Hand_2020-01	0.21	1.00	0.34	16
DE_PH_Bein_2018-09	0.68	0.68	0.68	22
DE_PH_Bein_2019-02	0.78	0.64	0.71	28
DE_PH_Bein_2020-01	1.00	0.59	0.74	143
Accuracy			0.43	1162
Macro average	0.71	0.67	0.58	1162
Weighted average	0.85	0.43	0.43	1162

Table 6.3: Classification report generated after the classifier is trained on faxified document images, its classification performance evaluated on the annotated real document images.

6.3 Results

In this chapter, the qualitative and quantitative results are discussed. In section 6.3.2, the qualitative results are illustrated to understand the quality of CycleGAN generated document images. Also, the typical failure cases of the proposed method to the problem statement are illustrated in section 6.3.3. In figure 6.14, some images of synthetic document images that are transformed into realistic document images by the proposed image-to-image translation application are illustrated. The figure 6.15 illustrates the proposed image-to-image translation application using CycleGAN has failed to transform handwritten crops present in the synthetic document images. The figure 6.16 illustrates the transformed image has undesired noisy artifacts. Figure 6.17 illustrates transformed image has a dark border and noisy artifacts. Figure 6.18 illustrates transformed image has random artifacts that are not present in the synthetic document image.

In section 6.3.1, the quantitative results are illustrated using a table and bar plot. In which the performance of the classifier trained using different distributions is evaluated and compared. The evaluation metrics like accuracy, macro average f1-score, and weighted average f1-score are used. As the testing dataset of annotated real document images is unbalanced we consider weighted average f1-score for comparison. Results show the CycleGAN generated document images were 25% similar to the real document images. But, the faxified document images were 43% similar to the real document images. Results reveal that CycleGAN could not achieve the expected results quantitatively. The faxified data distribution created using the faxification process was satisfactorily matched to the real data distribution. The qualitative results (figure 6.3.2) look very promising. The proposed image-to-image translation application can be improved further in the future using different methods and loss functions. The future work and discussion about the results achieved in this thesis are explained in the chapter 7 in more detail.

6.3.1 Quantitative Results

Data distributions	Accuracy	Weighted average f1-score	Macro average f1-score
Synthetic document images	25%	31%	27%
CycleGAN generated document images	27%	25%	34%
Faxified document images	43%	43%	58%

Table 6.4: The accuracies and f1-scores when the classifiers trained on different data distributions and evaluated on annotated real document images.

Comparison of accuracy and f1-scores

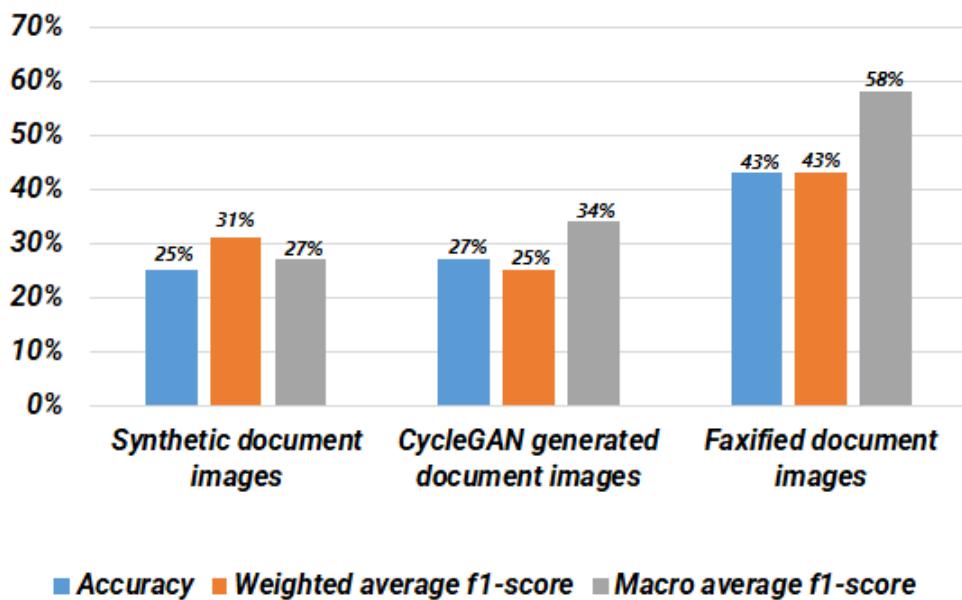
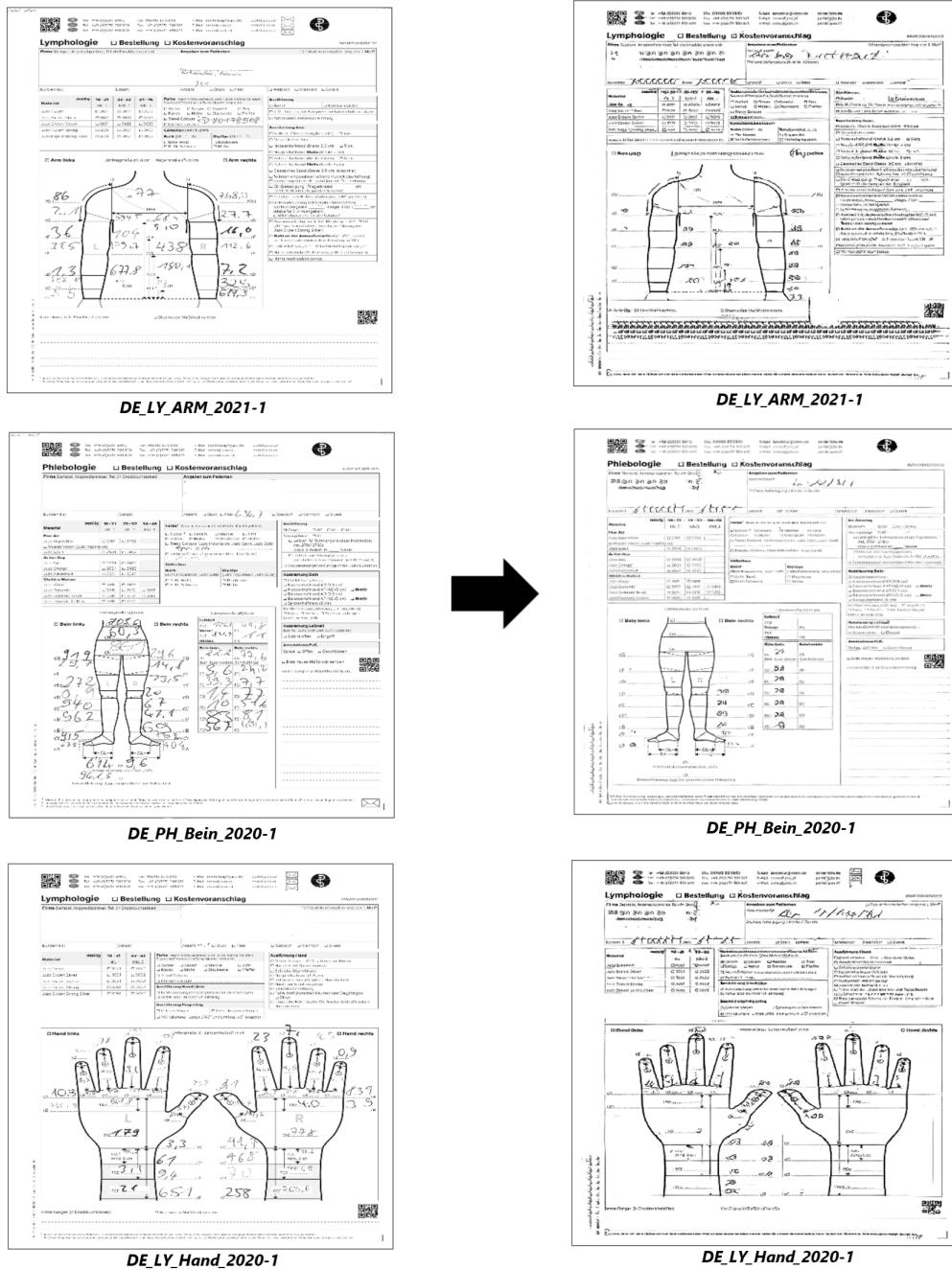


Figure 6.13: Comparison of accuracies and f1-scores, when the classifiers trained on different data distributions and evaluated on annotated real document images.

6.3.2 Qualitative Results



Synthetic document images

CycleGAN generated document images

Figure 6.14: Synthetic document images transformed into realistic document images by the image-to-image translation application (figure reproduced from elevait GmbH & Co. KG with permission).

6.3.3 Failure Cases

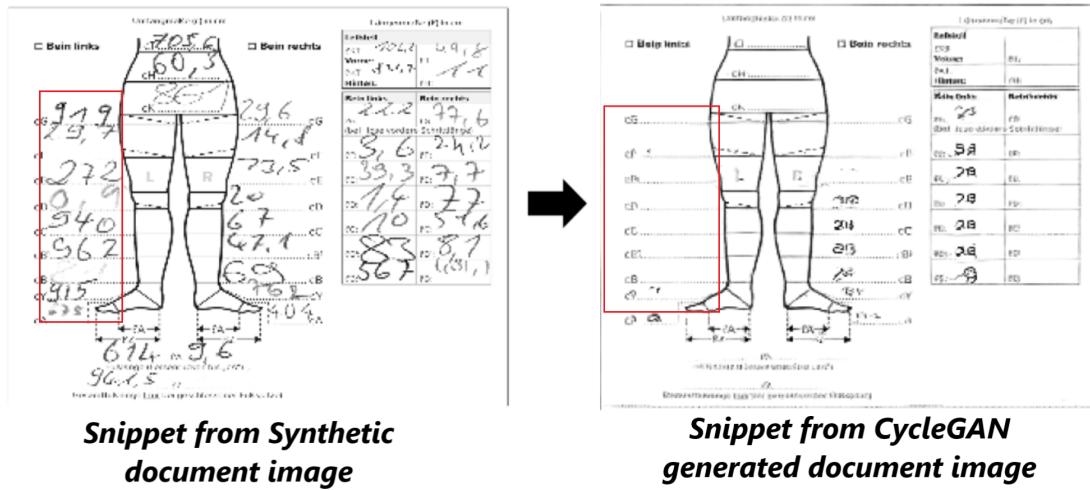


Figure 6.15: The snippet from CycleGAN generated document image illustrates that the proposed image-to-image translation application did not transform or reconstruct handwritten crops in the target domain from the synthetic document image in the source domain (figure reproduced from elevait GmbH & Co. KG with permission).

