



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, November 23, 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, November 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 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 reduce the domain gap between synthetic 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 realistic 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 to match the real data distribution significantly. 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 and sufficient data is generated in the target domain, the performance of the real document image classifier eventually can be improved. This thesis is limited to the study of unpaired image-to-image translation method CycleGAN. The remaining methods and comparisons with them are left for future work. In the future, CycleGAN can be used to generate high-quality 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 gives strength to his family.

# Contents

<b>List of Figures</b>	<b>3</b>
<b>List of Tables</b>	<b>5</b>
<b>List of Algorithms</b>	<b>6</b>
<b>List of Abbreviations</b>	<b>7</b>
<b>1 Introduction</b>	<b>8</b>
1.1 Overview . . . . .	8
1.2 Motivation . . . . .	9
1.3 Problem Statement and Proposed Solution . . . . .	13
1.4 Thesis Objectives . . . . .	17
1.5 Thesis Limitations and Structure . . . . .	17
<b>2 Related Works</b>	<b>18</b>
2.1 Literature Survey . . . . .	18
2.2 Summary . . . . .	23
<b>3 Fundamentals</b>	<b>24</b>
3.1 Generative Adversarial Networks (GANs) . . . . .	24
3.1.1 GAN Training . . . . .	26
3.2 Convolution Neural Networks . . . . .	29
3.2.1 Convolution Layer . . . . .	29
3.2.2 Pooling Layer . . . . .	33
3.2.3 Fully connected layer . . . . .	34
<b>4 Methodology</b>	<b>36</b>
4.1 Cycle-Consistent Adversarial Networks . . . . .	37
4.1.1 Cycle-Consistency Loss . . . . .	37
4.1.2 Least-Square Loss . . . . .	38
4.1.3 Identity Mapping Loss . . . . .	39
4.1.4 Complete Objective Function . . . . .	39
4.2 CycleGAN Training Algorithm . . . . .	40
<b>5 Implementation</b>	<b>41</b>
5.1 Dataset Preparation . . . . .	41
5.2 Network Architecture . . . . .	44
5.2.1 CycleGAN . . . . .	44
5.2.2 Classifier . . . . .	46
5.3 Training Details . . . . .	47
5.3.1 CycleGAN . . . . .	47
5.3.2 Classifier . . . . .	48
<b>6 Experiments and Evaluation</b>	<b>50</b>
6.1 Evaluation Metrics . . . . .	50
6.2 Experiments . . . . .	51
6.2.1 Experiment Steps . . . . .	52
6.2.2 Training a Classifier on Synthetic Document Images . . . . .	52
6.2.3 Training a Classifier on Faxified Document Images . . . . .	54
6.2.4 CycleGAN Training . . . . .	57
6.2.5 Training a Classifier on CycleGAN Generated Document Images . . . . .	58
6.3 Results . . . . .	61
6.3.1 Quantitative Evaluation . . . . .	61
6.3.2 Qualitative Evaluation . . . . .	63

6.3.3 Failure Cases . . . . .	65
<b>7 Conclusion and Future Work</b>	<b>67</b>
7.1 Overview . . . . .	67
7.2 Conclusion . . . . .	67
7.3 Future Work . . . . .	68
<b>A Appendix</b>	<b>70</b>
A.1 GAN Training . . . . .	70
A.2 Examples of Document Images . . . . .	71
A.3 Classifier Architecture Diagram . . . . .	74
A.4 Generator Model Summary . . . . .	75
A.5 Discriminator Model Summary . . . . .	81

# List of Figures

1.1	Relationship between Artificial Intelligence, Machine Learning and Deep Learning. . . . .	8
1.2	The transfer of knowledge from one domain to another domain. . . . .	9
1.3	An illustration of difference between domain adaptation and traditional machine learning.	10
1.4	Simple example of domain adaptation from Street View House Numbers (SVHN) dataset to Modified National Institute of Standards and Technology database (MNIST) dataset for the digit recognition task. . . . .	11
1.5	Example of an real document image from the Arm class. . . . .	12
1.6	Example of an real document image from the Bein class. . . . .	12
1.7	Examples of the paired training dataset and the unpaired training dataset. . . . .	13
1.8	Example of a template. . . . .	14
1.9	Examples of handwritten crops. . . . .	14
1.10	An illustration of the problem that this thesis will attempt to answer. . . . .	15
1.11	An illustration of the proposed solution for reducing the domain gap between synthetic and real document images. . . . .	16
2.1	Evolution of GANs Over the Years. . . . .	19
2.2	An illustration of training a Conditional Adversarial Network (cGAN) to map edges → photo transformation. . . . .	20
2.3	An illustration of CycleGAN transforming the noisy images into clean images and vice versa. . . . .	21
2.4	An illustration of CycleGAN transforming an image from one into the other and vice versa. .	22
3.1	Intuitive example of GAN training progress. . . . .	24
3.2	An illustration of GAN architecture. . . . .	25
3.3	An illustration of GAN converging to match generated data distribution $p_g$ to real data distribution $p_{data}$ . . . . .	26
3.4	An illustration of the training of the discriminator $D$ using backpropagation. . . . .	27
3.5	An illustration of the training of the generator $G$ using backpropagation. . . . .	28
3.6	An illustration of Convolutional Neural Network (CNN) architecture and its training process. . . . .	29
3.7	A Convolution Operation With Zero Padding. . . . .	30
3.8	An illustration of Convolution Operation. . . . .	31
3.9	An illustration of artificial neuron. . . . .	32
3.10	Simple neural network. . . . .	33
3.11	Most common nonlinear activation functions used while constructing Neural Networks.	33
3.12	An illustration of Max Pooling Operation. . . . .	34
4.1	An illustration of the proposed image-to-image translation application using CycleGAN, for transforming synthetic document images into real document images and vice versa. .	36
4.2	An illustration of CycleGAN model with its mapping functions $G$ and $F$ , and respective discriminators $D_Y$ and $D_X$ with depiction forward and backward cycle-consistency loss.	37
5.1	Examples of handwriting crops from the handwriting number dataset. . . . .	41
5.2	Inserting handwritten crops on empty form templates. . . . .	42
5.3	Templates with minor differences. . . . .	43
5.4	Templates with major differences. . . . .	43
5.5	An illustration of ResNet blocks in CycleGAN generator architecture. . . . .	44
5.6	Steps involved in preprocessing of training images of CycleGAN. . . . .	48
5.7	Steps involved in preprocessing of training images (Only synthetic and faxified document images) of classifiers. . . . .	49
6.1	Epochs vs. Accuracy plot during training a classifier on synthetic document images. . . . .	52
6.2	Epochs vs. Loss plot during training a classifier on synthetic document images. . . . .	52

6.3	Confusion matrix to determine the performance of the classifier trained on synthetic document images and evaluated using real annotated document images (test dataset) . . . . .	53
6.4	Epoch vs. Accuracy plot during training a classifier on faxified document images. . . . .	54
6.5	Epoch vs. Loss plot during training a classifier on faxified document images. . . . .	54
6.6	An illustration of faxification process applied on synthetic document images. . . . .	55
6.7	An illustration of faxified document images to conclude that faxification process is a random process, the input images are faxified randomly to create distinct output. . . . .	55
6.8	Confusion matrix to determine the performance of the classifier trained on faxified document images and evaluated using real annotated document images (test dataset). . . . .	56
6.9	Epochs vs. Loss plot during Generator $G$ is training. . . . .	57
6.10	Epochs vs. Loss plot during Discriminator $D_Y$ is training. . . . .	57
6.11	Epochs vs. Loss plot during Generator $F$ is training. . . . .	57
6.12	Epochs vs. Loss plot during Discriminator $D_X$ is training. . . . .	57
6.13	Epochs vs. Accuracy plot during training a classifier on CycleGAN generated document images. . . . .	59
6.14	Epochs vs. Loss plot during training a classifier on CycleGAN generated document images. . . . .	59
6.15	Examples of DE_LY_Bein_2019-01 and DE_LY_Bein_2019-07 real document images. . . . .	59
6.16	Confusion matrix to determine the performance of the classifier trained on CycleGAN generated document images and evaluated using real annotated document images (test dataset). . . . .	60
6.17	Comparison of accuracies and F1-scores, when the classifiers trained on different data distributions and evaluated on testing dataset (Annotated real document images). . . . .	62
6.18	Synthetic document images are transformed into realistic document images by the proposed image-to-image translation application. . . . .	63
6.19	Synthetic document images transformed into realistic document images by the image-to-image translation application. . . . .	64
6.20	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. . . . .	65
6.21	The CycleGAN generated document image consists of noisy artifacts that are unrealistic. . . . .	65
6.22	The CycleGAN generated document image consists of dark border. . . . .	66
6.23	There are some artifacts appear in the generated image that are not present in the synthetic image. . . . .	66
A.1	An illustration of training of the GAN as per the algorithm. . . . .	70
A.2	Example of template. . . . .	71
A.3	Example of real document image. . . . .	72
A.4	Examples of faxified document image. . . . .	73
A.5	Classifier model summary. . . . .	74
A.6	Generator model summary. Continue to next page. . . . .	75
A.7	Generator model summary. Continue to next page. . . . .	76
A.8	Generator model summary. Continue to next page. . . . .	77
A.9	Generator model summary. Continue to next page. . . . .	78
A.10	Generator model summary. Continue to next page. . . . .	79
A.11	Generator model summary. Ends here. . . . .	80
A.12	Discriminator model summary. . . . .	81

# List of Tables

5.1	Size of datasets used for training CycleGAN and classifiers. . . . .	42
5.2	Number of images in each class of annotated real document images dataset. . . . .	42
5.3	Generator architecture . . . . .	45
5.4	Discriminator architecture . . . . .	46
5.5	Classifier architecture . . . . .	47
6.1	Classification report, evaluating a classifier on the testing dataset after training with synthetic document images. . . . .	53
6.2	Classification report, evaluating a classifier on the testing dataset after training with faxified document images. . . . .	56
6.3	Classification report, evaluating a classifier on the testing dataset after training with CycleGAN generated document images. . . . .	60
6.4	The accuracies and F1-scores when the classifiers trained on different data distributions and evaluated on testing dataset (Annotated real document images). . . . .	61

# List of Algorithms

1	GAN training algorithm [1]. . . . .	28
2	CycleGAN training algorithm. . . . .	40

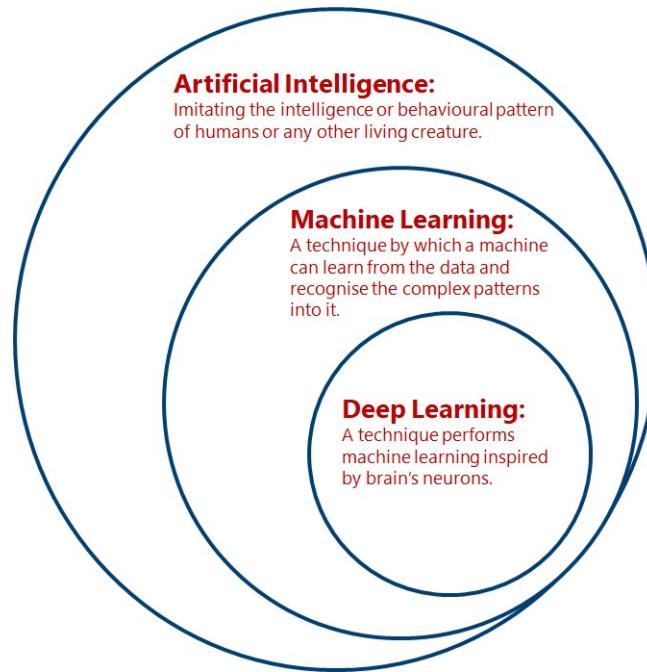
# List of Abbreviations

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

# 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 [2]. AI has a presence in many sectors like Healthcare [3], Autonomous Vehicles [4], Robotics [5], Space Exploration [6] and Computer Vision [7]. This is largely due to the research in Machine Learning (ML) and Deep Learning (DL). Machine Learning is a subclass 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 subclass 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.

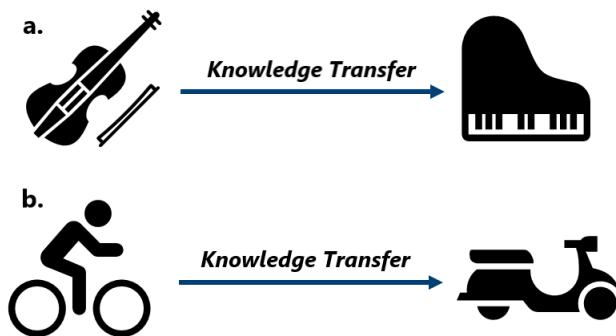
The concept of deep learning was invented as early as the 1950s, although it was largely ignored until the 1980s and 1990s. However, since the last decade, it has become a popular research topic among leading AI research institutions, organizations and startups because of fast GPUs and abundant training data. 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 Artificial Neural Networks (ANNs). ANNs are the multi-layered structure of algorithms and 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., Amazon's Alexa, Microsoft's Cortana, Apple's Siri 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. Two main things that can cause difficulties are bad algorithms and bad data. Let's start with some examples of bad data. Insufficient training data is one of the major problems faced while training deep learning models. Collecting training data is difficult and time-consuming. A large amount of training data is required for the training of most deep learning models to perform efficiently and accurately. 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 and perform well if the training data has more relevant features than irrelevant features. Hence, it is often worth putting effort and spend time cleaning up training data. Most of the Machine Learning Engineers spend significant amount of time doing the same.

Now, after some examples of bad data, let's have a look at some examples of bad algorithms. Overfitting and underfitting are one of the main problems of deep learning. In the case of overfitting, the 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 the methods used to decrease the risk of overfitting is regularization [8]. Regularization helps models to generalize better to the new examples. In the case of underfitting, the model neither learns training data nor generalize to the unseen data. The problem of underfitting is resolved by choosing a powerful model with more parameters and providing better features during the model learning process [9]. The training of the deep learning model is highly influenced by the quality and quantity of the data 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

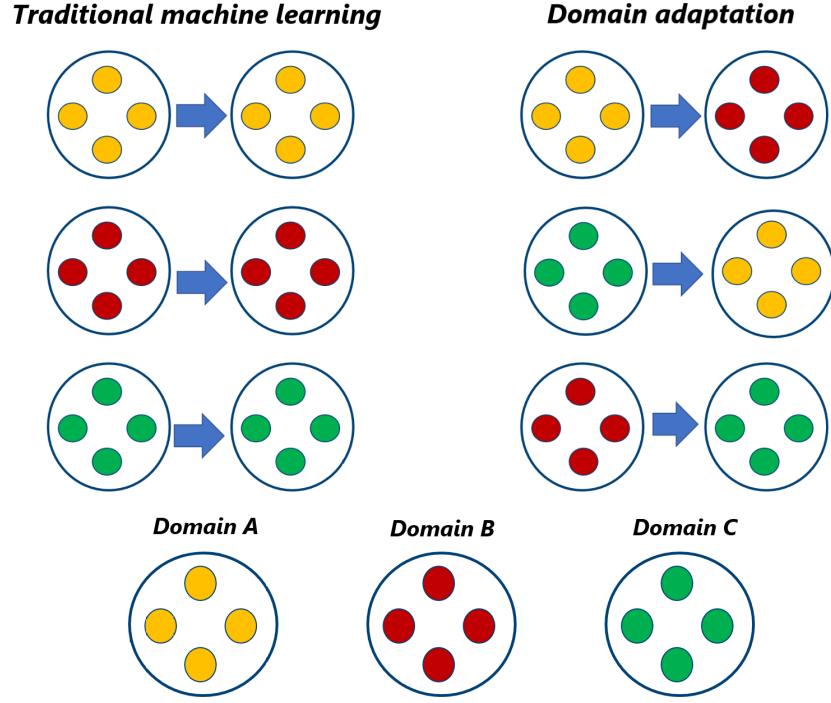
Deep learning methods have many applications in several fields, including natural language processing and computer vision. However, these methods are still limited by poor generalization due to the insufficient quantity of training data [10]. Annotated data are scarce while developing deep learning models 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 classes of data. 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 [11], Data Augmentation [12], Transfer Learning [13] and Domain Adaptation [14].



**Figure 1.2:** The transfer of knowledge from one domain to another domain. a) Transferring the knowledge of playing the violin to play the piano. b) Transferring the knowledge of riding the bicycle to ride the scooter.

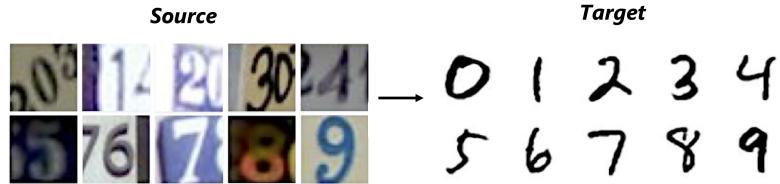
This thesis aims to solve the data scarcity problem using the domain adaptation method. Domain adaptation is a subclass of transfer learning in which the task is to transfer the knowledge from the source domain to the target domain. Transfer learning is a capability of a system that applies learned knowledge from one domain to another [13]. 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. An intuitive example of transfer learning illustrated in figure 1.2. Transfer learning is inspired by the learning process of humans. Humans are capable to transfer knowledge from one domain to another domain. The transfer learning grasps knowledge from one domain to improve the learning performance in another 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. [15] have performed survey on transfer learning, more information about the transfer learning can found in their research paper.



**Figure 1.3:** An illustration of difference between domain adaptation and traditional machine learning.

The source and target domains, and the tasks in the source and target domains, are the same in traditional machine learning. In domain adaptation, the source and target domains are distinct but related, and tasks required to be solved in the source and target domains are the same [15]. The difference between traditional machine learning and domain adaptation is illustrated in figure 1.3. The domain adaptation is also called transductive transfer learning [15]. It is distinguished depending upon the similarity or dissimilarity of feature space and availability of labeled data in the source domain and target domains. The domain adaptation has two categories, if the feature space is the same between the source 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 and unsupervised domain adaptation. In the supervised domain adaptation, the samples in the target domains are labeled. Semi-supervised domain adaptation has a small set of labeled and unlabeled samples in the target domain. And, in unsupervised domain adaptation, the samples in the target domain are not labeled [15]. A simple example of domain adaptation of SVHN transformed into handwritten digits shown in figure 1.4.



**Figure 1.4:** Simple example of domain adaptation from SVHN dataset [16] to MNIST dataset<sup>1</sup> for the digit recognition task.<sup>2</sup>

To understand how domain adaptation works, let's discuss a simple example. Consider the source domain represented by the SVHN dataset. It is a collection of house numbers images and the target domain is represented by the MNIST dataset, which is a collection of handwritten digits images. When the CNN is trained and evaluated on the source domain SVHN dataset for the task of identifying numbers, it will achieve high accuracy. However, the same classifier will perform worst when evaluated on 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 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 sufficient amount of images are 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) [17], Image Classification [15], Style Transfer [18] and Optical Character Recognition (OCR) [19] to solve the problem of scarcity of data. Making such applications robust, accurate and efficient is a big task, which requires years of research. The scope of this thesis is limited to improving document image classification using the proposed domain adaptation technique. The proposed domain adaptation technique uses image-to-image translation method to transform synthetic document images into realistic document images.

---

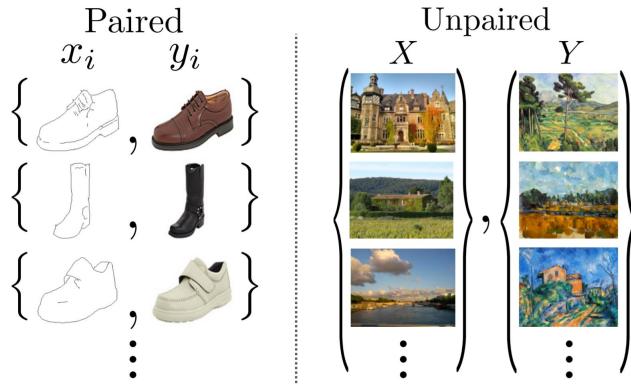
<sup>1</sup><http://yann.lecun.com/exdb/mnist/> last access: November 23, 2021

<sup>2</sup><https://machinelearning.apple.com/research/bridging-the-domain-gap-for-neural-models> last access: November 23, 2021



### 1.3 Problem Statement and Proposed Solution

As described earlier, a huge chunk of training data is required to train a neural network, but annotated data is scarce and data labeling is a costly and tedious job. For example, in this thesis, it is being addressed that collecting several and distinct real document images (figures 1.5 and 1.6) with different types of handwriting is expensive and difficult job. In such cases, machine learning engineers have to inevitably generate synthetic document images. However, deep learning models trained using synthetic document images will not generalize well on unseen real document images (figure 1.10), because the synthetic document images lack realism [10]. They don't possess a similar noise distribution, characteristics and artifacts as real document images [10]. Hence, in the last two decades, numerous domain adaptation methodologies have been introduced to perform image-to-image translation [19]. The image-to-image translation is a method of transforming images from source to a target domain by learning the underlying characteristic of target domain images. Most of the time, the image-to-image translation models are trained using the paired training dataset. However, for many tasks, paired training datasets do not exist. Example of paired and unpaired training datasets is illustrated in figure 1.7.



**Figure 1.7:** The paired training dataset consists of training examples, in which there is a correspondence between the source domain and the target domain (Left, Edges  $\leftrightarrow$  Images). But in the unpaired training dataset, there is no correspondence between the source domain and the target domain (Right, Photographs  $\leftrightarrow$  Paintings) [20].

The image-to-image translation technique has several applications. They are used to transform images of zebra into the horse, images of cats into dogs, images of the day into night and images summer into winter images. They can also be used to transform synthetic images into realistic images by learning to reduce the gap between the distribution of synthetic data and real data. In this thesis, an image-to-image translation application is developed using CycleGAN to reduce the domain gap between synthetic data distribution and real data distribution [20]. The CycleGAN is an extended variant of Generative Adversarial Network (GAN) [21]. It is a method to perform unpaired image-to-image translation, in which the model is trained using the collection of images from the source and target domain that are not related to each other<sup>3</sup>.

The proposed image-to-image translation application is developed 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 them. To make those systems robust and efficient, a large number of document images are required. The goal is to produce a large number of synthetic document images in order to provide a significant amount of annotated data. However, as mentioned earlier, the synthetic document images will not generalize well when they have to process real document images because real document images have different characteristics. Furthermore, real document images possess artifacts such as salt-and-pepper, background noise, blur due to camera motion or shake, watermarks, stains, wrinkles and fading text, introduced during the scanning process [23]. To address this problem, in this thesis, the image-to-image translation application is developed to transform synthetic

---

<sup>3</sup><https://machinelearningmastery.com/what-is-cyclegan/> last access: November 23, 2021

document images into realistic document images to reduce the domain gap between real data distribution and synthetic data distribution. In this way, a large number of realistic document images can be generated to reduce the scarcity of annotated real document images in the target domain.

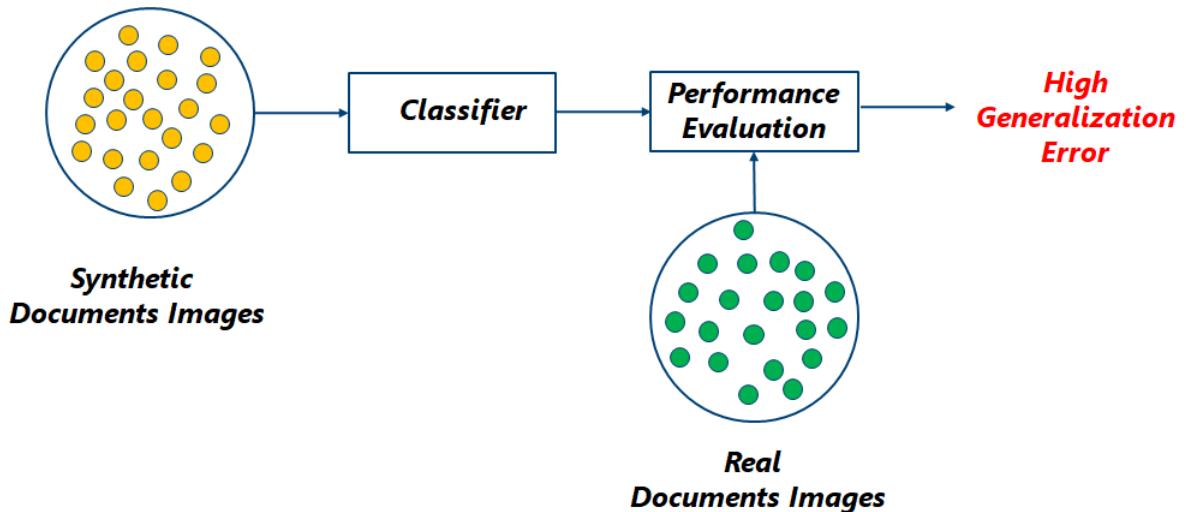
Bestellung <input type="checkbox"/> Nur Kostenvoranschlag		Phlebology
<b>Firma Stempel, Ansprechpartner, Tel. (in Druckschriftziffern)</b>  <b>Angaben zum Patienten</b> Kommission <sup>1</sup>  Frühere Anfertigung / KV-Nr. / Datum:  <b>Anzahl:</b> <input type="checkbox"/> Stück <input type="checkbox"/> Paar <input type="checkbox"/> Weiblich <input type="checkbox"/> Männlich <b>Seitenzahl:</b>		
<b>Phlebologie</b> <input type="checkbox"/> Zutreffendes bitte ankreuzen		
<b>Material</b> <b>18 - 21</b> <b>23 - 32</b> <b>34 - 46</b> <b>mmlg</b> <b>mmlg</b> <b>mmlg</b> <b>KKL 1</b> <b>KKL 2</b> <b>KKL 3</b>		
<b>Fine-Art</b> <b>Jazz Inspiration</b> <input type="checkbox"/> 2701 <input checked="" type="checkbox"/> 2702 <input type="checkbox"/> 2901 <input type="checkbox"/> 2902 <b>Active Day</b> <b>Jazz Soft</b> <input type="checkbox"/> 2001 <input type="checkbox"/> 2002 <input type="checkbox"/> 2021 <input type="checkbox"/> 2022 <b>Jazz Energy</b> <b>Jazz Adventure</b> <input type="checkbox"/> 3521 <input type="checkbox"/> 3522 <b>World in Motion</b> <b>Jazz Dynamic</b> <input type="checkbox"/> 3511 <input type="checkbox"/> 3512 <input type="checkbox"/> 3513 <input type="checkbox"/> 3511 <input type="checkbox"/> 3512 <input type="checkbox"/> 3513 <b>Jazz Dynamic Silver</b> <b>Jazz Dynamic Cotton</b> <input type="checkbox"/> 3511 <input type="checkbox"/> 3512		
<b>Ausführung</b> <input type="checkbox"/> AD <input type="checkbox"/> AG <input type="checkbox"/> AG <input type="checkbox"/> Strumpfgeschirr (AT) <input type="checkbox"/> Reitstrumpf Schwangerschaft <input type="checkbox"/> Mutter erwacht im Monat <input type="checkbox"/> Lebt wie Strumpfgerüst (Möglich bei Jazz Dynamic KKL 2 und 3) <input type="checkbox"/> Einheitsstrumpfphase <sup>2</sup>		
<b>Farbe</b> Wenn nichts vemerkt, wird Farbe Mandi gewählt. Weiss = Weiss, Schwarz = Schwarz, Umriss = Umriss <input type="checkbox"/> Zuck <input type="checkbox"/> Creme <input type="checkbox"/> Melone <input type="checkbox"/> Markt <input type="checkbox"/> Zint <input type="checkbox"/> Kakao <input type="checkbox"/> Möhre <input type="checkbox"/> Blaubeere <input type="checkbox"/> Wacholder <input type="checkbox"/> Pfeffer <input type="checkbox"/> Salz/Pfeffer		
<b>Trend Colour</b> <b>Fashion Colour</b> <b>Batik Collection</b> <input type="checkbox"/> Batik-Weiß <input type="checkbox"/> Batik-Schwarz <b>Dip Dye Collection</b> <input type="checkbox"/> Blaubeere <input type="checkbox"/> Möhre		
<b>Fußspitze</b> <input type="checkbox"/> Offen <input type="checkbox"/> Geschlossen		
<b>Abschluss / Befestigung Strümpe</b> <input type="checkbox"/> Trichterabschluss <input type="checkbox"/> Noppenabschluss AD (höhe 15 cm) <input type="checkbox"/> Balanceabschluss AD (höhe 15 cm)		
<b>Zusätzlich für AF/AG:</b> <input type="checkbox"/> Noppenabschluss 5 cm <input type="checkbox"/> Noppenabschluss 10 cm <input type="checkbox"/> Sprenkelauftrand (höhe 5 cm) <input type="checkbox"/> Sprenkelauftrand (höhe 10 cm) <input type="checkbox"/> Confortabschluss (höhe 5 cm - nur für Fine Art und Jazz Soft) <input type="checkbox"/> Balanceabschluss 5 cm <input type="checkbox"/> Balanceabschluss 10 cm Hartbefestigung (Anwendung „P“ angeben) <input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Als Paar zu tragen Nichts = Keine Befestigung und Jazz Soft		
<b>Angewählte Leibteil (aus Loyer)</b> <input type="checkbox"/> KKL 1 <input type="checkbox"/> KKL 2 <b>Nach Maß</b> <b>Serial (für neue Modelle - Größen)</b> <b>Individuelle Sonderausstattungen (AT)</b> <input type="checkbox"/> Schnitt offen <input type="checkbox"/> Schnitt (Nur für Jazz Inspiration und Jazz Sprint)		
<b>Zubehör</b> <input type="checkbox"/> Easy Fit <input type="checkbox"/> Easy Fit XL <input type="checkbox"/> Adhäsionsstreifen <input type="checkbox"/> Vital Balsam? <input type="checkbox"/> Spezialwaschmittel <b>Spezialhandschuhe</b> Größe <input type="checkbox"/> S (S) <input type="checkbox"/> M (L) <input type="checkbox"/> L (XL) <input type="checkbox"/> XL (XXL) <input type="checkbox"/> Jazo Arion Easy Slide Bein <input type="checkbox"/> Jazo Arion Magne		
Anmerkungen (in Druckschriftziffern):  <small>1 Wird der Patientenname angegeben, bestätigt die bestellende Firma, dass die rechtskonforme Erweiterung zur Weitergabe und Verarbeitung der Daten von dem betroffenen Patienten zuvor eingeholt werden ist. 2 Nur für Dynamic, Dynamic Silver und Dynamic Cotton erhältlich</small>		

Figure 1.8: An example of a template (figure reproduced from elevait GmbH & Co. KG with permission).

