Haakon Johansen Jonsson

Cycle GANs for Forgery Detection

master thesis, spring 2021

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

With the increasing capabilities of deep learning models concerns are being raised about the use of machine learning as a tool for generating fake media. This usually done using using generative neural network architectures such as generative adversarial networks (GANs).

The topic of this thesis is one such issue, the potential of the generative deep learning architecture CycleGAN for generating and detecting fake handwriting.

The thesis demonstrate the ability of the CycleGAN architecture to generate text in the style of handwriting using style transfer and establish some of the advantages and limitations of using CycleGANs for generating handwriting and detecting forged handwriting.

This is done by training CycleGAN models to map between domains of handwriting in different styles. This yields generators trained to map handwriting samples from one style domain to an equivalent sample in the other style domain, and discriminators trained to detect if handwriting samples are real or generated by the generators. These models are then tested in their ability to generate and detect fake handwriting respectively.

Preface

This project is the master thesis of Haakon J. Jonsson conducted at the Department of Computer and Information Science at NTNU. The project is supervised by Keith Downing.

Haakon Johansen Jonsson
Trondheim, June 11, 2021

Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the problem that will be addressed by the paper, what the motivation behind choosing this problem is, how the paper intend to approach this problem in a unique way, and the specific goals and research questions the paper intend to answer relating to the problem.

1.1 Background and Motivation

Handwriting is often used to identify the writer since individuals have a unique handwriting style. Handwriting is commonly used as proof of identity and intent in the form of signatures, and autographs are valued by collectors. This makes it important to understand the techniques that can be used to generate handwriting and makes the detection of forged handwriting an important problem.

The problem of handwriting generation and detection of forged handwriting has been extensively studied with many proposed solutions, however this paper intend to take a unique approach by using a CycleGAN architecture.

Motivated by the success of the CycleGAN architecture in image-to-image translation problems this paper intend to address the problem of generating handwriting in the style of a specific individual by building a system consisting of a CycleGAN which will be trained to map between the handwriting of two individuals. The generators should then be trained to generate the handwriting of one individual from the handwriting of the other.

The handwriting forgery detection problem is a binary classification problem asking if a handwriting sample is forger or not. But, this paper intend to address the handwriting forgery detection problem using the discriminator component of the CycleGAN trained to solve the related image-to-image translation problem.

What makes this approach so unique is that the classifier is not trained on

actual samples of forgery, rather, the system is trained to do it's own forging which is used to simultaneously train the classifier.

1.2 Goals and Research Questions

The main goal of this thesis is to explore how the advances made by the CycleGAN architecture in the area of unpaired image-to-image translation can be used for generating handwriting and handwriting forgery detection and what the advantages and disadvantages of this architecture is. Due to structure of the CycleGAN architecture it is natural to explore both the generation of handwriting and the detection of forgery.

Goal *To explore the potential of the CycleGAN architecture for generating text in the style of handwriting and detection of forged handwriting.*

More specifically, to meet this research goal the thesis will answer these tree research questions:

Research question 1 *What are the advantages, disadvantages and limitations of using CycleGAN style transfer to generate text and detect handwriting forgery?*

The goal with this research question is to establish the practical advantages and limitations of the CycleGAN architecture as a way of generating text of a specific style and detect handwriting forgery.

Research question 2 *Can a discriminator trained by a CycleGAN architecture achieve reasonable accuracy in handwriting forgery detection?*

The goal with this research question is not to necessarily beat the state-of-the-art, rather, a "reasonable" accuracy would need to indicate that the CycleGAN system has potential and might compete with other solutions with enough tweaking. Accuracy close to random guess will be considered too low.

Research question 3 *Can a generator trained by a CycleGAN architecture achieve reasonable results in handwriting generation?*

Like research question 2 the goal with this research question is not to achieve state-of-the-art results, rather, it is to demonstrate the potential of the architecture.

While the capability of forging handwriting is interesting in its own right this question also says something about the training of the discriminator. If the

generator can't forge handwriting decently then the discriminator has no good forgery data to train on. If the generator does forge handwriting decently, but the discriminator fails on real forgery examples, it indicates that the discriminator is being trained by the CycleGAN but this training does not generalise to real forgery examples.

1.3 Research Method

My approach to address this research goal will be to implement, train and test style transfer CycleGAN models on domains of text in the style of handwriting to determine their capabilities and limitations. This process will be compared to other architectures from the literature to determine their practical advantages and disadvantages.

1.4 Thesis Structure

The thesis will first cover some background information about the machine learning concepts that will be used for the CycleGAN models and the problem domain of handwritten forgery detection. Then it will present some related paper concerning either handwriting forgery detection or using deep learning for generating text in the style of handwriting. Then it will cover model architecture and how the models will be used in the experiments. Then finally the results of these experiments will be analyzed and discussed to address the research questions.

Chapter 2

Background Theory

This chapter will give an overview of the problem area of handwriting generation and handwriting forgery detection and explain The CycleGAN model, some related models, and some of the key machine learning concepts used in this paper.

2.1 Handwriting Forgery Detection

Handwriting forgery detection can, in general, be stated as a binary classification problem:

Given a sample of handwriting, determine if it is genuine or forged.

However, there is still a lot of room for interpreting exactly what this problem implies leading to many variations of the handwriting forgery detection problem Which puts different constraints on the system intended to solve it. This section will cover some of these variations and explain which variation will be tackled by the system proposed in this paper.

However many of the papers presented in chapter 3 focuses on other versions of the problem and many papers focuses on signatures specifically, not handwriting in general.

2.1.1 Original Document

Whether the handwriting sample is an original physical document or a reproduction has major effect on what kind of systems can be used. If the system has access to the original document then it opens up possibilities for physical and chemical examination of the document. However, since all solutions covered

in this paper are digital we will assume the handwriting data is digital as well, which excludes the possibility of a original physical document.

2.1.2 Types of Handwriting Forgery

There is several ways to forge handwriting which presents varying degree of difficulty for the detection process. This is some ways to forge handwriting in increasing degree of difficulty:

- Blind forgery. This is the case when the forger does not have access to samples of genuine handwriting. This is the easiest case since the forger would be unable to mimic the style of the target individual.
- Simulated forgery. This is the case when the forger have access to samples of genuine handwriting and is mimicking the style by hand. The difficulty in this case would depend on the forgers ability to mimic the style.
- Tracing. This is the case when the forger is tracing an exact copy of existing handwriting by hand. This can be extremely difficult to detect without access to the original document.
- Optical transfer. This is the case when the forger transfers a photocopy of existing handwriting to a target document. This type of forging is not possible to detect without the original document or other auxiliary information.

Additionally there is the question of whether the forger is reproducing the entire document from scratch or tampering with/adding text to an existing document.

The system proposed in this paper will assume the forger is using simulated forgery and forging the document from scratch.

2.1.3 Online and Offline Forgery Detection

In online handwriting forgery detection the system is classifying text while it is being written. That means the system has access to information about the handwriting process itself, not just the final result.

In offline handwriting forgery detection the system is classifying text after it has been written, which means it only has access to the final result. The system proposed in this paper will be doing offline detection.

2.1.4 Writer Dependence

A writer dependent model is trained to classify the handwriting of a single individual, while a writer independent model is trained to classify handwriting in

general. This makes the model more flexible, but it must be capable of generalizing better. The system proposed in this paper will have a writer dependent classification model.

2.2 Machine Learning Concepts

This section will explain some of the machine learning concepts used in this paper.

2.2.1 The CNN Architecture

a convolutional neural network (CNN) is class of neural networks commonly used when working with image data that is inspired by the brain's visual cortex. Convolutional neural network are networks whose layers are primarily convolutional layers. These layers takes advantage of two properties image data: Firstly, if a feature is worth computing for one region of the image it is probably worth computing for the entire image. Secondly, when computing features for a region the most useful features from the previous layer is the features for nearby regions. The structure of a typical CNN is shown in figure 2.1.

A convolutional layer works by preforming a convolution on the previous layer with a convolution kernel for each new feature. The values in this kernel is the trainable parameters. Since the kernel is space invariant it computes the same features for the entire image, and since the kernel usually is small only nearby features are used to compute new features. For a convolutional layer the number of trainable parameters depend only on the number of features and the size of the kernels, not on the dimensions of the data. Since images tend to be relatively large this will significantly reduces the number of parameters compared to a fully connected layer which has a regularizing effect on the model.

Convolutional neural networks usually also have pooling layers. Pooling layers are layers that reduce the dimensions of the data by combining clusters of features. This is intended to generate higher level features covering a larger part of the original image.

2.2.2 The RNN Architecture

A recurrent neural network (RNN) is a class of neural networks commonly used for sequential data. Recurrent neural networks differs from normal feed-forward networks by incorporating memory units. When running on a sequence of data points the network updates the state of these memory units on each run, this state is then feed as input to the run on the next data point as shown in figure 2.2. This allows recurrent neural networks to retain memory. This is useful for

tasks such as handwriting recognition as it allows the model to have context from previous characters when process the next character.

2.2.3 Supervised-, Unsupervised- and Transfer Learning

Supervised learning is machine learning using labeled data. The goal is to learn a mapping from an input domain to an output domain from a dataset of (input, output) samples.

Unsupervised learning is machine learning using unlabeled data. The goal is to learn patterns in a distribution of input data.

Transfer learning involves using knowledge a model learned from solving one problem to help solve another problem. For example an encoder can be trained to create a word embedding which can be used to simplify a more complex text processing task.