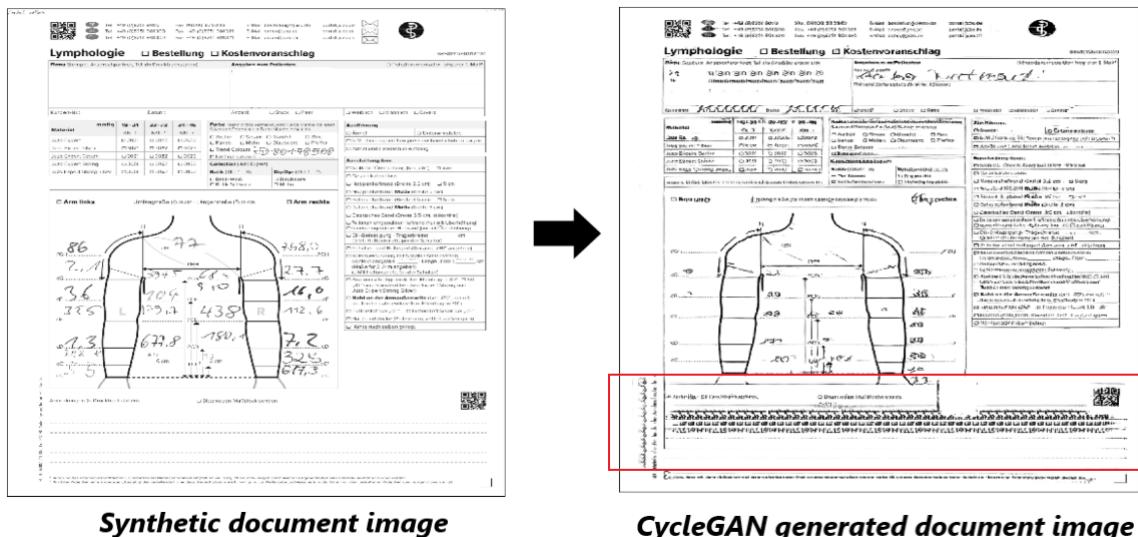
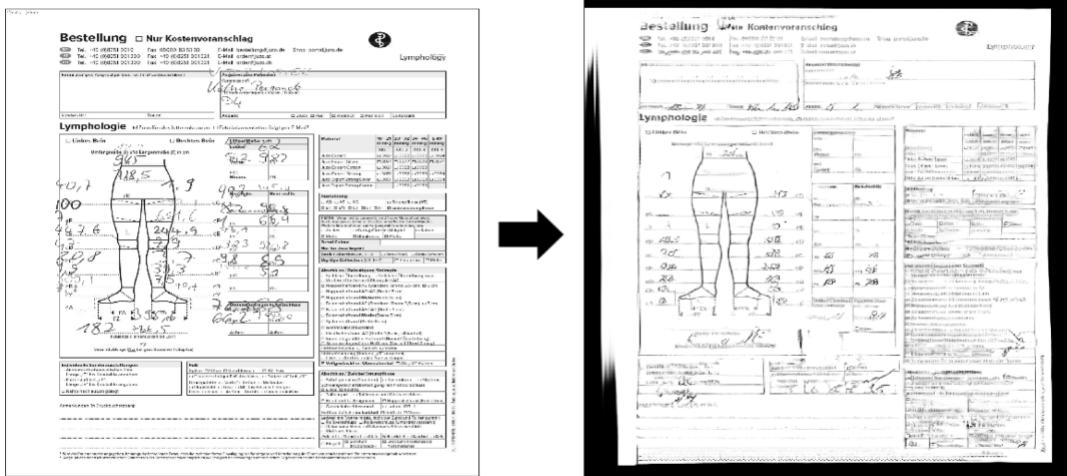


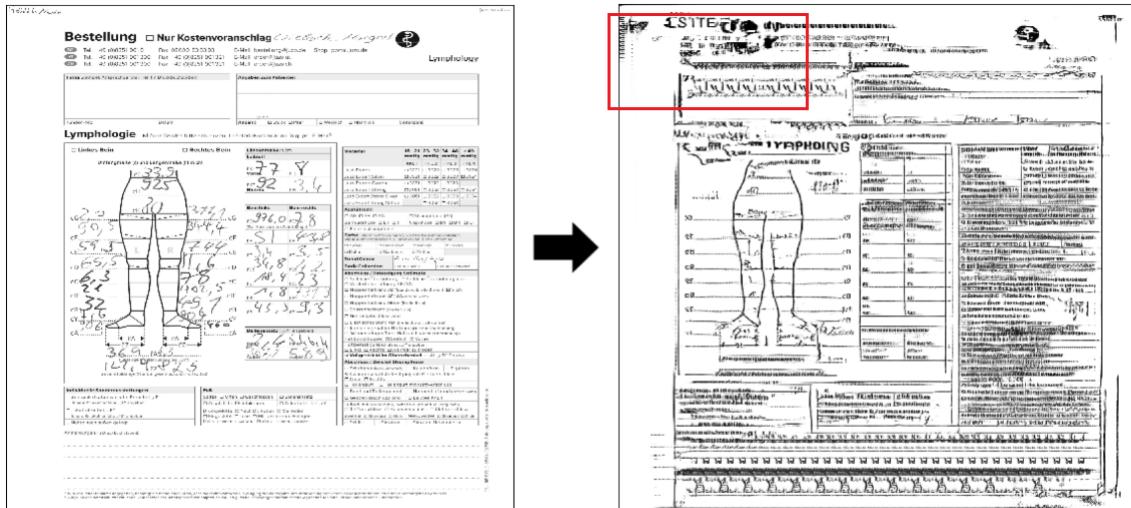
Figure 6.16: The CycleGAN generated document image consists of noisy artifacts that are unrealistic (figure reproduced from elevait GmbH & Co. KG with permission).



Synthetic document image

CycleGAN generated document image

Figure 6.17: The CycleGAN generated document image consists of dark border (figure reproduced from elevait GmbH & Co. KG with permission).



Synthetic document image

CycleGAN generated document image

Figure 6.18: The CycleGAN generated document image consists of noisy artifacts that are unrealistic and the visual quality of the generated image is not good. There are some artifacts (marked in red box) appear in the generated images that are not present in the synthetic image (figure reproduced from elevait GmbH & Co. KG with permission).

7. Conclusion and Future Work

In this chapter, the overview of the thesis is described by outlining the problem statement, related works, and the proposed solution in section 7.1. The results of the experiments conducted by the proposed image-to-image translation application to reduce the domain gap between synthetic data distribution and real data distribution are concluded in section 7.2. Finally, in section 7.3 the limitations and future work are discussed.

7.1 Overview

Since the last decades, there are many success stories about deep learning are written. Also, deep learning has evolved tremendously because of the introduction of neural networks to learn complex patterns present in videos, images, and speeches. Nowadays, neural networks are widely used in autonomous vehicles, language translations, object detection, face recognition, speech recognition, and many other sophisticated applications. However, there are limitations to deep learning models. In traditional deep learning models, neural networks are trained and tested on similar data distribution or the same domain to achieve the final objective by minimizing the error. In real-world scenarios, the data could arrive from different feature spaces, data distributions, and different domains. In such situations, neural networks don't generalize well to the data arrived from new data distribution or domain; for example, the model to detect pedestrian trained images of a pedestrian in summer might fail to detect pedestrians in winter. This problem of performance degradation causes because domain shift [81].

A large amount of training data is required to train neural networks to achieve low generalization error and high performance. However, training data are scarce and it is a very difficult job to collect and annotate the data. Hence, machine learning engineers required to create synthetic data, but neural networks trained using synthetic data will not generalize well on real data. So, over the year several domain adaption methodologies are developed by machine learning researchers to solve the problem of domain shift [81]. For example, transforming the image of a zebra from the source domain into the image of a horse in the target domain, and vice versa. Such kind of domain adaptation can be achieved by numerous image-to-image translation applications.

7.2 Conclusion

This thesis aims to solve the problem of data scarcity of real document images in the target domain using an image-to-image translation application. The proposed image-to-image translation application is developed using CycleGAN to transform synthetic document images into realistic document images to close the domain gap between synthetic data distribution and real data distribution. The synthetic document images created using templates and handwritten crops are transformed into realistic document images to tackle the problem of data scarcity in the target domain further to improve the classifier of real document images. The quality of images generated using CycleGAN is computed on the basis of how a classifier trained using images generated by the CycleGAN generalizes to the test dataset of annotated real document images.

Quantitatively the images generated using CycleGAN did not match the real data distribution. CycleGAN has been tried with document images for the first time, such form document images consist of small fields like checkboxes, small artifacts, and different handwritings, which increases the complexity in the learning of the neural networks. Also, most of the images used for generating realistic document images are of type “Bein”, and they are very similar to each other. As described in section 6.2.4, small artifact transformation failure has caused the wrong classification because

learning such distinct minor features present in training datasets by CycleGAN is very challenging. The images of the type “Arm” and “Hand” are very distinct from the image type “Bein”. Hence, they are classified significantly well A.6. Further, experiment conducted with dataset of synthetic document images, faxified document images, and CycleGAN generated document images to understand domain gap between these distributions and real data distribution. The quantitative results show the faxified data distribution has matched real data distribution significantly well compared to synthetic data distribution and CycleGAN generated data distribution.

Qualitatively the images generated using CycleGAN are promising. In section 6.14, qualitative results are illustrated. Though qualitative results are promising, some typical failure cases have been identified, and it requires further attention in the future to improve the quality of CycleGAN generated images. Failure cases like the transformation of handwriting crops from synthetic document images to realistic document images are failed (figure 6.15), and the generated realistic document images have unrealistic noise, dark borders, and unrealistic artifacts (figures 6.16), 6.17), and 6.18)). In the next section 7.3, a discussion about improving the proposed image-to-image translation application is discussed, along with its limitations.

7.3 Future Work

Every research work has a specified amount of time assigned to it. This thesis also has inevitable time constraints and limitations. So many experiments are left to be for future investigation. In this thesis, the proposed image-to-image translation application is implemented only using CycleGANs. Comparison between other methodologies and chosen methodology CycleGAN to transform synthetic document images into realistic document images is left for future research. For example CUT [45], FastCUT [45], Adversarial Consistency Loss Generative Adversarial Network (ACL-GAN) [82], Variational Autoencoder (VAE) [83], and DCGAN [36] are recommended for future research. GANs are a great success in generating realistic images, but the training process is not easy; the training is slow and unstable. GANs are hard to train. It has problems like mode collapse, vanishing gradients, lack of proper evaluation metric, and hard to converge. The most important problem with GANs is there no proper evaluation metric, without a good evaluation metric, it is like working in darkness. GANs have complex object functions observing training progress is difficult, there is no signal to where to stop the training, also, there no single common and good performance indicator to evaluate numerous types of GANs. In future, finding a proper performance indicator for GANs can a part of research.

In this thesis, CycleGAN’s generators and discriminators are optimized using least-square loss function [33] to avoid mentioned problems. The quality of generated images possibly would have been better if generators and discriminators of CycleGAN would have optimized using the Wasserstein metric. Wasserstein distance metric is popularly used in WGANs. WGANs are intended to improve GAN’s training by adopting a smooth Wasserstein distance metric for measuring the distance between two probability distributions. Shrivastava et al.[55] proposed a strategy for stable training. In which the discriminators are updated using a history of previously generated images instead of the ones produced recently by the generators. An image buffer of maintained to store 50 previously generated images. In the future, this strategy can be used to improve stability during the training and reduce model oscillations to produce better results. Qualitative results show the handwritten crops from synthetic document images were not reconstructed or transformed in the generated images (figure 6.15). In the future, this thesis work can be extended to solve this problem, so the generated images are used to improve the handwriting recognition models. Training CycleGAN sequentially is consuming numerous hours. Hence in the future, distributed training across multiple GPUs would be preferred while training the CycleGANs.

Data analysis and data cleaning are the essential steps taken before training any neural network. The dataset used for the training the CycleGAN consists of synthetic document images dataset and real document images dataset. This means the source domain is represented by synthetic document images and the target domain is represented by real document images. The dataset of synthetic document images is equally distributed. Every selected class has the same number of samples. The dataset of real document images had random real images from different classes, which are unknown and disorganized. It is just a collection large number of real document images. In the future, to

obtain better results the real document images dataset can be organized similarly to the synthetic document images dataset by performing data analysis and data cleaning at the initial stages of the research.

A. Appendix

A.1 MNIST Handwritten Numbers Dataset

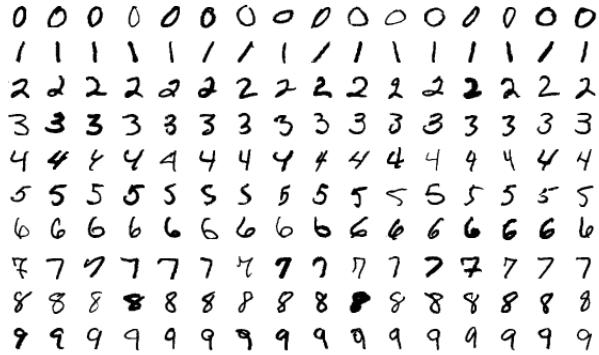


Figure A.1: Examples of Handwritten Numbers from the MNIST Dataset.¹

A.2 CycleGAN Models Training

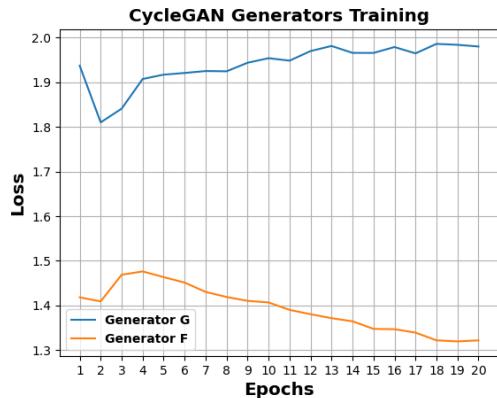


Figure A.2: CycleGAN generators G and F training epochs vs loss plot.