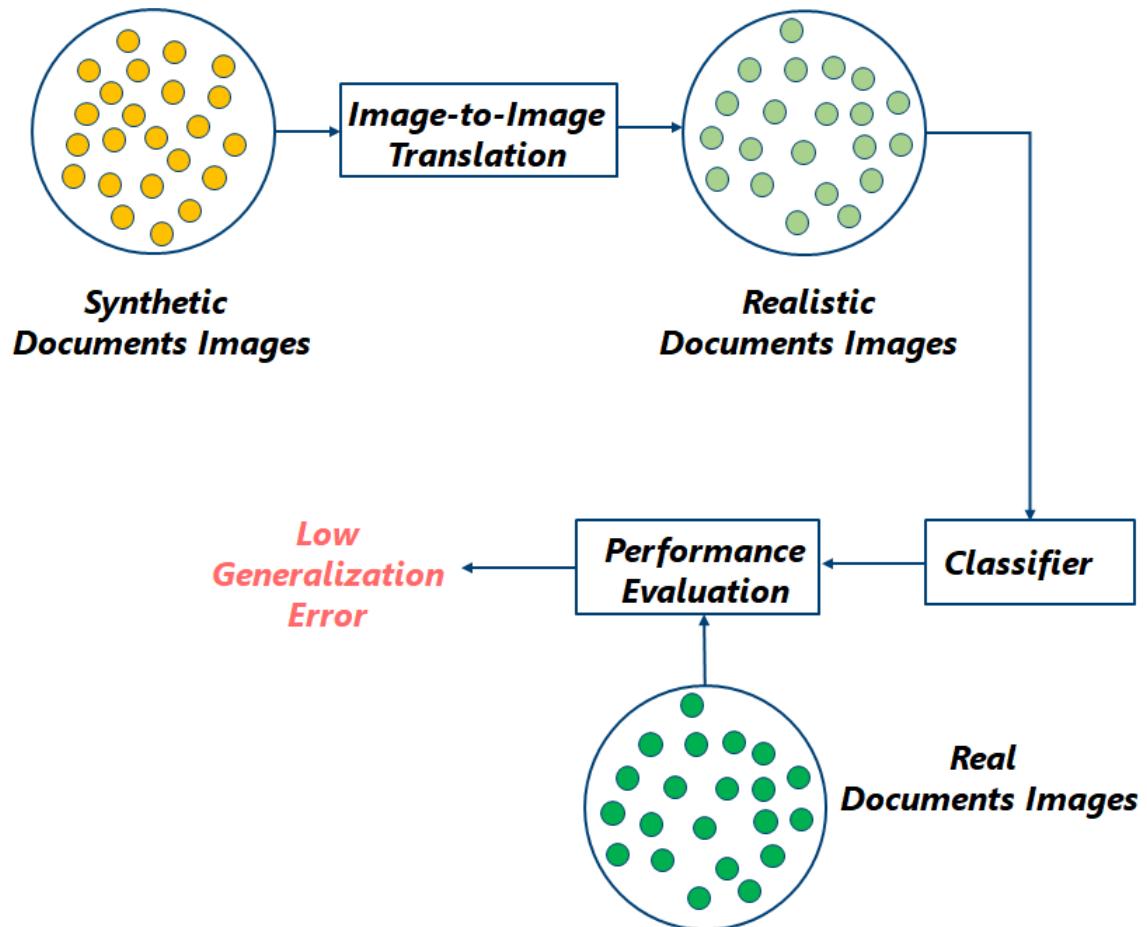
67 467 116 964  
 48,9 6,8 65,8  
 54,5 38,3 0,4  
 46,1 156 17  
 99,7 4869 463  
 38,4 7,4 205,7

Figure 1.9: Examples of handwritten crops (figure reproduced from elevait GmbH & Co. KG with permission).

The synthetic document images are created using templates (figure 1.8) and handwritten crops (figure 1.9) retrieved from handwriting datasets. These templates contain fields like customer number, name and other information. These fields are represented by using bounding boxes. The bounding boxes are annotated using Common Objects in Context (COCO) annotations [24]. The handwritten crops are inserted over the empty templates to generate numerous synthetic document images. The proposed image-to-image translation application solves the following problems. a) Once a large amount of labeled synthetic document images are created in the source domain, next, an image-to-image translation application is trained using synthetic and real document images. Further, the collection of synthetic document images is transformed into realistic document images in the target domain. So the problem of data labeling and data scarcity is solved in the target domain. b) Consider, a large dataset of real document images is not labeled. If the simple classifier is trained on realistic document images that are generated using proposed image-to-image translation application. Further, the dataset 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 1.10 illustrates if the classifier is trained using synthetic document images, it will not generalize well over the unseen real document images. Further, it leads to high generalization error, hence represented in dark red color for better understanding. In figure 1.11 data distribution in light green color represents realistic document images which are generated by the proposed image-to-image translation application. Dark green represents the real document images. The classifier is trained using realistic document images, 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 when compared with figure 1.10.



**Figure 1.10:** An illustration of the problem that this thesis will attempt to answer. The classifier trained using synthetic document images (Yellow) will not generalize well on real document images (Green), producing high generalization error (Red).



**Figure 1.11:** An illustration of the proposed solution for reducing the domain gap between synthetic and real document images. The classifier trained using realistic document images (Light green) produces low generalization error (Light red) when evaluated using real document images (Green).

## 1.4 Thesis Objectives

The goal of this thesis is to reduce the domain gap between synthetic data distribution and real data distribution by developing an image-to-image translation application using CycleGAN, which 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 main objectives of this thesis are listed below:

1. Conduct a literature survey, in which different types of image-to-image translation methods are discussed and briefly explained. In addition, also discuss a theoretical comparison between existing methods, which lead to the reason for choosing CycleGAN to solve the problem statement.
2. Create a dataset of synthetic document images, faxified document images and real document images. In addition, a testing dataset of annotated real document images is collected for evaluation of the models.
3. Implement proposed image-to-image translation application using CycleGAN. Next, train the application model using a dataset of synthetic document images and real document images. A dataset of synthetic document images represents the source domain and a dataset of real document images represents the target domain.
4. Train three different classifiers on different data distributions (Synthetic document images, Faxified document images and CycleGAN generated document images) and evaluate their performance on unseen annotated real document images (testing dataset) and analyze the domain gap between these distributions and real data distribution, using metrics like accuracy, macro average F1-score and weighted average F1-score. These experiments are listed briefly below:
  - (a) Train a classifier using synthetic document images and evaluate its performance over testing dataset.
  - (b) Train a classifier using faxified document images and evaluate its performance over testing dataset.
  - (c) Train a classifier using CycleGAN generated document images and evaluate its performance over testing dataset. This experiment reveals how well the CycleGAN generated document images are generalizing to the real document images, eventually determining the quality of images generated by the proposed image-to-image translation application.
5. Record all details regarding this thesis, for example, literature survey, selected methodology, fundamentals required to understand this thesis, implementation, experiments performed, obtained results, analysis of results, limitations and scope of the research, 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 constraints. The solution of the defined problem and analysis performed in this thesis investigated only using CycleGAN. The comparison with other methodologies is kept apart for future research. The scope of this thesis is limited to the improvement of real document images classification by increasing the quality and quantity of annotated images in the target domain using the proposed image-to-image translation application. The thesis is organized as follows. Chapter 2 describes related literature and existing methods to solve the problem. Chapter 3 discusses essential topics GANs and CNNs. Chapter 4 describes the method and loss functions used to solve the defined problem. Chapter 5 describes the details about the dataset, the architecture of the neural networks implemented in this thesis. Chapter 6 illustrates the evaluation metrics, experiments performed and obtained results. Chapter 7 concludes the research and analysis of results along with the limitations and the future work of the thesis.

## 2. Related Works

In the last decades, the field of computer vision and computer graphics has made a tremendous amount of progress, especially in the field of domain adaptation. The image-to-image translation is a popular and classic example of domain adaptation, in which the goal is to learn the mapping between a source image and a target image using a training set of aligned image pairs [25]. GANs were among the initial methods used to perform image-to-image translation [26] [21]. But, GANs suffers from unstable learning and mode collapse [27]. Also, for many tasks, aligned or paired training data will not be available, collecting and annotating paired document images is tedious, difficult, time-consuming and costly. Hence, over the years, several supervised and unsupervised image-to-image translation methods are developed. This chapter discusses some of those popular image-to-image translation methods. Also, a brief theoretical comparison of these existing methods is discussed. The motivation behind selecting the approach to solve the addressed problem statement of this thesis is summarized in section 2.2.

### 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 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 discriminator penalizes the generator 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 [28], Toronto Face Database (TFD) [29] and CIFAR-10 [30], also compared against existing methods like Deep Belief Network (DBN) [31], Stacked Convolutional Autoencoder (Stacked-CAE) [31] and Deep Generative Stochastic Networks (Deep-GSN) [32]. The authors do not claim that the samples generated by GANs are better than samples generated by already existed methods [21]. Authors believe at least the obtained results are competitive when compared with the better generative models in the literature, which highlights the potential of the generative adversarial framework [21]. The first and foremost advantage of GANs is computational when compared to other generative models. 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 [2]. Authors mentioned, to prevent the Helvetica Scenario [33], special care must be taken while training the GANs. The generator must not be trained too much without updating the discriminator, training of both models should happen simultaneously. The discriminator must be updated well to provide maximum error to update the generator to produce high-quality images. Usually, GANs should produce a wide variety of outputs. In the Helvetica Scenario, the generator collapses, repeatedly producing the same output or a small set of outputs. The Helvetica Scenario is also called Mode Collapse [27].

Xudong Mao et al. [35] proposed another variant of GANs called Least Squares Generative Adversarial Networks (LSGANs). In regular GANs, the discriminator is hypothesized to be a classifier with the sigmoid cross-entropy loss function [35]. They realized the cross-entropy loss function can cause vanishing gradients problems during the learning process [35]. Also, regular GANs are unstable during the learning process. Hence, the authors proposed LSGANs to solve the problem of vanishing gradi-

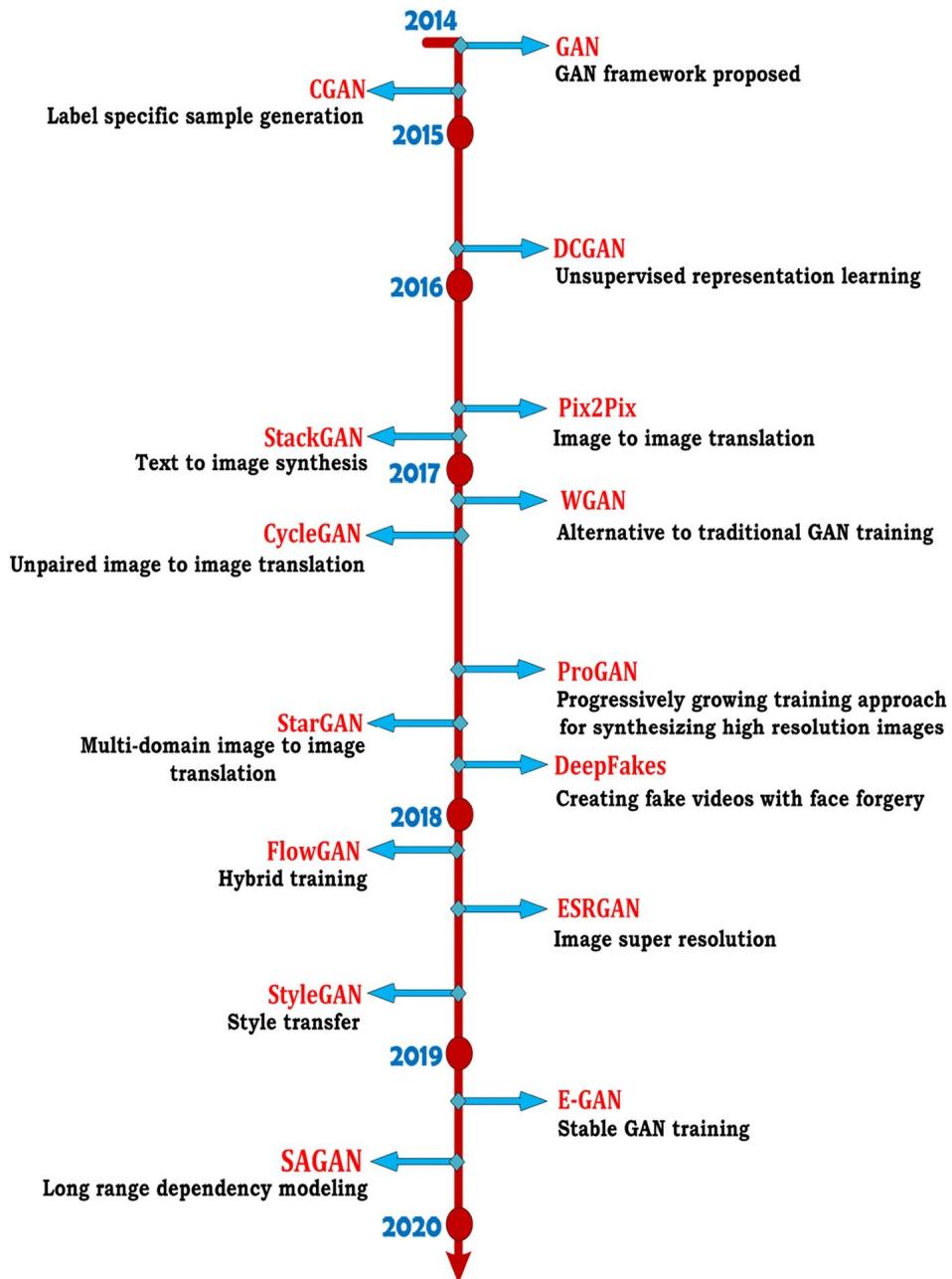
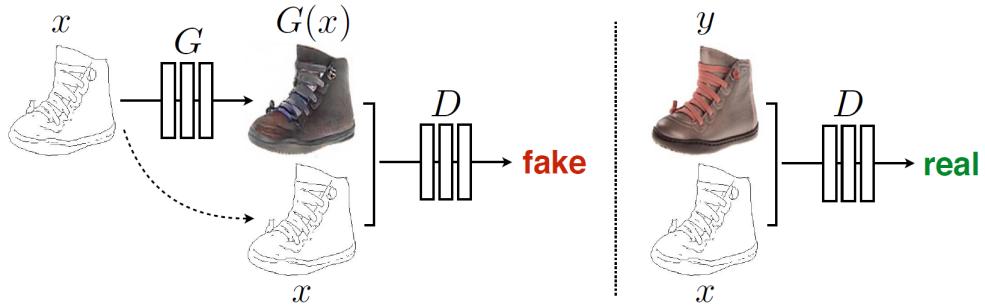


Figure 2.1: Evolution of GANs over the years [34].

ents and instability, during the learning process. LSGANs use the least-squares loss function in the discriminator to solve the problem of vanishing gradients. In LSGANs, the fake samples are penalized by the least-squares loss function, which forces the generator to generate samples closer towards to the decision boundary [35]. The authors evaluated LSGANs using two datasets LSUN [36] and HWDB1.0 (Handwritten Chinese Character Dataset) [37]. They discovered, when trained on the LSUN dataset [36], LSGANs generates better quality images than regular GANs, Deep Convolutional Generative Adversarial Networks (DCGANs) [38] and Energy-based Generative Adversarial Networks (EBGANs) [39]. 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. [40]. In this experiment, a regular GAN and LSGANs trained on a Two-dimensional (2D) mixture of 8 Gaussian datasets using a simplistic architecture of both generator and the discriminator, which have three fully-connected layers [35]. LSGANs learn the Gaussian mixture distribution effectively and generate a wide variety of samples, but regular GANs suffer from mode collapse and generate the single type of samples of the data distribution [35] and the learning process of LSGANs is stable compared to the regular GANs.

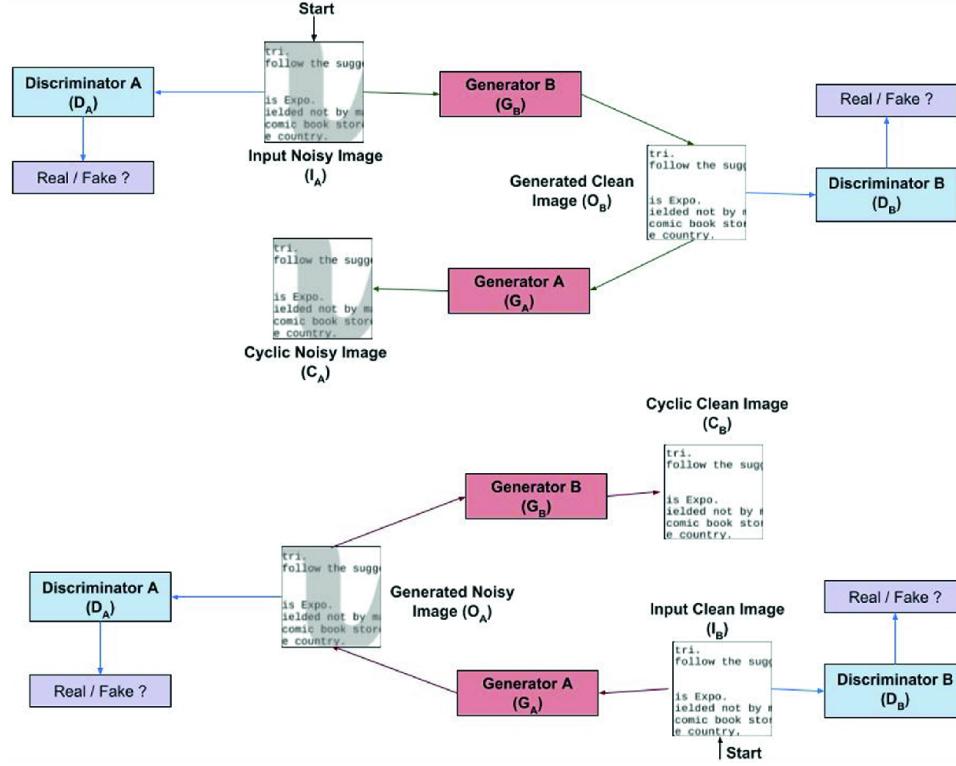
Phillip Isola et al. [41] proposed cGANs as a generic solution to numerous image-to-image translation problems. Authors wanted to provide a single solution for multiple types of image-to-image translation problems. cGANs learn to map input images to output images using paired training dataset. Traditionally different loss formulations are required to solve different image-to-image translation problems, but cGANs can be used as a common generic method to solve distinct image-to-image translation problems [41]. For example, synthesizing photos from label maps [42], reconstructing objects from edge maps [43] [44] and colorizing images [45]. In cGAN the generative model uses U-Net-based architecture, the U-Net is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks [46]. The discriminative model is constructed using a convolutional PatchGAN classifier architecture [47]. PatchGAN is simply a CNN in which each overlapping patch in an image is estimated to be fake or real. It penalizes structure of an image at the scale of image patches [48]. In cGANs, both generative and discriminative models are conditioned on some additional information, such as input images and class labels. But unconditional GANs are not conditioned on any auxiliary information [41]. Authors found, L1 loss encourages less blurring compared to the L2 loss [41]. 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 [41]. Authors have released cGAN as the pix2pix software and mentioned, cGAN has widespread applications and it can be adopted without ease, to solve different image-to-image translation problems without significant parameter modifications. 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 additional information like input edge map [41].



**Figure 2.2:** An illustration of the training a cGAN to map edges → photo transformation. In cGANs, both discriminator and generator observe the input edge map, unlike Unconditional GANs [41].

Taesung Park et al. [49] proposed Contrastive Unpaired Image-to-Image Translation (CUT) for encouraging content preservation in unpaired image-to-image translation problems by maximizing the mutual information between input and output with patchwise contrastive learning [50]. Authors stated, in patchwise contrastive learning, the patchwise contrastive loss is used to minimize the dif-

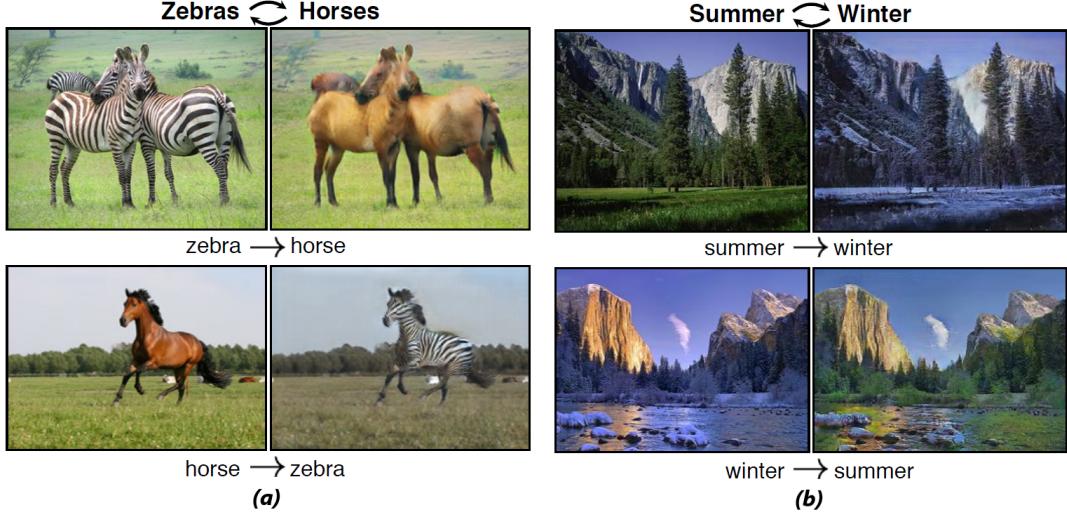
ferences between generated image and the input image, at the level of small patches. The task of patchwise contrastive learning is to bring a patch from the generated image, closer to its corresponding input patch in the input image compared to other patches that are present in the same input image. The patchwise contrastive learning forces to preserve context and content between the input image and generated image by learning from patches from the same input image rather than other images in the training dataset [49]. They demonstrated, the framework enables one-sided translation in the unpaired image-to-image translation setting while improving quality, consuming less memory and reducing training time [49]. The prime advantage of this method is it does not require lots of memory [51] [52] or specialized architecture [53] [54]. The contrastive representation is formulated within the same image, hence this method can even be trained on single images, where each domain is having only a single image [49]. The several prior methods like CycleGAN [20], Multimodal Unsupervised Image-to-image Translation (MUNIT) [55], Diverse Image-to-Image Translation (DRIT) [56] and Geometry-Consistent Generative Adversarial Networks (GCGAN) [57] 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. Furthermore, the authors also introduced faster and lighter variant Fast Contrastive Unpaired Translation (FastCUT). FastCUT is computationally lighter and produces competitive results. The code and models of CUT are available at GitHub.



**Figure 2.3:** An illustration of CycleGAN - It has two generators  $G_A$  and  $G_B$  and two discriminators  $D_A$  and  $D_B$ . The generator  $G_A$  transforms the noisy images into clean images and  $G_B$  transforms clean images into noisy images using cycle-consistency loss [20]. The discriminators  $D_A$  and  $D_B$  rejects samples generated by  $G_A$  and  $G_B$  respectively acting as an adversary [23].

Monika Sharma et al. [23] are 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 introduce several readability challenges in modern text recognition algorithms, decreasing their performance and accuracy [23]. The current denoising techniques are based on a supervised learning approach using paired training datasets, in which it contains noisy documents with their corresponding cleaned versions of the same noisy document [23]. However, in the real world, such a paired training dataset is not available to train a model, to generate clean documents from noisy versions [20][23]. Hence, in this literature, the authors used CycleGAN an unsupervised image-to-image translation technique to transform noisy document images into clean

document images. CycleGAN is a method to learn a mapping between two data distributions in the absence of paired training dataset [20] [23]. They have compared the performance of CycleGAN for document cleaning tasks with 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)<sup>1</sup> as a evaluation metric to evaluate the quality of transformed denoised images. Experiments were performed on 4 separate public document datasets, one each for deblurring, watermark removal, background noise removal and defading. Finally, they concluded CycleGAN learns a more robust mapping from the space of noisy to clean documents compared to cGAN [23]. The complete architecture of CycleGAN for denoising documents illustrated in figure 2.3.



**Figure 2.4:** An illustration of CycleGAN transforming an image from one into the other and vice versa. For example, (a) zebra image transformed into horse image and vice versa and (b) the view of Yosemite mountains in summer transformed into winter and vice versa [20].

Jun-Yan Zhu et al. [20] noticed the scarcity of paired training data to perform image-to-image translation. Hence, they proposed CycleGAN as an unpaired image-to-image translation method that is trained in the absence of paired training dataset to transform the images from source domain  $X$  to target domain  $Y$ . This method learns to capture underlying characteristics of the collection of images in the source domain and figuring out how these characteristics could be transformed into the collection of images in the target domain, all in the absence of any paired training examples [20]. In this method, the goal is to learn a mapping  $G : X \rightarrow Y$  so that the generated distribution of images  $G(X)$  matches the target distribution  $Y$  using an adversarial loss. However, the adversarial losses alone are not enough to map the images from the source domain to the target domain. Hence, authors introduced inverse mapping  $F : Y \rightarrow X$  to propose a cycle-consistency loss to make  $F(G(X)) \approx X$  and vice versa to regularize the output and keep it highly under-constrained and within the context [20]. The cycle-consistency property of an image-to-image translation can be understood by a simple example, if we translate a sentence of English to French, again back from French to English, one should arrive back to the original sentence. In this manner, CycleGAN learns to produce images within the context instead of generating random images<sup>2</sup>. Authors investigated several prior methods like Bi-GAN/ALI [58] [59], CoGAN [60], SimGAN [61] were unable to achieve compelling results. However, CycleGAN often produces images that are equivalent quality to the fully supervised pix2pix software [41]. Authors provided PyTorch implementation of CycleGAN. The figure 2.4 illustrates, examples of CycleGAN transforming an image from one into the other and vice versa.

Chris Tensmeyer et al. [10] 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. There have been efforts to create synthetic data for binarization using image processing techniques but, they generally lack realism [10]. Hence, the authors proposed DGT-CycleGAN to produce realistic synthetic data using an image-to-image translation model CycleGAN which is adversarially trained

<sup>1</sup>Peak Signal-to-Noise Ratio: <http://www.ni.com/white-paper/13306/en/> last access: November 23, 2021.

<sup>2</sup><https://machinelearningmastery.com/what-is-cyclegan/> last access: November 23, 2021

[10]. Authors modified the CycleGAN model to be conditioned on the ground truth binarization mask as it transforms synthetic images into realistic images [10]. They found 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 [10]. Authors pretrained deep learning models on realistic synthetic datasets generated by CycleGAN, DGT-CycleGAN and image compositing [10]. They found pretraining a model on realistic synthetic data generated by the DGT-CycleGAN model achieves the best performance for solving binarization tasks compared to data generated by CycleGAN and image compositing [10]. Also, they evaluated both pretrained only and finetuned models on each of the Document Image Binarization Competition (DIBCO) datasets<sup>3</sup> to measure their performance [10]. Finetuning a model on DIBCO dataset after pretraining on the realistic synthetic images generated by the DGT-CycleGAN has lead to a 13% reduction of F-measure error compared to no pretraining and to 6% error reduction compared to pretraining on non-realistic synthetic data [10]. Authors concluded, pretraining deep neural networks on the realistic synthetic data generated using DGT-CycleGAN lead to better predictive performance both before and after finetuning on real data.