2.2.4 The Encoder-Decoder Architecture

The encoder-decoder architecture consists of an encoder which maps the input domain to a low dimensional latent space. This latent space is usually of fixed length one dimensional vectors. A decoder then maps these vectors to the output domain.

The idea is that the encoder generates a vector of high level features which the decoder can use to produce the output. Since a lot of fine detail is lost in this process image processing tasks often uses skip connections which bypasses the latent space and gives the decoder access to the fine detail of the input.

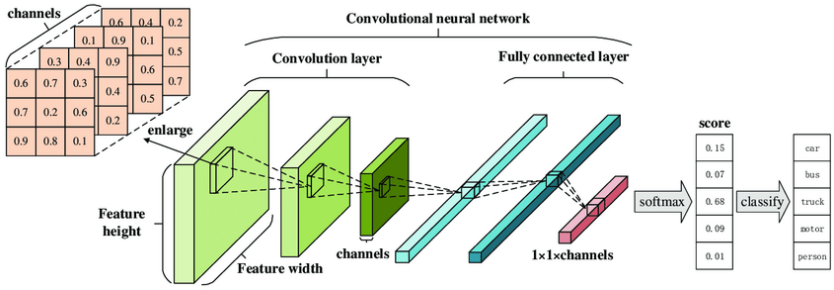

Figure 2.1: Structure of a typical CNN. Adapted from Kang et al. [2019].

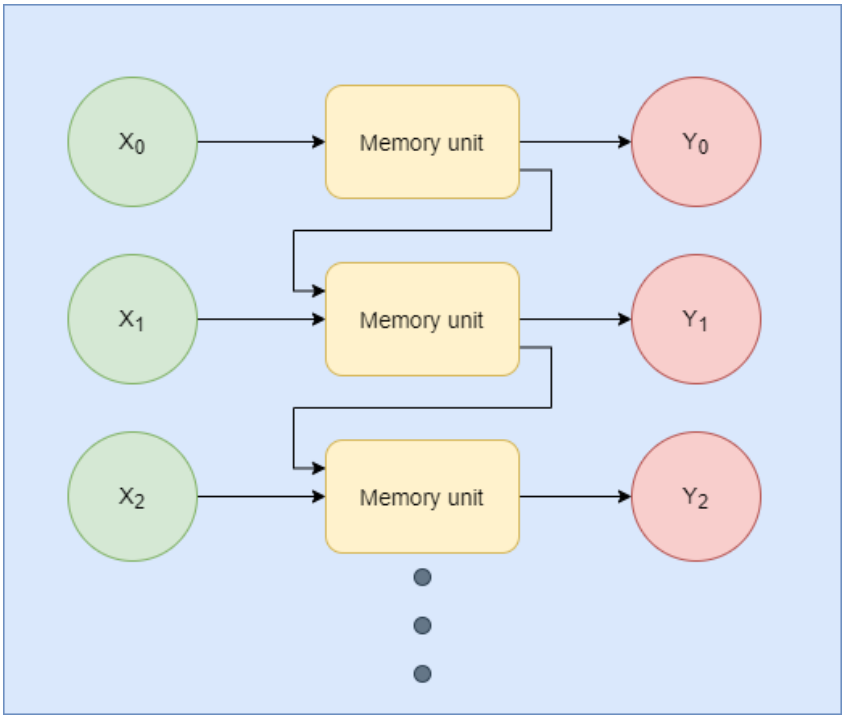

Figure 2.2: Information flow of a RNN memory unit run on multiple data points.

2.2.5 Normalization

Normalization is a common machine learning technique that standardizes the distribution of the data passing through the network. This is done to prevent layers from shifting the distribution of the data as they are trained as this would require downstream layers to adjust to the new distribution.

One common normalization technique is batch normalization introduced by Ioffe and Szegedy [2015] which is shown to improve training speed and acting as a regularizer.

Batch normalization is done by computing the mean and standard deviation for each channel in each batch, which for channel c for a batch of size N with samples of size $H \times W$ is given by:

$$\mu_c = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{ijkc}$$

$$\sigma_c^2 = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{ijkc} - \mu_c)^2$$

Then the batch is normalized by shifting by the batch mean and scaling by the batch standard deviation:

$$\hat{x_c} = \frac{x_c - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}$$

Where ϵ is a small value to prevent division by zero.

Instance normalization is a normalization technique introduced by Ulyanov et al. [2017] who showed improvements over batch normalization on stylization tasks. It works similarly to batch normalization, but it computes the mean and standard deviation for each sample in the batch independently. This is given by:

$$\mu_{ic} = \frac{1}{HW} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{jkc}$$

$$\sigma_{ic}^2 = \frac{1}{HW} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{jkc} - \mu_{ic})^2$$

$$\hat{x_{ic}} = \frac{x_{ic} - \mu_{ic}}{\sqrt{\sigma_{ic}^2 + \epsilon}}$$

2.3 The GAN Framework

A GAN is a type of machine learning framework originally proposed by Goodfellow et al. [2014] as a way to train generative models using unsupervised learning. A GAN consists of two models, a generative model G that tries to learn a target distribution, and a discriminative model D that tries to estimate the probability that a sample is taken from the target distribution and is not generated by G. Given data taken from the target distribution the models are trained simultaneously by having them compete in a minimax game.

Given data $\mathbf{x}$ sampled from a data space with the target distributing p_{data} the minimax game can be constructed the following way. The generator typically takes the form of a differentiable function $G(z)$ in the form of a neural network that maps seed values z taken from a latent space with distribution p_z to candidates in data space. The discriminator is a separate network that takes

candidates x from data space and outputs a scalar $D(\mathbf{x})$ representing the probability that the candidate came from the real data and not the generator. The minimax game is then defined as:

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

Which is the basis for the GAN loss function and gradient feedback during training. That is the generator is trained to minimize the discriminator's accuracy on it's candidates $log(1-D(G(z)))$, and the discriminator is trained to maximize it's accuracy in classifying real and fake candidates $\log(D(x))+log(1-D(G(z)))$.

An overview of this structure is shown in figure 2.3. The models can be trained using gradient decent methods by back propagating through both networks and only require unlabeled data.

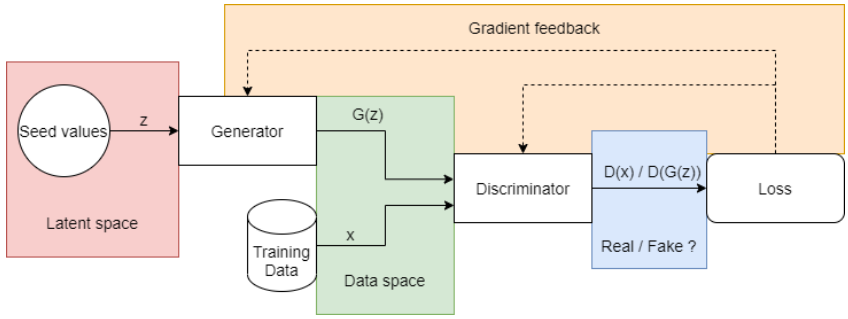

Figure 2.3: Overview of typical GAN structure.

2.4 Image-to-Image Translation

image-to-image Translation is a class of problems often encountered in image processing where images from a source domain needs to be mapped to images in a target domain while preserving some or all of the underlying information or structure. This is similar to how written information can be conveyed in different languages and one may therefore want to translate text in one language to another.

For example, translating a sentence from English to French can be viewed as a mapping of text from the English language domain to the French language domain, analogously one may want to translate an edge map to a photo that conserves the edge information.

Isola et al. [2018] proposes the "pix2pix" model to tackles the image-to-image Translation problem using a conditional GAN with a generator architecture based on U-Net. Zhu et al. [2020] build on this with the CycleGAN model for unpaired image-to-image translation. This section will explain these model types in more detail.

2.4.1 The U-Net Architecture

The U-Net architecture, which forms the basis for the pix2pix generator, is an architecture proposed by Ronneberger et al. [2015] shown in figure 2.4. It is a a convolutional neural network that is built around a encoder-decoder. The encoder incrementally reduces the spacial dimension and increases the feature dimension of the data to generate high level features, the decoder can then use the reverse process to construct the target image from the high level features. Most low level detail is lost during the encoding process and since this information is often shared between the source and target image U-Net adds skip connections between encoder layers and decoder layers with similar levels of spacial detail to give the decoder access to this information directly.

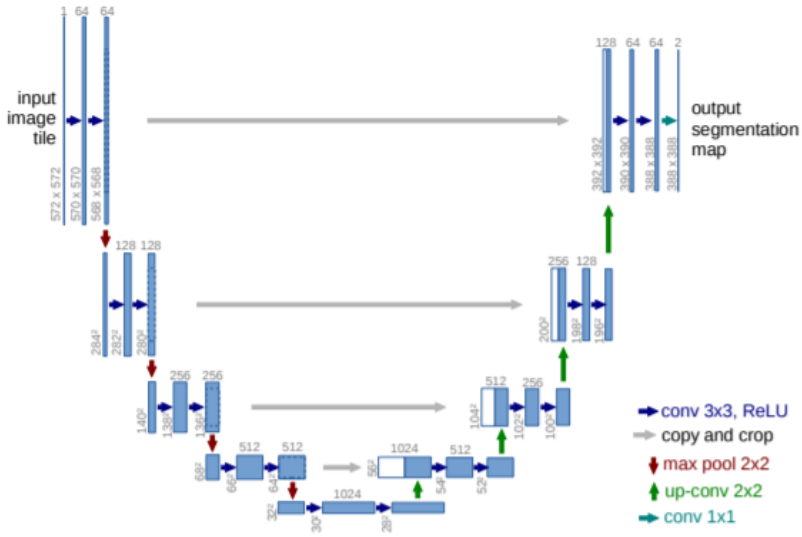

Figure 2.4: Structure of the U-Net architecture. Adapted from Ronneberger et al. [2015].