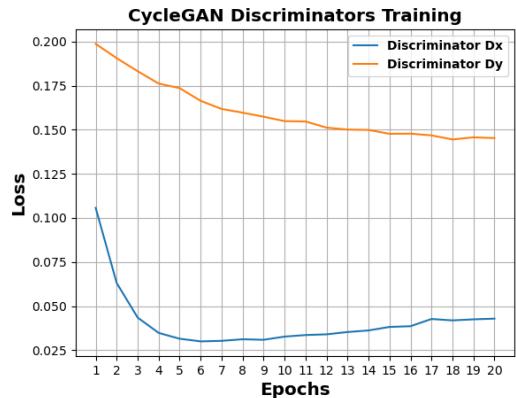


Figure A.3: CycleGAN discriminators D_X and D_Y training epochs vs loss plot.

¹<http://yann.lecun.com/exdb/mnist/> last access: September 3, 2021

A.3 GAN Training

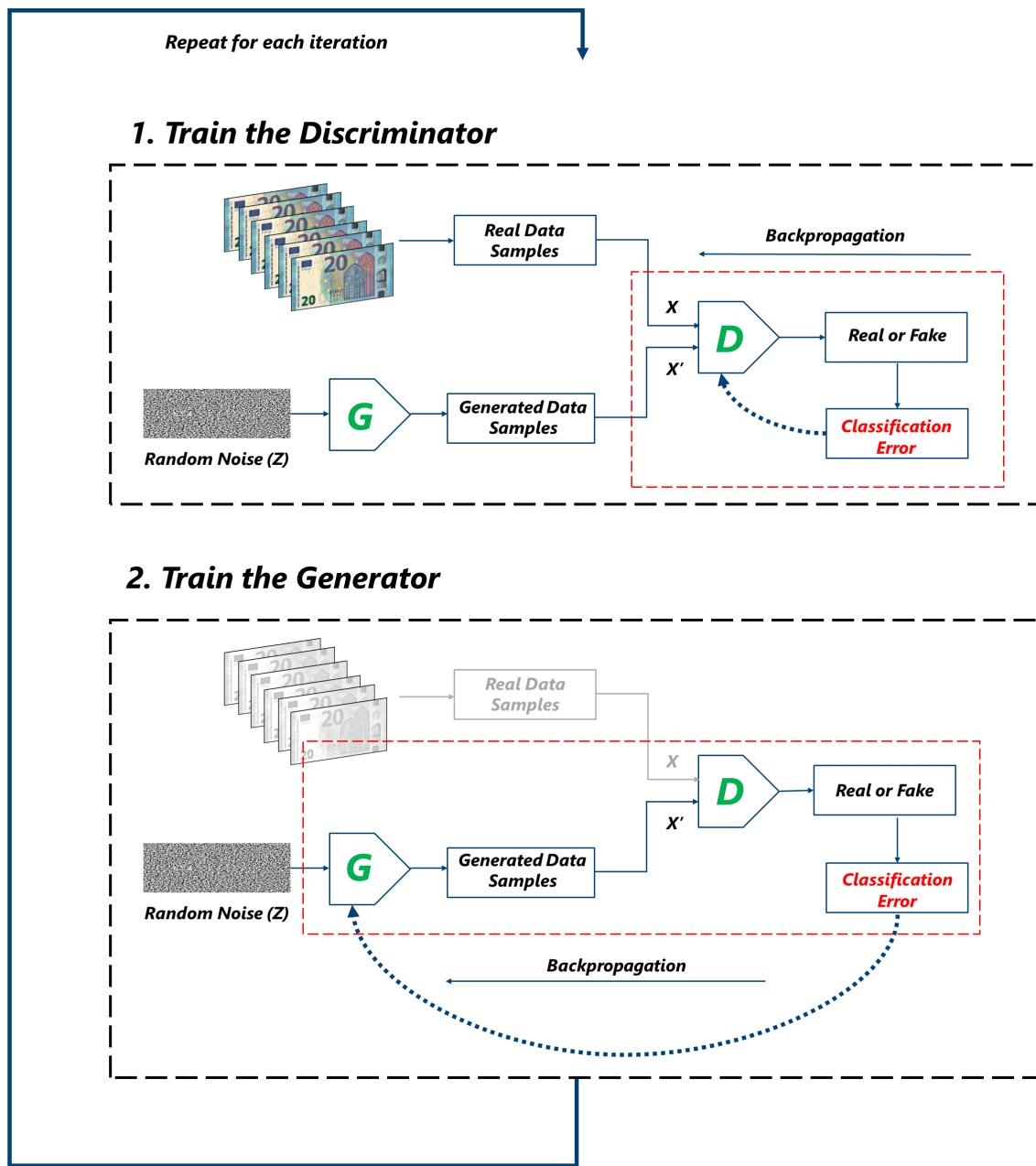


Figure A.4: Illustration of training of the GAN as per the algorithm.

A.4 Confusion Matrices

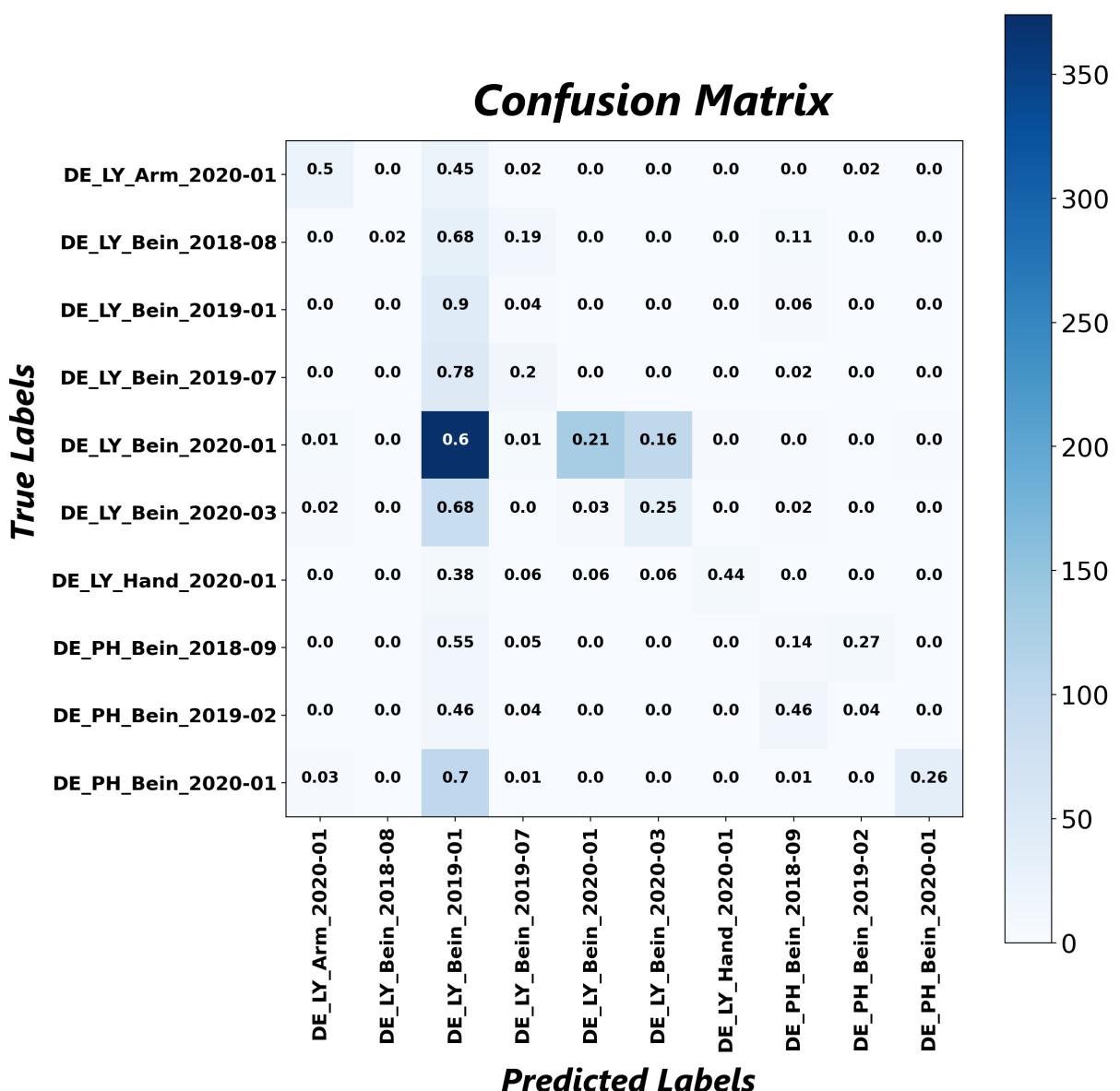


Figure A.5: Confusion matrix to analyze the performance of the classifier trained on synthetic document images and evaluated using real annotated document images (test dataset).

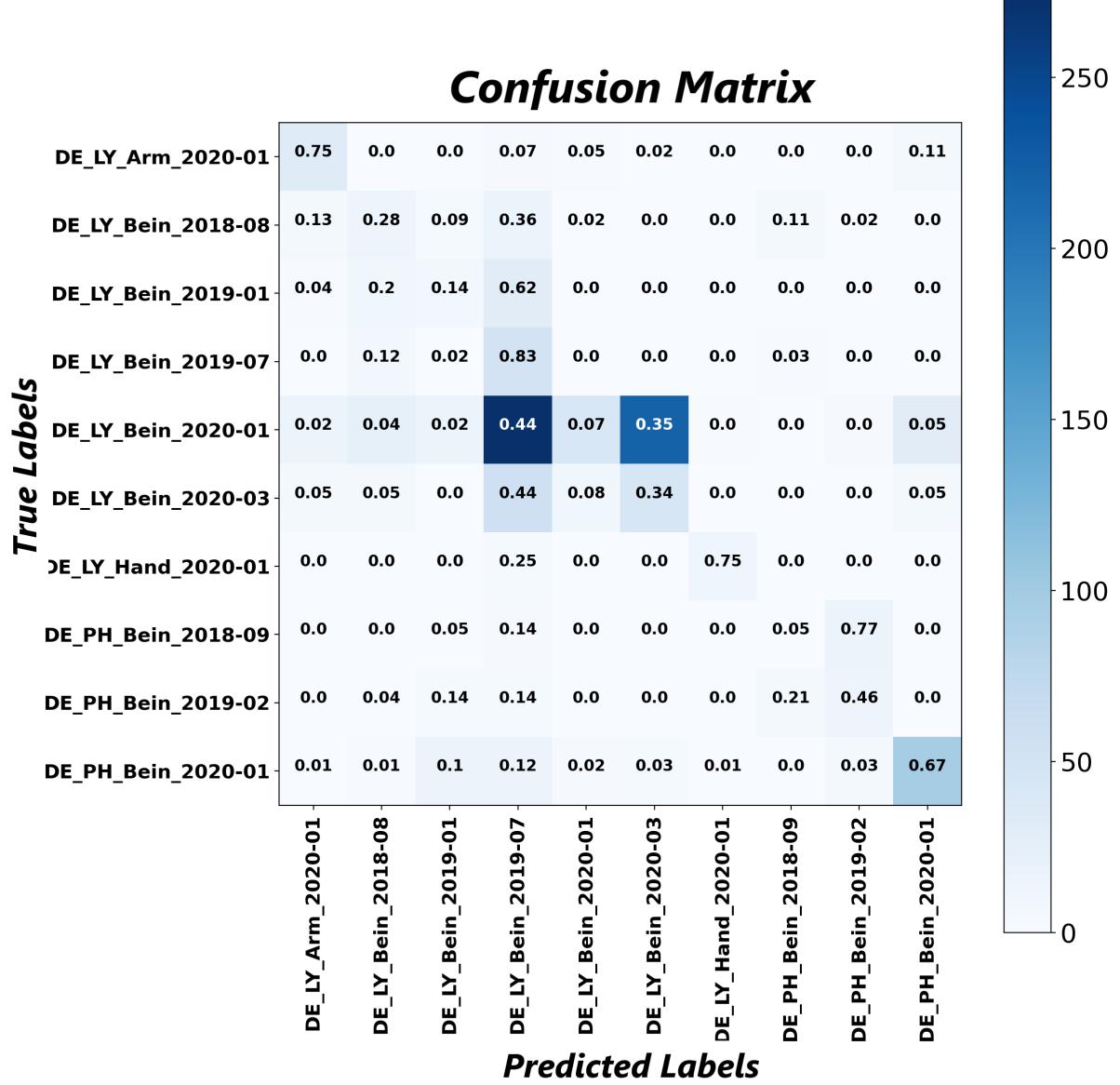


Figure A.6: Confusion matrix to analyze the performance of the classifier trained on CycleGAN generated document images and evaluated using real annotated document images (test dataset).