## 2.2 Summary

Over the years, GANs have evolved to solve different kinds of problems (figure 2.1). General GANs suffer from unstable training, vanishing gradients problem and mode collapse. Therefore, Martin Arjovsky et al. [62] proposed Wasserstein GANs (WGANs) to solve problems with GANs. WGANs solve problems like mode collapse, improve learning stability and provide meaningful learning curves beneficial for hyperparameter searches and debugging [62]. Although the implementation of WGAN is straightforward, the theory behind it is difficult and requires modifications, for example, implementation of Weight Clipping [63]. Hence, Xudong Mao et al. [35] proposed a simple and more intuitive method compared to WGAN, called LSGAN. First, LSGANs generate higher quality images than GANs [35]. Second, its learning process is stable compared to GANs [35]. Monika Sharma et al. [23] demonstrated CycleGAN can transform noisy document images into denoised and clean document images. They demonstrated CycleGAN transforms noisy document images into clean document images superior to cGAN. Chris Tensmeyer et al. [10] proposed DGT-CycleGAN, a modified version of CycleGAN, which is adequate to transform synthetic images into realistic images to solve binarization tasks. Taesung Park et al. [49] proposed CUT. They have demonstrated that CUT is a better, faster and memory-efficient approach to perform unsupervised image-to-image translation. Further, Jun-Yan Zhu et al. [20] proposed an unsupervised image-to-image translation approach CycleGAN to address the problem of scarcity of paired training data while training image-to-image translation applications. Compared to GANs, CycleGANs also can deal more meticulously with the problems like unstable training, vanishing gradients problems and mode collapse. In this thesis, the image-to-image translation model is developed using LSGAN and CycleGAN. The generators and discriminators are optimized using cycle-consistency loss [20] and least-square loss [35]. The proposed method is simple and capable to solve problems in general GANs. Also, it learns a function of image-to-image translation in the absence of paired training data using cycle-consistency property [20].

---

<sup>3</sup>DIBCO 2019: <https://vc.ee.duth.gr/dibco2019/> last access: November 23, 2021

### 3. Fundamentals

This chapter focuses on improving the understanding of the fundamental concepts required to understand this thesis. Section 3.1 describes architecture of GANs. Section 3.2 describes neural network layers, namely convolution layer, activation layer, pooling layer and fully connected layer.

#### 3.1 Generative Adversarial Networks (GANs)

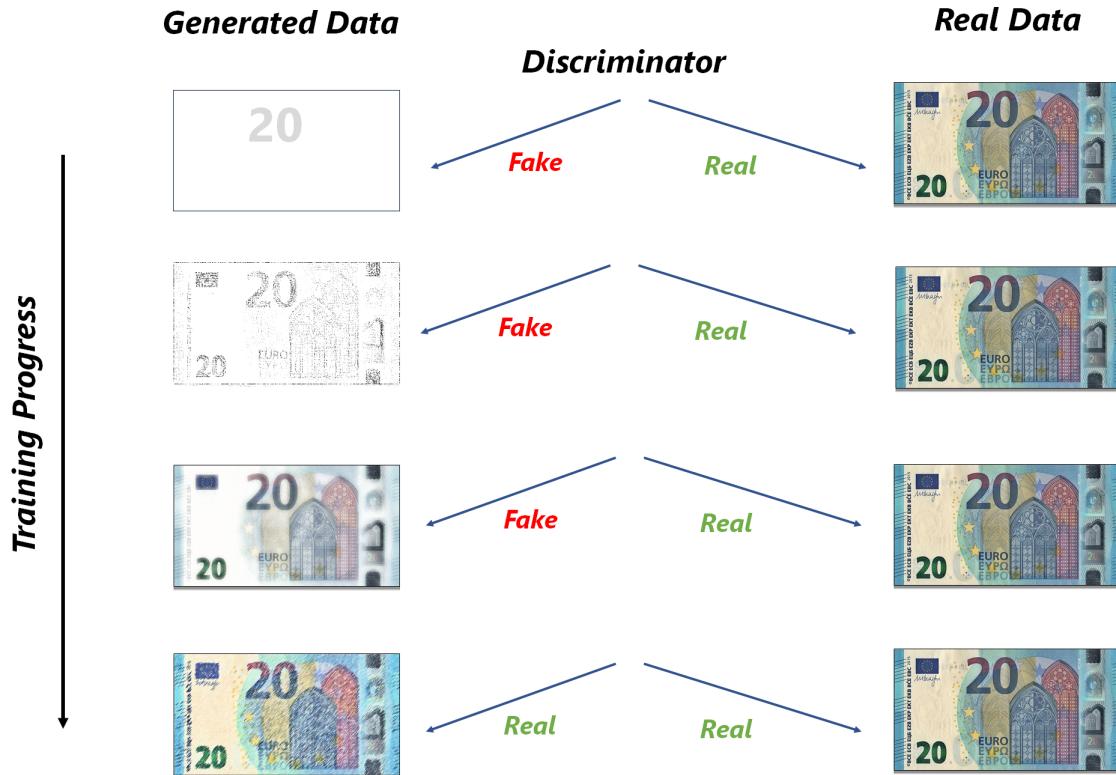
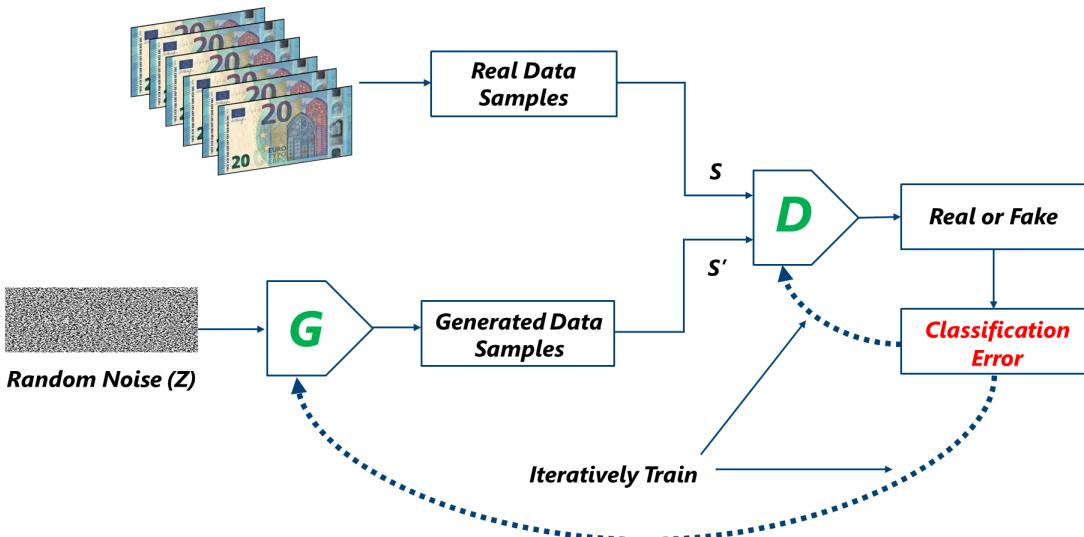


Figure 3.1: Intuitive example of GAN training progress.

GANs have two models in their architecture first is the generator and the second is the discriminator and both models can be implemented using multilayer perceptrons [21]. Let's try to understand the function of GANs intuitively. Consider generator is a forger who creates fake currencies, with the intention of making them realistic as much as possible. Further, the discriminator is an expert who distinguishes forged and real currencies. 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. The discriminator learns by the feedback error after misclassifying real images as fake and fake images as real. The generator also learns by the feedback error given by the discriminator, after it successfully classifies image produced by the generator as fake. To produce a better quality of fake images, the generator model is updated using the feedback error given by the discriminator. It is a setup where both generator and discriminator are competing with each other. An intuitive example of GAN training process is illustrated in figure 3.1. It depicts, during the initial stages of the training, the generated images by the generator are easily distinguished as fake by the discriminator. After training iterations, the quality of the generated images by

the generator increases. The generator learns using feedback error given by the discriminator. Finally, the generator learns to produce realistic images and the discriminator starts to classify them as real.

Now, let's have a look into the GAN's architecture. As described earlier, the task of the GAN is to generate fake samples that are realistic. The training dataset of real data samples  $S$  serves as an input to the discriminator  $D$ . The random noise vector  $Z$  retrieved from the random noise distribution serves as an input to the generator  $G$  to produce fake samples. The generator  $G$  takes  $Z$  as an input and outputs a fake sample  $S'$ . Its goal is to create fake samples that are realistic as much as possible and indistinguishable from the real samples from the training dataset. The discriminator  $D$  takes input from real data sample  $S$  that are present in the training dataset and fake samples  $S'$  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  $S$  as real and  $S'$  as fake. The generator's weights are updated to produce quality samples and maximize the probability that the discriminator makes mistake and misclassifies  $S'$  as real. GAN in action is illustrated in figure 3.2.



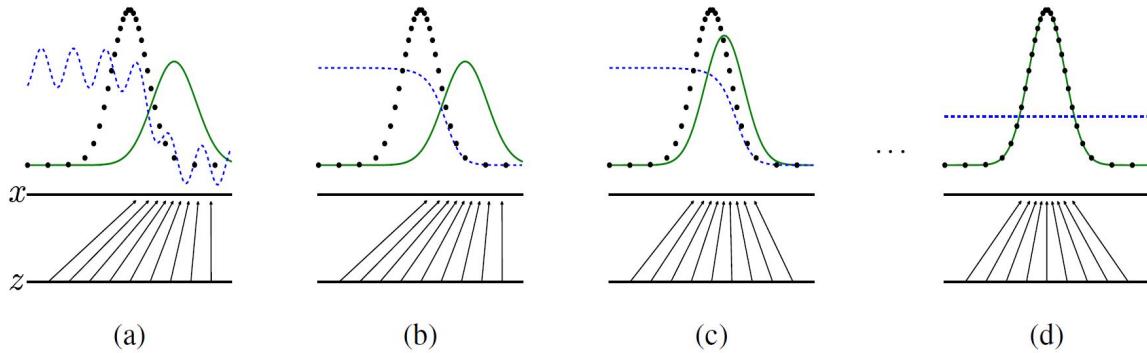
**Figure 3.2:** An illustration of GAN architecture along with the generator's and discriminator's inputs, outputs and their interactions.

Let's try to understand the mathematics behind 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 differential functions. They are represented by multilayer perceptrons with parameters  $\theta_g$  and  $\theta_d$  respectively [21]. The mapping function from some representation space, called, the latent space to the space of data, is represented by  $G(z; \theta_g)$ . The  $D(x; \theta_d)$  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. For a fixed  $G$ , 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 a certain number of training iterations, when the discriminator is optimally optimized, it can be fixed and the generator  $G$  advances to learn by the error provided by the discriminator to produce plausible data. At the end of the training, if the generator can match the real data distribution, then the discriminator is maximally confused, will predict 0.5 probability for its inputs and mathematically  $G$  is being trained to minimize  $\log(1 - D(G(z)))$ . It can be said  $D$  and  $G$  compete with each other just like a two-player

minimax game to achieve the objective  $\mathcal{L}_{GAN}(G, D)$ :

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

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 [21]. Hence, Authors proposed to train  $G$  to maximize  $\log D(G(z))$  rather than training it to minimize  $\log(1 - D(G(z)))$ . This modified objective function leads to the same fixed point of the dynamics of  $G$  and  $D$  by providing much stronger gradients early in learning [21]. The GAN converging to match generated data distribution  $p_g$  to real data distribution  $p_{data}$  is illustrated in the figure 3.3.



**Figure 3.3:** An illustration of GAN converging to match generated data distribution  $p_g$  to real data distribution  $p_{data}$ . While training GANs, discriminative distribution is simultaneously updated so it will discriminate or distinguish samples from the real data distribution  $p_x$  from those of the generated data distribution  $p_g$ . In the figure, the discriminative distribution is represented by ( $D$ , dashed line, blue), the real data distribution  $p_x$  represented by (dotted line, black) and the generated data distribution  $p_g$  represented by ( $G$ , green, solid line).

The lower horizontal line represents the domain of 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,  $G$  and  $D$  are on the verge of convergence,  $p_g$  is almost similar to  $p_{data}$  and  $D$  is a partially accurate classifier. (b) Slowly,  $D$  is trained further to distinguish real data samples from generated data samples. For a fixed  $G$ , there is an optimal discriminator,  $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$ . (c) The  $G$  is updated using gradients that are backpropagated from  $D$  and makes  $G(z)$  to flow into the regions that are most probably classified as real data, the parameters of the  $G$  are updated, while the parameters of the  $D$  are fixed. (d) After several steps of training,  $G$  is optimal where  $p_g = p_{data}$ , which is equivalent to the optimal  $D$  predicting 0.5 for all samples drawn from  $x$ , at this stage  $D$  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 discusses the training algorithm of the GAN. The training algorithm is divided into two sections. These two sections are discriminator training and generator training. It illustrates the discriminator and generator training process together. It also illustrates, while training the generator, the discriminator is not training, basically, it means, a fixed discriminator is used while training the generator. Hence, the training dataset of real data samples is grayed out in the section where the generator training is illustrated, and while training discriminator both generated samples from the generator and real samples are used as an input. In the following sections generator's and discriminator's training process is described along with intuitive figures. Further, the GAN training algorithm has been described in the form of pseudo-code. The algorithm of the GAN is pictorially illustrated in the figure A.1

## Discriminator Training

In GANs, the discriminator is a binary classifier. It attempts to classify real data samples from fake data samples generated by the generator. It provides the likelihood that its input is a real data sample. In computer vision tasks, the discriminator classifies images, hence, it is implemented using CNN [38]. 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. 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 weights of the discriminator network are updated using backpropagated discriminator loss. The training process of the discriminator  $D$  using backpropagation illustrated in figure 3.4.

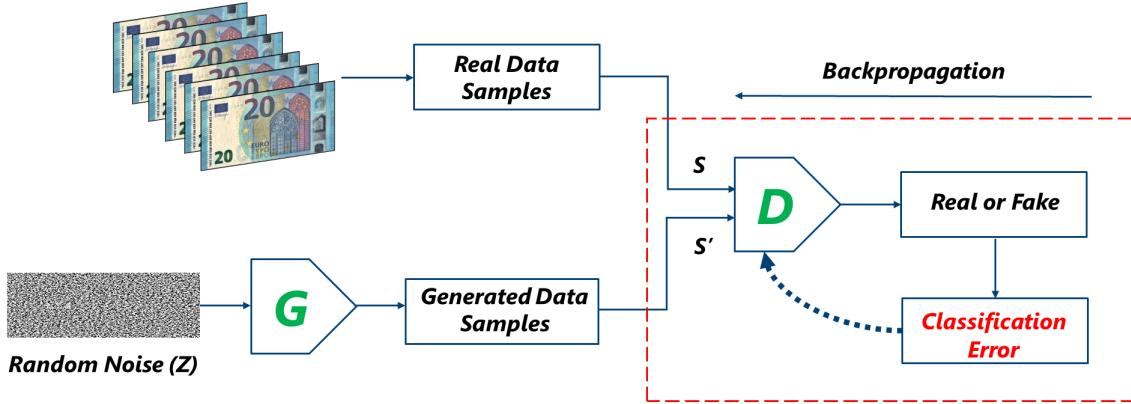
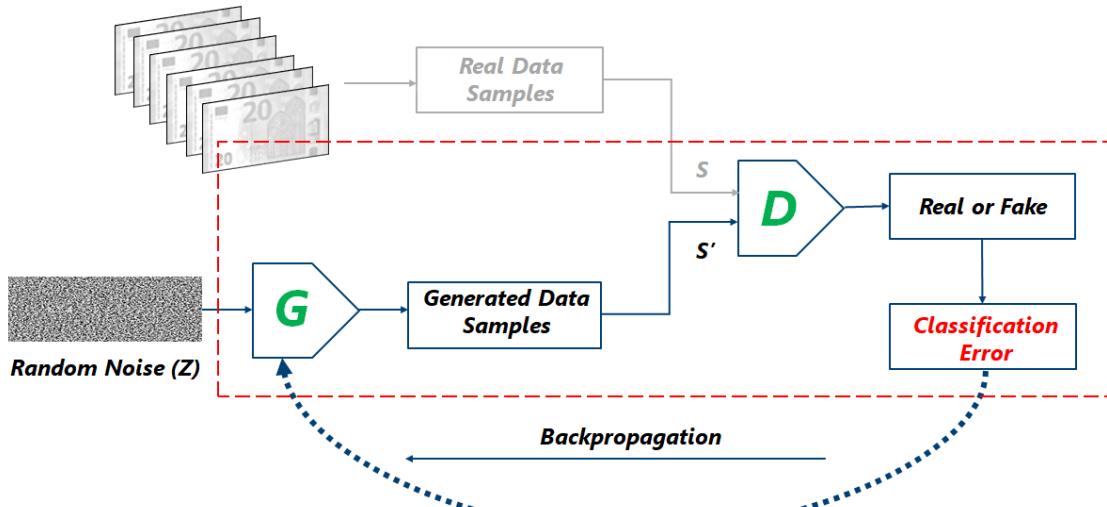


Figure 3.4: An illustration of the training of the discriminator  $D$  using backpropagation.

## Generator Training

The generator produces fake samples using random noise vectors sampled from the random noise distribution. In computer vision tasks, the generator usually produces fake images, hence, it is implemented using CNN [38]. The generator aims to produce fake samples that are as realistic as possible, but if the generator fails to fool the discriminator, in other words, if the discriminator recognizes and classifies the generated samples as fake, then it is penalized with classification error, such classification error is called generator loss. The weights of the generator are updated using backpropagated generator loss. The training process of the generator  $G$  using backpropagation is illustrated in figure

3.5.