2.4.2 PatchGAN

With the pix2pix model Ronneberger et al. [2015] introduces a new type of convolutional discriminator called PatchGAN.

The PatchGAN discriminator is an alternative to the standard binary classifier for images and it works by classifying patches of the input image. PatchGAN therefore outputs a two dimensional tensor of scores each of which corresponds to an NxN patch in the input image instead of a single score for the entire image.

During training the loss function only penalizes structure at the patch scale which assumes that patches can be classified independently by PatchGAN. This independence between patches reduces the number of parameters in the model and thus speeds up training.

2.4.3 Conditional GAN

A basic GAN suffer from a limitation when it comes to multi-class data in that they don't explicitly differentiate between the different classes within data space. This means that the output from the GAN may be of arbitrary classes found in the data set, or it may converge to a solution of only generating output from a few, possibly only one, of the classes it was trained on. It may also suffer from mixing features from different classes.

A conditional GAN is a type of GAN that tries to address the problem of multi-class data by incorporating additional information to condition it to generate output of a particular class. A overview of typical Conditional GAN structure is shown in figure 2.5. Mirza and Osindero [2014] introduced the conditional version of GANs with a model where both the generator and discriminator is conditioned by adding a one-hot vector representing the desired class label to their input. In the pix2pix framework conditioning is done by adding a source image to the input. Thus the generator maps from (source image, seed) pairs to target images while the discriminator works on (source image, target image) pairs and tries to determine if the target images is from training data or generated. A limitation of this approach is that it requires the training data to be of (source image, training image) pairs.

2.4.4 The CycleGAN Framework

A major challenge for machine learning when tackling image-to-image Translation problems is the lack of paired training data, unpaired image-to-image Translation is a special case of the image-to-image Translation problem where training is done using unpaired data. CycleGAN proposed by Zhu et al. [2020] is an approach intended to address the unpaired image-to-image Translation problem. It builds on the pix2pix framework but introduces cycle consistency loss to further

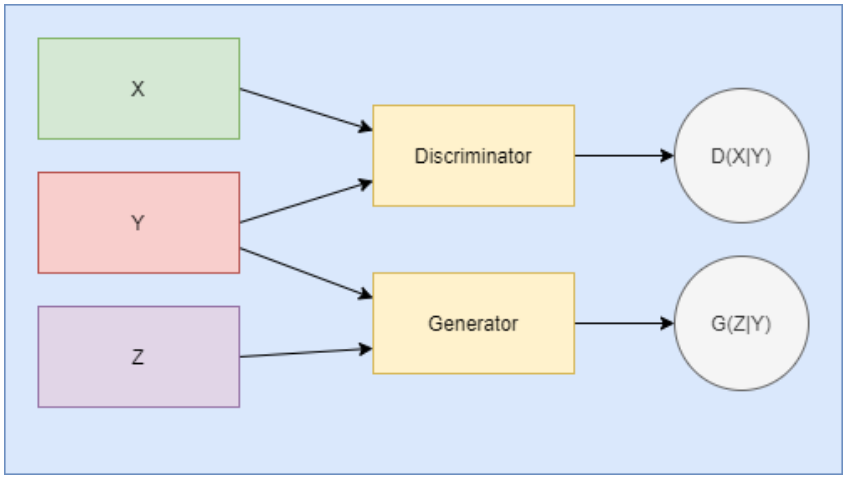

Figure 2.5: Overview of typical Conditional GAN structure. X is output data, Z is seed data and Y is conditioning data.

regularize the mapping. A horse-zebra translation achieved using this system is shown in figure 2.6.

Given two image domains X and Y the CycleGAN framework uses of two GANs to learn a mapping between the two image domains. That means we have two generators that performs the mapping $G : X \rightarrow Y$ and $F : Y \rightarrow X$ with their respective discriminators D_Y and D_X. The discriminators are only given candidate images for their respective domain, not image pairs, removing the need for paired training data. The generators play the GAN minimax game with their respective discriminator giving us two GAN loss functions $\mathcal{L}_{GAN}(G, D_Y, X, Y)$ and $\mathcal{L}_{GAN}(F, D_X, Y, X)$. However this is not enough to ensure a desired mapping between the two domains as G could map elements in X to arbitrary elements in Y and still pass D_Y and similarly for F. To enforce a mapping between equivalent elements in the two domains CycleGAN introduces an additional loss function, cycle consistency loss:

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

The cycle consistency loss is given a weight λ relative to the GAN losses which makes the total loss function for the CycleGAN:

$$\mathcal{L} = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$

Cycle consistency loss is based on the assumption that the image mapping be-

tween two domains should be bijective and the inverse of each other. This is analogous to assuming that if a sentence is translated from English to French then translating it back to English should yield the original sentence. An overview of the CycleGAN structure is shown in figure 2.7.

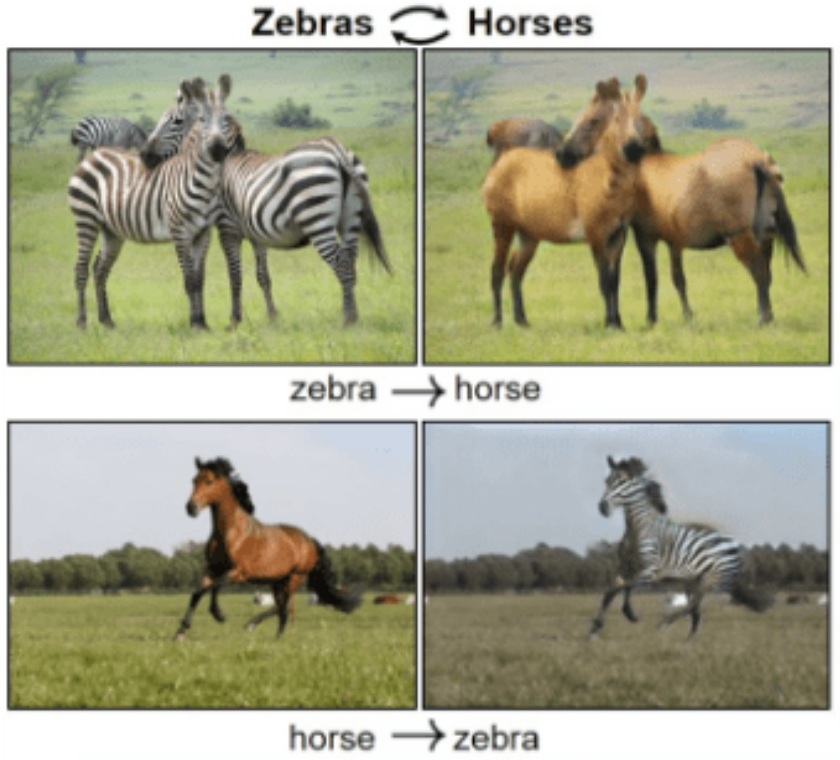

Figure 2.6: Unpaired image-to-image translation between horse domain and zebra domain using a CycleGAN. Adapted from Zhu et al. [2020].

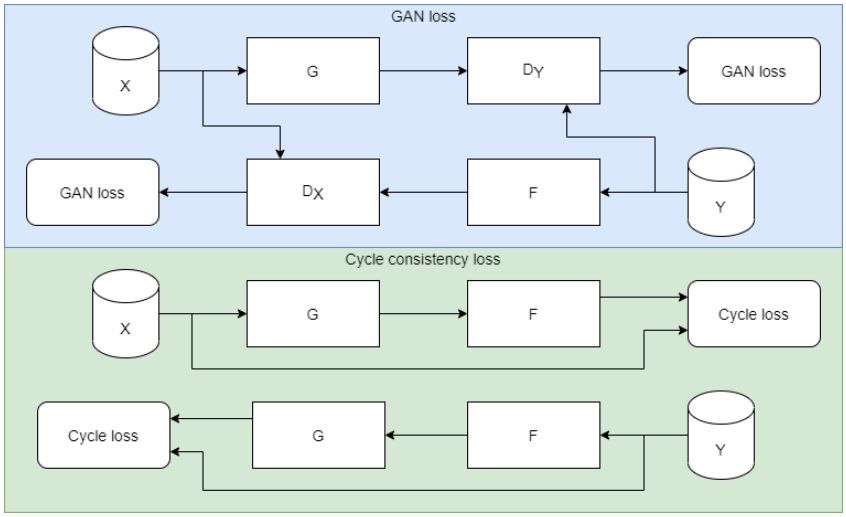

Figure 2.7: Overview of CycleGAN structure.

Chapter 3

Related Work

This chapter will present some papers I think is relevant to my research. The papers presented here are related to my research either because the system they present uses a GAN to synthesise handwritten text, which would mean the system faces similar challenges to the system I intend to develop. Or the papers deals with the same problem domain, that is handwriting forgery detection, but approaches it in a different way. I only considered papers published after 2016 due to the significant developments in the field of machine learning in recent years.