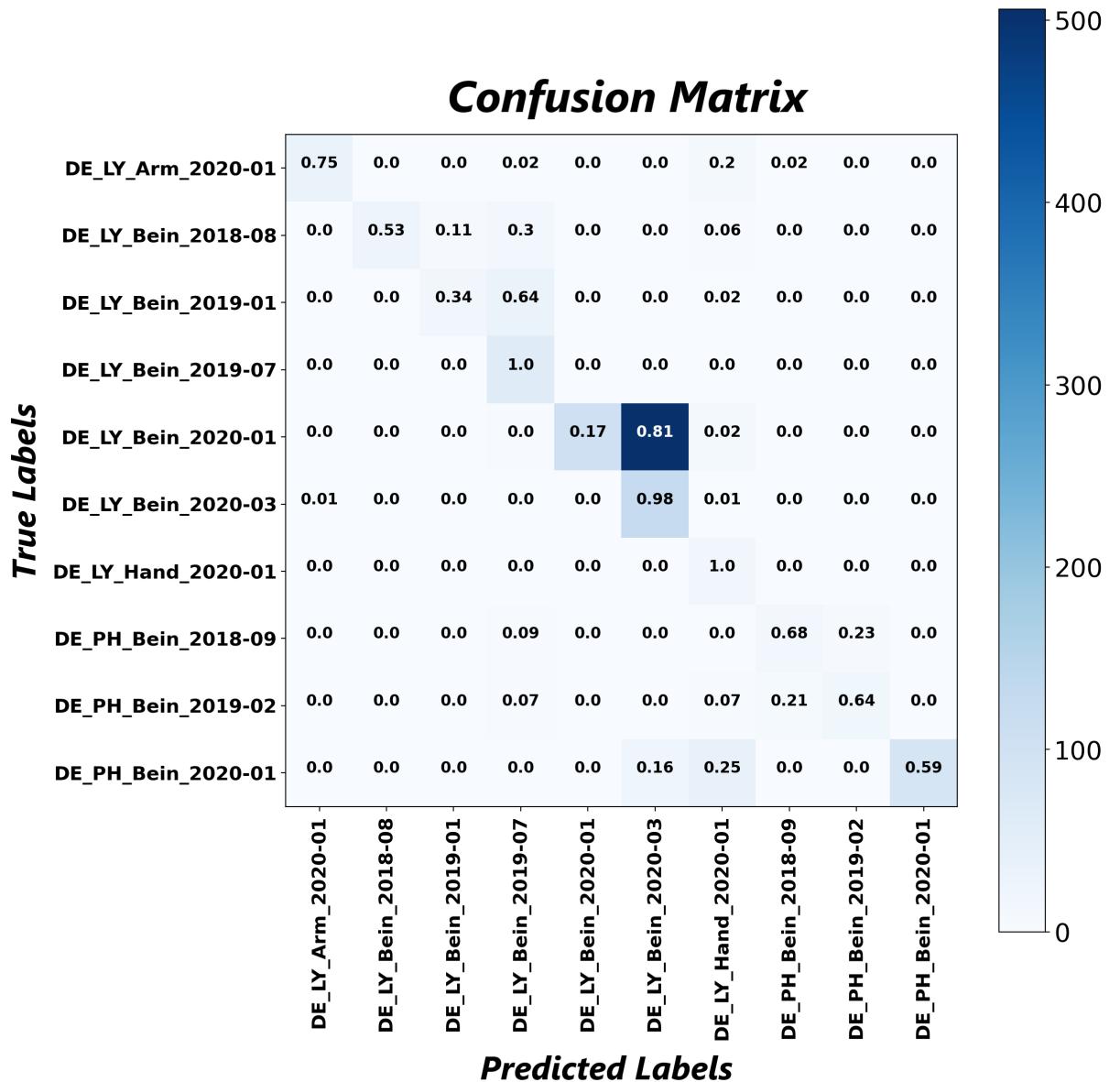


Figure A.7: Confusion matrix to analyze the performance of the classifier trained on faxified document images and evaluated using real annotated document images (test dataset).

A.5 Examples of Document Images

Bestellung Nur Kostenvoranschlag

Lymphology

Firma Stempel, Ansprechpartner, Tel. (in Druckbuchstaben)		Angaben zum Patienten																																														
		Kommission ¹ :																																														
		Frühere Anfertigung / KV-Nr. / Datum:																																														
Kunden-Nr.:	Datum:	Anzahl: <input type="checkbox"/> Stück <input type="checkbox"/> Paar		<input type="checkbox"/> Weiblich <input type="checkbox"/> Männlich Seitenzahl:																																												
Lymphologie <input checked="" type="checkbox"/> Zutreffendes bitte ankreuzen <input type="checkbox"/> Fotodokumentation folgt per E-Mail ²																																																
Zubehör <input type="checkbox"/> Lymphpad Line <input type="checkbox"/> Lymphpad Square		Abschluss / Befestigung Rundstrick <input type="checkbox"/> Gestrickabschlussrand <input type="checkbox"/> Noppenhaftstrand (Breite 3,5 cm) <input type="checkbox"/> Balancehaftstrand (Breite 3,5 cm - ab 01.2019) <input type="checkbox"/> BH-Befestigung - Trägerbreite: _____ cm <input type="checkbox"/> Mit Haftuntertritt (an der Schulter) <input type="checkbox"/> Schuster- und Haltegurt (Umfang „cHI“ angeben)																																														
Anmerkungen (in Druckbuchstaben):		Flachstrick <input type="checkbox"/> Gestrickabschlussrand <input type="checkbox"/> Noppenhaftstrand (Breite 3,5 cm) <input type="checkbox"/> 5 cm <input type="checkbox"/> Noppenhaftstrand Motiv (Breite 5 cm) <input type="checkbox"/> Elastisches Band (Breite 3,5 cm, silikonfrei) <input type="checkbox"/> Überhöhung (bei „cG“) <input type="checkbox"/> Überhöhung max. (bei „cG“) <input type="checkbox"/> $\frac{1}{4}$ innen eingenähter Haftrand (nur mit Überhöhung) <input type="checkbox"/> Innen eingenähter Haftrand (nur mit Überhöhung)																																														
		<input type="checkbox"/> BH-Befestigung - Trägerbreite: _____ cm <input type="checkbox"/> Mit Haftuntertritt (an der Schulter) <input type="checkbox"/> Schuster- und Haltegurt (Umfang „cHI“ angeben)																																														
		<input type="checkbox"/> Boleroverbindung mit Ärmeln / Armansätzen Konfektionsgröße: _____ - Länge „FH“: _____ cm (Maße für 2. Arm angeben) <input type="checkbox"/> Mit Haftuntertritt (an der Schulter)																																														
Material <table border="1"> <thead> <tr> <th></th> <th>18 - 21 mmHg</th> <th>23 - 32 mmHg</th> <th>34 - 46 mmHg</th> </tr> </thead> <tbody> <tr> <td>Rundstrick</td> <td>KKL 1</td> <td>KKL 2</td> <td>KKL 3</td> </tr> <tr> <td>Juzo Soft</td> <td><input type="checkbox"/> 2001</td> <td><input type="checkbox"/> 2002</td> <td>-</td> </tr> <tr> <td>Juzo Dynamic</td> <td><input type="checkbox"/> 3511</td> <td><input type="checkbox"/> 3512</td> <td>3513</td> </tr> <tr> <td>Juzo Dynamic Silver</td> <td><input type="checkbox"/> 3511</td> <td><input type="checkbox"/> 3512</td> <td>3513</td> </tr> <tr> <td>Flachstrick</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Juzo Expert</td> <td><input type="checkbox"/> 3021</td> <td><input type="checkbox"/> 3022</td> <td>3023</td> </tr> <tr> <td>Juzo Expert Silver</td> <td><input type="checkbox"/> 3021</td> <td><input type="checkbox"/> 3022</td> <td>3023</td> </tr> <tr> <td>Juzo Expert Cotton</td> <td><input type="checkbox"/> 3021</td> <td><input type="checkbox"/> 3022</td> <td>3023</td> </tr> <tr> <td>Juzo Expert Strong</td> <td><input type="checkbox"/> 3051</td> <td><input type="checkbox"/> 3052</td> <td>3053</td> </tr> <tr> <td>Juzo Expert Strong Silver</td> <td><input type="checkbox"/> 3051</td> <td><input type="checkbox"/> 3052</td> <td>3053</td> </tr> </tbody> </table>						18 - 21 mmHg	23 - 32 mmHg	34 - 46 mmHg	Rundstrick	KKL 1	KKL 2	KKL 3	Juzo Soft	<input type="checkbox"/> 2001	<input type="checkbox"/> 2002	-	Juzo Dynamic	<input type="checkbox"/> 3511	<input type="checkbox"/> 3512	3513	Juzo Dynamic Silver	<input type="checkbox"/> 3511	<input type="checkbox"/> 3512	3513	Flachstrick				Juzo Expert	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023	Juzo Expert Silver	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023	Juzo Expert Cotton	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023	Juzo Expert Strong	<input type="checkbox"/> 3051	<input type="checkbox"/> 3052	3053	Juzo Expert Strong Silver	<input type="checkbox"/> 3051	<input type="checkbox"/> 3052	3053
	18 - 21 mmHg	23 - 32 mmHg	34 - 46 mmHg																																													
Rundstrick	KKL 1	KKL 2	KKL 3																																													
Juzo Soft	<input type="checkbox"/> 2001	<input type="checkbox"/> 2002	-																																													
Juzo Dynamic	<input type="checkbox"/> 3511	<input type="checkbox"/> 3512	3513																																													
Juzo Dynamic Silver	<input type="checkbox"/> 3511	<input type="checkbox"/> 3512	3513																																													
Flachstrick																																																
Juzo Expert	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023																																													
Juzo Expert Silver	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023																																													
Juzo Expert Cotton	<input type="checkbox"/> 3021	<input type="checkbox"/> 3022	3023																																													
Juzo Expert Strong	<input type="checkbox"/> 3051	<input type="checkbox"/> 3052	3053																																													
Juzo Expert Strong Silver	<input type="checkbox"/> 3051	<input type="checkbox"/> 3052	3053																																													
Ausführung																																																
Rundstrick <input type="checkbox"/> Ärmel																																																
Flachstrick <input type="checkbox"/> Ärmel <input type="checkbox"/> Unterarmstulpe <input type="checkbox"/> In Verbindung mit Kompressionshandschuh zu tragen <input type="checkbox"/> Ärmel und Handschuh einteilig																																																
Farbe Wenn nichts vermerkt, wird Farbe Mandel geliefert. Ausführungen in Silver und Cotton erhältlich in Farbe Mandel. Weitere Informationen zur Farbauswahl siehe Umschlag.																																																
Rundstrick - Juzo Soft <table border="1"> <tbody> <tr> <td><input type="checkbox"/> Zucker</td> <td><input type="checkbox"/> Sesam</td> <td><input type="checkbox"/> Mandel</td> <td><input type="checkbox"/> Muskat</td> </tr> <tr> <td><input type="checkbox"/> Zimt</td> <td><input type="checkbox"/> Kakao</td> <td><input type="checkbox"/> Mohn</td> <td><input type="checkbox"/> Blaubeere</td> </tr> <tr> <td><input type="checkbox"/> Pfeffer</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>					<input type="checkbox"/> Zucker	<input type="checkbox"/> Sesam	<input type="checkbox"/> Mandel	<input type="checkbox"/> Muskat	<input type="checkbox"/> Zimt	<input type="checkbox"/> Kakao	<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere	<input type="checkbox"/> Pfeffer																																			
<input type="checkbox"/> Zucker	<input type="checkbox"/> Sesam	<input type="checkbox"/> Mandel	<input type="checkbox"/> Muskat																																													
<input type="checkbox"/> Zimt	<input type="checkbox"/> Kakao	<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere																																													
<input type="checkbox"/> Pfeffer																																																
Trend Colour <table border="1"> <tbody> <tr> <td>Rundstrick</td> <td><input type="checkbox"/> Batik-Weiß</td> <td><input type="checkbox"/> Batik-Schwarz</td> </tr> <tr> <td>Batik Collection</td> <td><input type="checkbox"/> Batik-Weiß</td> <td><input type="checkbox"/> Batik-Schwarz</td> </tr> </tbody> </table>					Rundstrick	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz	Batik Collection	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																						
Rundstrick	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																														
Batik Collection	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																														
Rundstrick - Juzo Dynamic <table border="1"> <tbody> <tr> <td><input type="checkbox"/> Sesam</td> <td><input type="checkbox"/> Mandel</td> <td><input type="checkbox"/> Mohn</td> <td><input type="checkbox"/> Blaubeere</td> </tr> <tr> <td><input type="checkbox"/> Pfeffer</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>					<input type="checkbox"/> Sesam	<input type="checkbox"/> Mandel	<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere	<input type="checkbox"/> Pfeffer																																							
<input type="checkbox"/> Sesam	<input type="checkbox"/> Mandel	<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere																																													
<input type="checkbox"/> Pfeffer																																																
Flachstrick - Juzo Expert / Juzo Expert Strong <table border="1"> <tbody> <tr> <td><input type="checkbox"/> Zucker</td> <td><input type="checkbox"/> Kardamom</td> <td><input type="checkbox"/> Mandel</td> <td><input type="checkbox"/> Kakao</td> </tr> <tr> <td><input type="checkbox"/> Mohn</td> <td><input type="checkbox"/> Blaubeere</td> <td><input type="checkbox"/> Pfeffer</td> <td></td> </tr> </tbody> </table>					<input type="checkbox"/> Zucker	<input type="checkbox"/> Kardamom	<input type="checkbox"/> Mandel	<input type="checkbox"/> Kakao	<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere	<input type="checkbox"/> Pfeffer																																					
<input type="checkbox"/> Zucker	<input type="checkbox"/> Kardamom	<input type="checkbox"/> Mandel	<input type="checkbox"/> Kakao																																													
<input type="checkbox"/> Mohn	<input type="checkbox"/> Blaubeere	<input type="checkbox"/> Pfeffer																																														
Trend Colour <table border="1"> <tbody> <tr> <td>Batik Collection (Juzo Expert)</td> <td><input type="checkbox"/> Batik-Weiß</td> <td><input type="checkbox"/> Batik-Schwarz</td> </tr> </tbody> </table>					Batik Collection (Juzo Expert)	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																									
Batik Collection (Juzo Expert)	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																														
Individuelle Sonderausstattungen																																																
Flachstrick <input type="checkbox"/> Anatomisch abgewinkelte Form bei „cE“ 30° <input type="checkbox"/> 50° (30° sind Standard bei Juzo Expert Strong und Juzo Expert Strong Silver) <input type="checkbox"/> Naht an der Armaußenseite (bei „cG“, nur mit anatomisch abgewinkelten Form 30°) <input type="checkbox"/> Trikofutter bei „cE“ <input type="checkbox"/> Silver <input type="checkbox"/> Nähte nach außen gelegt <input type="checkbox"/> Haftstrandstopper (Platzierung seitlich außen quer)																																																