**Figure 3.5:** An illustration of the training of the generator  $G$  using backpropagation.

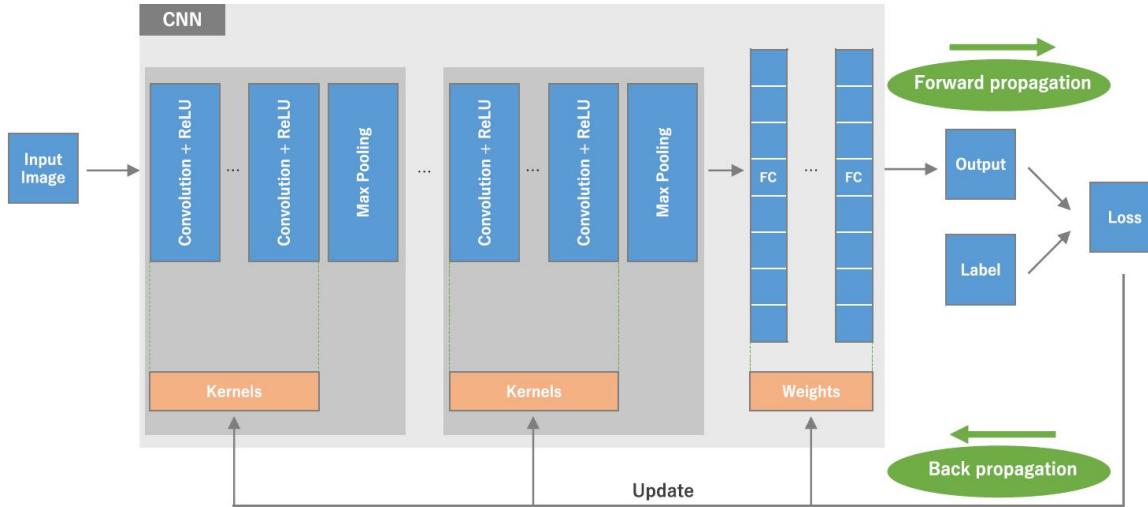
## GAN Training Algorithm

```
1 for each training iteration do
2   1. Train the Discriminator:
3     a) Get a real data sample  $S$  from the training dataset.
4     b) Get a noise vector  $Z$  from the random noise distribution, pass it thorough the
        generator  $G$  and create a fake sample  $S'$ .
5     c) Classify  $S$  and  $S'$  using the discriminator  $D$ .
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 noise vector  $Z$  from the random noise distribution, pass it thorough the
        generator  $G$  and create a fake sample  $S'$ .
9     b) Classify  $S'$  using the discriminator  $D$ .
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 [1].

## 3.2 Convolution Neural Networks

In recent years deep learning is evolved with the rise of ANNs. ANNs are biologically inspired and capable to solve complex problems [64]. CNNs is the most successful image-driven pattern recognition technique among ANNs [64]. 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 and videos. CNNs are inspired by the early findings in the study of biological vision, especially, organization of the animal visual cortex [65] [66] [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 and they perform feature extraction. Whereas the third, a fully connected layer, maps the extracted and downsampled features from the convolution layers and pooling layers into the final output, such as probabilities for classification tasks [67]. Let's have a look at each layer in the following sections.

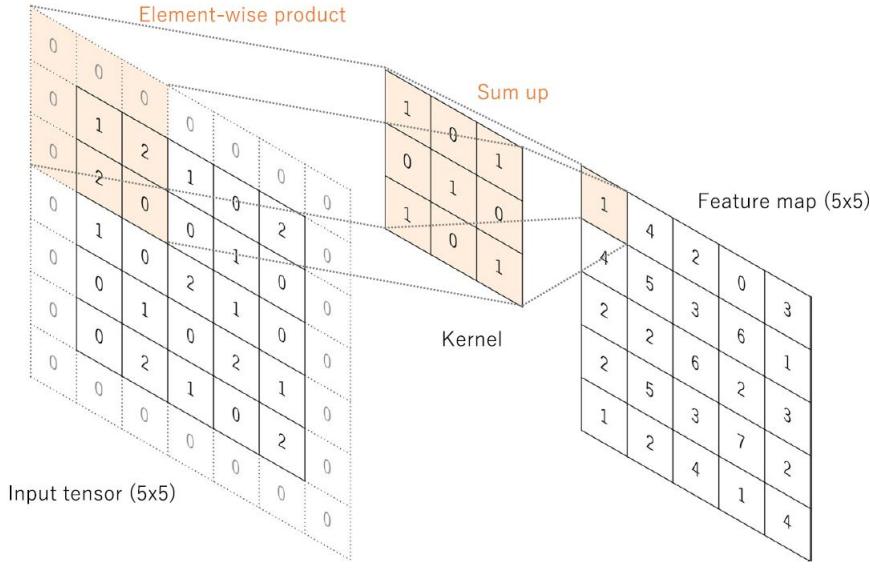


**Figure 3.6:** An illustration of CNN architecture and its training process. Convolution layers, pooling layers and fully connected layers are the basic building blocks of CNN. These layers are combined to build complex CNN architectures to solve different types of problems. These building blocks are stacked upon each other. The CNN is trained using the training dataset and the input data is fed into the network in a forward direction. The process of feeding data in the forward direction to the CNN is called forward propagation. Each layer receives the input data and processes it through an activation function such as Rectified Linear Unit (ReLU) (figure 3.11), then sends it on to the next successive layer. The model's performance is improved by updating the learnable parameters, weights and kernels using the loss value through backpropagation with gradient descent optimization algorithm [68] [67].

### 3.2.1 Convolution Layer

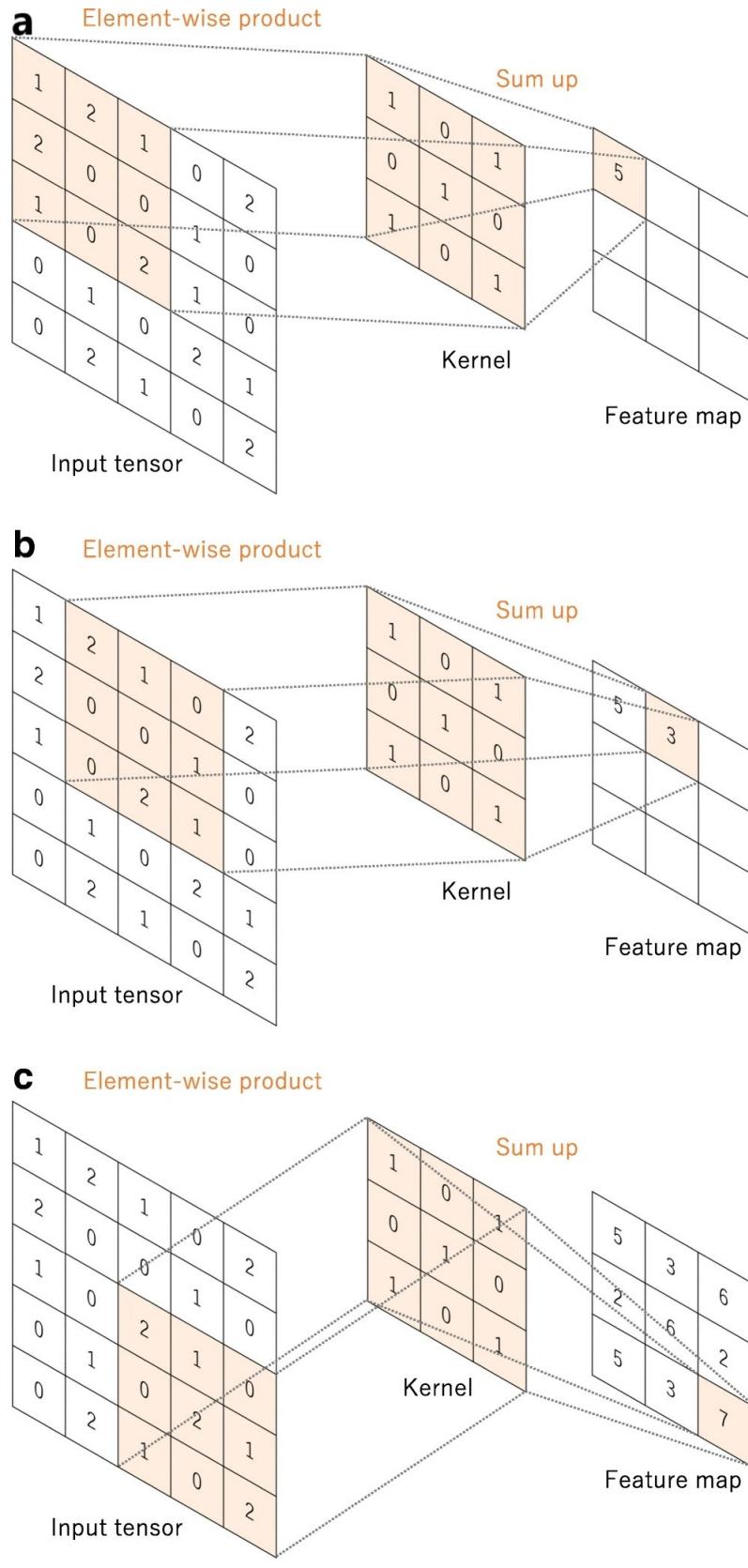
The convolution layer is a basic component of the CNN that performs feature extraction. It is typically a combination of linear and nonlinear operations like convolution operation and activation function [67]. It is one of the core building blocks of CNNs. Also, it computes for most heavy and complex computations. A convolution layer performs mathematical operations like convolution, a specialized type of linear operation [67]. The digital images store pixel values in a 2D grid, i.e., an array of numbers as shown in figure 3.8. The features from an image are extracted using a filter or kernel. A kernel is a small array of numbers applied across an input image, covering all pixels, using a given stride size. As features could exist at any location, the kernel is slid across an input, to perform an element-wise dot product between kernel elements and the input tensor, which is an array of pixel values. An element-wise dot product is performed between kernel's elements and an input tensor and results are added to

obtain an output value at the corresponding location of the output tensor, such output tensor is called a feature map (figure 3.8a-c) [67]. This process extracts all vital features present in an image, making CNN highly efficient for image processing and feature extraction tasks [67].



**Figure 3.7:** An illustration of a convolution operation with zero padding to retain spatial dimensions. Note that an input dimension of  $5 \times 5$  is retained in the generated output feature map. In this example, kernel size is set to  $3 \times 3$  and stride is set to 1 [67].

This procedure is repeated by applying multiple kernels to create an arbitrary number of feature maps, which describe different characteristics of the input tensors [67]. The different kernels act as separate feature extractors. Two important hyperparameters that represent the convolution operation are the size and number of kernels [67]. The size is typically  $3 \times 3$ , but sometimes  $5 \times 5$  or  $7 \times 7$ , 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. Hence, the output feature map's height and width are reduced compared to the input tensor [67]. This problem is solved by the zero-padding. In this technique, extra rows and columns of zeros are added around the input tensor so that the kernel's center element overlaps with an outermost element of an input tensor. Zero padding helps to preserve the original spatial dimension throughout convolution operation (figure 3.7) [67]. 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 [67]. 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 is an alternative technique for downsampling. It is described briefly in section 3.2.2.

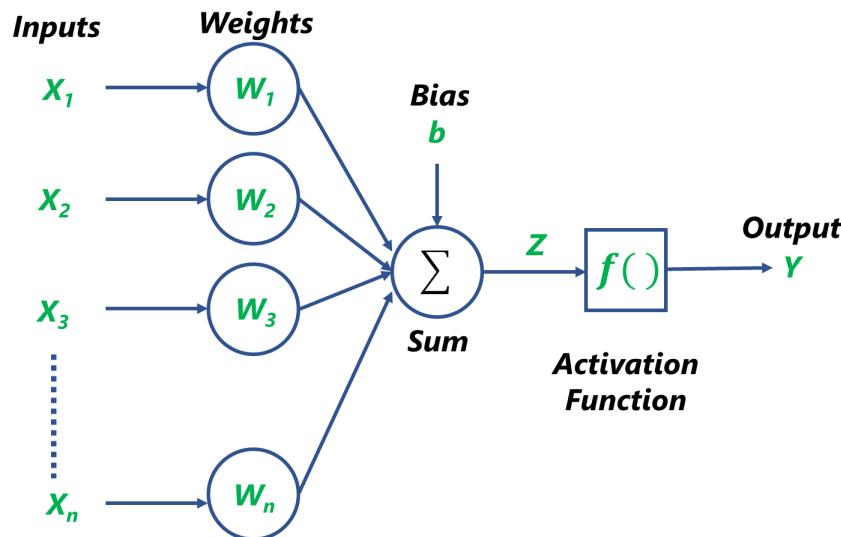


**Figure 3.8: a–c** An illustration of convolution operation with no padding, kernel size  $3 \times 3$  and stride 1 [67].

## Activation Functions

The artificial neuron is a mathematical function inspired by the neurons inside the brain and they are the basis of ANNs. The artificial neurons receive one or more weighted inputs and bias, and their sum is passed through an activation function. An artificial neuron's schematic diagram is illustrated in figure 3.9, 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 and  $b$  is the bias. The summation symbol represents the addition unit. Inputs are fed to the neuron, they perform a linear transformation on these inputs using the weights and biases. 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:** An illustration of artificial neuron.

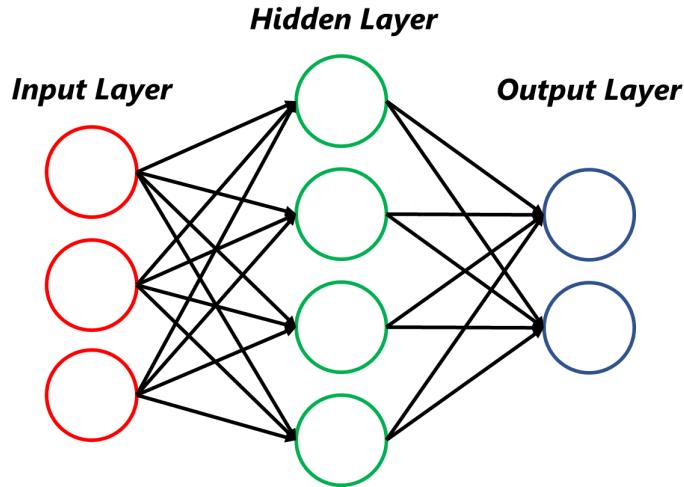
The function  $f$  is the activation function applied on  $Z$ .

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

The data is fed to the input layer of the neurons in neural network, undergoes the linear operation, later, activation functions are applied in the hidden layers and output is produced. The produced output is propagated forward to next hidden layers, this is movement is called forward propagation<sup>1</sup>. In neural networks the hidden layer lies between input layer and output layer. A simple neural network is illustrated in figure 3.10. The main purpose of the activation function is to introduce non-linearity in the neural network. The 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. The neural networks learn from the errors calculated at the output layer, a differentiable nonlinear activation function is required to perform backpropagation to compute loss with respect to the weights of the neurons and optimize them using gradient descent optimization algorithm or any other optimization algorithm.

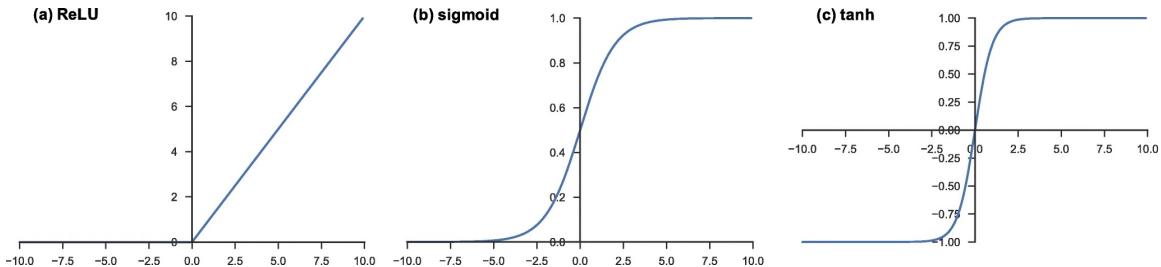
---

<sup>1</sup><https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/> last access: November 23, 2021



**Figure 3.10:** Simple neural network

The backpropagation minimizes the error, updates the weights of neurons and enhances the accuracy and performance of the neural network. The nonlinear functions, sigmoid, hyperbolic tangent( $\tanh$ ) functions are mathematical representations of biological neuron behavior. The ReLU is now the most commonly used nonlinear activation function, it computes the function:  $f(x) = \max(0, x)$ . These activation functions along with their plots are illustrated in figure 3.11 [69] [70].



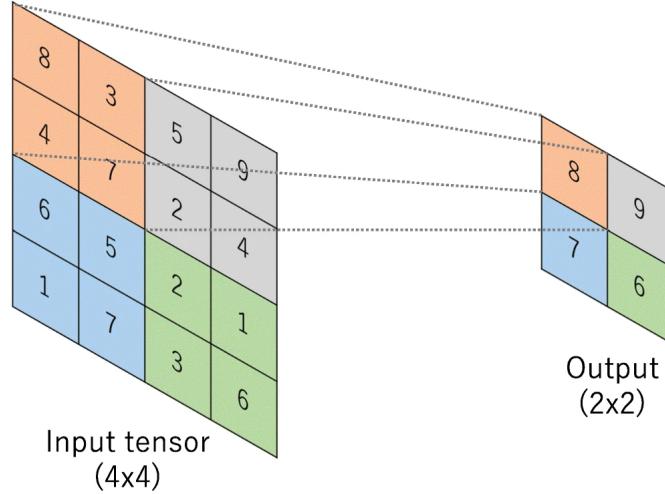
**Figure 3.11:** Most common nonlinear activation functions used while constructing neural networks: a) ReLU, b) sigmoid and c) hyperbolic tangent ( $\tanh$ ) [67].

### 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 computational complexity and number of parameters of the neural network and controlling overfitting. Downsampling introduces translation invariance to minor shifts and distortions [2]. Downsampling of feature maps can also 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 [2]. Similar to convolution operations, it uses 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 in each input patch of the feature map. The average pooling averages values in each input patch of the feature map [71].

## 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, max-pooling filters are of size  $2 \times 2$  and 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) [2]. The spatial dimension of feature maps is reduced by a factor of two by discarding 75% of its activations [72]. 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 and it's worth noting that max-pooling layers do not have learnable parameters [67].



**Figure 3.12:** An illustration of max pooling operation with a filter size of  $2 \times 2$ , no padding and stride 2. Max pooling operation extracts  $2 \times 2$  patches from the input tensors and outputs the maximum value in each patch by ignoring the remaining values. The spatial dimension of an input tensor is downsampled by a factor of 2 [67].

## 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  $height \times width \times depth$  into a  $1 \times 1 \times depth$  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 [67] [73]. c) It is intended to replace fully connected layers, the idea is to generate one feature map for each class, in the classification task, as they provide robust spatial information and the resulting vector can easily be fed to the softmax layer<sup>2</sup>.

### 3.2.3 Fully connected 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. 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 [67]. A fully connected layer performs multiplication between the input and weight matrix then adds a bias vector [67]. Equation 3.3 represents the task of each neuron in a fully connected layer. 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 features present in an image are extracted using convolution layers and downsampled by pooling layers. Next, extracted features are mapped through fully connected layers to

<sup>2</sup><https://paperswithcode.com/method/global-average-pooling/> last access: November 23, 2021

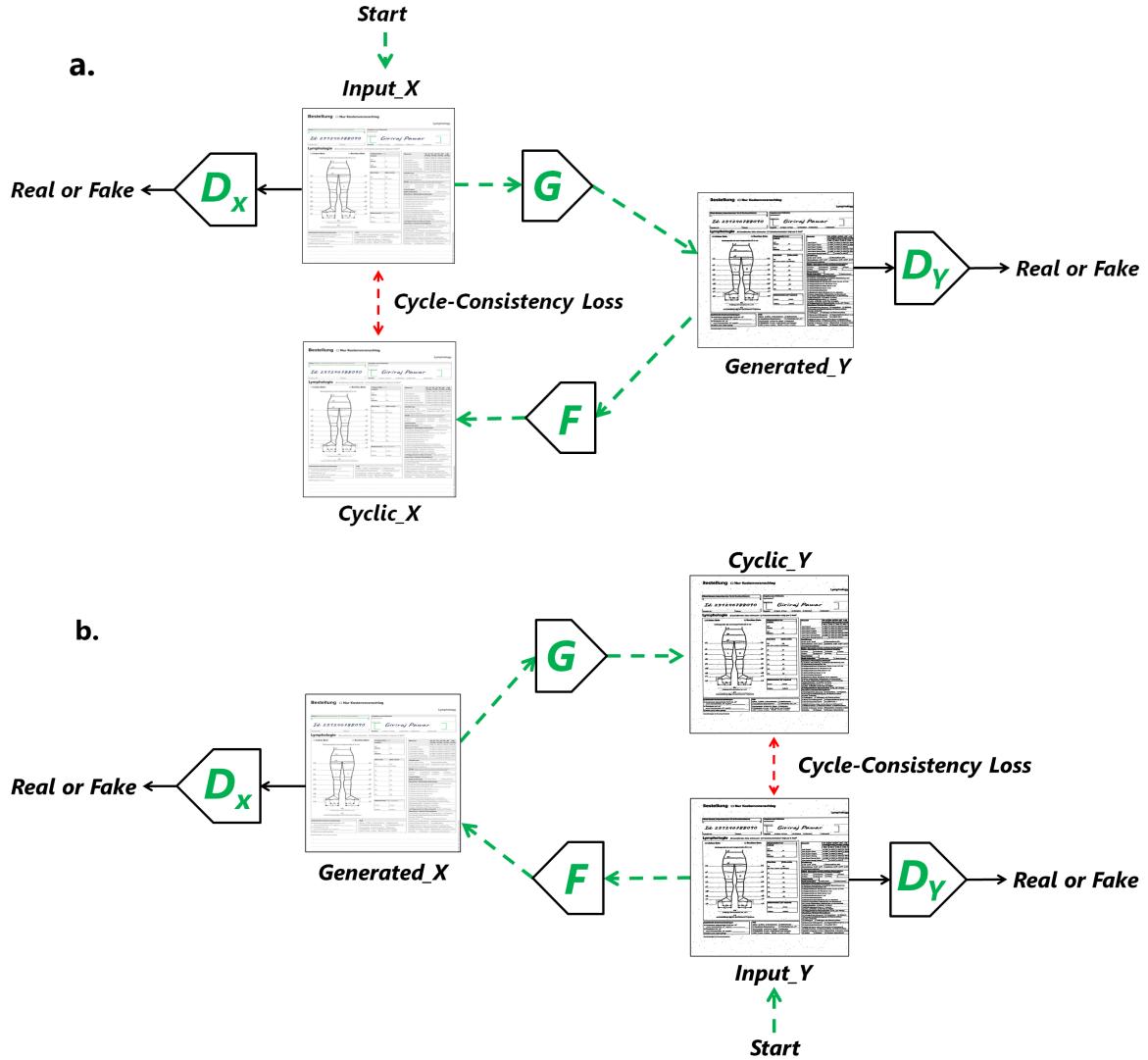
the final output of the neural network, determining the probability for each class in classification tasks. Usually, the number of output nodes in the fully connected layer is equal to the number of classes [67]. The last fully connected layer's activation function is usually different than others. Each task chooses a suitable activation function. Often, in binary classification tasks sigmoid activation function is used. For example, logistic regression, models the probabilities for binary classification tasks using the sigmoid activation function<sup>3</sup>. The softmax activation function is widely used in multiclass classification tasks. The output of the last fully connected layers are normalized into target class probabilities by the softmax activation function, in the classification tasks. The probability of each class ranges between 0 and 1 and the sum of all target class probabilities is 1 [67].

---

<sup>3</sup><https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/> last access: November 23, 2021

## 4. Methodology

This chapter describes the unpaired image-to-image translation method CycleGAN. In this thesis, CycleGAN is used to implement an image-to-image translation application. This application transforms synthetic document images into realistic document images. The application model is optimized using cycle-consistency loss, least-square loss and identity mapping loss. These loss functions are briefly described in this chapter and section 4.2 presents the algorithm of CycleGAN.



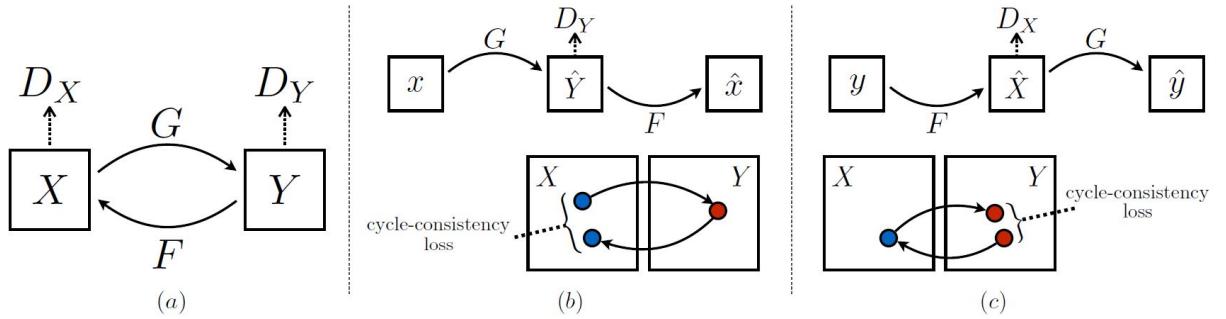
**Figure 4.1:** An illustration of the proposed image-to-image translation application using CycleGAN, for transforming synthetic document images into real document images and vice versa. It consists of two generators,  $G$  and  $F$  which map synthetic document images to real document images and real document images to synthetic document images, respectively, minimizing cycle-consistency loss [20]. It also contains two discriminators  $D_X$  and  $D_Y$  which acts as the adversary and reject images generated by respective generators [20].

## 4.1 Cycle-Consistent Adversarial Networks

In CycleGAN, the aim is to learn the mapping functions between two domains  $X$  and  $Y$ . The collection of synthetic document images represents source domain  $X$  and the collection of real document images represents target domain  $Y$ . 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  $y \sim p_{data}(y)$  [74]. The model includes two mappings functions  $G : X \rightarrow Y$  and  $F : Y \rightarrow X$  as illustrated in figure 4.1, both are the generators. Along with generators, two adversarial discriminators  $D_X$  and  $D_Y$  are introduced.  $D_X$  learns to distinguish images  $\{x\}$  from translated images  $\{F(y)\}$ . In the same way,  $D_Y$  learns to distinguish images  $\{y\}$  from  $\{G(x)\}$ .

The architecture of the CycleGAN is illustrated in figure 4.1. Figure (a) depicts the flow of transformation from synthetic document images to real document images. Figure (b) depicts the flow of transformation from real document images to synthetic document images. It depicts CycleGAN has two generators  $G$  and  $F$ , two discriminators  $D_X$  and  $D_Y$ . The discriminator  $D_Y$  takes input  $Input\_Y$  from the target domain  $Y$  and fake image  $Generated\_Y$  generated by the generator  $G$ . The discriminator  $D_Y$  learns to distinguish target domain images from the fake images generated by the  $G$  and penalizes itself in the event of misclassification. When the discriminator  $D_Y$  determines the image generated by the generator  $G$  is fake. It gives feedback to generator  $G$  through backpropagation to update its weights to produce good quality images. Next, to achieve cycle-consistency the fake image  $Generated\_Y$  passed thorough generator  $F$  to produce image  $Cyclic\_X$  back in the source domain  $X$  and the cycle-consistency loss is minimized between  $Input\_X$  and  $Cyclic\_X$ . The discriminator  $D_X$  takes input  $Input\_X$  from the source domain  $X$  and fake image  $Generated\_X$  generated by the generator  $F$ . The discriminator  $D_X$  learns to distinguish source domain images from fake images generated by the generator  $F$  and penalizes itself in the event of misclassification. When the discriminator  $D_X$  determines the image generated by generator  $F$  is fake. It gives feedback to generator  $F$  through backpropagation to update its weights to produce good quality images. Next, to achieve cycle-consistency the fake image  $Generated\_X$  passed thorough generator  $G$  to produce image  $Cyclic\_Y$  back in the target domain  $Y$  and the cycle-consistency loss is minimized between  $Input\_Y$  and  $Cyclic\_Y$ . The transformation flow of synthetic document images into real document images and vice versa (marked in green dotted arrows) by minimizing the cycle-consistency loss (marked in red dotted arrows) between the input image and cyclically generated image.

### 4.1.1 Cycle-Consistency Loss



**Figure 4.2:** An illustration of CycleGAN model with its mapping functions  $G$  and  $F$ , and respective discriminators  $D_Y$  and  $D_X$  with depiction forward and backward cycle-consistency loss. (a) CycleGAN model with two mapping functions  $G : X \rightarrow Y$  and  $F : Y \rightarrow X$ , and the respective adversarial discriminators are  $D_Y$  and  $D_X$ . (b) for each image  $x$  from source domain  $X$ , the cyclic image-to-image translation brings  $x$  back to the original image, i.e.,  $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ , this is called forward cycle-consistency loss [20]. (c) for each image  $y$  from target domain  $Y$ , the cyclic image-to-image translation brings  $y$  back to the original image, i.e.,  $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ , this is called backward cycle-consistency loss [20].

The cycle-consistency loss function is the basis of CycleGAN. Theoretically, adversarial training can learn mappings that produce outputs identically distributed as target domains  $Y$  and  $X$ , respectively [75]. However, a learned neural network maps input images to any random permutations of images in the target domain. In such a case, any random mapping should not be considered as an output distribution, possibly that would not match the target distribution. As the transformed images could be simply noise and there would not be any context or relationship between the input image and output image [20] [75]. 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 [75]. To further narrow down and reduce the space of possible outcomes using mapping functions, attempt has been made to argue that learned mapping functions should be cycle-consistent [20]. The property cycle-consistency can be understood by an example of sentence translation in different language. 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 [75]. The cycle-consistency loss in the form of a mathematical equation:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}(F(G(x)) - x) + \mathbb{E}_{y \sim p_{data}(y)}(G(F(y)) - y). \quad (4.1)$$

#### 4.1.2 Least-Square Loss

In general GANs the discriminator is a binary classifier that adopts the sigmoid cross-entropy loss function. However, the sigmoid cross-entropy loss function causes vanishing gradients problems while updating the generator for the samples that are on the correct side of the decision boundary but are still far away from the real data, because, the calculated error gradients are not sufficient to update the weights of the generator [35]. Also, the sigmoid cross-entropy loss function causes difficulty to stabilize the model training procedure [35]. First,  $\mathcal{L}_{GAN}$  (equation 3.1), the negative log-likelihood objective replaced by a least-squares loss functions [35]. The least-squares loss is more stable during training and generates higher quality results [35]. It is used to optimize the generator and discriminator adversarially. In the below equations,  $a$ ,  $b$  and  $c$  are the coding scheme for the equations of the discriminator, where  $a = 0$ ,  $b = 1$  and  $c = 1$ . The fake data is represented by 0 and real data is represented by 1. Then the improved objective functions using least-squares loss can be defined as follows:

$$\min_D \mathcal{L}_{ls}(D_Y) = \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_Y(y) - b)^2 + \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)) - a)^2 \quad (4.2)$$

$$\min_G \mathcal{L}_{ls}(G) = \mathbb{E}_{x \sim p_{data}(x)} [(D_Y(G(x)) - c)^2] \quad (4.3)$$

$$\mathcal{L}_{ls}(G, D_Y, X, Y) = \min_D \mathcal{L}_{ls}(D_Y) + \min_G \mathcal{L}_{ls}(G) \quad (4.4)$$

$$\min_D \mathcal{L}_{ls}(D_X) = \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_X(x) - b)^2 + \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - a)^2 \quad (4.5)$$

$$\min_F \mathcal{L}_{ls}(F) = \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - c)^2 \quad (4.6)$$

$$\mathcal{L}_{ls}(F, D_X, Y, X) = \min_D \mathcal{L}_{ls}(D_X) + \min_F \mathcal{L}_{ls}(F) \quad (4.7)$$

### 4.1.3 Identity Mapping Loss

The identity mapping loss encourages the mapping to preserve color composition between the input and output [76]. As described earlier, generator  $G$  transforms synthetic document images into realistic document images. The identity mapping loss regularizes the generator  $G$  to produce samples identical to the real document images when real samples are provided as an input. It is similar with the generator  $F$ , but vice versa, the identity mapping loss regularizes it to produce samples identical to the synthetic document images when synthetic samples are provided as an input. It means providing an input to the generator from the same domain in which it is transforming into, the output should be identical to the given input. The identity mapping loss in the form of a mathematical equation:

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

### 4.1.4 Complete Objective Function

Combining equations 4.7, 4.1 and 4.8 the final objective is achieved. The model is optimized using least-squares loss, cycle-consistency loss and identity mapping loss. In equation 4.9, the terms  $\lambda_{cyc}$  and  $\lambda_{identity}$  control the relative importance of the term in the objective. The complete objective function in the form of a mathematical equation:

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

## 4.2 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

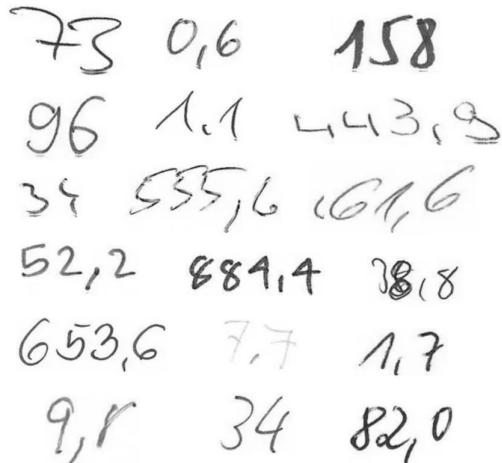
```

**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 and to evaluate the quality of images generated by the CycleGAN. The architecture of CycleGAN and classifiers discussed in section 5.2 and in section 5.1 dataset preparation is described. The training details of CycleGAN and classifiers described in section 5.3. How the classifiers are used to analyze the domain gap between distributions will be discussed in chapter Evaluation 6. The experiments are visualized using Tensorboard<sup>1</sup>. 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 [77]. 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



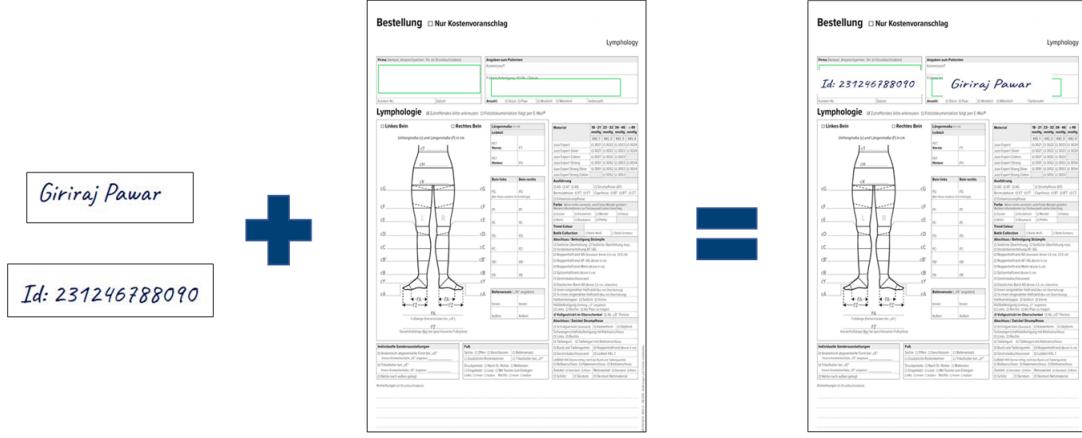
**Figure 5.1:** Examples of handwriting crops from the handwriting number dataset (figure reproduced from elevait GmbH & Co. KG with permission).

The dataset preparation is one of the vital step before training any neural network. Bad quality data leads to a poor generalization of neural networks [9]. Ten types of documents that were considered to work with this image-to-image translation application. The CycleGAN is trained using a stash of synthetic document images and real document images stored in a memory. In the source domain, there are 100,000 synthetic document images and an equal amount of real document images are in the target domain. The synthetic document images are generated using templates (figure A.2) 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 [78]. For each class of template 10,000, synthetic document images are created. The dataset of 100,000 synthetic document images is formed. These synthetic document images are used while training CycleGAN, just they are stashed at the single location collectively. The created 100,000 synthetic document images were also faxified and a faxified dataset of 100,000 images is created. It has 10 classes and each class has 10,000 images, similar to the synthetic document images dataset.

The faxification process uses several image transformations to make a clean gray-scale image look like it was sent via fax. A sample faxified image can be seen in figure A.4. The faxification process is

<sup>1</sup><https://www.tensorflow.org/tensorboard> last access: November 23, 2021

described briefly in section 6.2.3. Also, the faxification process can be visualized in figure 6.6. 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).

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).

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". Apart from minor differences, these templates are very similar to each other. 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, (a) is the template of "Arm", (b) is the template of "Bein" and (c) is the template of "Hand". The list of classes or templates is listed in the table 5.2.

**Bestellung**  Nur Kostenvorschlag

**BESTELLUNG**  Nur Kostenvorschlag

(a) (b) (c)

**Figure 5.3:** Templates with minor differences. All the templates (a), (b) and (c) are of type “Bein” (figure reproduced from elevait GmbH & Co. KG with permission).

**Bestellung**  Nur Kostenvorschlag

**BESTELLUNG**  Nur Kostenvorschlag

**Bestellung**  Nur Kostenvorschlag

(a) (b) (c)

**Figure 5.4:** Templates with major differences. (a) Template is of type “Arm”, (b) Template is of type “Bein” and (c) Template is of type “Hand” (figure reproduced from elevait GmbH & Co. KG with permission).

## 5.2 Network Architecture

### 5.2.1 CycleGAN

Johnson et al. [18] proposed the architecture of CycleGAN. In which the generator has three sequences of blocks one is downsampling, transformation and upsampling. The sequence of 2 downsampling 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, they resolve the vanishing gradient problem in deep neural networks [79]. The discriminator classifier network is designed using PatchGAN architecture [41] [48]. 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 as if each  $M \times M$  patch in an image is real or fake.

Johnson et al. [18] have provided naming conventions to define the architecture of generator and discriminator used in CycleGAN. The complete generator network with 9 residual blocks can be described as:  $c7s1-64$ ,  $d128$ ,  $d256$ ,  $R256$ ,  $u128$ ,  $u64$ ,  $c7s1-1$ .  $c7s1-k$  denotes a  $7 \times 7$  Convolution-InstanceNormlization-ReLU layer with  $k$  filters and stride 1. The downsampling block  $dk$  is denoted by a  $3 \times 3$  Convolution-InstanceNormlization-ReLU layer with  $k$  filters and stride 2. To reduce artifacts reflection padding is used.  $Rk$  denotes a single residual block that has two  $3 \times 3$  convolution layers with the same number of filters  $k$  on both layers and stride 1. The upsampling block  $uk$  denoted a  $3 \times 3$  TransposedConvolution-InstanceNormlization-ReLU layer with  $k$  filters and stride 2. The last layer  $c7s1-1$  denotes a  $7 \times 7$  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.6. The architecture of the generator is illustrated in table 5.3. Also, in the figure 5.5, the architecture of ResNet block in the generator network is illustrated. In which, at every ResNet block, the output from the previous layer is passed through two convolution layers (each with  $c3s1-256$ ). Further, it concatenated with the convolution layer's output and forwarded to the following layers.