3.1 GAN for Text Synthesis

This section covers papers using variations of the GAN framework to tackle problems within handwritten text synthesis. The models presented here performs image-to-image translation, or is conditioned in some other way.

3.1.1 Generating Handwritten Chinese Characters Using CycleGAN

Creating personalized font for the Chinese language is a major challenge due to the large number of characters. Chang et al. [2018] proposes to use a CycleGAN architect to generate a personalized font using unpaired training data.

Two sets of Chinese characters in different styles makes up the two domains that the CycleGAN is trained to map between. One set is of handwritten Chinese characters and the other is from a commonly used Chinese font. The font characters do not need to have corresponding handwritten characters in the training data sine a CycleGAN is trained on unpaired data.

Even though the problem addressed by this paper is very different from what I will be addressing I consider this paper to still be very relevant to my research since it implements a system very close to what I intend to. However, one limitation is that the system presented here acts on discrete characters, meaning it does not need to process text as sequential data, while i intend to try sequences of data as well.

3.1.2 Chinese Handwriting Imitation with Hierarchical Generative Adversarial Network

Another method of generating handwritten fonts is HGAN proposed by Chang et al. [2018]. HGAN is a image-to-image translating GAN conditioned on a target image similar to pix2pix, which means it require paired training data. HGAN consists of a content encoder, hierarchical generator and hierarchical discriminator. The content encoder maps the source image to a embedding and the hierarchical generator is a decoder that maps this embedding to the target character, it also has skip connections to the encoder. Unlike a normal generator the hierarchical generator also outputs intermediate characters from different stages of decoding, which is also send to the hierarchical discriminator generating a loss. This is intended to make the generator preserve the structure of the character in the hidden layers. The hierarchical discriminator has a encoder architecture that produces a loss at multiple levels to help emphasise local discrepancies in the character.

When compared to CycleGAN, HGAN must be trained using supervised learning while CycleGAN has the benefit of being unsupervised. Both HGAN and CycleGAN can successfully learn and apply a handwritten style to a font, but HGAN preforms better at reproducing the structure of the character stokes. A comparison of result from CycleGAN and HGAN is shown in figure 3.1.

I think this paper is interesting since it demonstrates a potential limitation of the CycleGAN approach when it comes to learning to reproduce the structure of characters in the target domain. Of course, this paper deals with Chinese characters which has a much more complex structure than Latin characters, so this limitation might be less of an issue for a system working with Latin writing.

3.1.3 Adversarial Generation of Handwritten Text Images Conditioned on Sequences

Alonso et al. [2019] proposes a system for generating handwritten text using a convolutional GAN and introduces an recurrent encoder network and a recogniser network to condition the GAN on a source text instead of a source image.

The recurrent network takes the source text and encodes it as an embedding

Figure 3.1: Results from CycleGAN and HGAN. Adapted from Chang et al. [2018].

vector. The GAN then takes this vector as input in addition to the seed value and produces a handwritten text image that should correspond to the source text. To condition the GAN during training the recogniser network takes the GAN output and tries to map it back to the original source text. This is used as an additional loss function to encourage the GAN to produce handwriting the recogniser can recognise and should therefore correspond to the source text. This structure is shown in figure 3.2.

What makes this paper relevant for my research is that, unlike the other papers in this section, this paper deals with the problem of having to condition a GAN on sequential text data. This is a problem I will be facing and I think this papers approach is interesting. Though, one major difference is that this paper condition the GAN text directly while I will have to condition on images of text.

3.2 Handwriting Forgery Detection

This section covers papers proposing solutions to the handwriting forgery detection problem using techniques other than CycleGAN, mainly other deep learning approaches.

3.2.1 Handwritten Signature Forgery Detection using Convolutional Neural Networks

Gideon et al. [2018] proposes a solution to offline signature forgery detection using a CNN classifier. The problem is stated as a binary classification problem where the classifier is trained using supervised learning on a dataset of 6000 signatures prepared by the authors which are labeled as either genuine or forged. The model is then trained to classify signatures as genuine or forged.

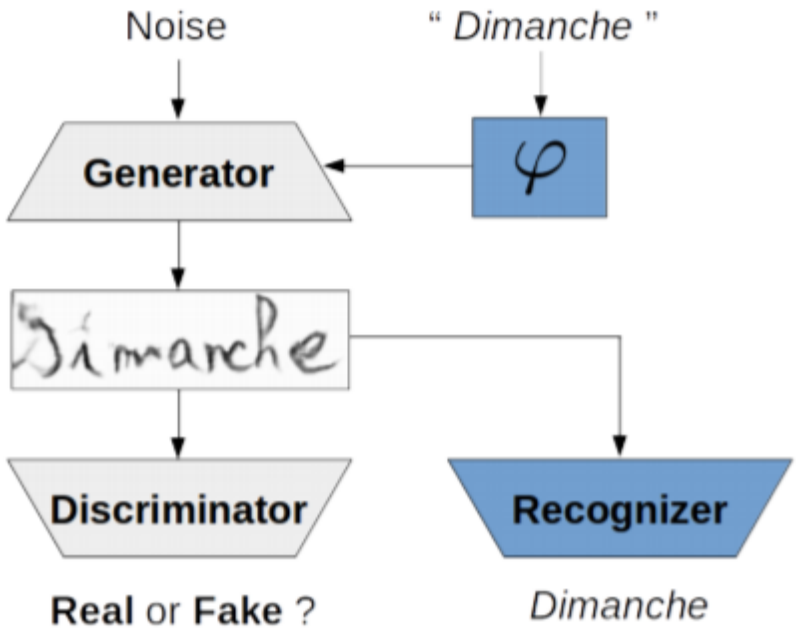

Figure 3.2: Structure of the sequence conditioned GAN. Adapted from Alonso et al. [2019].

The model achieves good results with a classification accuracy of 96% on the validation set.

The version of the handwriting forgery detection problem that is dealt with in this paper is almost the same as the problem I intend to take on in my paper. This makes the system presented here useful as a baseline to compare my system against and for establishing how well a supervised system can preform on the task.

3.2.2 Forgery Numeral Handwriting Detection based on Convolutional Neural Network

Chen and Gao [2020] proposes a solution to a variation of handwriting forgery detection that deals with altered handwritten digits. The task of this problem is to distinguish between genuine handwritten digits and digits that has been altered to look like a different digit than they where originally. While this problem is

different from detecting when someone is trying to replicate a handwriting style the underlying task is similar, that is to detect anomalies in handwritten text.

A large dataset of handwritten digits are generated by 50 volunteers for 6 different digits, forged digits are then created by altering one type of digit to look like another. A CNN model based on AlexNet is then trained using supervised learning to recognize digits that has been modified and achieves an average classification accuracy of 95.35% across all digits.

This paper presents a solution to an interesting version of the handwriting forgery detection problem. Training a model to detect if handwritten digits has been modified is similar to how my system will be trained to detect if handwriting has been translated.

3.2.3 Detection of Handwritten Document Forgery by Analyzing Writers' Handwritings

Roy and Bag [2019] proposes a method of detecting if a document has been tampered with using sliding window feature extraction and a bagging meta-classifier. The goal of the classifier is to determine if the document contain handwriting from more than one person, in which case it is deemed to be a forged document.

While the system presented here deals with a version of hand forgery detection that is quite different than what I will be dealing with I find the paper interesting because the system is designed to analyze documents that are much larger than a single signature which is what most systems works on.

3.2.4 Learning features for offline handwritten signature verification using deep convolutional neural networks

Hafemann et al. [2017] deals with the problem of offline handwritten signature verification and proposes a framework that achieves state of the art results in distinguishing genuine and forged signatures by leveraging transfer learning.

The framework has two phases, first a writer independent feature learning phase then a writer dependent classification phase. In the first phase a CNN model is trained to do writer independent feature extraction on signatures. This is done because the authors recognise that doing writer specific supervised learning directly is often impractical due to the limited data available from a single writer. So in this phase the framework tries to leverage a large dataset from many writers by learning features that are general for all writers. These general features can then help in the writer dependent phase.

In the writer dependent classification phase the trained CNN model is used to extract features from writer dependent data. These features are then used to train a writer dependent SVM classifier.

What I find interesting with this paper is that it addressed the problem of limited writer specific data. The use of transfer learning from a broader dataset to learn general handwriting features is an interesting technique to address the limitation of lacking data.

Chapter 4

Methodology

This chapter will give an overview of the systems that I built to tackle the problem of text generation and handwriting forgery detection, how the systems are trained and tested, and how they are used to generate text and detect forged handwriting.

4.1 Experimental Plan

Due to the complexity of the CycleGAN architecture the approach I intend to take is to start with a simple version of the problem and setup and gradually increase the complexity of the setup to answer the research questions.

4.1.1 Character by Character Model

The first step will be a setup where the model is mapping between text on a character by character level and using a dataset generated from handwriting looking fonts. This is to remove complexity related to processing sequential data and issues related to the consistency and quality of the input data. This step is to test if CycleGAN style transfer will work on text at all.

4.1.2 Character Sequence

The next step will be a setup where the model works on sequences of characters generated from fonts. This will reveal strengths and limitations of using CycleGAN style transfer to generate text data with a specific style and should answer much of research question 1.

4.1.3 Real Handwriting

The next step will be a setup where the model works on real handwriting. Since the style of handwriting is less consistent than fonts this will likely be more difficult for the model than the last step. This step should answer research question 3.