¹ Wird der Patientenname angegeben, bestätigt die bestellende Firma, dass die rechtskonforme Einwilligung zur Weitergabe und Verarbeitung der Daten von dem betroffenen Patienten zuvor eingeholt worden ist.

² Aufgrund des datenschutzrechtlichen Grundsatzes der Datensparsamkeit empfehlen wir, lediglich bei schwierigen anatomischen Gegebenheiten eine Fotodokumentation zu übersenden.

DEU-140730/17/8 8650 LY 09/2018 Änderungen und Drucke vorbehalten.

Figure A.8: Examples of template (figure reproduced from elevait GmbH & Co. KG with permission).

Bestellung Nur Kostenvoranschlag

Lymphology

Firma Stempel, Ansprechpartner, Tel. (n Druckbuchstaben)	Angaben zum Patienten		
	Frühere Anfertigung / KV Nr. / Diagn.		
Kunden-Nr.	Datum:	Anzahl: <input checked="" type="checkbox"/> 1 Stück <input type="checkbox"/> Paar <input checked="" type="checkbox"/> Weiblich <input type="checkbox"/> Männlich	Schuhzahl:
Lymphologie <input checked="" type="checkbox"/> Zutreffendes bitte ankreuzen <input type="checkbox"/> Fotodokumentation folgt per E-Mail ²			
Linkes Bein		Rechtes Bein	
Umfangmaße (c) und Längenmaße (l) in cm 		Längenmaße in cm Leibteil PKT Vorne: <input type="checkbox"/> Hinter: <input type="checkbox"/> PKT Vorne: <input type="checkbox"/> Hinter: <input type="checkbox"/>	
		Material 18-21 22-32 34-46 ≥ 49 mmHg mmHg mmHg mmHg KK 1 : KKL 2 KK 3 : KKL 4 <input checked="" type="checkbox"/> Juzo Expert <input type="checkbox"/> 3021 <input type="checkbox"/> 3022 <input type="checkbox"/> 3023 <input type="checkbox"/> 3024 <input type="checkbox"/> Juzo Expert Silvia <input type="checkbox"/> 13021 <input type="checkbox"/> 3022 <input type="checkbox"/> 3023 <input type="checkbox"/> 3024 <input type="checkbox"/> Juzo Expert Cotton <input type="checkbox"/> 3021 <input type="checkbox"/> 3022 <input type="checkbox"/> 3023 <input type="checkbox"/> Juzo Expert Strong <input type="checkbox"/> 3051 <input type="checkbox"/> 3052 <input type="checkbox"/> 3053 <input type="checkbox"/> 3054 <input type="checkbox"/> Juzo Expert Strong Silver <input type="checkbox"/> 3051 <input type="checkbox"/> 3052 <input type="checkbox"/> 3053 <input type="checkbox"/> 3054 <input type="checkbox"/> Juzo Expert Strong Cotton <input type="checkbox"/> 13052 <input type="checkbox"/> 3053	
		Ausführung <input type="checkbox"/> AD <input type="checkbox"/> N <input checked="" type="checkbox"/> AG <input type="checkbox"/> Stumpfrose (AT) <input type="checkbox"/> Bern-Jahose <input type="checkbox"/> ET <input type="checkbox"/> LIT <input type="checkbox"/> Caprimos <input type="checkbox"/> B <input type="checkbox"/> IBI <input type="checkbox"/> C <input type="checkbox"/> Fibrabeschichtung	
		Farbe Wenn nichts vorgenommen wird, steht Monochromatik: <input type="checkbox"/> Weitere Farben möglich nur Farbauflage steht die Umgebung <input type="checkbox"/> Zucke <input checked="" type="checkbox"/> Karneol <input type="checkbox"/> Vande <input type="checkbox"/> Lachs <input type="checkbox"/> Mohr <input type="checkbox"/> Blaurose <input type="checkbox"/> Pfirsich	
		Trend Colour	
		Batik Collection <input type="checkbox"/> Batik Weiss <input type="checkbox"/> Batik Schwarz	
Abschluss / Befestigung Strümpfe <input type="checkbox"/> Seitlich Überhöhung <input type="checkbox"/> Seitliche Überhöhung max. <input type="checkbox"/> Vorderseitige Verstärkung AF/AG <input type="checkbox"/> Nonstop Haftband AD (Standard Breite 3,5 cm) <input type="checkbox"/> 5 cm <input checked="" type="checkbox"/> Nonstop Haftband AH/AG (Breite 5 cm) <input type="checkbox"/> Nonstop Haftband Moritz (Breite 5 cm) <input type="checkbox"/> Spitzenthaftband (Breite 3 cm) <input type="checkbox"/> Gestrickabschlussrand <input type="checkbox"/> Elastisches Band AD (Breite 3,5 cm, silikonisiert) <input type="checkbox"/> Innen eingezwicke "Haftranc Nur mit Überhöhung" <input type="checkbox"/> % in einem eingewickelten Hatband über der entsprechenden Überhöhung Haftrabdstopper <input type="checkbox"/> Seilach <input type="checkbox"/> Vorne Haftrabdstopper <input type="checkbox"/> Seilach <input type="checkbox"/> Vorne Fixierung <input type="checkbox"/> Ringnäht <input type="checkbox"/> Nahtgarn <input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Auf <input type="checkbox"/> Ab Vollgestrickt im Oberschenkel <input type="checkbox"/> Aa „CD“ Porosa			
Abschluss / Zwickel Strumpfhose <input type="checkbox"/> Bund und Tüllegründung <input type="checkbox"/> Nonstop Haftband (Breite 5 cm) <input type="checkbox"/> Schrägerwirren (Schrägad) <input type="checkbox"/> Kastenform <input type="checkbox"/> Stiform Schwangerschaftsbefestigung mit Klettverschluss <input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Tüllegründung <input type="checkbox"/> Tüllegründung mit Klettverschluss <input type="checkbox"/> Bund und Tüllegründung <input type="checkbox"/> Nonstop Haftband (Breite 5 cm) <input type="checkbox"/> Gestrickabschlussrand <input type="checkbox"/> Tüllegründung KK 1 <input type="checkbox"/> Leibteil mit Vorne (N), nicht sie kann die Tüllegründung <input type="checkbox"/> Reißverschluss <input type="checkbox"/> Hakenverschluss <input type="checkbox"/> Klettverschluss usw. <input type="checkbox"/> Zwickel Standard <input type="checkbox"/> Klein <input type="checkbox"/> Netzzwickel <input type="checkbox"/> Spannung <input type="checkbox"/> ohne <input type="checkbox"/> Schlitze <input type="checkbox"/> Scrotum <input type="checkbox"/> Skrotum Netzhakenfal			
Individuelle Sonderausstattungen <input type="checkbox"/> Anatomisch angepasste Form bei „CF“ <input type="checkbox"/> Inne & Klettabschlüsse „CE“ angezogen <input type="checkbox"/> Trikotfutter bei „CF“ <input type="checkbox"/> innere Kreukleinkante „CE“ angezogen <input type="checkbox"/> Nähte nach a.ßen gelegt			
Fuß <input type="checkbox"/> Spitzo <input type="checkbox"/> Offen <input checked="" type="checkbox"/> Geschlossen <input type="checkbox"/> Balleneinsatz <input type="checkbox"/> 7 zusätzliche Histostecken <input type="checkbox"/> Innenhälfte der „CY“ <input type="checkbox"/> Druckpolster <input type="checkbox"/> Nach Dr. Rötter <input type="checkbox"/> Malteulen <input type="checkbox"/> Eingeklebt <input type="checkbox"/> Löse <input type="checkbox"/> Mit Tasche zum Einlegen <input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Außen <input type="checkbox"/> Innen <input type="checkbox"/> Außen			
<small>¹ Wird der Patientenname eingesetzt, bestätigt die bestellende Firma, dass die rote rote Farbe Erweiterung zur Weiterleitung und Verarbeitung der Daten von den bestellten Artikeln durchgeführt werden soll. <small>² Aufgrund des unterschiedlichen Kundensegments darf dies speziell empfohlen werden. Insolgebei schwieriger Reaktionen ist eine Dokumentation erforderlich.</small> </small>			

Figure A.9: Example of real document image (figure reproduced from elevait GmbH & Co. KG with permission).