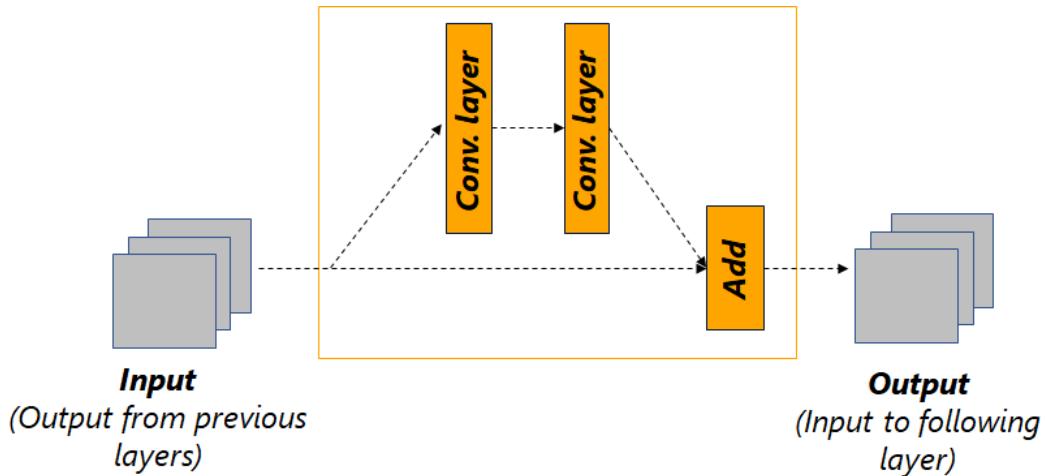


Figure 5.5: An illustration of ResNet blocks in CycleGAN generator architecture.

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	$7 \times 7$	$1 \times 1$
Convolution Layer Instance Normalization ReLU	128	$3 \times 3$	$2 \times 2$
Convolution Layer Instance Normalization ReLU	256	$3 \times 3$	$2 \times 2$
<b>9 Residual Blocks</b>			
2 Convolution Layers <i>(each with)</i> Instance Normalization ReLU	256	$3 \times 3$	$1 \times 1$
Transposed Convolution Layer Instance Normalization ReLU	128	$3 \times 3$	$2 \times 2$
Transposed Convolution Layer Instance Normalization ReLU	64	$3 \times 3$	$2 \times 2$
Convolution Layer tanh	1	$7 \times 7$	$1 \times 1$
Output ( $256 \times 256 \times 1$ )	-	-	-

Table 5.3: Generator architecture

The discriminator uses  $70 \times 70$  PatchGAN classifier architecture [41]. It is also called a Markovian discriminator [48]. The L1 and L2 loss functions produce blurry results while solving image generation problems [80]. These losses fail to encourage high-frequency crispness [80]. To model high frequencies, more attention is given to the structure in local image patches [41]. Therefore Markovian discriminator is called the PatchGAN discriminator. The design of discriminator network is: C64-C128-C256-C512-C512-C1. Ck denotes a  $4 \times 4$  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 C64 layer. After the last layer C512, the convolution operation is applied with filter 1 to produce an output of depth 1 using  $4 \times 4$  kernel and stride 1. All of the layers in the discriminator can be seen in figure A.12. The architecture of the classifier is illustrated in table 5.4.

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

Table 5.4: Discriminator architecture

For more information on the architectures of generators and discriminators, Github repository<sup>2</sup> 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 faxified data distribution and CycleGAN generated data distribution respectively. Their classification performance was evaluated on annotated real document images (testing dataset) using metrics like weighted average F1-score, macro average F1-score and accuracy to investigate the domain gap between real data distribution and mentioned three data distributions. Chapter Evaluation 6 discusses more about evaluation metrics. The classifier architecture is simple and easy to implement. The architecture of the classifier is illustrated in table 5.5. Also, all the layers in the classifier can be seen in figure A.5.

---

<sup>2</sup><https://github.com/junyanz/CycleGAN> last access: November 23, 2021

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

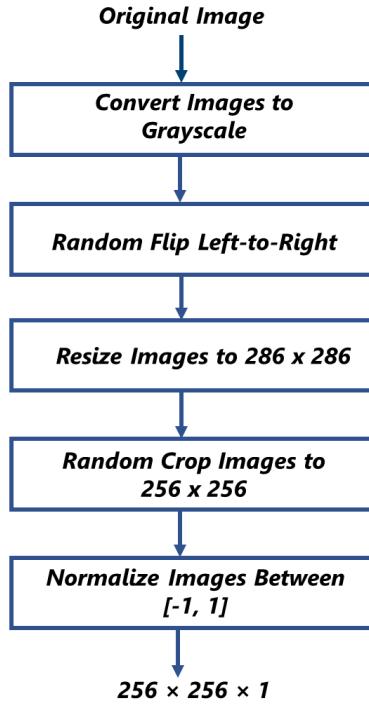
Table 5.5: Classifier architecture

## 5.3 Training Details

### 5.3.1 CycleGAN

Designing an efficient and fast input pipeline is challenging. The proposed 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 tf.data APIs to load large datasets spontaneously. To learn more about how to load large datasets efficiently in TensorFlow refer to this tutorial. 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  $F$ . For the stable training of the CycleGAN model, the general GAN objective function 3.1 has been modified. In which, the negative log-likelihood objective is replaced by least-squares loss [35]. The GAN loss function  $\mathcal{L}_{ls}(G, D_Y, X, Y)$  of the forward generator  $G$  minimizes  $\mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)) - 1)^2$  and discriminator  $D_Y$  minimizes  $\frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_Y(y) - 1)^2 + \frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_Y(G(x)))^2$ . Similarly, the GAN loss function  $\mathcal{L}_{ls}(F, D_X, Y, X)$  of the backward generator  $F$  minimizes  $\mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)) - 1)^2$  and discriminator  $D_X$  minimizes  $\frac{1}{2} \mathbb{E}_{x \sim p_{data}(x)} (D_X(x) - 1)^2 + \frac{1}{2} \mathbb{E}_{y \sim p_{data}(y)} (D_X(F(y)))^2$ .

The learning rate ( $\eta$ ) is set to 0.0002 while training CycleGAN. The weights neural networks (generators and discriminators) in the CycleGAN model are initialized by a Gaussian distribution with standard deviation ( $\sigma$ ) 0.02 and mean ( $\mu$ ) 0. In the final objective function 4.9,  $\lambda_{cyc}$  is set to 10 and  $\lambda_{identity}$  is set to 0.5, which means importance of cycle-consistency loss is higher than identity mapping loss 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 [81]. Also, while constructing generators and discriminators the instance normalization layers [82] are used and gamma weight is initialized by Gaussian distribution with mean ( $\mu$ ) 0 and standard deviation ( $\sigma$ ) 0.02.



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

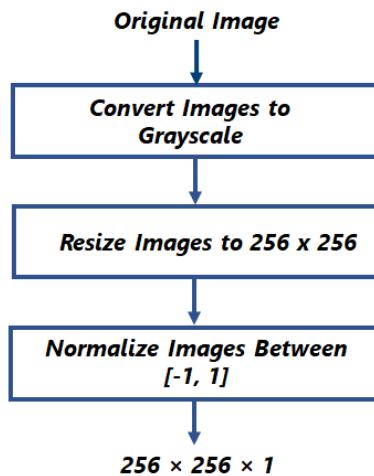
The CycleGAN model is trained with batch size 1 for 20 epochs. The complete CycleGAN model consists of forward generator  $G$ , backward generator  $F$ , discriminator  $D_Y$  and discriminator  $D_X$ . The training checkpoints<sup>3</sup> are saved at the end of every epoch. The stack of 100,000 synthetic document images (Domain  $X$ ) and 100,000 real document images (Domain  $Y$ ) are used to train the CycleGAN model. The training images are preprocessed and the preprocessing process is illustrated in figure 5.6. Initially, the training images are converted into grayscale. Next, random mirroring is applied, in which the image is randomly flipped horizontally from left to right. Further, random mirroring is applied, in which the image is resized to  $286 \times 286$  and then randomly cropped to  $256 \times 256$ . Random jittering and mirroring are image augmentation techniques that avoid overfitting [20]. Finally, the images are normalized to a value between -1 and 1.

### 5.3.2 Classifier

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, the training images are converted into grayscale. Next, resized to  $256 \times 256$  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  $256 \times 256$  dimension and already normalized between  $[-1, 1]$ . Because at the final layer of the generator, the tanh activation function is used, this can be seen in the architecture of the generator in table 5.3. All three classifiers are trained for 10 epochs with 100,000 images and evaluated using 1162 annotated real document images (testing dataset).

---

<sup>3</sup><https://www.tensorflow.org/guide/checkpoint> last access: November 23, 2021



**Figure 5.7:** Steps involved in preprocessing of training images (Only synthetic and faxified document images) of classifiers.

# 6. Experiments and Evaluation

This chapter discusses the experiments conducted in this thesis, evaluation metrics and results. The evaluation metrics like accuracy, precision, recall, confusion matrix and F1-score are discussed in section 6.1. These metrics are used to evaluate the domain gap between the distributions and quality of CycleGAN generated document images. In section 6.2 conducted experiments are explained along with training plots, confusion matrices and classification reports. Finally, in section 6.3, quantitative and qualitative results are discussed thoroughly along with the typical failure cases.

## 6.1 Evaluation Metrics

In machine learning, there are numerous performance metrics to evaluate neural networks. Each evaluates different aspects of neural network performance. Hence, It is needed to have a specific set of performance metrics for a particular problem solved using neural networks. It's vital to evaluate the neural network's performance after training using testing data to determine its actual performance and generalization error on unseen data [83]. In this thesis, the performance of classifiers trained upon different data distributions determined using annotated real document images (testing dataset) and evaluation metrics like accuracy, precision, recall, confusion matrix and F1-score are used. 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. Accuracy is the most common and widely used metric to evaluate classifiers<sup>1</sup>. It is the ratio of number of correct predictions to the total number of predictions (equation 6.1). However, accuracy is not suitable metric for performance evaluation when testing dataset is unbalanced. Hence, along with accuracy and F1-score, precision and recall are also considered during the performance evaluation. 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 the ratio of the number of true positives divided by the sum of true positives and false negatives (equations 6.3). In below equations 6.1, 6.2 and 6.3 the  $TP$  means true positives,  $TN$  means true negatives,  $FP$  means false positives and  $FN$  means false negatives.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6.1)$$

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

$$Recall = \frac{TP}{TP + FN} \quad (6.3)$$

The 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<sup>2</sup>. 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<sup>2</sup>. This leads to bigger penalization when the classifier does not

<sup>1</sup><https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/> last access: November 23, 2021

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 here.

$$F1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (6.4)$$

$$\text{Macro average F1-score} = \frac{F1_{\text{class1}} + F1_{\text{class2}} + \dots + F1_{\text{classN}}}{N} \quad (6.5)$$

$$\text{Weighted average F1-score} = \frac{F1_{\text{class1}} \times W_1 + F1_{\text{class2}} \times W_2 + \dots + F1_{\text{classN}} \times W_N}{N} \quad (6.6)$$

The confusion matrix is the most intuitive metric to determine the accuracy of the classifiers. It is a table that describes how well a classifier performs on a test dataset that is labeled or annotated. It is useful when the classifier has to classify more than two classes. 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. In this thesis, the confusion matrix is extensively used to evaluate classifiers trained on different data distributions. Once the classifier is trained on one data distribution, it is evaluated using annotated real document images (testing dataset). The confusion matrix provides an insight of class level classification performance to conclude similarity or dissimilarity between the data distribution on which classifier is trained and real data distribution. For example, if a classifier is trained on CycleGAN generated document images, then it will be evaluated using a testing dataset and the confusion matrix is plotted. Such a visualization technique shows which class of generated document images is closer to real document images and which are not. In this way, class level quality of CycleGAN generated document images is determined. More information about the confusion matrix can be found here.

## 6.2 Experiments

In this thesis, three data distributions are considered for the experiments. The synthetic data distribution, faxified data distribution and CycleGAN generated data distribution. Experiments are performed to analyze the domain gap between these data distributions and real data distribution. The experiment is conducted to analyze the domain gap between CycleGAN generated data distribution and real data distribution, determines the quality of CycleGAN generated document images, revealing how much they are close to the real document images. Now let's describe the conducted experiments briefly, a) Train a classifier on synthetic document images and its performance is evaluated on testing dataset (Annotated real document images, table 5.2). b) Another classifier with the same architecture is trained on faxified document images and its performance is evaluated on the testing dataset. c) The CycleGAN is trained for 20 epochs, using synthetic document images (Source domain  $X$ ) and real document images (Target domain  $Y$ ) and the checkpoint is saved at the end of every epoch. Once CycleGAN training is finished, the saved model is loaded to generate images or transform synthetic document images into realistic document images. From the saved CycleGAN model, generator  $G$ , retrieved, as it transforms synthetic document images into realistic document images (Source domain  $X$  to Target domain  $Y$ ,  $G : X \rightarrow Y$ ). 100,000 synthetic document images transformed into realistic document images, generating a dataset of 100,000 CycleGAN generated document images, with 10 classes and each class has 10000 images. d) Next, another classifier, with same architecture is trained on the CycleGAN generated document images and its performance is evaluated on testing dataset. This experiment determines the quality of the images generated by CycleGAN. Also, it indicates how well CycleGAN was able to generate data distribution that matches real data distribution to reduce the domain gap between synthetic data distribution and real data distribution. To put it another way, how well the CycleGAN generated

---

<sup>2</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html) last access: November 23, 2021

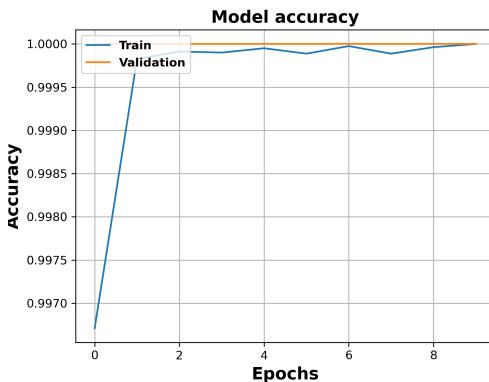
data distribution matched the real data distribution and concludes how similar CycleGAN generated document images are to the real document images. In this thesis, the evaluation metrics like accuracy, weighted average F1-score and macro average F1-score [84] are used to determine the performance of classifiers on testing dataset. The architecture of the classifier is illustrated in table 5.5.

### 6.2.1 Experiment Steps

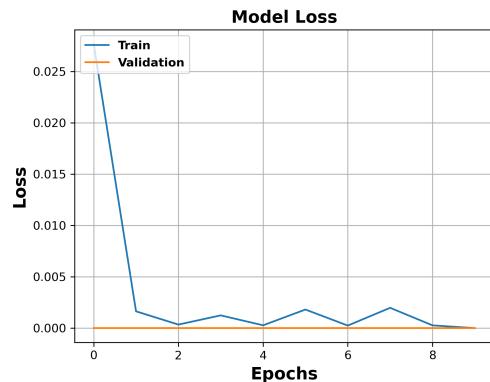
1. Train a classifier using synthetic document images and evaluate its performance over testing dataset.
2. Train a classifier using faxified document images and evaluate its performance over testing dataset.
3. Train CycleGAN using synthetic document images and real document images.
4. Generate realistic document images using generator  $G$  retrieved from the trained and saved CycleGAN model.
5. Train a classifier using CycleGAN generated document images and evaluate its performance over testing dataset. This experiment reveals how well the CycleGAN generated document images are generalizing to the real document images, eventually determining the quality of images generated by CycleGAN.
6. Compare the classification performance of the above three classifiers (trained on 3 different data distributions) on testing dataset and determine which distribution is closer to the real data distribution.

### 6.2.2 Training a Classifier on Synthetic Document Images

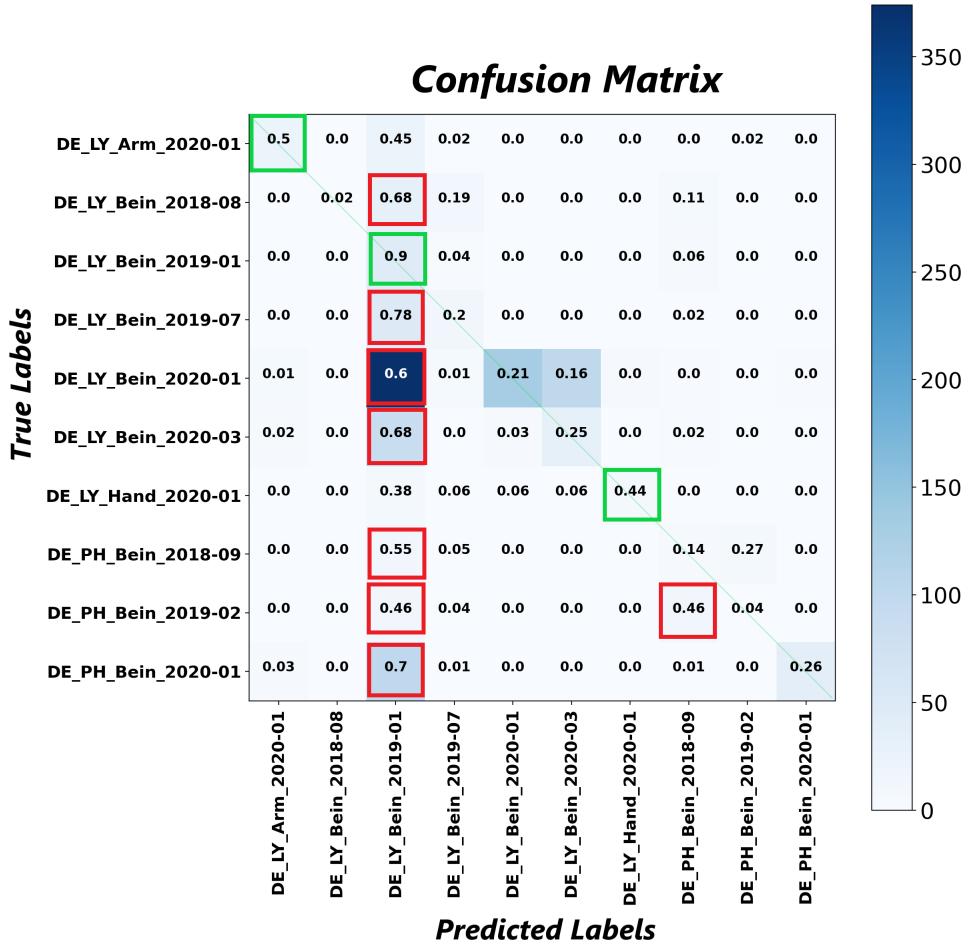
The dataset of synthetic document images is created using templates (figure of sample A.2) and handwritten crops (figure of sample 5.1) using the process mentioned in figure 5.2. This dataset consists of 100,000 document images, 10 classes and each class has 10000 images. In this experiment, a classifier is trained on synthetic document images dataset. The performance of this classifier is evaluated on testing dataset (Annotated real document images) 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%. The obtained results conclude that there is a huge gap between real data distribution and synthetic data distribution. The confusion matrix is illustrated in figure 6.3. In which, it is visible that most of the images from the testing dataset were classified wrongly. Only test images from classes DE.LY\_Arm\_2020-01 (support 44), DE.LY\_Bein\_2019-01 (support 50) and DE.LY\_Hand\_2020-01 (support 16) are classified correctly with a recall score of 50%, 90% and 44% respectively. The epochs against accuracy and epochs against loss plots while training classifier on synthetic document images is illustrated in figure 6.1 and 6.2 respectively. The epochs against accuracy and epochs against loss plots while training classifier on synthetic document images is illustrated in figure 6.1 and 6.2 respectively.



**Figure 6.1:** Epochs vs. Accuracy plot during training a classifier on synthetic document images.



**Figure 6.2:** Epochs vs. Loss plot during training a classifier on synthetic document images.



**Figure 6.3:** Confusion matrix to determine the performance of the classifier trained on synthetic document images and evaluated using real annotated document images (testing dataset). Green boxes represent classes that are correctly classified and red boxes represent classes that are misclassified.

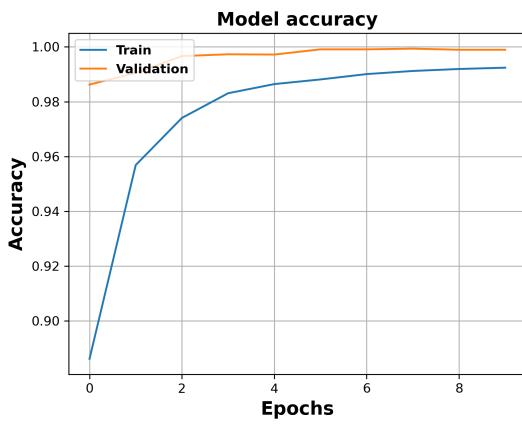
	Precision	Recall	F1-score	Support
DE_LY_Arm_2020-01	0.65	0.50	0.56	44
DE_LY_Bein_2018-08	0.50	0.02	0.04	47
DE_LY_Bein_2019-01	0.06	0.90	0.11	50
DE_LY_Bein_2019-07	0.36	0.20	0.26	60
DE_LY_Bein_2020-01	0.96	0.21	0.34	624
DE_LY_Bein_2020-03	0.24	0.25	0.24	128
DE_LY_Hand_2020-01	0.70	0.44	0.54	16
DE_PH_Bein_2018-09	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, evaluating a classifier on the testing dataset after training with synthetic document images.

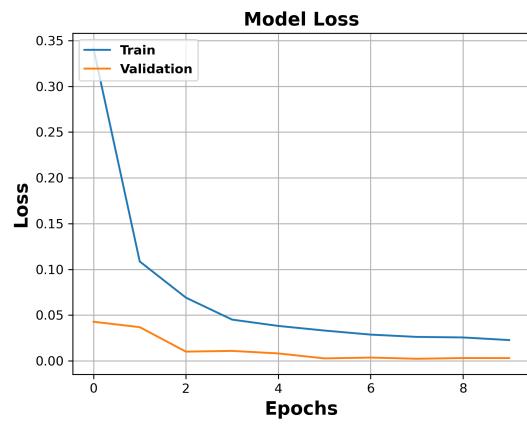
### 6.2.3 Training a Classifier on Faxified Document Images

The faxification process mimics the way the fax machine works. Usually, the fax machines transmit only black-and-white images, but the transferred images might be dirty and not aligned perfectly. This leads to several common artifacts being introduced into transferred images. The 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. However, the faxification process is not deterministic, it 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 verticle lines and conversion to a grayscale image. The faxification process can be visualized in figure 6.6. In figure 6.7, it is visible a snippet from the synthetic document image that has transformed into randomly distinct image transformations when it has sent through the faxification process. The faxification process is implemented in Python using a traditional programming approach.

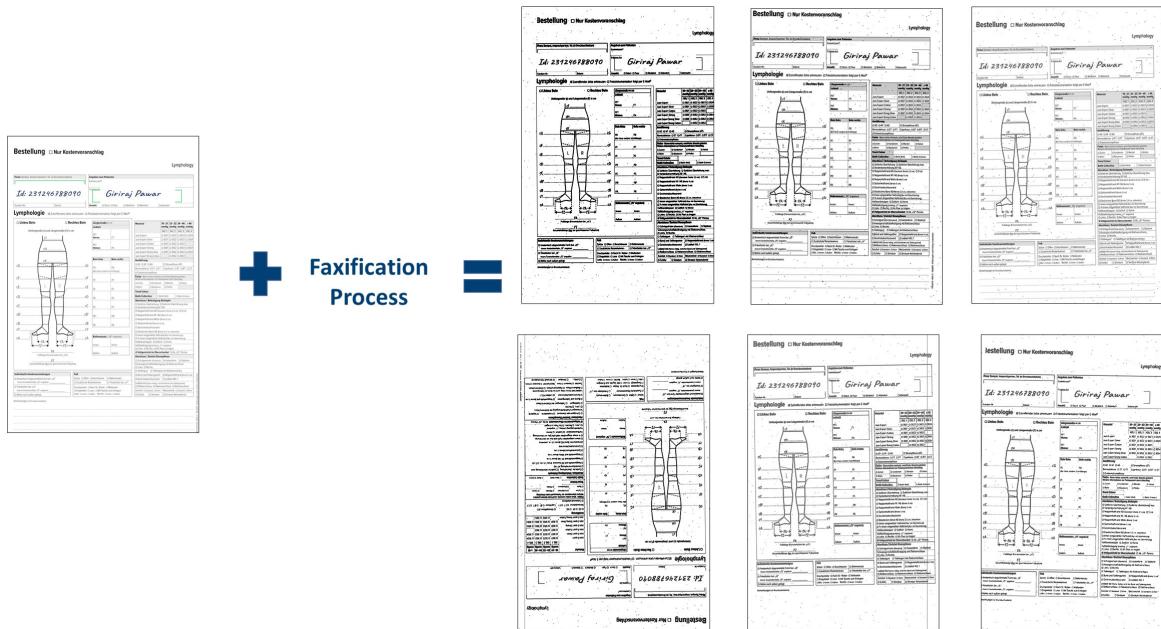
The dataset of faxified document images is created by transforming synthetic document images using the faxification process (figure 6.6). In this experiment, the classifier is trained using the dataset of faxified document images. The performance of this classifier is evaluated on the testing dataset to understand the domain gap between real data distribution and faxified data distribution. The classification report in table 6.2 states that the accuracy of this classifier on testing dataset is 43%, macro average F1-score is 58% and weighted average F1-score is 43%. The results conclude that the faxified data distribution matched the real data distribution better than the synthetic data distribution. The confusion matrix is illustrated in figure 6.8. In which, it is visible that document images of class DE\_LY\_Bein\_2019-01 (support 50) are being wrongly classified as DE\_LY\_Bein\_2019-07. Also, the document images of class DE\_LY\_Bein\_2020-01 (support 624) are wrongly classified as DE\_LY\_Bein\_2020-03. The remaining images from the testing dataset are classified convincingly. The epochs against accuracy and epochs against loss plots while training classifier on faxified document images is illustrated in figure 6.4 and 6.5 respectively.



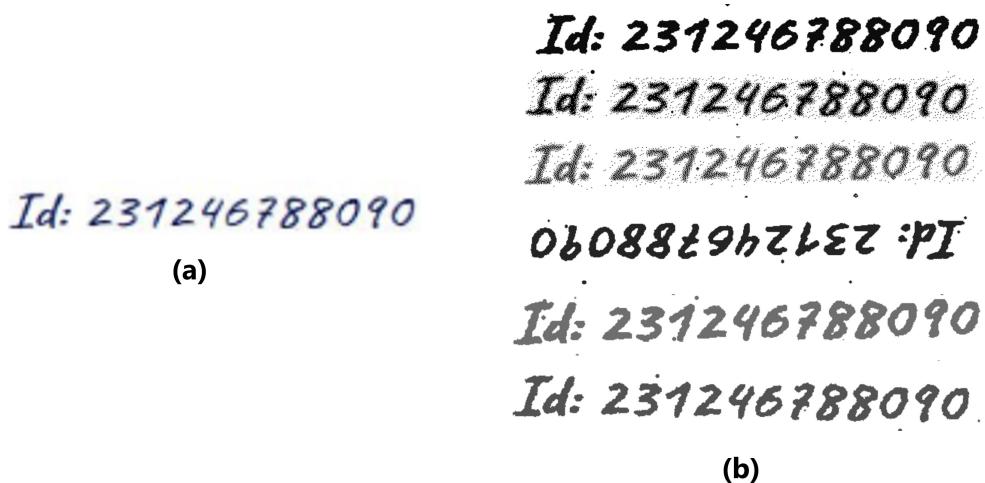
**Figure 6.4:** Epoch vs. Accuracy plot during training a classifier on faxified document images.