4.1.4 Signature Forgery Detection

The final step will be to train the model on a dataset of real signatures and test the discriminators on a set of forged signatures. this step should answer much of research question 2.

4.2 Architecture

The architecture of the system presented in this paper is build around a CycleGAN framework where text written in different styles, for example handwriting from different individuals, serve as the two domains of the CycleGAN.

The generators and discriminators of the CycleGAN are convolutional neural networks based on existing models. The pix2pix model described in section 2.4 is used as a basis for the generators and the discriminators are based on the PatchGAN model described in section **??**. These models where chosen because they have been shown to be capable of achieving good results on other Image-to-Image translation task, for example by Ronneberger et al. [2015].

Instance normalization as described in 2.2.5 is used instead of batch normalization as it has been shown to improve performance in stylization tasks Ulyanov et al. [2017].

The structure of the generators and discriminators used for each step in the the experimental process is mostly the same as the base model, but some modifications are made to accommodate the different datasets, thus slightly different variations of the model is uses. This is mainly to account for difference in the input size.

The system is implemented in TensorFlow, a open source machine learning platform developed by google and mostly uses the high level API Keras. A link to the full code can be found in the appendix 6.4.

4.2.1 Generators

The generators are image-to-image translators that will be trained to translate text in one style domain to text in the other style domain when training the CycleGAN. This constitute the handwriting forging process in the system.

The generators are implemented as convolutional encoder-decoders with skip connections. The convolutional encoder is a stack of convolutional down-sampling layers which reduces the spatial dimensions of the data and increases the feature dimension to extract the high level feature vector. The decoder is a stack of convolutional up-sampling layers which produces the output image from the high level feature vector and the low level data from the skip connections. The number of down-sampling and up sampling layers is changed from the original pix2pix architecture to suit the size of the datasets.

A diagram of the generator structure for the character by character Model is shown in figure 4.1.

When processing the character sequence data a slightly different version of the generator structure is used. This version reduces the data to a 32x512 feature vector thus keeping some of the spatial data in the feature vector. The reason for this is that the style of each character is mostly independent of the rest of the text so features extracted from one part of the image at the scale of characters is mostly only relevant to that part so we can reduce the number of parameters by reducing image down to a feature vectors for patches roughly at the scale of characters instead of a single vector for the entire image.

Since the style of characters are not entirely independent and the patches may not align with the patches a 1D convolution across the spatial dimension of the feature vector is preformed before the decoder to allow access to the neighbouring patches. an alternative to this would be the an RNN layer as discussed in section 2.2.2, but a convolutional layer was deemed more suitable since longer distance relations between patches is likely not important.

A diagram of the generator structure for the character sequence Model is shown in figure 4.2.

The generator for the handwriting dataset and the signature dataset has mostly the same structure as the one character sequence model, but the number of down-sampling and up-sampling layers is increased from 5 to 7 to account for the larger input size.

4.2.2 Discriminators

The discriminators are PatchGANs that will be trained to detect if text is genuine or has been translated from the other domain when training the CycleGAN. This will be used as the handwriting forgery detection process in the system. This means that, unlike most applications of GANs which are mostly interested in their generative ability, in this system the discriminators trained by the CycleGAN are also important when addressing the research goal.

The discriminators are implemented as stack of convolutional down-sampling layers which outputs a tensor with the classification of each patch. The patch

size chosen is 32x32 to roughly match the scale of character.

Since the PatchGAN structure works on arbitrary sized input the the same same discriminator structure is used for all datasets without modification.

A diagram of the discriminator structure is shown in figure 4.3.

4.3 Experimental Setup

The experimental setup should include all data - parameters etc, that would allow a person to repeat your experiments.

4.3.1 Training Data

A significant limitation of the system is that is trained between two specific writing styles which means it has to be trained on writer specific data. This makes the system less practical in many situations due to lack of large quantities of such data. However, a significant advantage is that the CycleGAN structure allows is trained using unpaired data which means it is not necessary to have matching samples in the two domains. In fact, the samples do not need to be labeled at all, but having labels allows for some additional options.

Interestingly this system would not be trained on data of actual forgery, instead it relies on the assumption that the generators image-to-image translation will forge handwriting in a manner similar enough to real forgery attempts that the discriminator can generalise between them. One major factor in answering research question 2 is whether this assumption holds true.

For the system I use three different sources of data with different advantages and disadvantages.

Dataset Generated From Fonts

The first dataset that is used is generated from fonts that imitate handwriting. Two different font families are used for the two domains "Dancing Script" and "Indie Flower" 6.4. samples are generated by rendering the font and adding small variations to prevent overfitting. These variations include changing offset, fontsize, font weight (for Dancing Script) and adding noise. Two types of noise is added, Gaussian noise with $\sigma = 0.1$ and salt and pepper noise with 1% of pixel set to 0 and 1% set to 1. A version of all the datasets with no noise is also generated.

For the character by character model a dataset is generated of English upper and lower case letters and digit with a uniform distribution. The symbols are rendered to 32x32 pixels and include labels. For each domain 10,000 samples are generated which is split into a training set of 9,900 and a test set of 100. Samples

of this dataset is shown in figure 4.4 and 4.5 for the Dancing Script font and Indie Flower font respectively.

For the character sequence model a dataset is generated by fist generating a random text string of 26 symbols. Period, comma, exclamation mark, have a 18%, 5% and 2% chance of being generated respectively. They are followed by a space and Period and exclamation mark are followed by an upper case letter. A digit has a 0.5% chance of being generated. Otherwise a lower case letter is generated. This text is then rendered to 32x256 pixels and include labels. For each domain 10,000 samples are generated which is split into a training set of 9,900 and a test set of 100. Samples of this dataset is shown in figure 4.6 and 4.7 for the Dancing Script font and Indie Flower font respectively.

Handwriting Dataset

The handwriting dataset used is a subset of of the IAM Handwriting Database Marti and Bunke [2002]. It consists of samples from from the "sentences" data in the IAM Database which is pictures of handwritten sentences. Sentences from two writers is used, writer with id 0 and writer with id 552, which constitutes the x and y domain respectively. These writers where chosen because they where found to have a large number of sentences in the database. For each writer 127 sentences where sampled and padded to 116x1829 pixels. These samples are then resized to 140x2048 and patches of 128x512 are randomly crop and and possibly flipped horizontally. This increases the number of samples in the dataset and makes them a more convenient size.

Signature Dataset

The signatures used consist of a subset of a handwritten signature dataset 6.4.The training set for domain x consists of genuine signatures from person 1 and the y domain consists of genuine signatures from person 3. Some of the genuine signatures by person 1 are reserved for the test set where it is combined with an equal part signatures forged by person 3 to look like person 1.

The signature images are padded to 1600x800 pixels and scaled down to 128x256 pixels. To increase the amount of training data and prevent the CycleGAN from overfitting data augmentation is used. The dataset is expanded by creating multiple versions of each sample, this is done by including samples which has been scaled to 144x288 and randomly cropping to 128x256, additionally 50% of samples are flipped and all samples have added gaussian noise.

4.3.2 Loss function

The loss function used for the discriminator loss and cycle loss is binary cross-entropy loss, the discriminator uses this across all patches.

In addition to the losses described in 2.4.4 an additional identity loss is also used for the generators. This loss is given by

$$\mathcal{L}_{id}(F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(x) - x||]$$

This loss encourage the generators to map samples to them selves then given input from their output domain. Since this condition the generators to map symbols to them selves it helps prevent a situation where the generators maps a symbol in the input domain to a different symbol in the output domain.

For the character by character model an additional conditioning system that is tested is to train classifiers on the same dataset. This is possible since the datasets generated from fonts also generate labels. The output from the generator are then passed through the classifiers and the loss is added to the generator loss thus penalizing the generator for changing the label. A diagram of the classifier model is shown in figure 4.8.

4.3.3 Training

Training is done using TensorFlow's eager execution functionality. When using eager execution training is done by fetching sample batches from the training set and running the model on them with an active gradient tape. The gradient tape records the forward pass of the model which allows TensorFlow to compute the gradient by backpropagating through the model. An optimizer is then used to apply the gradient to the model parameters.

For this system the Adam optimizer is used. Adam is an improved version stochastic gradient descent introduced by Kingma and Ba [2017].

For the character by character model and character sequence model the CycleGAN is trained with batches of 60 samples, but for the handwriting and signature model the batch size is 32 due to memory limitations.

Training is done for 20 epochs of the training dataset.