Lymphologie Bestellung Kostenvoranschlag

8650LYDEU012020-

Firma Stempel, Ansprechpartner, Tel. (in Druckbuchstaben)				Angaben zum Patienten				<input type="checkbox"/> Fotodokumentation folgt per E-Mail*		
				Frühere Anfertigung / KV-Nr. / Datum:						
Kunden-Nr.:		Datum:		Anzahl:	<input type="checkbox"/> Stück	<input type="checkbox"/> Paar	<input type="checkbox"/> Weiblich	<input type="checkbox"/> Männlich	<input type="checkbox"/> Divers	
Material Juzo Expert Juzo Expert Silver Juzo Expert Cotton Juzo Expert Strong Juzo Expert Strong Silver	mmHg	18-21	23-32	34-46	Farbe Wenn nichts vermerkt, wird Farbe Mandel geliefert. Silver und Cotton nur in Farbe Mandel erhältlich. <input type="checkbox"/> Zucker <input type="checkbox"/> Sesam <input type="checkbox"/> Mandel <input type="checkbox"/> Zimt <input type="checkbox"/> Kakao <input type="checkbox"/> Mohn <input type="checkbox"/> Blaubeere <input type="checkbox"/> Pfeffet. <input type="checkbox"/> Trend Colours <input type="checkbox"/> Fashion Colours					
		KKL 1	KKL 2	KKL 3						
		3021	3022	3023						
		3021	3022	3023						
		3021	3022	3023						
		3051	3052	3053						
Collection (Juzo Expert) Batik (KKL 1 - 3) Dip Dye (KKL 1 - 2) <input type="checkbox"/> Batik-Weiß <input type="checkbox"/> Blaubeere <input type="checkbox"/> Batik-Schwarz <input type="checkbox"/> Mohn										
Ausführung <input type="checkbox"/> Ärmel <input type="checkbox"/> Unterarmstulpe <input type="checkbox"/> In Verbindung mit Kompressionshandschuh zu tragen <input type="checkbox"/> Ärmel und Handschuh einstellig										
Ausstattung Arm <input type="checkbox"/> Seitliche Überhöhung (bei „cG“) <input type="checkbox"/> max. <input type="checkbox"/> Gestrickabschluss <input type="checkbox"/> Noppenhastrand (Breite 3,5 cm) <input type="checkbox"/> 5 cm <input type="checkbox"/> Noppenhastrand Motiv (Breite 5 cm) <input type="checkbox"/> Balancehastrand (Breite 3,5 cm) <input type="checkbox"/> 5 cm <input type="checkbox"/> Balancehastrand Motiv (Breite 5 cm) <input type="checkbox"/> Elastisches Band (Breite 3,5 cm, silikonfrei) <input type="checkbox"/> ¼ innen eingenähter Hafrand (nur mit Überhöhung) <input type="checkbox"/> Innen eingenähter Hafrand (nur mit Überhöhung) <input type="checkbox"/> BH-Befestigung - Trägerbreite: _____ cm <input type="checkbox"/> Mit Haftuntertritt (an der Schulter) <input type="checkbox"/> Schulter- und Haltegurt (Umfang „cH“ angeben) <input type="checkbox"/> Boleroverbindung mit Ärmeln / Armsätszen <input type="checkbox"/> Konfektionsgröße: _____ Länge „PHH“: _____ cm <input type="checkbox"/> Maße für 2. Arm angeben <input type="checkbox"/> Mit Haftuntertritt (an der Schulter) <input type="checkbox"/> Anatomisch abgewinkelster Ellenbogen 30° <input type="checkbox"/> 50° (30° sind Standard bei Juzo Expert Strong und Juzo Expert Strong Silver) <input type="checkbox"/> Naht an der Armaußenseite (bei „cE“, nur mit anatomisch abgewinkeltem Ellenbogen 30°) <input type="checkbox"/> Futterstoff bei „cE“ <input type="checkbox"/> Futterstoff Silver bei „cE“ <input type="checkbox"/> Haftbandstücke (Platzierung seitlich außen quer) <input type="checkbox"/> Nähte nach außen gelegt										
Arm links Umfangmaße (c) in cm · Längemaße (l) in cm Arm rechts										

ED 14072010/01-8650LY 03/2020 - Änderungen und Irrtumer vorbehalten.

Anmerkungen (in Druckbuchstaben):

Bitte neuen Maßblock senden

* Aufgrund des datenschutzrechtlichen Grundsatzes der Datensparsamkeit empfehlen wir, lediglich bei schwierigen anatomischen Gegebenheiten eine Fotodokumentation zu über senden.

Figure A.10: Examples of faxified document image (figure reproduced from elevait GmbH & Co. KG with permission).

A.6 Classifier Architecture Diagram

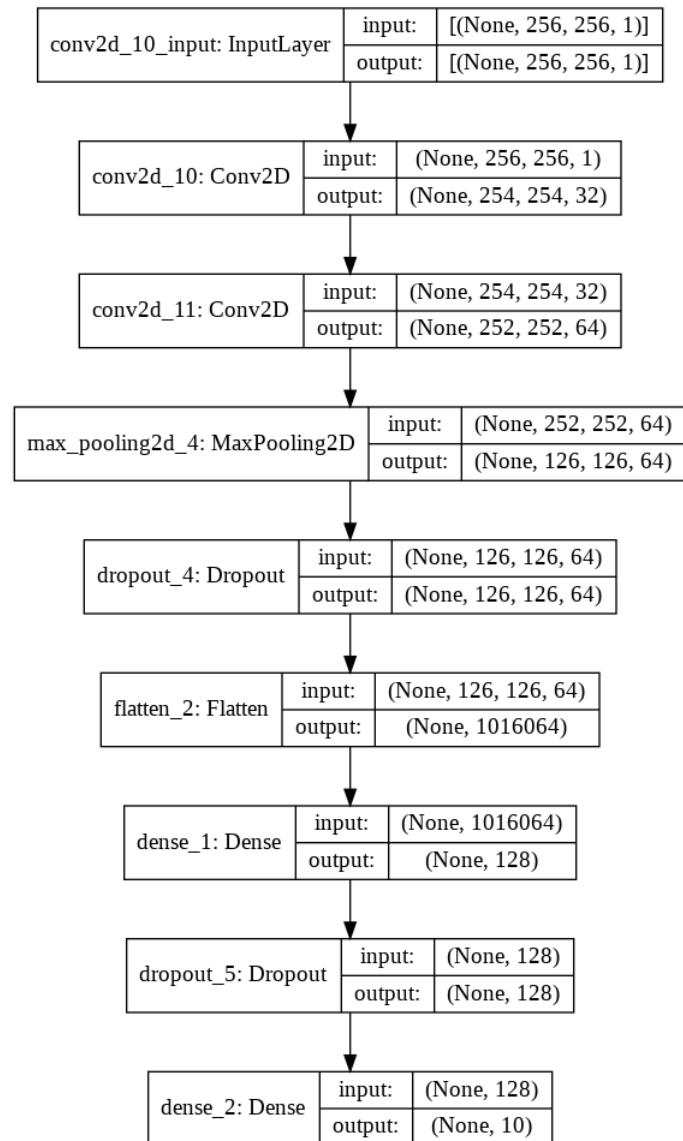


Figure A.11: Classifier Model Summary.

A.7 Generator Model Summary

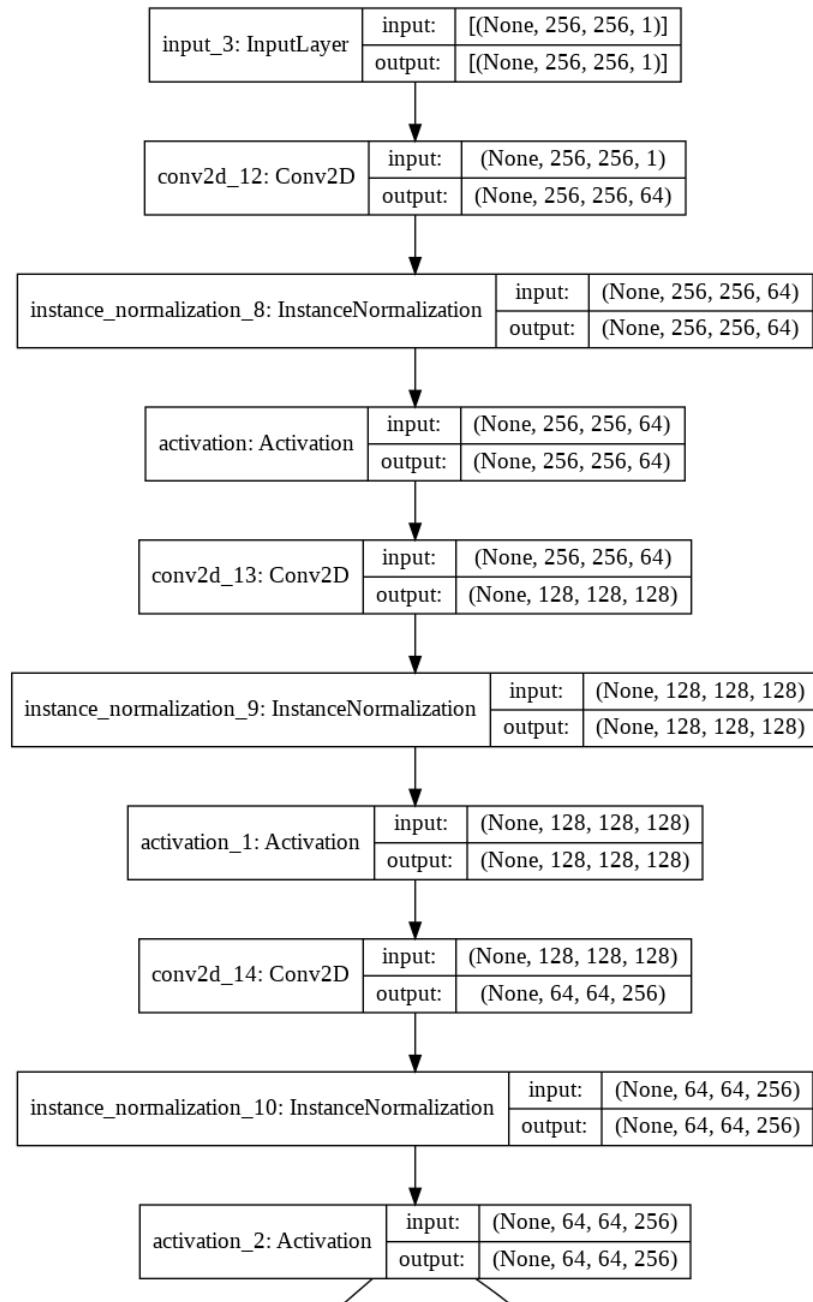


Figure A.12: Generator Model Summary. Continue to Next Page.

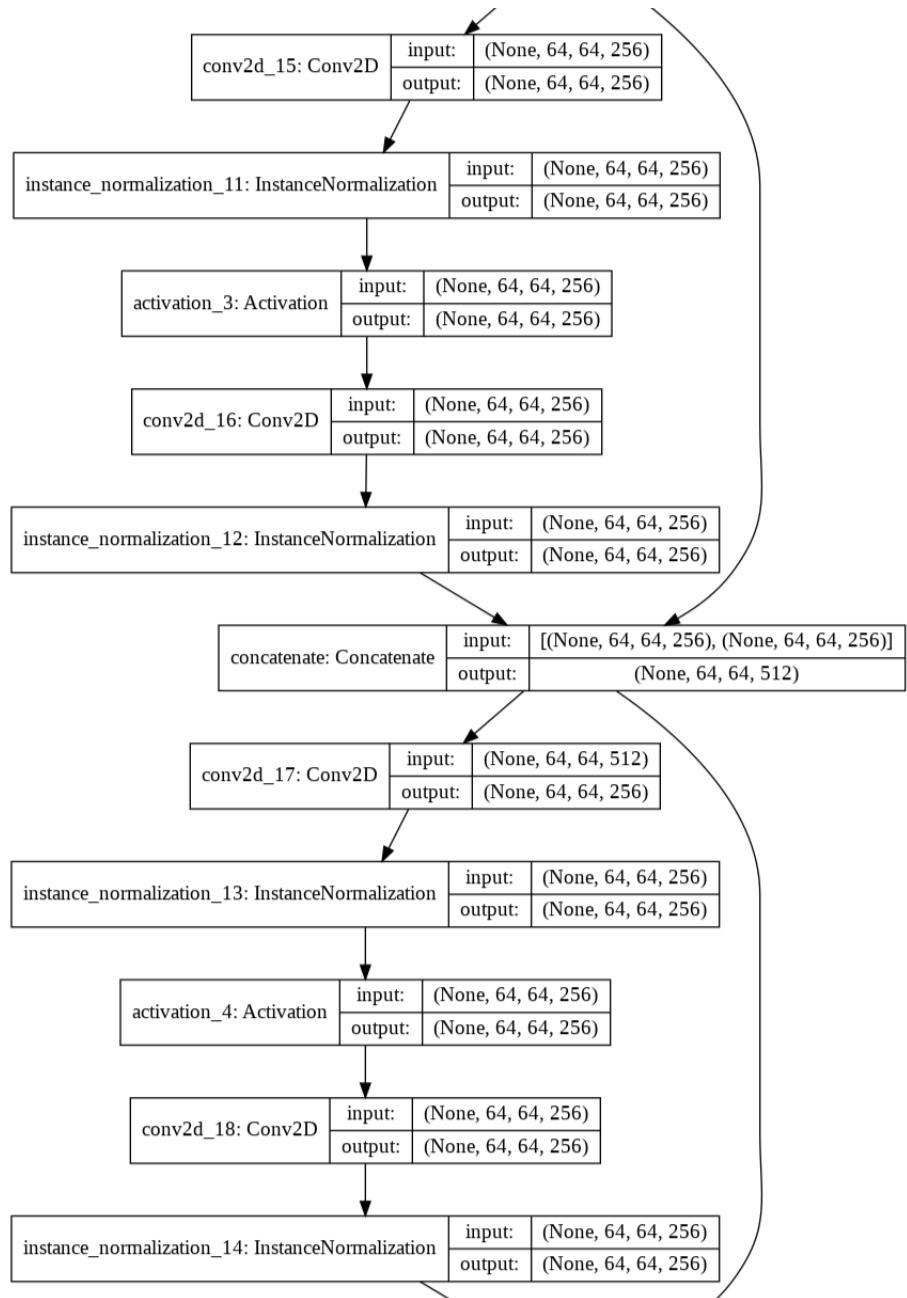


Figure A.13: Generator Model Summary. Continue to Next Page.