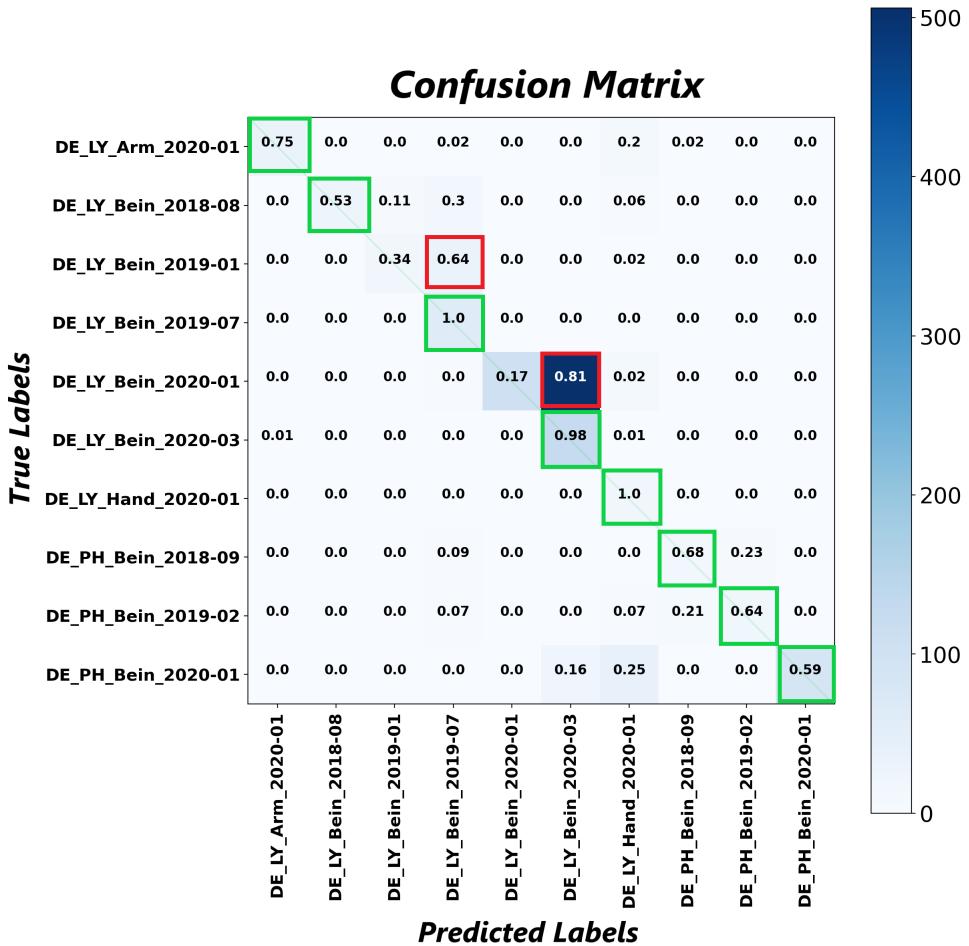
**Figure 6.5:** Epoch vs. Loss plot during training a classifier on faxified document images.



**Figure 6.6:** An illustration of faxification process applied on synthetic document images (figure reproduced from elevait GmbH & Co. KG with permission).



**Figure 6.7:** An 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.



**Figure 6.8:** Confusion matrix to determine the performance of the classifier trained on faxified document images and evaluated using real annotated document images (testing dataset). Green boxes represent classes that are correctly classified and red boxes represent classes that are misclassified.

	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 6.2: Classification report, evaluating a classifier on the testing dataset after training with faxified document images.

## 6.2.4 CycleGAN Training

The proposed image-to-image translation application is implemented using CycleGAN. The CycleGAN is trained using synthetic document images (Domain  $X$ ) and real document images (Domain  $Y$ ). The domain  $X$  represents the source domain and domain  $Y$  represents the target domain. The CycleGAN consists of two generators  $G$  and  $F$  and two discriminators  $D_X$  and  $D_Y$ . Further, for training, 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. 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 respective discriminator. Next, the 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. Please 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.

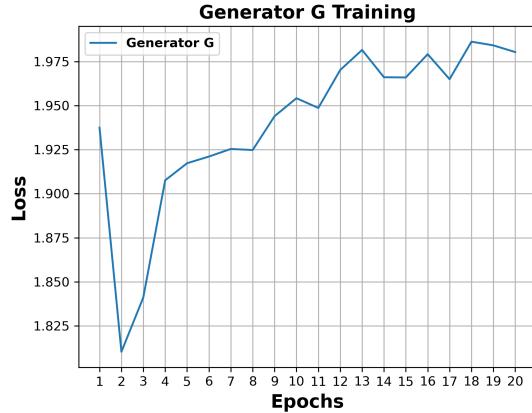


Figure 6.9: Epochs vs. Loss plot during Generator  $G$  is training.

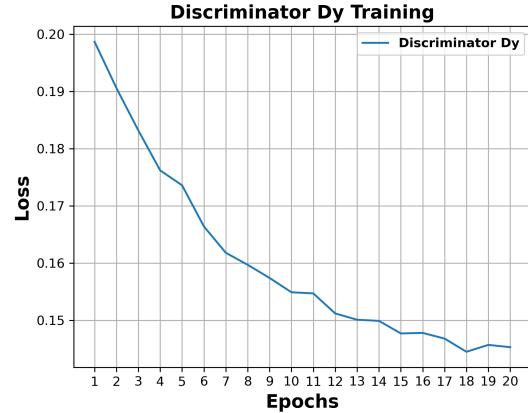


Figure 6.10: Epochs vs. Loss plot during Discriminator  $D_Y$  is training.

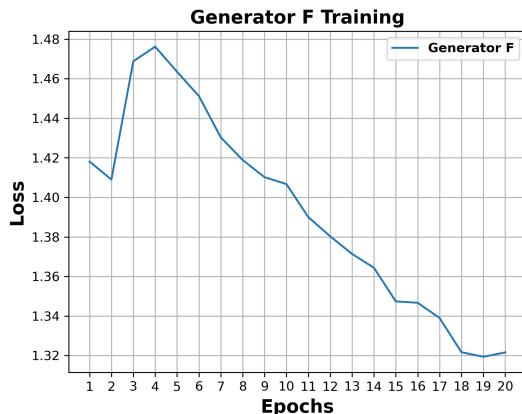


Figure 6.11: Epochs vs. Loss plot during Generator  $F$  is training.

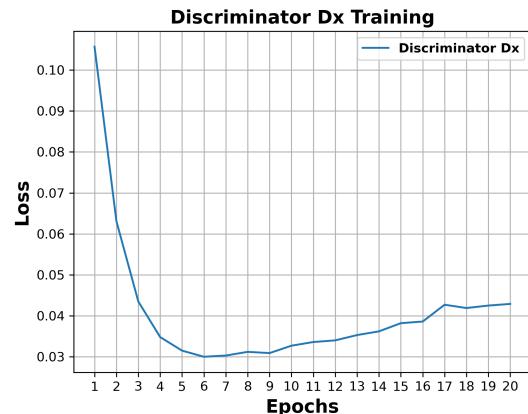


Figure 6.12: Epochs vs. Loss plot during Discriminator  $D_X$  is training.

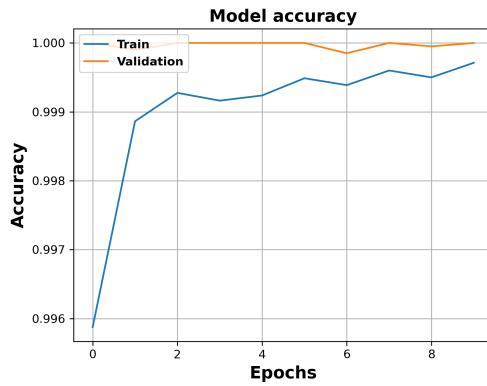
Further, identity mapping loss is calculated to preserve the color of images after domain translation

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  $y_i$  itself is input to  $G$ . And, the identity mapping loss for generator  $F$  is the difference between input image  $x_i$  and the transformed image  $Same\_X$  when  $x_i$  itself is input to  $F$ . 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 6.9 and 6.11 respectively. Also, the epochs against loss plot while training discriminators  $D_Y$  and  $D_X$  are shown in figure 6.10 and 6.12 respectively. The algorithm of CycleGAN is described in section 4.2. The complete architecture of the proposed image-to-image translation application using CycleGAN is illustrated in figure 4.1. For more CycleGAN training details refer section 5.3.1.

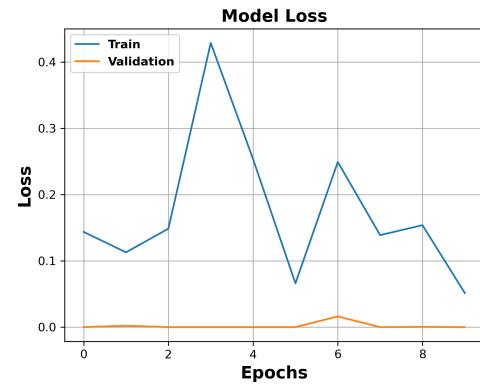
### 6.2.5 Training a Classifier on CycleGAN Generated Document Images

Once CycleGAN is trained, the generator  $G$  is retrieved from the saved CycleGAN model to transform synthetic document images to realistic document images. 100,000 synthetic document images are transformed into realistic document images. The classifier is trained using these transformed 100,000 realistic document images. The performance of this classifier is evaluated on the Annotated real document images (testing dataset) to understand the domain gap between real data distribution and CycleGAN generated data distribution. The purpose of this experiment is to determine how closely the CycleGAN generated data distribution matches the real data distribution. The quality of generated realistic document images can be determined using this experiment. The classification report in table 6.3 states that the accuracy of this classifier on the testing dataset is 27%, macro average F1-score is 34% and weighted average F1-score is 25%.

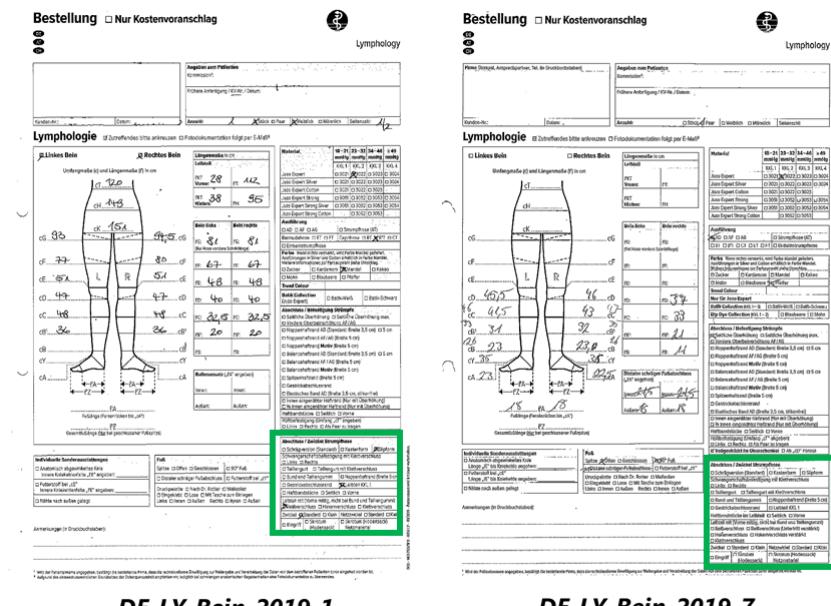
The results conclude that the faxified data distribution matched the real data distribution better than the CycleGAN generated data distribution. The confusion matrix is illustrated in figure 6.8. In which, the document images DE.LY\_Arm\_2020-01 (support 44 and recall 75%), DE.LY\_Bein\_2019-07 (support 60 and recall 83%), DE.LY\_Hand\_2020-01 (support 16 and recall 75%), DE.PH.Bein\_2019-02 (support 28 and recall 46%) and DE.PH.Bein\_2020-01 (support 143 and recall 67%) are classified satisfactorily, with convincing recall score. 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. As shown in the confusion matrix 6.16, most of the generated document images DE.LY.Bein\_2019-01 are misclassified as DE.LY.Bein\_2019-07. The real document images DE.LY.Bein\_2019-01 and DE.LY.Bein\_2019-07 are similar to each other with couple of minor differences. Examples of DE.LY.Bein\_2019-01 and DE.LY.Bein\_2019-07 real document images are illustrated in figure 6.15. It is challenging for CycleGAN to learn such minor differences and features from large training datasets. Also, the majority of document images are of the type "Bein" and they are similar to each other, learning minor differences and features between them from the training dataset is very challenging. The CycleGAN is failed to learn such small and complex differences, hence some of the generated images are more similar to different classes than the original class and they are misclassified. 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 6.13 and 6.14 respectively.



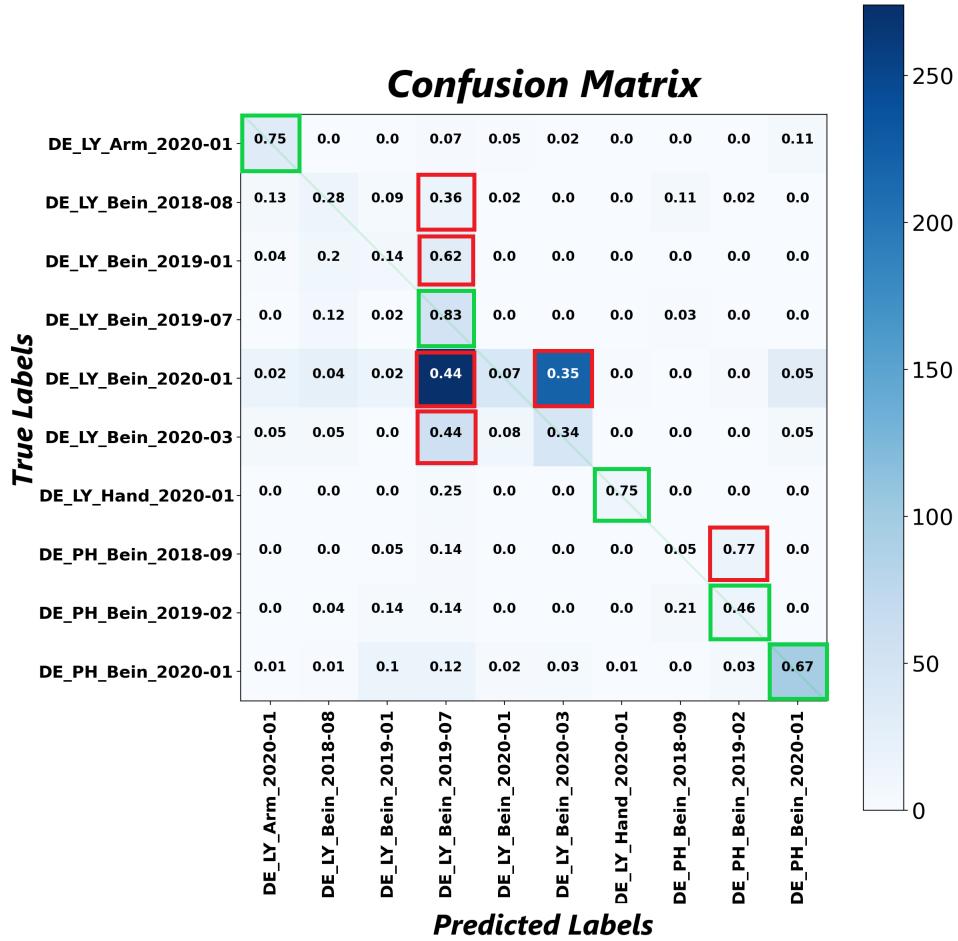
**Figure 6.13:** Epochs vs. Accuracy plot during training a classifier on CycleGAN generated document images.



**Figure 6.14:** Epochs vs. Loss plot during training a classifier on CycleGAN generated document images.



**Figure 6.15:** Examples of DE\_LY\_Bein\_2019-01 and DE\_LY\_Bein\_2019-07 real document images. Both real document images are similar to each other hence learning such minor differences from the training dataset is a challenging for CycleGAN.



**Figure 6.16:** Confusion matrix to determine the performance of the classifier trained on CycleGAN generated document images and evaluated using real annotated document images (testing dataset). Green boxes represent classes that are correctly classified and red boxes represent classes that are misclassified.

	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 6.3:** Classification report, evaluating a classifier on the testing dataset after training with CycleGAN generated document images.

## 6.3 Results

The qualitative and quantitative results of the proposed image-to-image translation application implemented using CycleGAN are discussed in this section. In experiments, three separate classifiers are trained on three data distributions, synthetic data distribution, faxified data distribution and CycleGAN generated data distribution. The performance of these classifiers that are trained using different distributions is evaluated and compared using the testing dataset (Annotated real document images) to analyze the domain gap between these distributions and real data distribution. The experiment of training a classifier on CycleGAN generated document images and evaluating on testing dataset helps to determine the quality of CycleGAN generated document images. It shows how close the generated document images are to the real document images.

The performance of the classifiers is evaluated using metrics like accuracy, macro average F1-score and weighted average F1-score. The accuracy of a classifier trained on synthetic document images is 25%, the accuracy of a classifier trained on faxified document images is 27% and the accuracy of a classifier trained on CycleGAN generated document images is 43%. The accuracy is not a suitable metric to evaluate the performance of the classifiers when imbalanced datasets are involved<sup>3</sup>. The performance evaluation of the classifiers trained on different data distributions is calculated using an imbalanced testing dataset. Hence, it will be beneficial to consider macro average and weighted average F1-score metrics to evaluate the performance of the classifiers. The macro average F1-score of a classifier trained on synthetic document images is 27%, the macro average F1-score of a classifier trained on faxified document images is 34% and the macro average F1-score of a classifier trained on CycleGAN generated document images is 58%. The weighted average F1-score states, 25% real document images were correctly classified when a classifier is trained using CycleGAN generated document images. 43% real document images were correctly classified when a classifier is trained using faxified document images. 31% real document images were correctly classified when a classifier is trained using synthetic document images. Finally, results conclude that the quantitatively CycleGAN generated data distribution could not match real data distribution thoroughly. The faxified data distribution produced by faxifying synthetic document images convincingly matched the real data distribution, therefore, the faxified document images are more similar to the real document images compared to CycleGAN generated document images. The quantitative results are illustrated using a table and bar plot in section 6.3.1. The proposed image-to-image translation application can be improved further in the future using different methods and loss functions.

### 6.3.1 Quantitative Evaluation

Data distributions	Accuracy	Macro average F1-score	Weighted average F1-score
Synthetic document images	25%	27%	31%
CycleGAN generated document images	27%	34%	25%
Faxified document images	43%	58%	43%

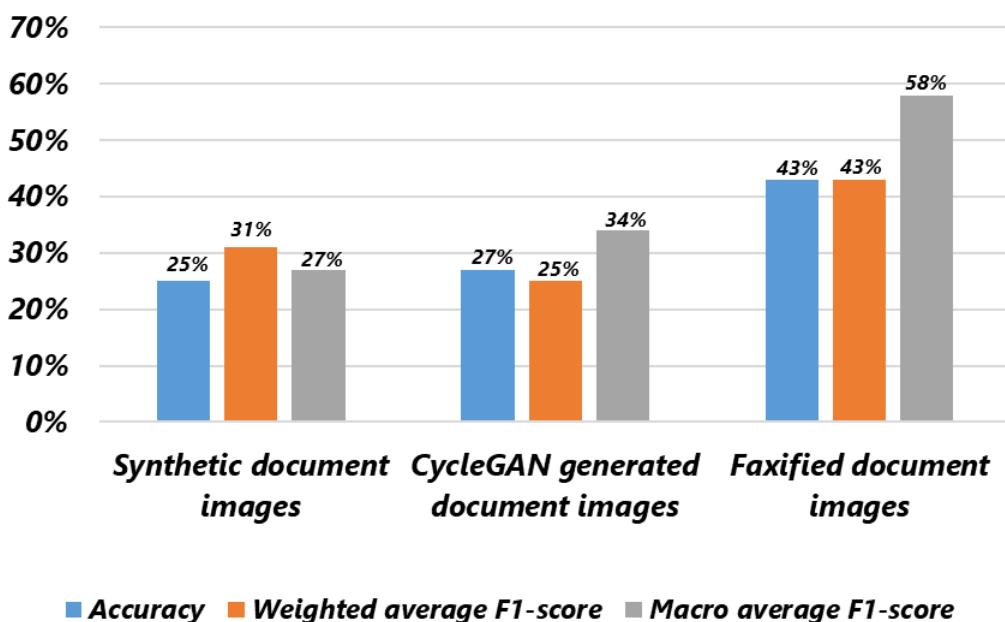
Table 6.4: The accuracies and F1-scores when the classifiers trained on different data distributions and evaluated on testing dataset (Annotated real document images).

The qualitative results are illustrated in section 6.3.2, to understand the visual quality of document images generated using CycleGAN. Some sample synthetic document images are transformed into realistic document images by using the proposed image-to-image translation application. Qualitatively the images generated are very distinct and mode collapse [27] is not observed. The generated realistic

<sup>3</sup><https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distribution/> November 23, 2021

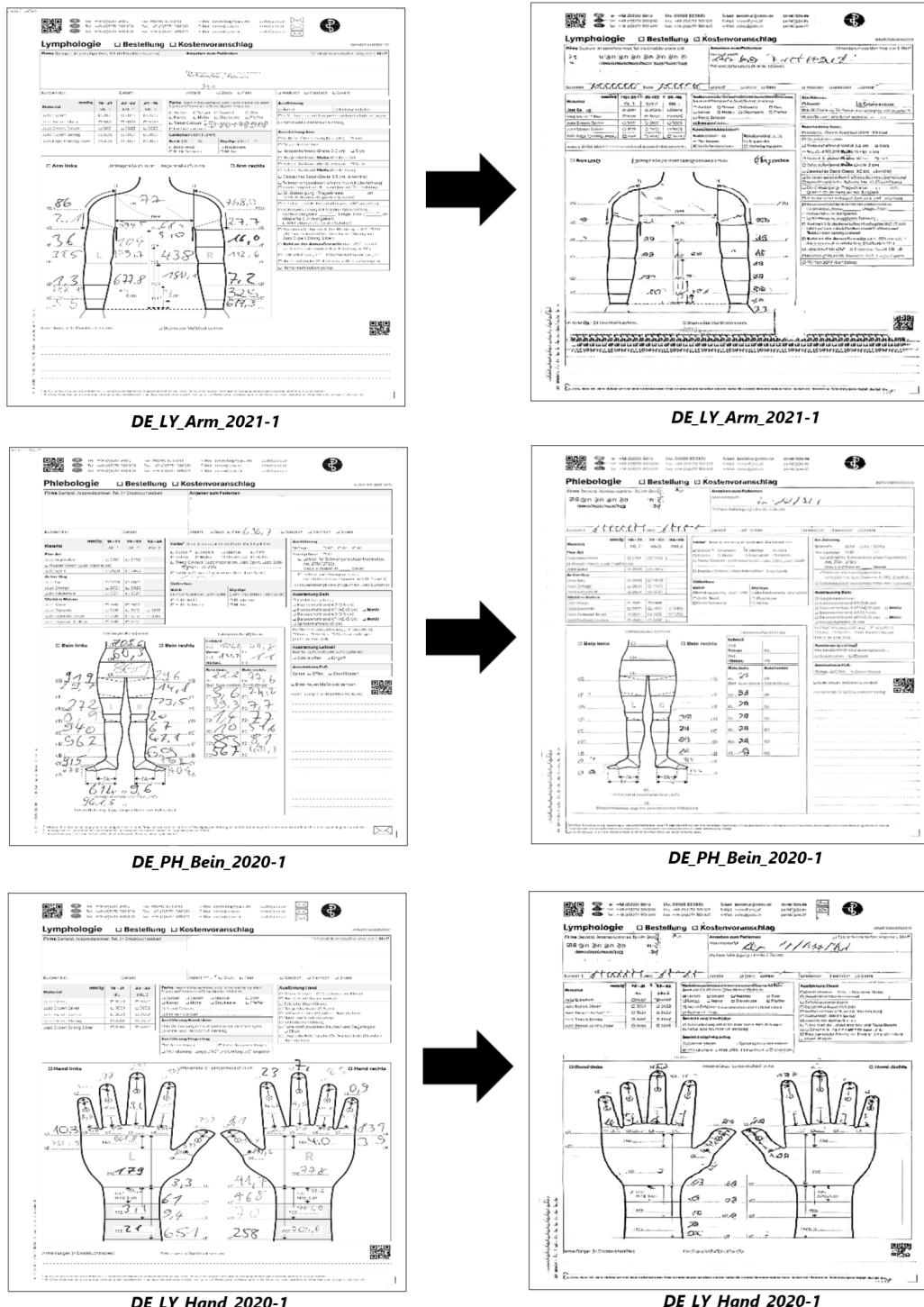
images are in the context, because of cycle-consistency loss. Major image artifacts from the synthetic document images are transformed successfully in the target domain and one can visibly determine the class of generated images. The bold strings, the logo and the QR code present in the synthetic document images are transformed and reconstructed properly in the generated images. The quality of generated document images is adequate for initial research. There is an area for improvement and further research. There are some typical failure cases of the proposed method. For example, figure 6.20 illustrates that, the proposed image-to-image translation application using CycleGAN has failed to transform handwritten crops present in the synthetic document images in the target domain. The handwritten crops are very minor artifacts. It is a challenging task to learn such features. The typical failure cases of the proposed image-to-image transformation are illustrated in section 6.3.3. For example, in figure 6.21 the transformed image has undesired noisy artifacts, in figure 6.22 transformed image has a dark border and noisy artifacts and figure 6.23 illustrates transformed image has random artifacts that are not present in the synthetic document image.

### **Comparison of Accuracies and F1-scores**



**Figure 6.17:** Comparison of accuracies and F1-scores, when the classifiers trained on different data distributions and evaluated on testing dataset (Annotated real document images).

### 6.3.2 Qualitative Evaluation



**Synthetic document images**

**CycleGAN generated document images**

**Figure 6.18:** Synthetic document images transformed into realistic document images by the image-to-image translation application (figure reproduced from elevait GmbH & Co. KG with permission).

DE\_PH\_Bein\_2020-01

DE\_PH\_Bein\_2020-1

DE LY\_Hand\_2020-1

DE LY\_Hand\_2020-1

DE LY\_Bein\_2019-1

DE LY\_Bein\_2019-1

*Synthetic document images**CycleGAN generated document images*

**Figure 6.19:** 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

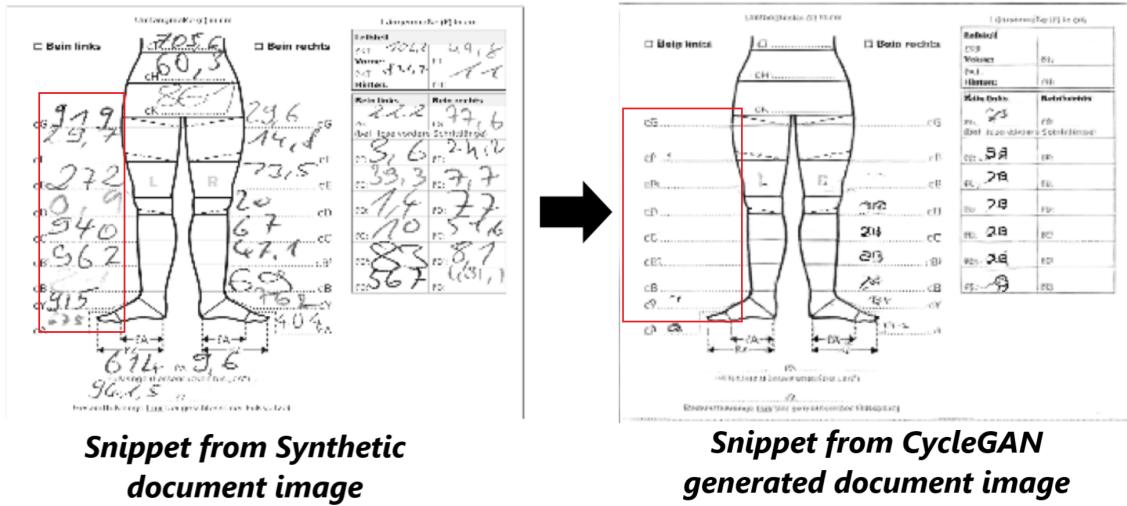


Figure 6.20: The snippet from CycleGAN generated document image illustrates that the proposed image-to-image translation application did not transform or reconstruct handwritten crops in the target domain from the synthetic document image in the source domain (figure reproduced from elevait GmbH & Co. KG with permission).