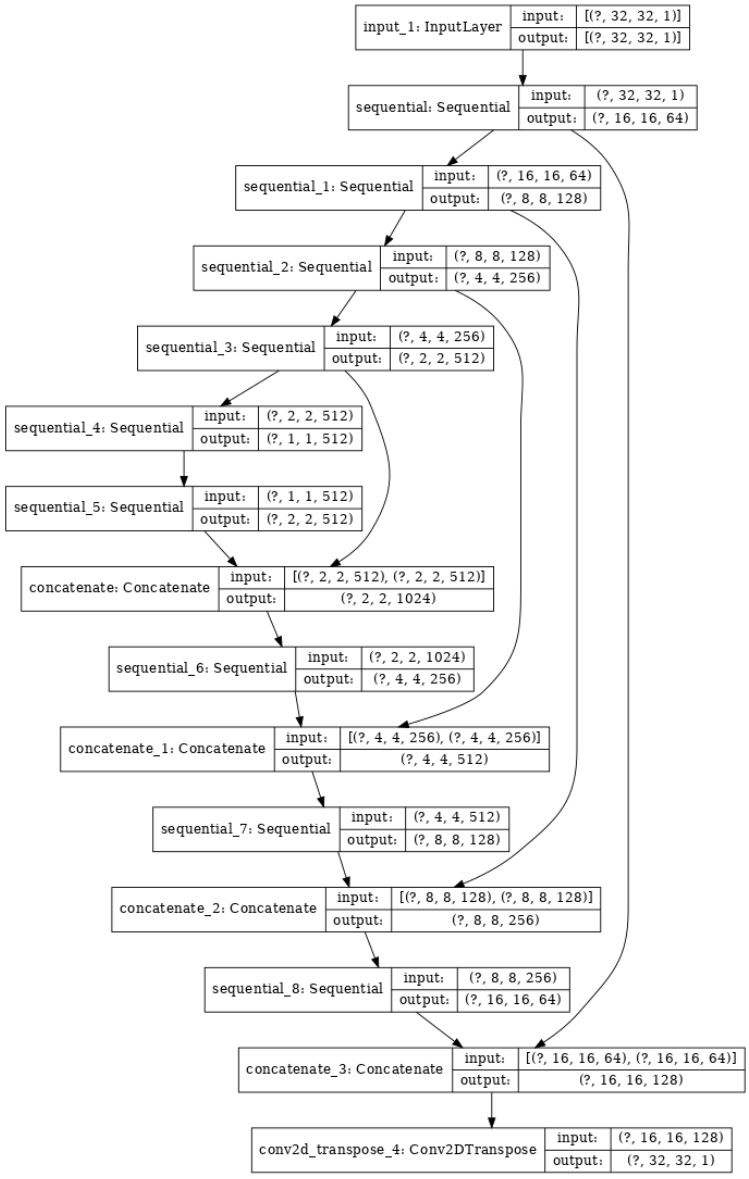

Figure 4.1: Character by Character model generator structure.

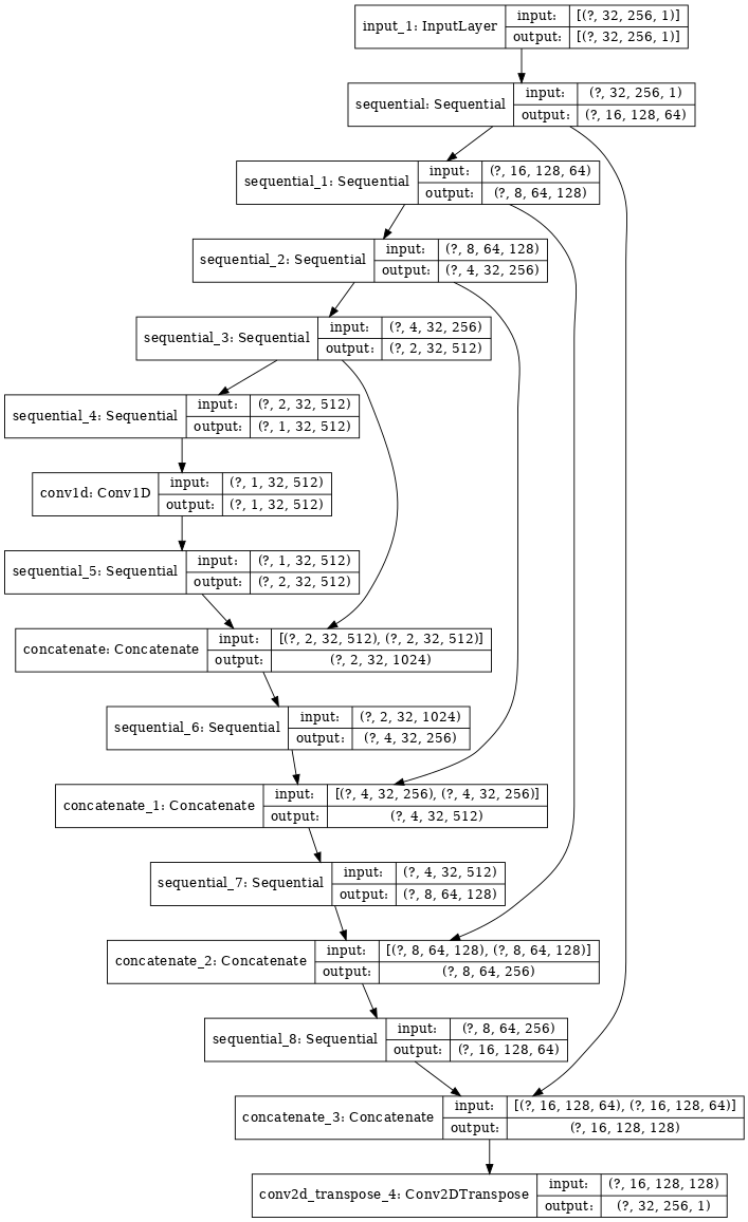

Figure 4.2: Character sequence model generator structure.

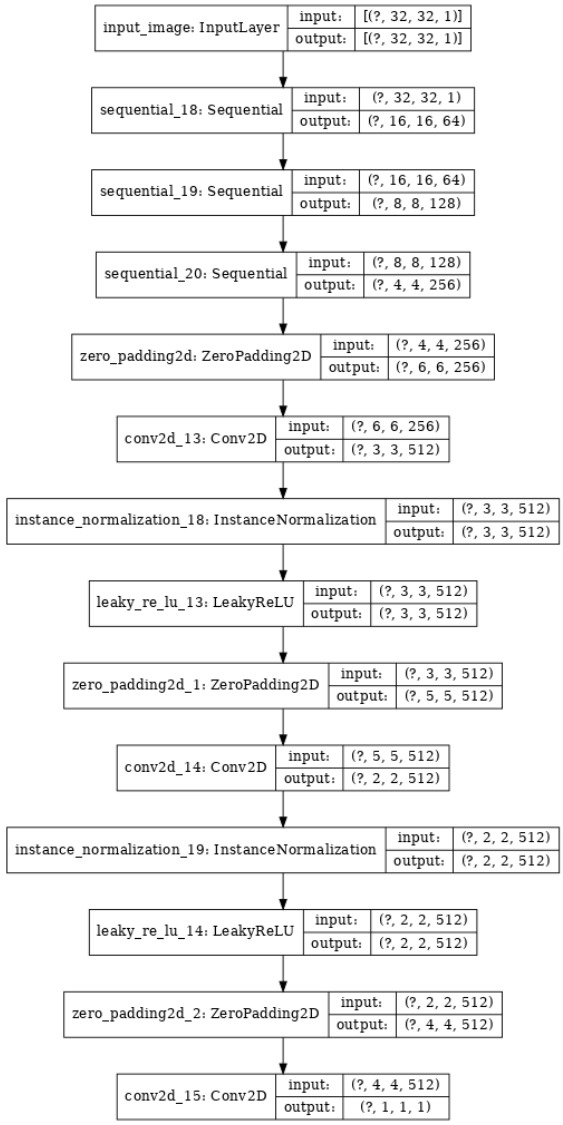

Figure 4.3: PatchGAN discriminator structure.

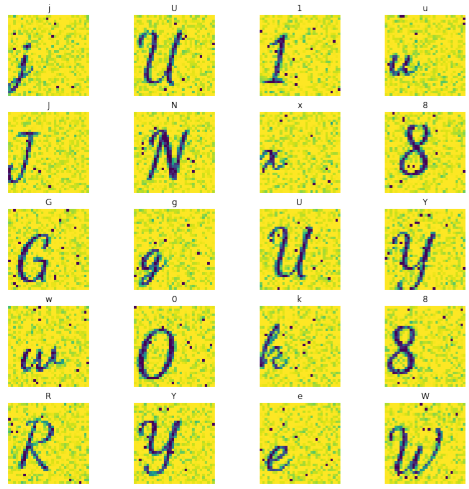

Figure 4.4: Character by Character data generated from Dancing Script font with noise.

Figure 4.5: Character by Character data generated from Indie Flower font with noise.

Figure 4.6: Character sequence data generated from Dancing Script font with noise.

Figure 4.7: Character sequence data generated from Indie Flower font with noise.

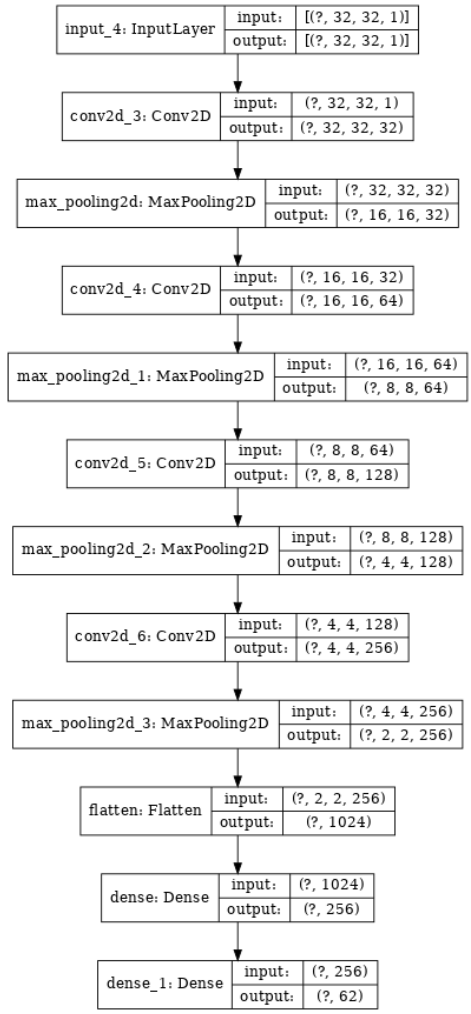

Figure 4.8: Classifier used to condition Character by Character model.