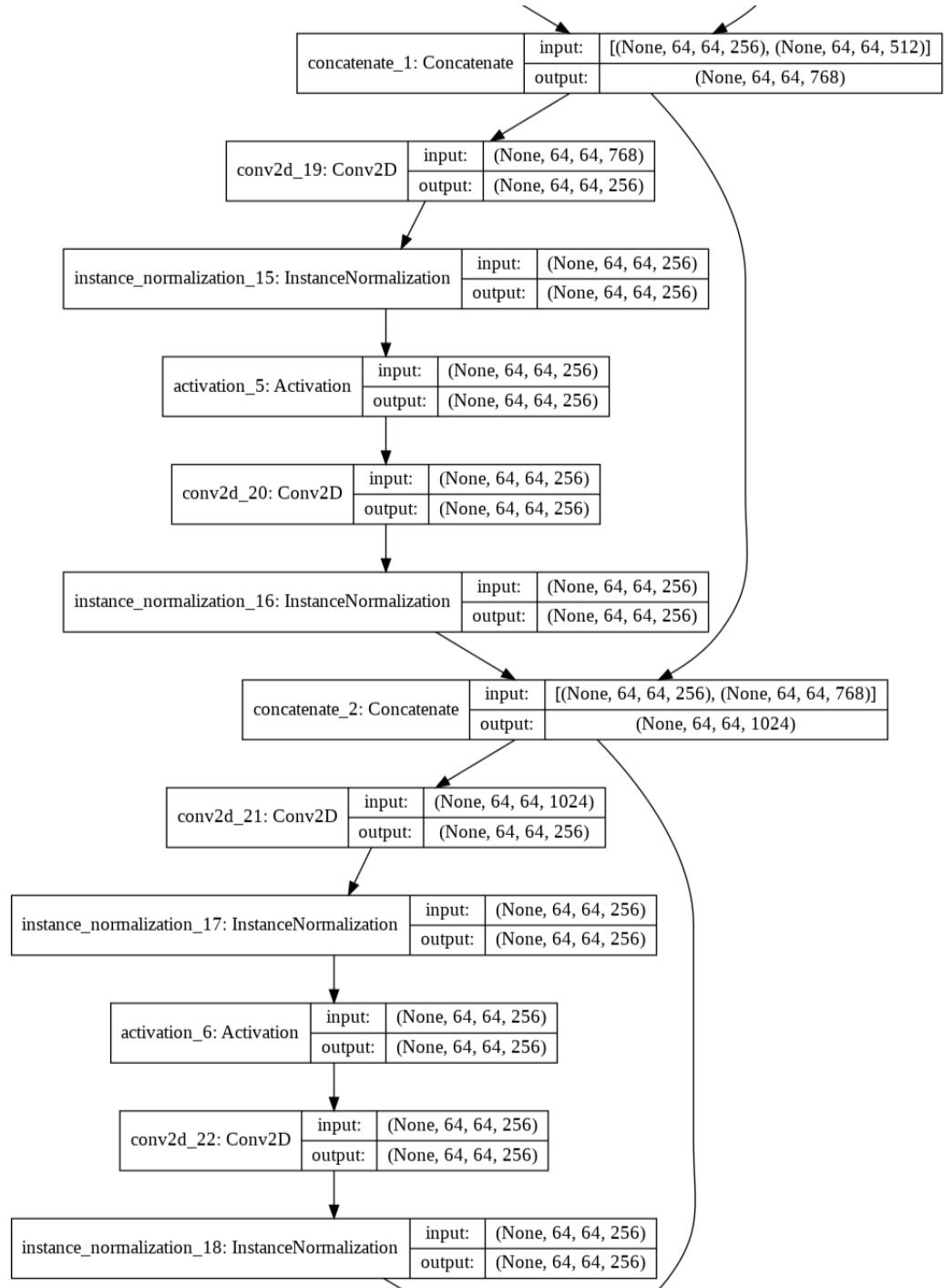


Figure A.14: Generator Model Summary. Continue to Next Page.

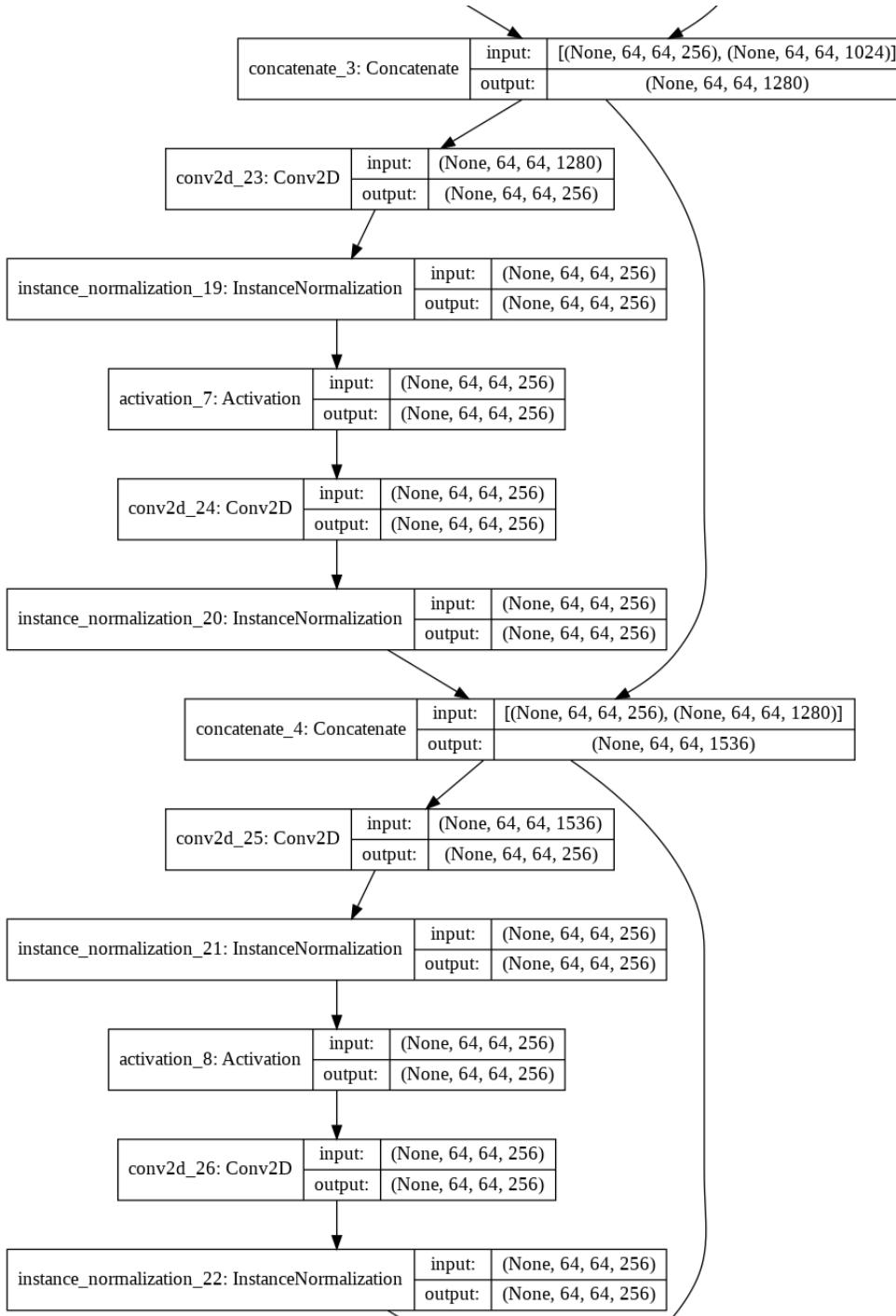


Figure A.15: Generator Model Summary. Continue to Next Page.

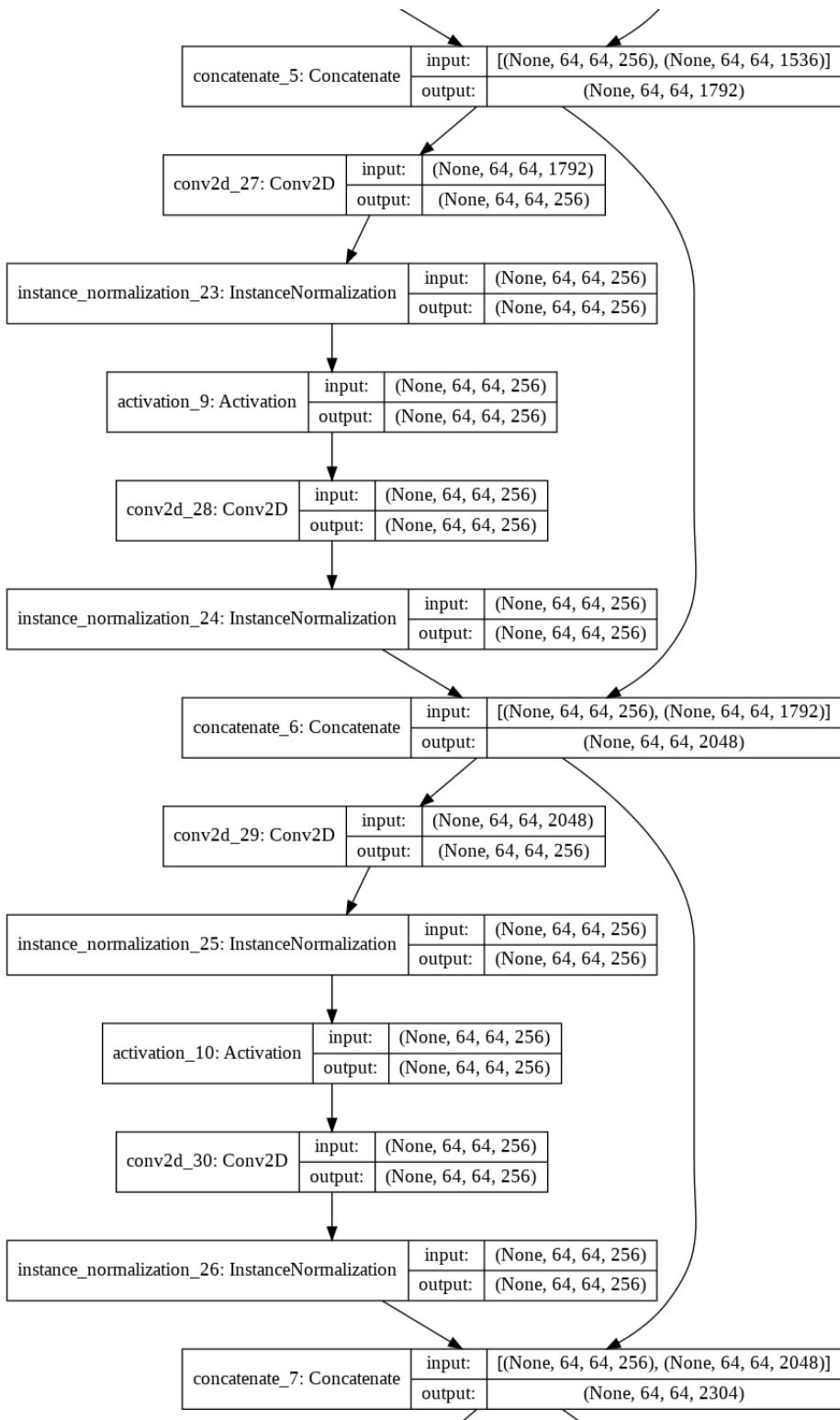


Figure A.16: Generator Model Summary. Continue to Next Page.