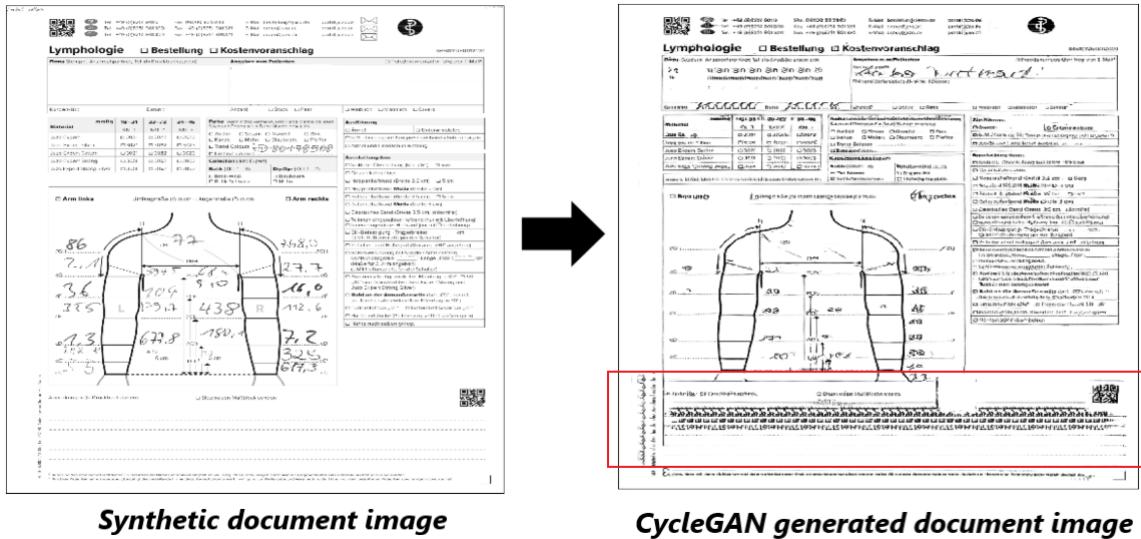
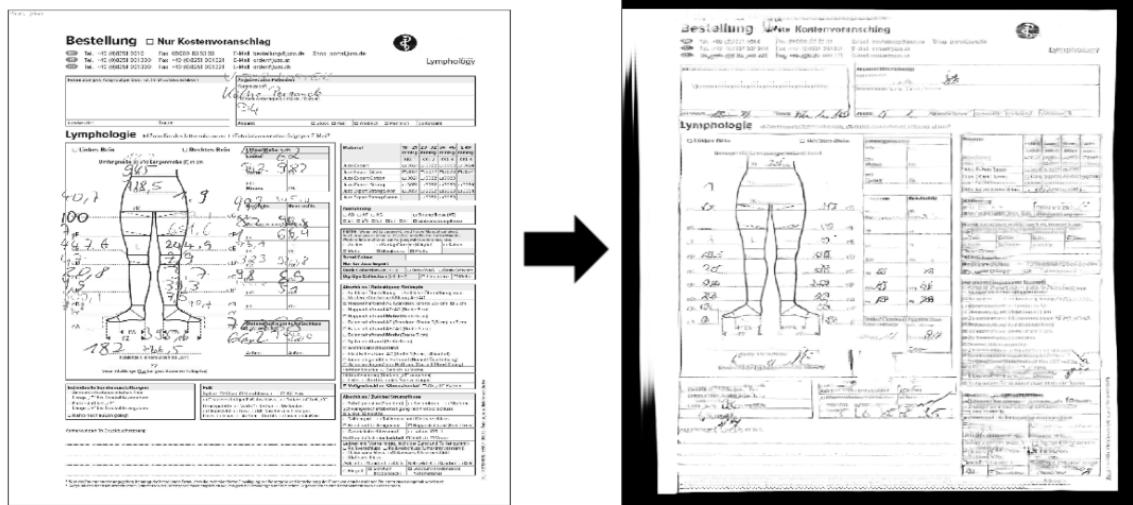


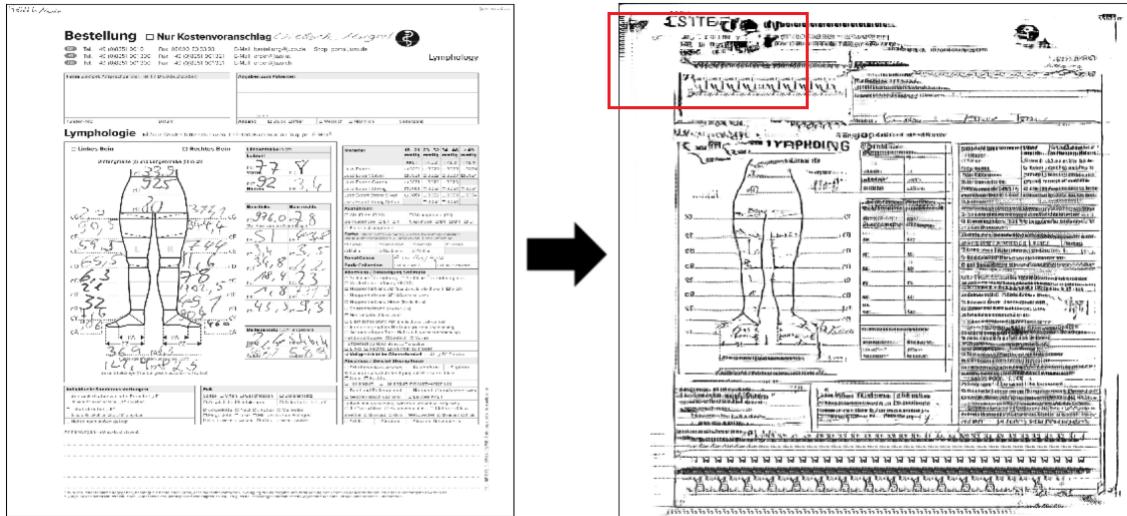
Figure 6.21: The CycleGAN generated document image consists of noisy artifacts that are unrealistic (figure reproduced from elevait GmbH & Co. KG with permission).



**Synthetic document image**

**CycleGAN generated document image**

**Figure 6.22:** 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 6.23:** 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 this thesis is described by outlining the problem statement and the proposed solution. In section 7.1 the overview of this thesis is briefly discussed. 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, many success stories have been written because of deep learning. In the field of computer vision, deep learning is the backbone of numerous sophisticated AI applications. Also, it has grown remarkably due to the introduction of neural networks, which are capable of learning 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, deep learning models have some limitations, like training data scarcity and domain gap between training and real-world data distribution. In traditional deep learning models, neural networks are trained and tested on similar data distribution or in 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 of domain gap between two different data distributions [85]. The domain gap is observed when a model is trained on one specific data distribution, but after deployment, it encounters another new and unseen data distribution, which leads to performance degradation of the model.

A large amount of training data is required to train neural networks to achieve low generalization error and high performance. However, as mentioned earlier, training data is scarce and it is a very difficult job to collect and annotate the data. Hence, inevitably, machine learning engineers are 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 reduce the domain gap between two different domains [85]. One of such domain adaptation methodologies is image-to-image translation, in which images from the source domain are transformed and mapped into the target domain by a function that learns the underlying characteristics extracted from the collection of images in both domains. 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. CycleGAN is one of the popular methods to perform image-to-image translation. The prior image-to-image translation method like cGAN requires paired training dataset from learning to transform and map images from one domain to another domain, but the CycleGAN learns the underlying characteristics of both domains and performs the unpaired, unsupervised mage-to-image translation.

## 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 reduce the domain gap between synthetic data distribution and real data distribution. The CycleGAN is trained using two datasets of document images, the source domain has synthetic document images and the target domain has real document images. The objective of this training would be

to learn to translate an image from the source domain to the target domain and vice versa. The synthetic document images created using templates and handwritten crops. They are transformed into realistic document images to tackle the problem of data scarcity of real document images in the target domain. Once sufficient amount of images are generated in the target domain, tasks like real document images classification can be improved. The quality of images generated using CycleGAN is computed based on how a classifier trained on images generated by the CycleGAN generalizes to the testing dataset of annotated real document images. This way it can be seen how close is the CycleGAN generated data distribution to the real data distribution.

In this thesis, CycleGAN is trained using document images of type form . The form document images consist of minute artifacts like check boxes, small fields and different handwritings, which increases the complexity in the learning of neural networks. Also, most of the synthetic document images used for generating realistic document images are of type "Bein" and they are very similar to each other. 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 6.16. Experiments were conducted by training classifiers with dataset of synthetic document images, faxified document images and CycleGAN generated document images respectively. These classifiers are evaluated using annotated real document images (testing dataset), to understand domain gap between these distributions and real data distribution. The experiment of training a classifier with CycleGAN generated document images and evaluated by using annotated real document images, determines the quality of generated images, also gives an impression of how closely the generated data distribution matches to the 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 and concluding results describe quantitatively the images generated using CycleGAN did not match well to the real data distribution, and it must be improved in future work.

Qualitatively the images generated using CycleGAN are promising for further research (figure 6.3.2). 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 is failed (figure 6.20) and the generated realistic document images have unrealistic noise, dark borders and unrealistic artifacts (figures 6.21, 6.22 and 6.23)). In the next section 7.3, a discussion about improving the proposed image-to-image translation application is discussed, along with thesis limitations.

### 7.3 Future Work

The proposed image-to-image translation application is implemented only using CycleGAN. 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 [49], FastCUT [49], Adversarial Consistency Loss Generative Adversarial Network (ACL-GAN) [74], Variational Autoencoder (VAE) [86] and DCGAN [38] 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 objective 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 [35] to avoid mode collapse and vanishing gradient problems. The quality of generated images possibly would have been better if generators and discriminators of CycleGAN would have optimized using the Wasserstein metric [62]. Wasserstein distance metric is popularly used in WGANs. WGANs improve GAN's training by using a smooth Wasserstein distance metric to minimize the distance between two probability distributions [62]. Furthermore, Shrivastava et al. [61] 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 [61]. Qualitative results show the hand-written crops from synthetic document images did not reconstructed and transformed in the generated images (figure 6.20). In the future, this research can be extended to solve this problem, so the generated images are used to improve the handwriting recognition models. Training CycleGAN sequentially is consumed 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 two datasets are used for the training the CycleGAN, which means the source domain is represented by the dataset of synthetic document images and the target domain is represented by the dataset of 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 GAN Training

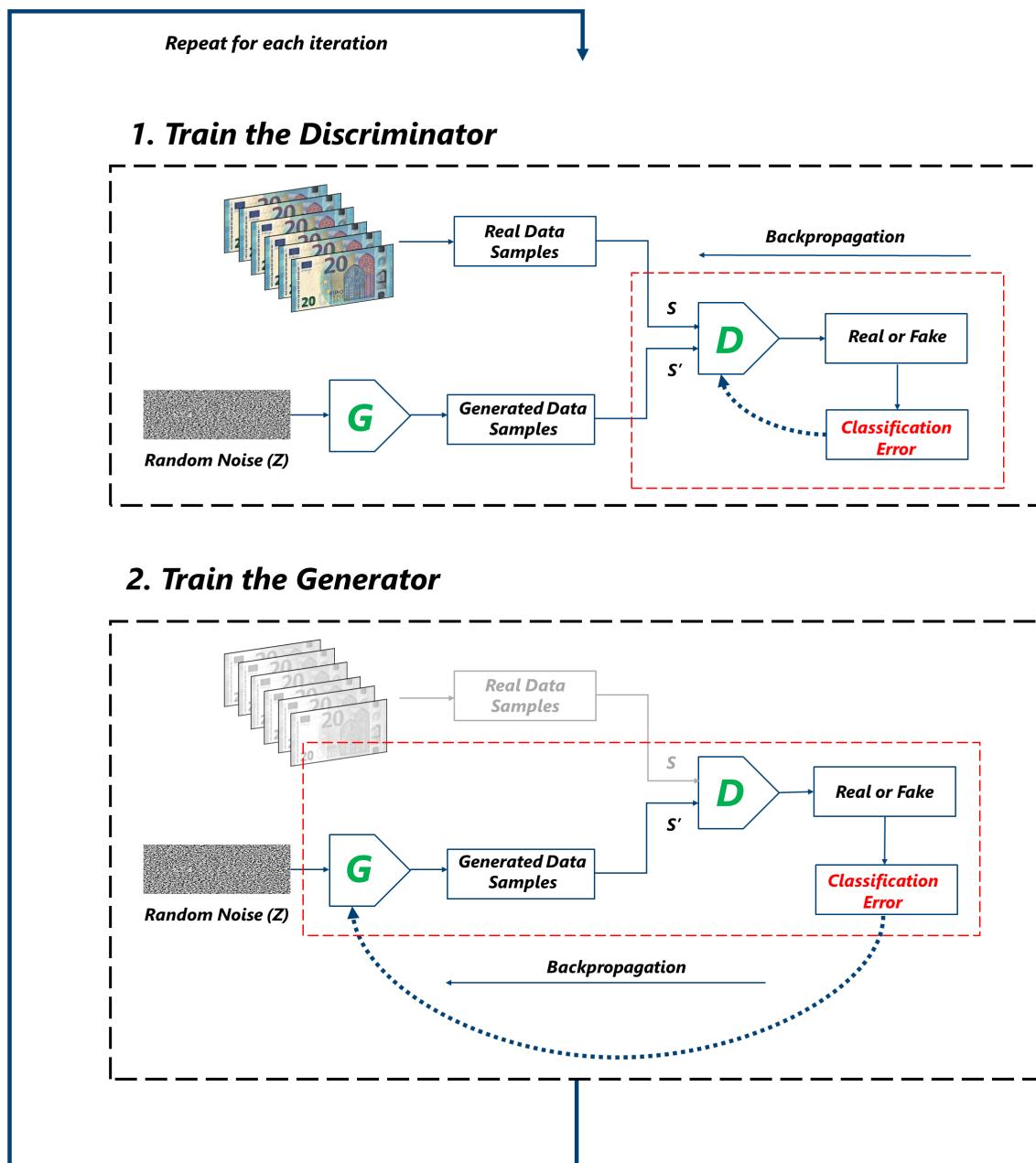


Figure A.1: An illustration of training of the GAN as per the algorithm.

## A.2 Examples of Document Images

**Bestellung**  Nur Kostenvoranschlag

Lymphology

Firma Stempel, Ansprechpartner, Tel. (in Druckbuchstaben)	Angaben zum Patienten																																														
Kommission <sup>1</sup> :																																															
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: _____																																															
<b>Lymphologie</b> <input checked="" type="checkbox"/> Zutreffendes bitte ankreuzen <input type="checkbox"/> Fotodokumentation folgt per E-Mail <sup>2</sup>																																															
Zubehör <input type="checkbox"/> Lymphpad Line <input type="checkbox"/> Lymphpad Square	<b>Abschluss / Befestigung</b> <b>Rundstrick</b> <ul style="list-style-type: none"> <li><input type="checkbox"/> Gestrickabschlussrand</li> <li><input type="checkbox"/> Noppenhafrand (Breite 3,5 cm)</li> <li><input type="checkbox"/> Balancehafrand (Breite 3,5 cm - ab 01.2019)</li> <li><input type="checkbox"/> BH-Befestigung - Trägerbreite: _____ cm Mit Haftuntertritt (an der Schulter)</li> <li><input type="checkbox"/> Schulter- und Haltegurt (Umfang „cHI“ angeben)</li> </ul> <b>Flachstrick</b> <ul style="list-style-type: none"> <li><input type="checkbox"/> Gestrickabschlussrand</li> <li><input type="checkbox"/> Noppenhafrand (Breite 3,5 cm) <input type="checkbox"/> 5 cm</li> <li><input type="checkbox"/> Noppenhafrand Motiv (Breite 5 cm)</li> <li><input type="checkbox"/> Elastisches Band (Breite 3,5 cm, silikonfrei)</li> <li><input type="checkbox"/> Überhöhung (bei „cG“) <input type="checkbox"/> Überhöhung max. (bei „cG“) ¾ innen eingenähter Hafrand (<b>nur mit Überhöhung</b>)</li> <li><input type="checkbox"/> Innen eingenähter Hafrand (<b>nur mit Überhöhung</b>)</li> <li><input type="checkbox"/> BH-Befestigung - Trägerbreite: _____ cm Mit Haftuntertritt (an der Schulter)</li> <li><input type="checkbox"/> Schulter- und Haltegurt (Umfang „cHI“ angeben)</li> <li><input type="checkbox"/> Boleroverbindung mit Ärmeln / Armansätzen Konfektionsgröße: _____ - Länge „PHH“: _____ cm (Maße für 2. Arm angeben) Mit Haftuntertritt (an der Schulter)</li> </ul>																																														
<b>Material</b> <table border="1"> <tr> <th></th> <th>18 - 21 mmHg</th> <th>23 - 32 mmHg</th> <th>34 - 46 mmHg</th> </tr> <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><b>Flachstrick</b></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> </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	<b>Flachstrick</b>				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																																												
<b>Flachstrick</b>																																															
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																																												
<b>Ausführung</b> <table border="1"> <tr> <td>Rundstrick</td> <td><input type="checkbox"/> Ärmel</td> </tr> <tr> <td>Flachstrick</td> <td><input type="checkbox"/> Ärmel <input type="checkbox"/> Unterarmstulpe</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> In Verbindung mit Kompressionshandschuh zu tragen</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> Ärmel und Handschuh einteilig</td> </tr> </table>				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																																					
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																																															
<b>Farbe</b> Wenn nichts vermerkt, wird Farbe Mandel geliefert. Ausführungen in Silver und Cotton erhältlich in Farbe Mandel. Weitere Informationen zur Farbauswahl siehe Umschlag.																																															
<b>Rundstrick - Juzo Soft</b> <table border="1"> <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> </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																																															
<b>Trend Colour</b> <table border="1"> <tr> <td>Rundstrick</td> <td><input type="checkbox"/> Batik-Weiß</td> <td><input type="checkbox"/> Batik-Schwarz</td> </tr> </table>				Rundstrick	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																									
Rundstrick	<input type="checkbox"/> Batik-Weiß	<input type="checkbox"/> Batik-Schwarz																																													
<b>Batik Collection</b> <input type="checkbox"/> Batik-Weiß <input type="checkbox"/> Batik-Schwarz																																															
<b>Rundstrick - Juzo Dynamic</b> <table border="1"> <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> </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																																															
<b>Flachstrick - Juzo Expert / Juzo Expert Strong</b> <table border="1"> <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> </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																																													
<b>Trend Colour</b> <table border="1"> <tr> <td>Batik Collection (Juzo Expert)</td> <td><input type="checkbox"/> Batik-Weiß</td> <td><input type="checkbox"/> Batik-Schwarz</td> </tr> </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																																													
<b>Individuelle Sonderausstattungen</b> <table border="1"> <tr> <td>Flachstrick</td> <td></td> </tr> <tr> <td colspan="2"><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)</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> Naht an der Armaußenseite (bei „cG“, nur mit anatomisch abgewinkelten Form 30°)</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> Trikofutter bei „cE“ <input type="checkbox"/> Silver</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> Nähte nach außen gelegt</td> </tr> <tr> <td colspan="2"><input type="checkbox"/> Hafrandstopper (Platzierung seitlich außen quer)</td> </tr> </table>				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"/> Hafrandstopper (Platzierung seitlich außen quer)																																	
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"/> Hafrandstopper (Platzierung seitlich außen quer)																																															

<sup>1</sup> 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.

<sup>2</sup> 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 Fehler vorbehalten.

**Figure A.2:** 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
		Sollenzahl: .....	

**Lymphologie**  Zutreffendes bitte ankreuzen  Fotodokumentation folgt per E-Mail<sup>2</sup>

<b>Linkes Bein</b>	<b>Rechtes Bein</b>	<b>Längenmaße in cm</b>	<b>Material</b>
Umfangmaße (c) und Längenmaße (l) in cm		Leibteil	18-21 22-23 24-25 26-27 28-29 29-30 31-32 33-34 35-36 37-38 39-40 41-42 43-44 45-46 47-48 49-50 51-52 53-54 55-56 57-58 59-60 61-62 63-64 65-66 67-68 69-70 71-72 73-74 75-76 77-78 79-80 81-82 83-84 85-86 87-88 89-90 91-92 93-94 95-96 97-98 99-100 ≥ 101
PKT Vorne:	PKT Hinten:	Juzo Expert	mmHg mmHg mmHg mmHg
cT.....	cH.....	Juzo Expert Silve	KK 1; KKL 2 KK 3; KKL 4
cK.....		Juzo Expert Cotton	<input checked="" type="checkbox"/> 3021 <input type="checkbox"/> 3022 <input type="checkbox"/> 3023 <input type="checkbox"/> 3024
cF.....	S.7.2. cG.....	Juzo Expert Strong	<input type="checkbox"/> 3051 <input type="checkbox"/> 3052 <input type="checkbox"/> 3053 <input type="checkbox"/> 3054
cE.....	49.3. cF.....	Juzo Expert Strong Silver	<input type="checkbox"/> 3051 <input type="checkbox"/> 3052 <input type="checkbox"/> 3053 <input type="checkbox"/> 3054
cD.....	43.0. cE.....	Juzo Expert Strong Cotton	<input type="checkbox"/> 3051 <input type="checkbox"/> 3052 <input type="checkbox"/> 3053
cC.....	37.0. cD.....		
cB.....	36.0. cC.....		
cB <sup>1</sup> .....	28.7. cB <sup>1</sup> .....		
cB <sup>2</sup> .....	22.5. cY.....		
cA.....	31.9. cA.....		
Fußlänge (Fersenhöhen s. „A“)		Ballensatz „Z“ angeben:	
22.5. cZ..... 22.5. cZ.....		Innen:..... Außen:.....	
Gesamtfußlänge (Nur bei geschlossener Fußspitze)		Innen:..... Außen:.....	
<b>Individuelle Sonderausstattungen</b>		<b>Fuß</b>	<b>Abschluss / Zwickel Strumpfhose</b>
<input type="checkbox"/> Anatomisch angewinkelte Form bei „c“ Inne & Klettverschlüsse „Z“ angebaut		<input type="checkbox"/> Spitzo <input checked="" type="checkbox"/> Offen <input type="checkbox"/> Geschlossen	<input type="checkbox"/> Seitl. Oberehöhung <input type="checkbox"/> Seitliche Oberehöhung max.
<input type="checkbox"/> Trikotfutter bei „c“ innere Klettverschlüsse „Z“ angebaut		<input type="checkbox"/> 7 zusätzliche Hinterbein	<input type="checkbox"/> Vorderbein neith. n. AF/AG
<input type="checkbox"/> Nähte nach a.ßen gelegt		<input type="checkbox"/> Druckpolster <input type="checkbox"/> Nach. Dr. Roter <input type="checkbox"/> Maileulen	<input type="checkbox"/> Noptex Haftband AD (Standard Breite: 3.5 cm) <input type="checkbox"/> 5 cm
		<input type="checkbox"/> Eingeklebt <input type="checkbox"/> Löse <input type="checkbox"/> Mit Tasche zum Einlegen	<input checked="" type="checkbox"/> Noppenhaftband Al-/AG (Breite 5 cm)
		<input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Außen <input type="checkbox"/> Innen <input type="checkbox"/> Außen	<input type="checkbox"/> Noppenhaftband Motiv (Breite: 5 cm)
			<input type="checkbox"/> Spitzenklettband (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 Übersteigung“
			<input type="checkbox"/> % in ein eingebettetes Haftrand und über der Übersteigung)
			<input type="checkbox"/> Haftrandsstopper <input type="checkbox"/> Seilach <input type="checkbox"/> Vorne
			<input type="checkbox"/> Fußbelastung (Unter „Z“) eingetragen
			<input type="checkbox"/> Links <input type="checkbox"/> Rechts <input type="checkbox"/> Fuß mit zu tragen
			<input type="checkbox"/> Vollgestrickt im Oberschenkel <input type="checkbox"/> Aa „CD“ Porosa
			<b>Abschluss / Zwickel Strumpfhose</b>
			<input type="checkbox"/> Schrägverrennung (Seitwand) <input type="checkbox"/> Kastenform <input type="checkbox"/> St. form
			<input type="checkbox"/> Schwangerschaftsbefestigung mit Klettverschluss
			<input type="checkbox"/> Links <input type="checkbox"/> Rechts
			<input type="checkbox"/> Teil langer <input type="checkbox"/> teil „Z“ gerollt mit Klettverschluss
			<input type="checkbox"/> Bund und „Z“ langumrand <input type="checkbox"/> Noppenhaftband (Breite 5 cm)
			<input type="checkbox"/> 11 Gestrickabschlussrand <input type="checkbox"/> 11 elbstell KKL 1
			<input type="checkbox"/> Leibteil mit Vorne („Z“), nicht sie kann die Füllergummie
			<input type="checkbox"/> Reißverschluss <input type="checkbox"/> Hakenverschluss <input type="checkbox"/> Klettverschluss usw.
			<input type="checkbox"/> Zwickel <input type="checkbox"/> Standart <input type="checkbox"/> Klein <input type="checkbox"/> Netzwickel <input type="checkbox"/> Spannung <input type="checkbox"/> 10 cm
			<input type="checkbox"/> Schlitz <input type="checkbox"/> Scrotum <input type="checkbox"/> Skrotum Netzhautschutz

<sup>1</sup> Wird der Patientenname eingesetzt, bestätigt die bestellende Firma, dass die rechts vorliegende Erstellung zur Weitergabe und Verarbeitung der Daten von den berührten Personen ausreichend geworden ist.  
<sup>2</sup> Aufgrund des diversen rechtlichen Grundzuges darf Datenspeicherung empfohlen werden. Insolgebei struktureller Sicherheitlichkeit ist diese Verwendung zu überlassen.

Figure A.3: Example of real document image (figure reproduced from elevait GmbH & Co. KG with permission).



## Lymphologie Bestellung Kostenvoranschlag

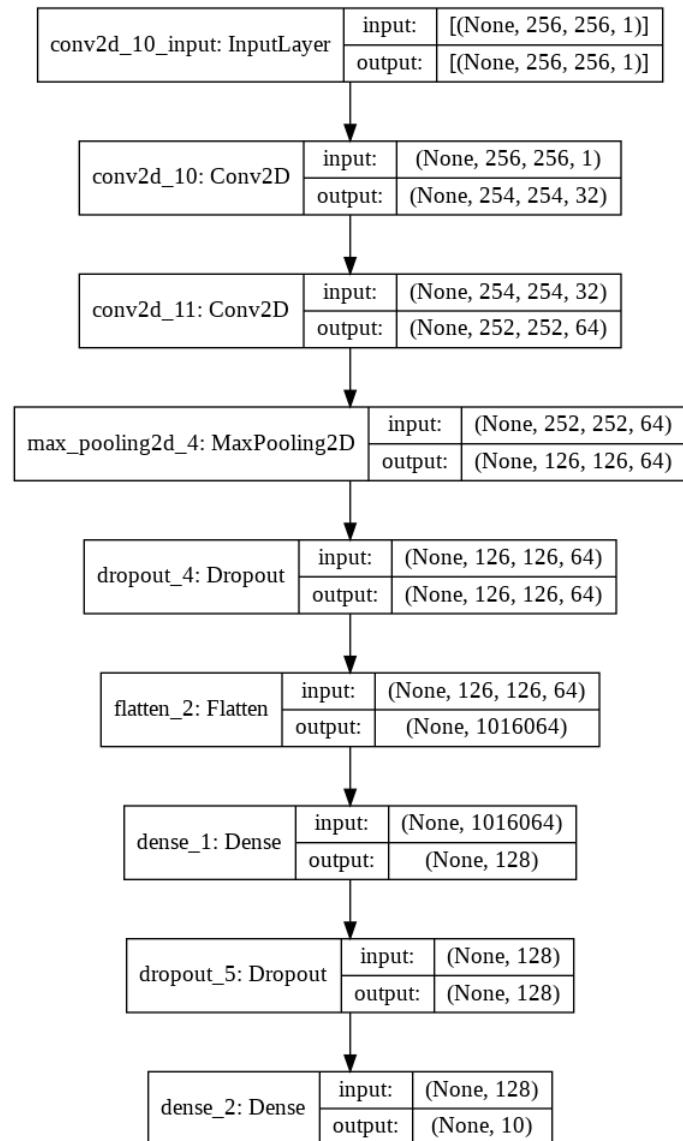
8650LYDEU012020

Firma Stempel, Ansprechpartner, Tel. (in Druckbuchstaben)				Angaben zum Patienten		<input type="checkbox"/> Fotodokumentation folgt per E-Mail!
				Komplikation <sup>2</sup> :		
				Frühere Anfertigung? KV-Nr. / Datum:		
				Kolodec, Sana		
				251		
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		mmHg	18-21	23-32	34-46	<b>Farbe</b> Wenn nichts vermerkt, wird Farbe Mandel geliefert. Silver und Cotton nur in Farbe Mandel erhältlich.
		KKL 1	KKL 2	KKL 3		
Juzo Expert		3021	3022	3023	<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"/> Pfeffer	
Juzo Expert Silver		3021	3022	3023	<input type="checkbox"/> Trend Colours <input type="checkbox"/> Fashion Colours	
Juzo Expert Cotton		3021	3022	3023		
Juzo Expert Strong		3051	3052	3053		
Juzo Expert Strong Silver		3051	3052	3053		
<b>Collection (Juzo Expert)</b>		Bank (KKL 1 - 3)		Dip Dye (KKL 1 - 2)		<b>Ausführung</b>
				<input type="checkbox"/> Batik-Weiß <input type="checkbox"/> Blaubeere <input type="checkbox"/> Batik-Schwarz <input type="checkbox"/> Mohn		
<input type="checkbox"/> Arm links		Umfangmaße (c) in cm - Längemaße (l) in cm				<input type="checkbox"/> Arm rechts
		H	CHI	PCH	PGH	
		86	72		758,0	
		2,1				
		cG				
		36				
		375	L	704		
		cE				
		13		73,7		
		cC				
		292,8		677,8		
		cC				
		55				
		cC				
		632	.6 cm			
		PCC				
		180,1				
		PCE				
		438	R			
		cE				
		112,6				
		cD				
		7,2				
		cD				
		325				
		cC				
		677,3				
		cC				
Anmerkungen (in Druckbuchstaben):						<input type="checkbox"/> Bitte neuen Maßblock senden

KONT301001 8650 LY 01/2020 - Änderungen und Irrtümer vorbehalten.