Chapter 5

Results and Analysis

This chapter presents the results from the experiments that where done in the four steps of the experimental and discussed what can be learned from them in regards to the research goal.

5.1 Experimental Results

5.1.1 Character by Character Model

The Character by Character Model seems to have converged by epoch 20 as shown by the loss in figure 5.1.

Samples from the Dancing Script and Indie Flower test set after 20 epochs is shown in figure 5.2 and 5.3 respectively.

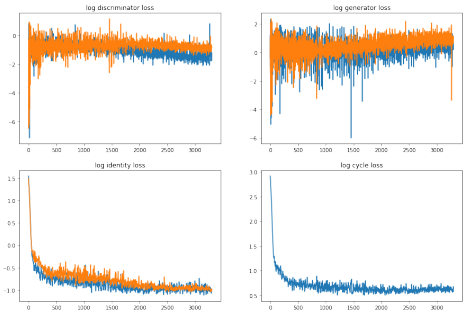

Figure 5.1: Loss for Character by Character Model.

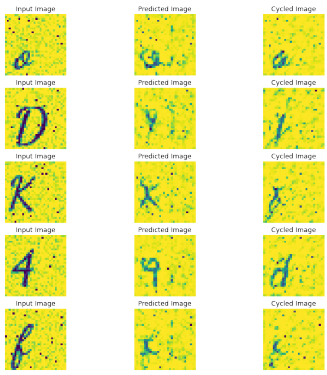

Figure 5.2: Dancing Script test after 20 epochs.

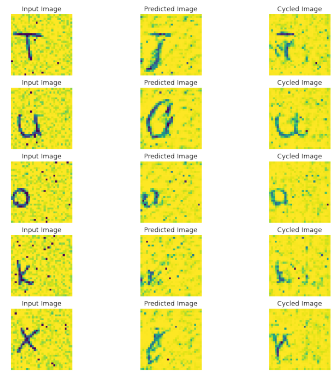

Figure 5.3: Indie Flower test after 20 epochs.

Both classifiers get accuracy above 98% but When using the classifier as part of the loss function the loss for the generators seems to converge differently as shown in figure 5.4. Test samples after 20 epochs is shown in figure 5.5 and 5.6.

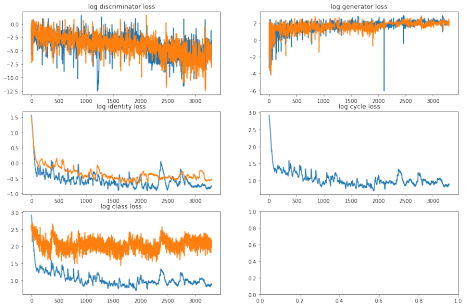

Figure 5.4: Loss for Character by Character Model with classifier.

When removing noise the model converges similarly. Test samples after 20 epochs is shown in figure 5.7 and 5.8.

5.1.2 Character Sequence

Based on the results from the character by character model the character sequence model was trained on samples without noise and without the classifier adding to the loss.

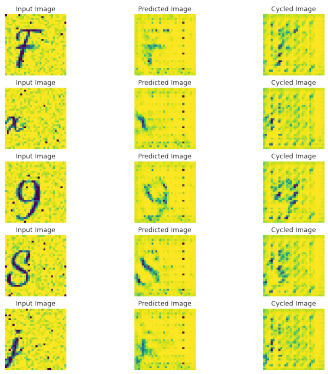

Figure 5.5: Dancing Script test after 20 epochs with classifier.

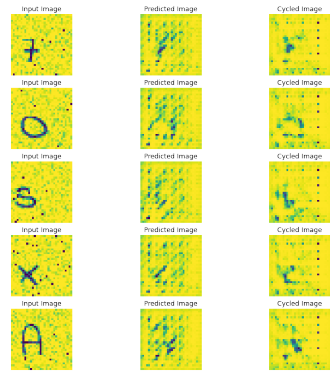

Figure 5.6: Indie Flower test after 20 epochs with classifier.

Samples from the Dancing Script and Indie Flower test set after 20 epochs is shown in figure 5.9 and 5.10 respectively.

Classifiers was trained on the Dancing Script and Indie Flower training set achieving 85.1% and 79.2% test accuracy respectively. When run on samples generated by the CycleGAN from the training set they achieved 21.0% and 19.5% accuracy respectively. This is across 66 categories of symbols in the text.

5.1.3 Real Handwriting

Samples of the x and y domain from the CycleGAN trained on the handwriting dataset for 20 epochs is shown in figure 5.11 and 5.12 respectively.

5.1.4 Signature Forgery Detection

Samples from the CycleGAN trained on the signature dataset for 20 epochs is shown in figure 5.13. On the test set the CycleGAN acheaves a 50% accuracy.

5.2 Evaluation

5.2.1 Character by Character Model

The result of the character by character models on the on the dataset with noise (figure 5.2 and 5.3) produces characters that are quite blurry and faded but usually legible.

Figure 5.7: Dancing Script test after 20 epochs without noise.

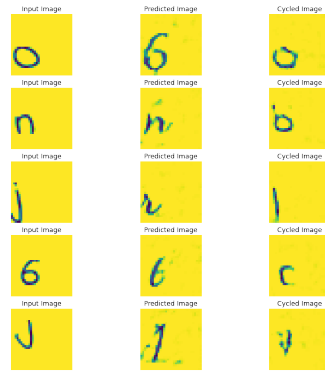

Figure 5.8: Indie Flower test after 20 epochs without noise.

The model seems to have captured the curvy style of the Dancing Script font and the simpler style of the Indie Flower font, but the blurriness indicates that the model struggles with the finer high frequency detail but get the general structure of the characters. Training for longer likely would not fix this issue since the loss shown by figure 5.1 have converged.

Interestingly, while the model will usually output a legible character it is not necessarily the same as the input character. This is shown in figure 5.2 on line 4 where "4" is mapped to "9" and "l" is mapped to "x". Even more interestingly, the model can sometimes cycle back to the original letter even though the first mapping was wrong. This is shown in figure 5.3 on line 1 where "T" is mapped to "J" and the cycled back to "T" and on line 2 where "u" is mapped to "O" and cycled back to "u".

The discriminator does not discourage this behaviour since it only evaluates if the generator output data that matches the domain which is defined by the style of the character and does not care which specific character it is. The cycle loss could discourage the wrong mapping, but if the two generators learn a wrong but inverse mapping of each other (shown in figure 5.14) then the cycle loss will not discourage it as the full cycle will output the correct character. This seems like what is happening in figure 5.3. This leaves only the identity loss which should discourage this behaviour, but the identity loss is based on the generators being feed input from what is usually their output domain so weighting this to highly cold make it hard to train the generators and may cause them to simply output the input without doing anything.

Figure 5.9: Dancing Script test after 20 epochs.

Using a classifier to regularize the CycleGAN

This problem indicates that when run on text character by character the generator need additional regularization to converge the correct mapping. This paper tested using a classifier to regularize the CycleGAN. The result from the test shown in figure 5.5 seems to suggest this regularization solved the problem of mapping to the incorrect character, but when compared to 5.2 the output is also less clear and much closer to the input failing to capture as much of the target domain style. Also the test result for the generator of the other domain (shown in figure 5.6) shows that it fails to generate legible characters at all. These limitations seems to suggest that the classifier regularization has similar trade offs as increasing the identity loss and is struggling with training as is shown by the loss in figure 5.4.

Character data without noise

A final test was done with the character by character model where the character data was generated without noise. The characters generated by this test (shown in figure 5.7 and 5.8) does not have the fading and blurriness issue of the characters generated with noise. However, this model may be overfitting since there is less variation in the characters in the dataset. The problem of mapping to the wrong character is also much worse with almost all the test samples mapping incorrectly. Again, this may be due to the model overfitting and outputting samples it has memorized from the training data.

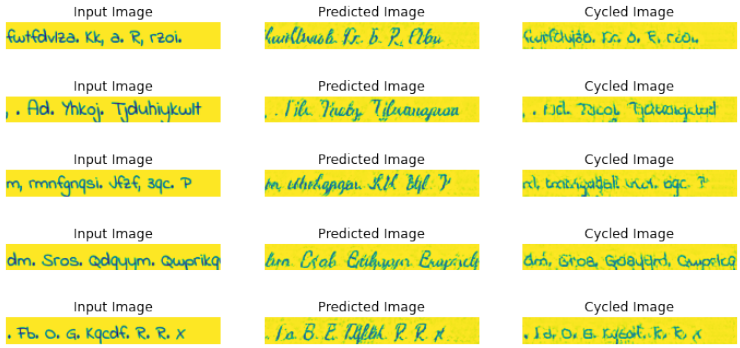

Figure 5.10: Indie Flower test after 20 epochs.

Figure 5.11: CycleGAN output for x domain after 20 epochs of training on real handwriting.

Figure 5.12: CycleGAN output for y domain after 20 epochs of training on real handwriting.