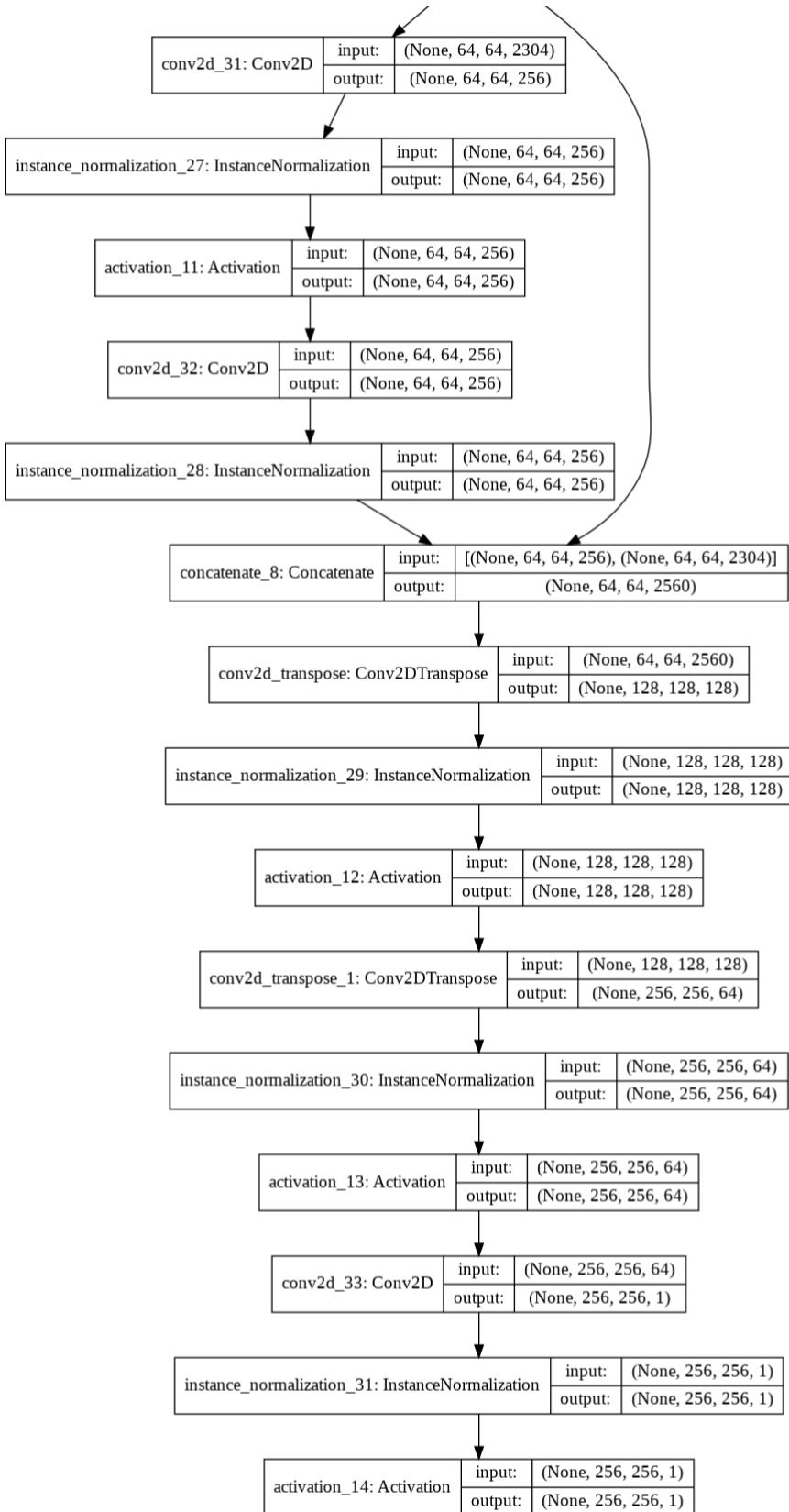


Figure A.17: Generator Model Summary. Ends Here.

A.8 Discriminator Model Summary

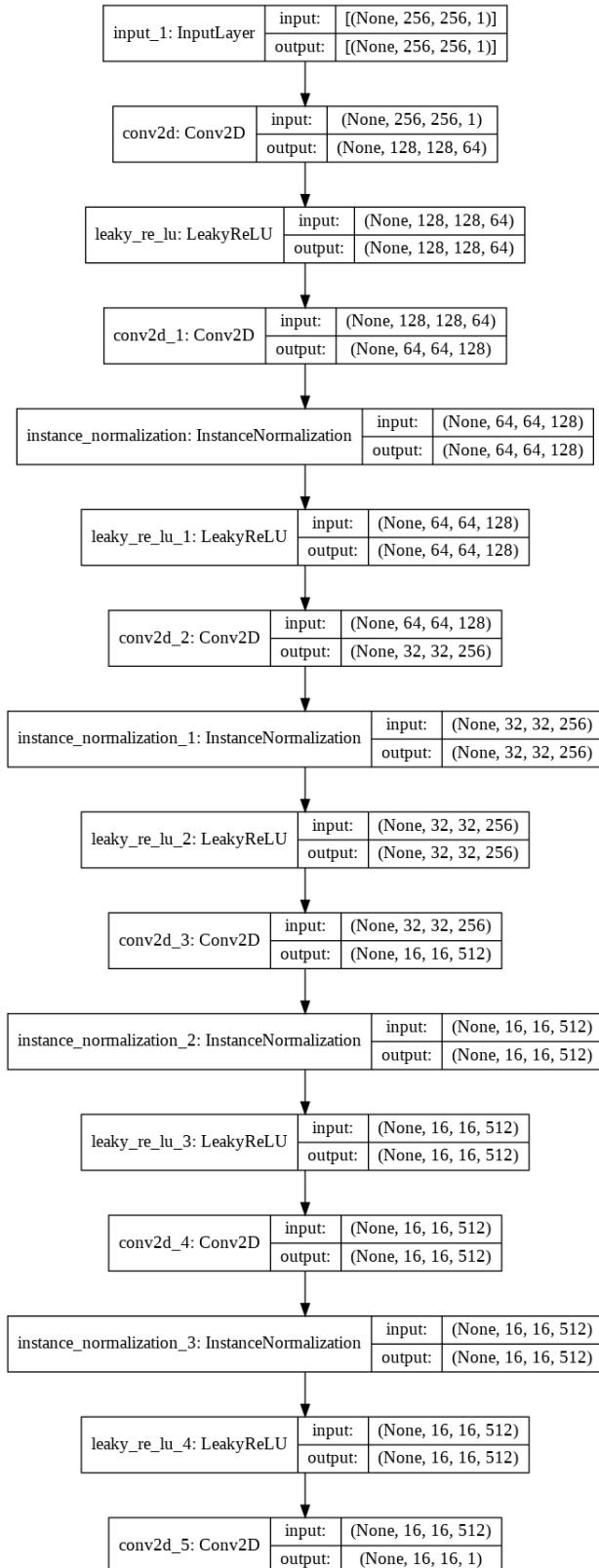


Figure A.18: Discriminator Model Summary.

Bibliography

- [1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- [2] J. Langr and V. Bok. *GANs in Action: Deep learning with Generative Adversarial Networks*. Manning Publications, 2019.
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning (adaptive computation and machine learning series). *Cambridge Massachusetts*, pages 321–359, 2017.
- [4] Kun-Hsing Yu, Andrew L. Beam, and Isaac S. Kohane. Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10):719–731, 2018.
- [5] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.
- [6] Michael Brady. Artificial intelligence and robotics. In Michael Brady, Lester A. Gerhardt, and Harold F. Davidson, editors, *Robotics and Artificial Intelligence*, pages 47–63, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.
- [7] Daniela Girimonte and Dario Izzo. *Artificial Intelligence for Space Applications*, pages 235–253. Springer London, London, 2007.
- [8] Aboul-Ella Hassanien, Ahmad Taher Azar, Tarek Gaber, Diego Oliva, and Fahmy M. Tolba, editors. *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*. Springer International Publishing, 2020.
- [9] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.
- [10] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy, 2017.
- [11] C. Tensmeyer, M. Brodie, D. Saunders, and T. Martinez. Generating realistic binarization data with generative adversarial networks. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 172–177, 2019.
- [12] Patrick Hemmer, Niklas Kühl, and Jakob Schöffer. Deal: Deep evidential active learning for image classification, 2020.
- [13] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.
- [14] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.
- [15] Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees, 2020.
- [16] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [18] Lei Kang, Marcal Rusinol, Alicia Fornes, Pau Riba, and Mauricio Villegas. Unsupervised adaptation for synthetic-to-real handwritten word recognition. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar 2020.

- [19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.
- [20] Q. A. Bui, D. Molland, and S. Tabbone. Automatic synthetic document image generation using generative adversarial networks: Application in mobile-captured document analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 393–400, 2019.
- [21] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
- [22] Giorgio Metta and Angelo Cangelosi. *Cognitive Robotics*, pages 613–616. Springer US, Boston, MA, 2012.
- [23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [24] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications, 2021.
- [25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [26] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010.
- [27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [28] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations, 2012.
- [29] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop, 2014.
- [30] P Manisha and Sujit Gujar. Generative adversarial networks (gans): What it can generate and what it cannot?, 2019.
- [31] Hoang Thanh-Tung and Truyen Tran. On catastrophic forgetting and mode collapse in generative adversarial networks, 2020.
- [32] M. R. Pavan Kumar and Prabhu Jayagopal. Generative adversarial networks: a survey on applications and challenges. *International Journal of Multimedia Information Retrieval*, 10(1):1–24, 2021.
- [33] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017.
- [34] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2016.
- [35] C. Liu, F. Yin, Q. Wang, and D. Wang. Icdar 2011 chinese handwriting recognition competition. In *2011 International Conference on Document Analysis and Recognition*, pages 1464–1469, 2011.
- [36] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.
- [37] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network, 2017.
- [38] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2017.

- [39] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.
- [40] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.
- [41] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold, 2018.
- [42] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.
- [43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [44] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.
- [45] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation, 2020.
- [46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.
- [47] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks, 2018.
- [48] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations, 2019.
- [49] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, Kun Zhang, and Dacheng Tao. Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping, 2018.
- [50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.
- [51] Monika Sharma, Abhishek Verma, and Lovekesh Vig. Learning to clean: A gan perspective, 2019.
- [52] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning, 2017.
- [53] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference, 2017.
- [54] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks, 2016.
- [55] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training, 2017.
- [56] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
- [57] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.
- [58] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, Jan 2018.
- [59] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.
- [60] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.
- [61] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

- [62] Rikiya Yamashita, Mizuho Nishio, Richard Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 06 2018.
- [63] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.
- [64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [65] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [67] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, page 807–814, Madison, WI, USA, 2010. Omnipress.
- [68] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.
- [69] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings.
- [70] Ashwani Kumar. Ordinal pooling networks: For preserving information over shrinking feature maps, 2018.
- [71] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2014.
- [72] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation, 2016.
- [73] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [74] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.
- [75] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.
- [76] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [77] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017.
- [78] David M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2020.
- [79] Meysam Vakili, Mohammad Ghamsari, and Masoumeh Rezaei. Performance analysis and comparison of machine and deep learning algorithms for iot data classification, 2020.
- [80] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. Thresholding classifiers to maximize f1 score, 2014.

- [81] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation, 2020.
- [82] Yihao Zhao, Ruihai Wu, and Hao Dong. Unpaired image-to-image translation using adversarial consistency loss, 2021.
- [83] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.



Name: Pawar	Bitte beachten:
Vorname: Giriraj Sukumar	1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.
geb. am: 13.04.1993	
Matr.-Nr.: 551205	

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende **Masterarbeit** selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: **31.08.2021**

Unterschrift:

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.