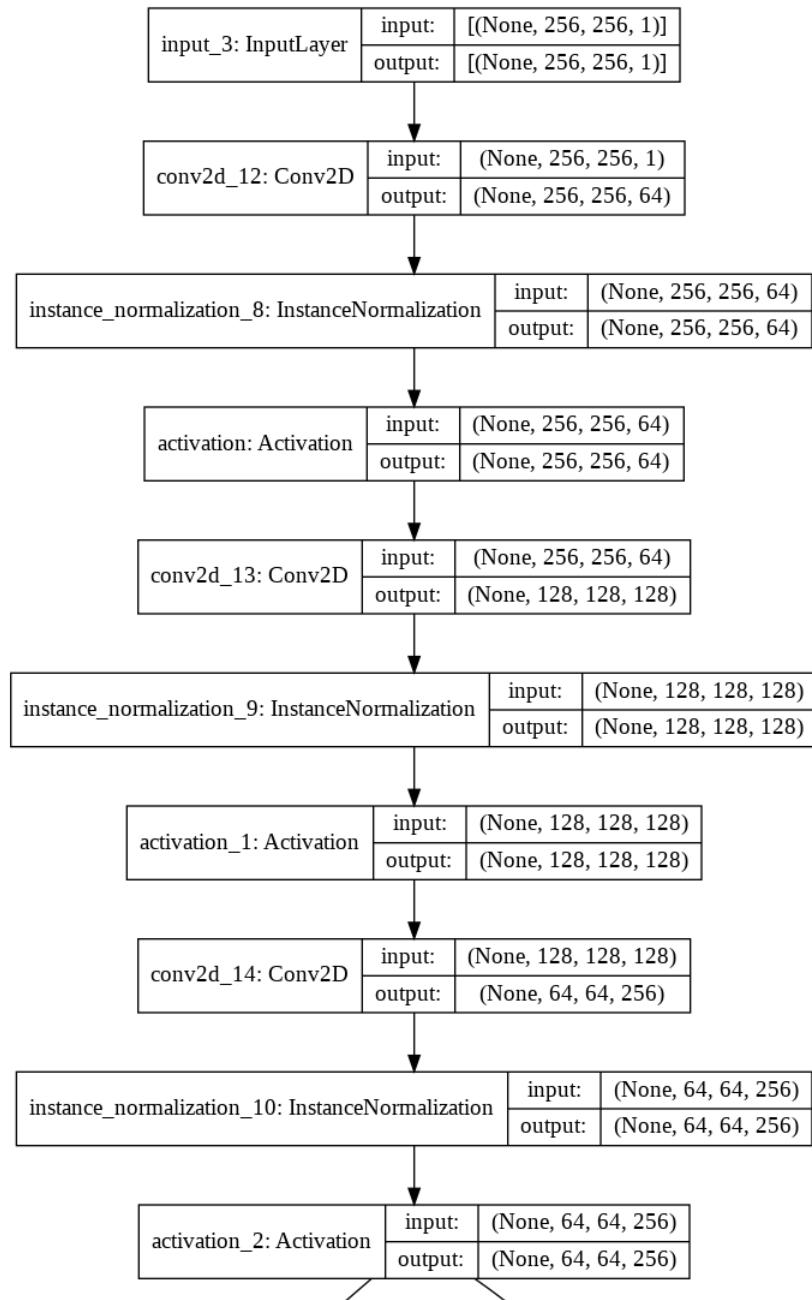
**Figure A.4:** Examples of faxified document image (figure reproduced from elevait GmbH & Co. KG with permission).

### A.3 Classifier Architecture Diagram

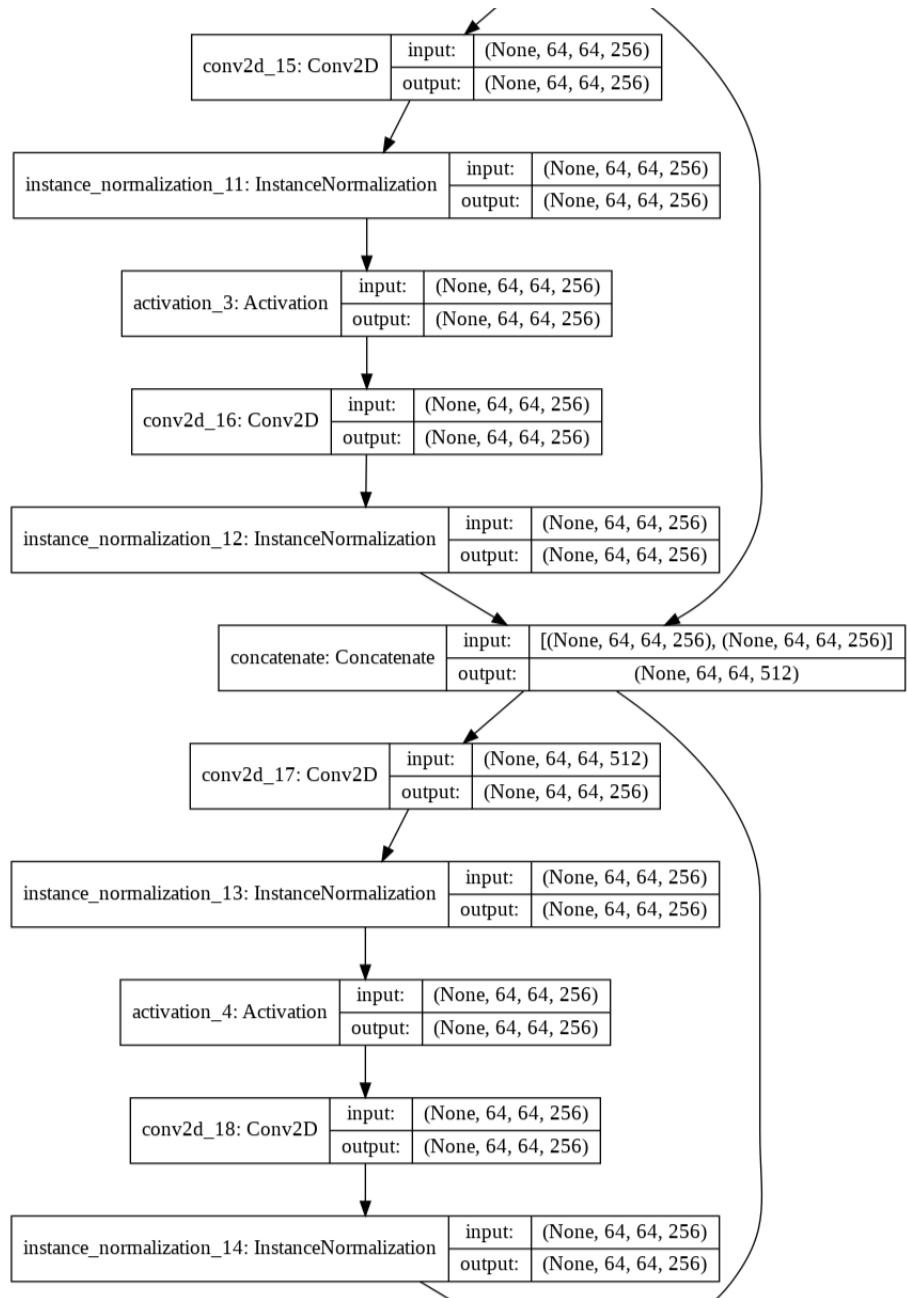


**Figure A.5:** Classifier model summary.

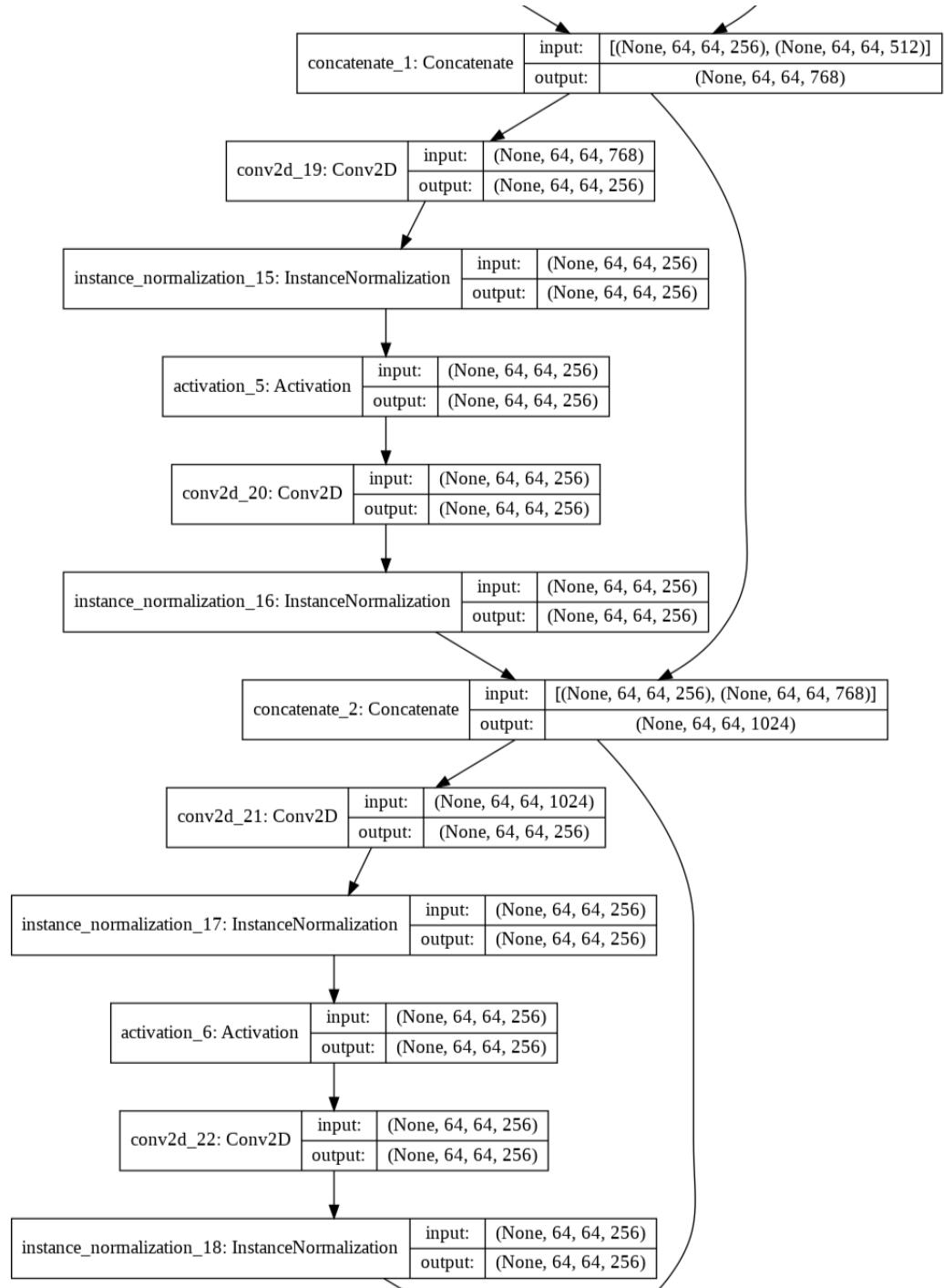
## A.4 Generator Model Summary



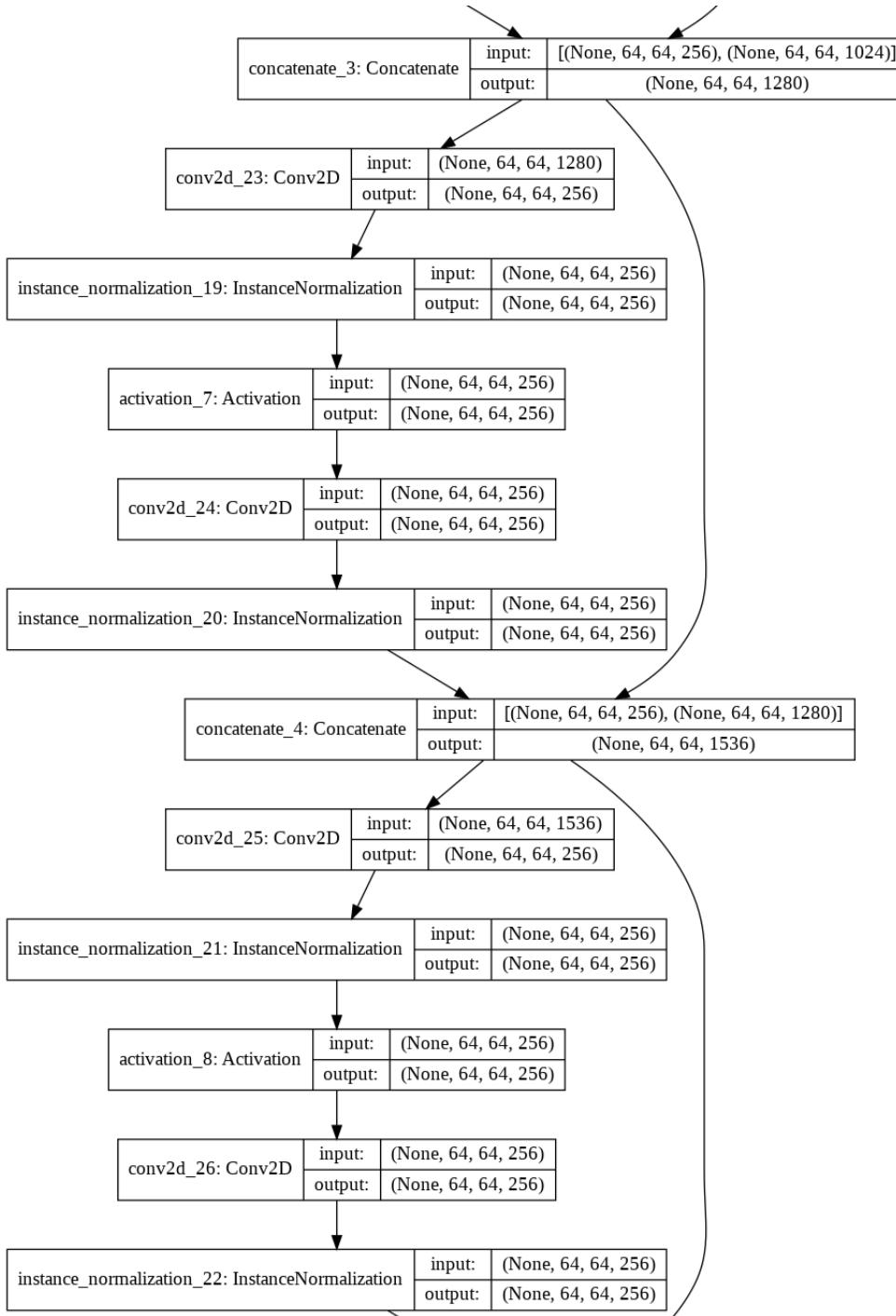
**Figure A.6:** Generator model summary. Continue to next page.



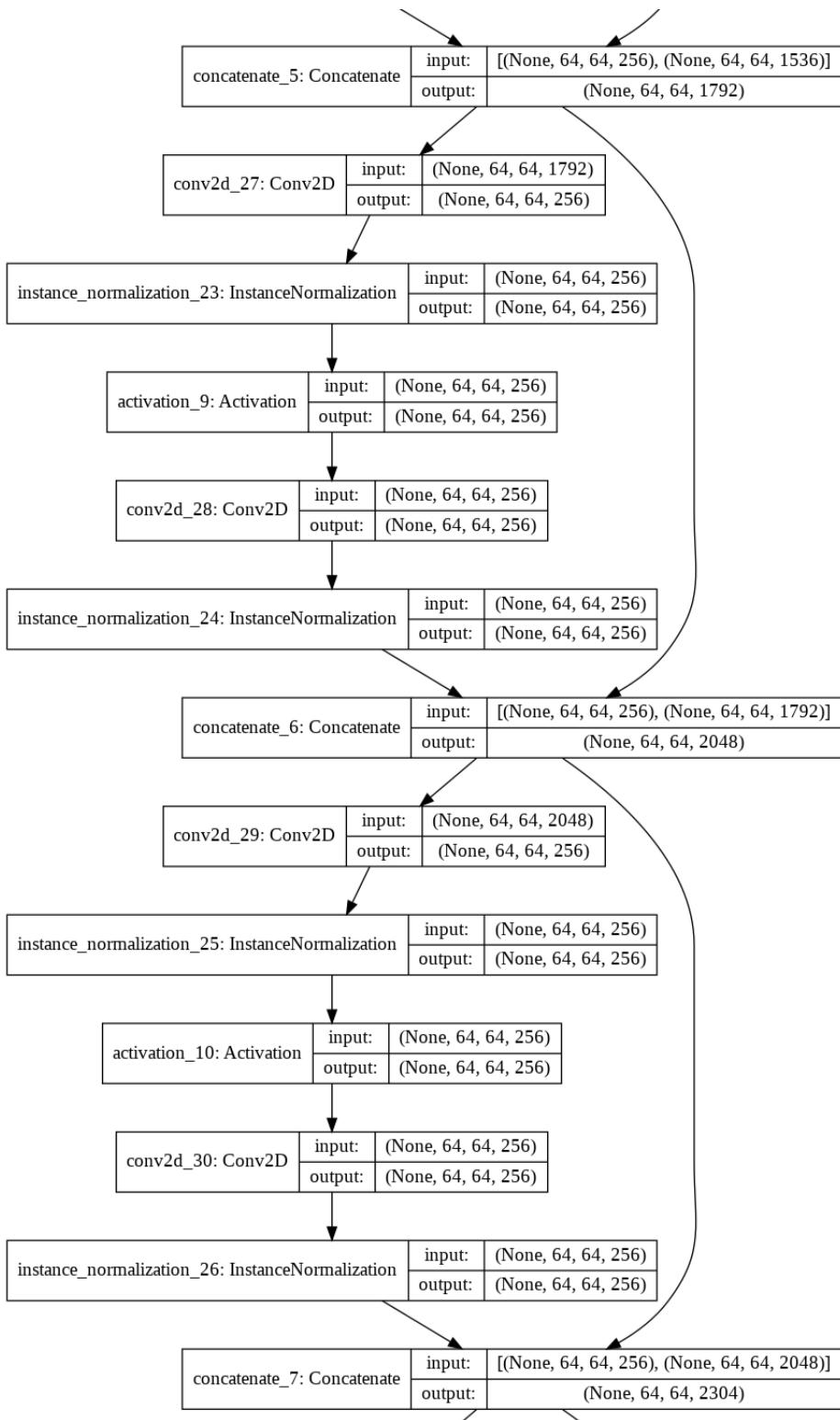
**Figure A.7:** Generator model summary. Continue to next page.



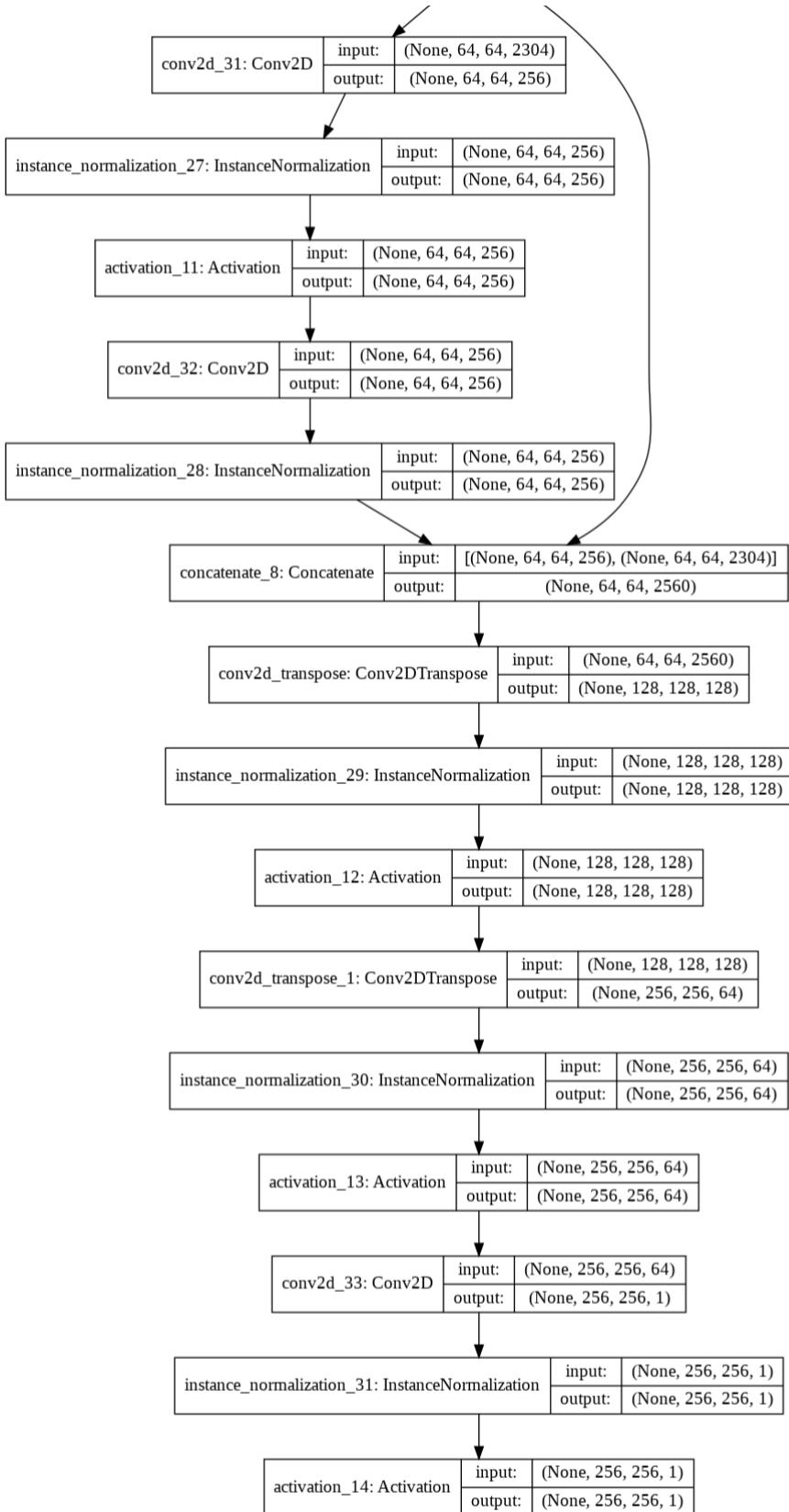
**Figure A.8:** Generator model summary. Continue to next page.



**Figure A.9:** Generator model summary. Continue to next page.

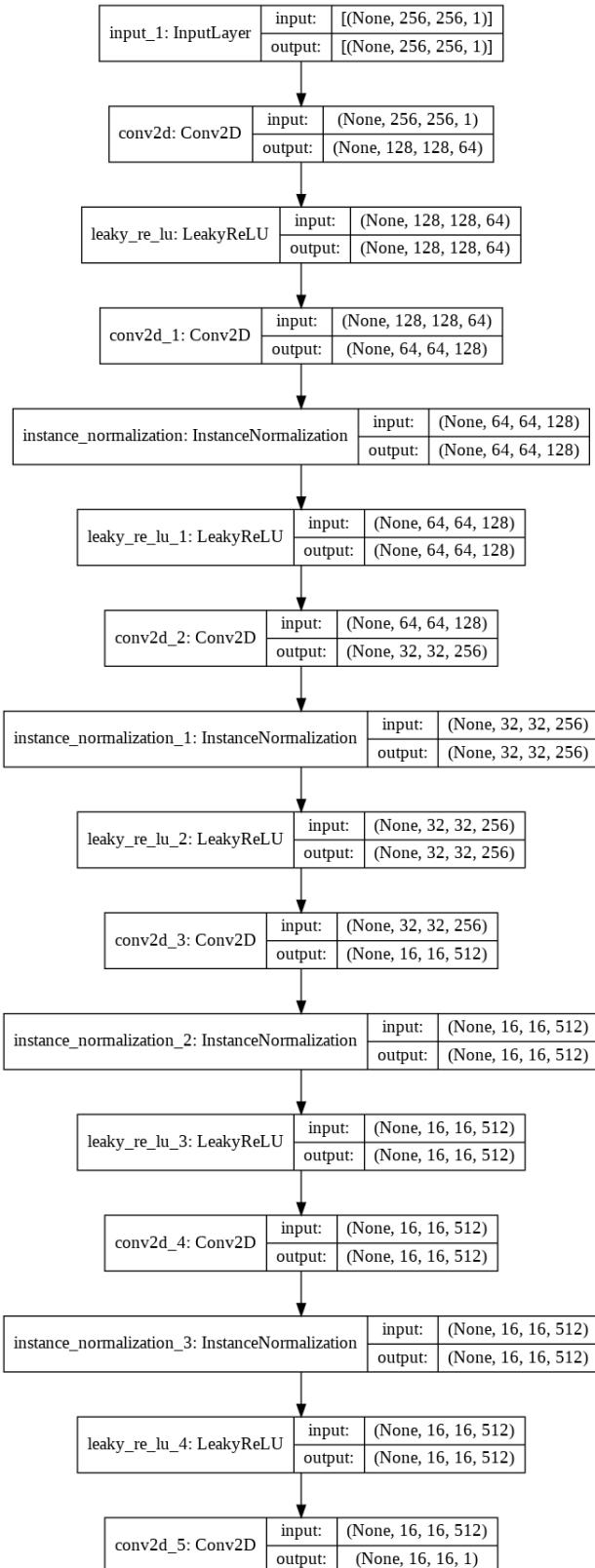


**Figure A.10:** Generator model summary. Continue to next page.



**Figure A.11:** Generator model summary. Ends here.

## A.5 Discriminator Model Summary



**Figure A.12:** Discriminator model summary.

# Bibliography

- [1] J. Langr and V. Bok. *GANs in Action: Deep learning with Generative Adversarial Networks*. Manning Publications, 2019.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning (adaptive computation and machine learning series). *Cambridge Massachusetts*, pages 321–359, 2017.
- [3] Kun-Hsing Yu, Andrew L. Beam, and Isaac S. Kohane. Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10):719–731, 2018.
- [4] 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.
- [5] 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.
- [6] Daniela Girimonte and Dario Izzo. *Artificial Intelligence for Space Applications*, pages 235–253. Springer London, London, 2007.
- [7] 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.
- [8] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy, 2017.
- [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] 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.
- [11] Patrick Hemmer, Niklas Kühl, and Jakob Schöffer. Deal: Deep evidential active learning for image classification, 2020.
- [12] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.
- [13] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.
- [14] Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees, 2020.
- [15] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [16] 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.
- [17] 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.
- [18] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

- [19] Q. A. Bui, D. Mollard, 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.
- [20] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- [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] Monika Sharma, Abhishek Verma, and Lovekesh Vig. Learning to clean: A gan perspective, 2019.
- [24] 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.
- [25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.
- [26] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications, 2021.
- [27] Hoang Thanh-Tung and Truyen Tran. On catastrophic forgetting and mode collapse in generative adversarial networks, 2020.
- [28] 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.
- [29] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010.
- [30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [31] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations, 2012.
- [32] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop, 2014.
- [33] P Manisha and Sujit Gujar. Generative adversarial networks (gans): What it can generate and what it cannot?, 2019.
- [34] 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.
- [35] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017.
- [36] 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.
- [37] 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.
- [38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.
- [39] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network, 2017.
- [40] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2017.

- [41] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.
- [42] 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.
- [43] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold, 2018.
- [44] 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.
- [45] Wesley Peisch, Luca Pistor, and Samarpreet Singh Pandher. Colorizing grayscale paintings using cgans, 2021.
- [46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [47] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks, 2018.
- [48] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.
- [49] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation, 2020.
- [50] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.
- [51] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.
- [52] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.
- [53] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding, 2020.
- [54] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views, 2019.
- [55] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks, 2018.
- [56] 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.
- [57] 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.
- [58] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning, 2017.
- [59] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference, 2017.
- [60] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks, 2016.
- [61] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training, 2017.
- [62] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

- [63] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.
- [64] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.
- [65] 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.
- [66] 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.
- [67] Rikiya Yamashita, Mizuho Nishio, Richard Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 06 2018.
- [68] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.
- [69] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [70] 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.
- [71] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, pages 92–101, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [72] Ashwani Kumar. Ordinal pooling networks: For preserving information over shrinking feature maps, 2018.
- [73] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2014.
- [74] Yihao Zhao, Ruihai Wu, and Hao Dong. Unpaired image-to-image translation using adversarial consistency loss, 2021.
- [75] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2017.
- [76] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation, 2016.
- [77] 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.
- [78] 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.
- [79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [80] 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.
- [81] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [82] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017.
- [83] David M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2020.

- [84] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. Thresholding classifiers to maximize f1 score, 2014.
- [85] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation, 2020.
- [86] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.



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

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: **22.11.2021**

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.