5.2.2 Character Sequence

The result of the character sequence test, shown in figure 5.9 and 5.10, shows that the character sequence model suffers a lot less from the wrong mapping problem when compared to the character by character model. Especially when considering it and has no additional regularization and is trained on text samples without noise. This is probably because the character sequences has a lot more variation between samples since they almost certainly display different text. This makes it less feasible for the model to memorize samples which results in less overfitting.

The classifiers accuracy of 21.0% and 19.5% on the generated text compared to a baseline of 85.1% and 79.2% means that the generators output is far from perfect. But if the output was not accurate or legible at all one must consider that a character can be one of 66 categories which gives an expected accuracy of $1/66 = 1.52\%$ for a random guess by the classifier. Since the accuracy achieved by the classifier is far better than this the text generated by the CycleGAN can

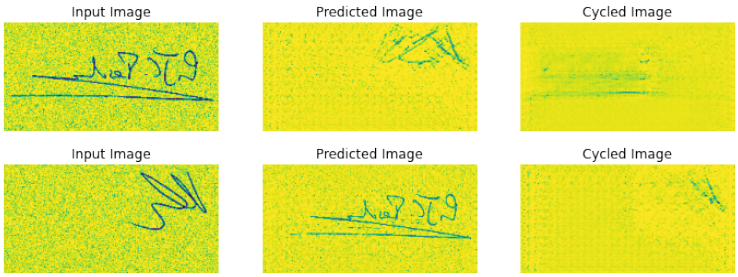

Figure 5.13: CycleGAN output after 20 epochs of training on signatures.

be considered decently accurate and legible.

The output is also not to blurry, faded or noisy compared to the character by character model and the it also seems to capture the style of the domains fairly well as it is clearly visible in the generator output which one is trying to imitate the Dancing Script and indie flower fonts.

Interestingly, this model working on text sequence seems to achieve better results and not suffer some of the problems found when working on text character by character even though is seems like the task of processing sequences should be more difficult.

5.2.3 Real Handwriting

The result of the model trained on the handwriting dataset is quite different from the character sequence model. The output is clear and does not have any issues with blurriness, fading or noise and there is no issues with incorrect mapping. In fact, the cycled text almost perfectly matches the original.

However, when comparing the CycleGAN output to the input one sees that they are almost identical except for the line width. It seems like the model has picked up on the difference in line width between the two domains and changes the input text to mach the line width of the target domain, but not much else. Thus any subtle difference in the handwriting style of the two writers is lost.

The model may also be overfitting to some degree. Since the dataset consists of random patches from sentences with multiple patches per sentence there may be some overlap in the data resulting in less variety in the data. also some of the samples are empty since the sentence where padded before patching.

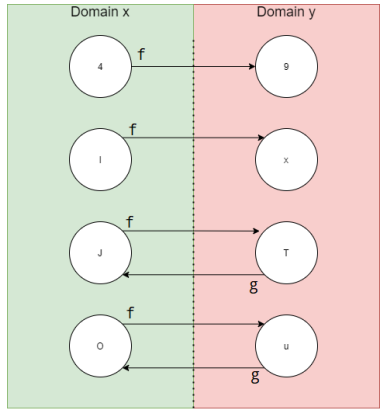

Figure 5.14: Generator g maps "T" to "J" and "u" to "O" while generator f maps "J" back to "T" and "O" back to "u" thus satisfying the CycleGAN cycle constraint.

5.2.4 Signature Forgery Detection

On the signature forgery detection task the discriminator achieve a 50% accuracy which is what would be expected for random guessing. This indicates that the discriminator has no significant ability to detect signature forgery. This could be because the ability to discriminate the generator output does not generalize to forgery by humans. It could also be that the forged signatures are to close to the genuine signatures that the discriminator is unable to detect the subtle differences. But it could also be that the CycleGAN is overfitting. The signature dataset generates a lot of samples by data augmentation, so it could also be that there is to little variety in the training data to generalize to the test set. The training data is also exclusively made up of signatures which reduces the variety further and a model trained on a broader dataset might have been able to generalize better.

Chapter 6

Discussion

6.1 Handwriting Generation

When it comes to handwriting generation the results seems to suggest that a CycleGAN generator using style transfer could be capable of generating text with a credible handwriting style. While the text generated in these tests where far from perfect the generator seems to be able to capture the aspects of the style and with some tuning the text could probably become clearer. However, a limitation of this result is the lack of objective criteria which makes it difficult to judge the quality of the generated text in a quantitative way. Another limitation of the result is data that is used. The result is based in large part on text that is not true handwriting but generated generated from handwriting like fonts, and the real handwriting dataset uses patches from writing which could overlap.

A significant limitation of the CycleGAN architecture is the tendency of mapping characters in the input domain to different characters in the output domain when processing single characters at a time. Though this problem was not encountered by Chang et al. [2018] so it may be limited to writing systems with relatively few character such as the Latin alphabet. Also, this seems to be less of an issue when processing text as a sequence of character.

The major advantage of the CycleGAN architecture is the ability to use unlabeled data for training. This is particularly advantageous when working with text since text labeling is a highly time consuming process.

However, if the main goal of the system is to generate text in a specific handwritten style then it may be easier conditioning the generator on a text vector instead of an image of text witch is what the system proposed by Alonso et al. [2019] does. Furthermore, this system can be trained on unlabeled data as well. This could mean that a CycleGAN may not be the best architecture for

generating handwriting.

6.2 Handwriting Forgery Detection

When it comes to handwriting forgery detection the discriminator of the CycleGAN tested in this paper could not achieve accuracy better than random guessing. There is ways in which this result may be improved, such as using a larger and more general dataset of handwriting for training instead of just signatures, or by using transfer learning to learn general features of handwriting from multiple writers. This paper can therefore not entirely exclude the possibility that a discriminator could be trained to detect handwriting Forgery with accuracy better than random guessing.

However, there are practical problems with using the discriminator of a CycleGAN to detect handwriting forgery that makes it seem unlikely that it is the best suited architecture for the task. The results seem to indicate that the CycleGAN is better picking up on large scale features and struggle more with small details. This is an issue for forgery detection since the forgery is made to look genuine and would thus need to be detected bases on smaller detail which would make it difficult to train an accurate enough CycleGAN. Additionally a CycleGAN is trained for two specific individuals which significantly limits its use case. This will also make finding suitable data for training difficult as it require a large amount of handwriting for the two specific individuals. And a persons handwriting style may not be consistent, for example signatures are often stylized differently. For these reasons it seems likely that other solutions will be more practical such as the one proposed by Gideon et al. [2018].

6.3 Contributions

The main contributions of this paper are:

- Demonstrating that style transfer with CycleGAN is capable of generating text stylized as handwriting.
- Determine the problem with CycleGAN of incorrectly mapping character when working on individual Latin characters.
- Establish practical advantages and limitations of the CycleGAN architecture for working with text.

6.4 Future Work

The result of this paper has two significant limitations which could be addressed in future work. One would be to test CycleGAN style transfer on tasks where it is easier to define objective and quantifiable criteria for determining the quality of the generated text.

Secondly is to address the quality and amount of handwritten training data, or try a technique like transfer learning to address the issue.

Another direction to take the research in would be to try to use CycleGAN style transfer for text that is not handwritten, for example by restyling digital documents such as pdfs or websites. This might be an area where the CycleGAN is more suitable and finding data is easier.

Bibliography

Alonso, E., Moysset, B., and Messina, R. (2019). Adversarial generation of handwritten text images conditioned on sequences. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 481–486.

Chang, B., Zhang, Q., Pan, S., and Meng, L. (2018). Generating handwritten chinese characters using cyclegan. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 199–207.

Chang, J., Gu, Y., Zhang, Y., and Wang, Y. (2018). Chinese handwriting imitation with hierarchical generative adversarial network. In *BMVC*.

Chen, Y. and Gao, S. (2020). Forgery numeral handwriting detection based on convolutional neural network. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 201–205.

Gideon, S. J., Kandulna, A., Kujur, A. A., Diana, A., and Raimond, K. (2018). Handwritten signature forgery detection using convolutional neural networks. *Procedia Computer Science*, 143:978 – 987. 8th International Conference on Advances in Computing & Communications (ICACC-2018).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Hafemann, L. G., Sabourin, R., and Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition*, 70:163 – 176.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2018). Image-to-image translation with conditional adversarial networks.

Kang, X., Song, B., and Sun, F. (2019). A deep similarity metric method based on incomplete data for traffic anomaly detection in iot. *Applied Sciences*, 9:135.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Marti, U.-V. and Bunke, H. (2002). The iam-database: An english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Roy, P. and Bag, S. (2019). Detection of handwritten document forgery by analyzing writers' handwritings. In Deka, B., Maji, P., Mitra, S., Bhattacharyya, D. K., Bora, P. K., and Pal, S. K., editors, *Pattern Recognition and Machine Intelligence*, pages 596–605, Cham. Springer International Publishing.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Instance normalization: The missing ingredient for fast stylization.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2020). Unpaired image-to-image translation using cycle-consistent adversarial networks.

Appendices

The project code can be found at:
https://github.com/haakojj/master_project

Tensorflow machine learning platform:
https://www.tensorflow.org/

Dancing Script font:
https://fonts.google.com/specimen/Dancing+Script

Indie Flower font:
https://fonts.google.com/specimen/Indie+Flower

Handwritten signatures:
https://www.kaggle.com/divyanshrai/handwritten-signatures