Improving Character Recognition Of Engraved/Embossed Steel Type Plates Using Generative Adversarial Network (GAN) Generated Image

Supervisors:
Prof. Dr. Dr. Lars Schmidt-Thieme
Lukas Brinkmeyer
External Supervisor:
Thomas Kessler
Genie Enterprise Inc

Author:
Kalaiselvan Panneerselvam
278920

2nd August 2020

Thesis submited for

Master of Science in Data Analytics

Wirtschaftsinformatik und Maschinelles Lernen
Stiftung Universität Hildesheim
Universitatsplätz 1, 31141 Hildesheim

Statement as to the sole authorship of the thesis:

Improving Character Recognition Of Engraved/Embossed Steel Type Plates Using Generative Adversarial Network (GAN) Generated Image.

I hereby certify that the master's thesis named above was solely written by me and that no assistance was used other than that cited. The passages in this thesis that were taken verbatim or with the same sense as that of other works have been identified in each individual case by the citation of the source or the origin, including the secondary sources used. This also applies for drawings. sketches, illustration as well as internet sources and other collections of electronic texts or data, etc. The submitted thesis has not been previously used for the fulfilment of a degree requirements and has not been published in English or any other language. I am aware of the fact that false declarations will be treated as fraud.

Date, City Signature

Abstract

Generative Adversarial Networks (GANs) have gained much research attention ever since their initial invention. They have been shown to produce impressive results for image generation with great success. Yet, there has been no work or only a little done in using images generated by GAN for enhancing classification tasks especially in areas of character recognition. This thesis attempt to explore, in the context of the images of nameplates/steel type plates attached to power supply equipment, whether it is possible to improve their character recognition accuracy using GAN generated image. Nowadays Optical Character Recognition (OCR) algorithms are powerful enough to read any type of text such as printed, written or even embossed information present in the images. But still, the greatest difficulty in adopting OCR is that every use case is unique and requires appropriate preprocessing methods. In this study, we have approached the problem as a translation of noisy images into clear images for improving character recognition accuracy. A carefully designed pipeline is proposed that includes an image-to-image translation GAN model to optimize the noisy images. Then these optimized images are fed into the OCR engine for performing accurate character recognition. We investigated three kinds of GAN, one that trains in a supervised manner (pix2pix), one in an unsupervised manner (CycleGAN), and lastly in a semi-supervised manner (FactorGAN). The experimental results show that pix2pix GAN performs well in eliminating the noisy factors present in the image such as rust, corrosion, difficult lighting conditions, reflections or shadows and enhance the contrast between the text and background. Most significantly, it can be seen that the character recognition of steel type plates has been increased by 37 % for the proposed OCR using GAN generated images.

Keywords: nameplates/steel type plates, GAN, image-to-image translation, pix2pix, CycleGAN, FactorGAN, OCR.

Acknowledgements

I would like to thank and express my gratitude to Prof. Dr. Dr. Lars Schmidt-Thieme, Lukas Brinkmeyer, Regina Kessler and Thomas Kessler for helping me to bring this work to fruition.

I am grateful to Regina Kessler and Genie Enterprise Inc for the warm welcome and financial support all through the development of this thesis.

To Thomas Kessler, I wish to express my gratitude for the many stimulating conversations which led to the inception of the core idea behind the thesis. He was also instrumental in giving constructive criticism and feedback, as well as technical support during this period.

I also wish to particularly thank Lukas Brinkmeyer for his constant supportive comments, valuable insights and knowledge which helped in shaping the thesis.

Lastly, I am thankful to my friends and family, especially my parents, for their perennial support and encouragement.

A special thanks to Adaobi Onyeakagbu for providing her feedback on the report.

Contents

List of Figures

List of Tables

List of Abbreviations

ADAM Adaptive Moment Estimation

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

CNN Convolution Neural Network

CRAFT Character Region Awareness For Text detection

CTC Connectionist Temporal Classification

CTPN Connectionist Text Proposal Network

DL Deep Learning

EAST Efficient and Accurate Scene Text Detector

FID Frechet Inception Distance

GAN Generative Adversarial Network

GPU Graphics Processing Units

HP Horse Power

KNN K - Nearest Neighbour

LSTM Long Short Term Memory

MAE Mean Absolute Error

ML Machine Learning

MSC Metal Stamping Character

MSE Mean Square Error

OCR Optical Character Recognition

PSENet Progressive Scale Expansion Network

PSNR Peak Signal to Noise Ratio

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROI Region Of Interest

RPM Revolutions Per Minute

SE Squeeze and Excitation Network

SSD Single Shot Detector

SSIM Structural Similarity Index Measure

STN Spatial Transformation Network

TPS Thin Plate Spline transformation

Chapter 1

Introduction

This master thesis was done at the University of Hildesheim in collaboration with Genie Enterprise Inc, Ludwigshafen. It investigates the use of image-to-image translation GAN for improving the character recognition of steel type plates/nameplates. This chapter introduces the problem statement with motivation, research questions, contributions and outline.

1.1 Background

Computer vision is the field of Artificial Intelligence (AI) that enables computers to see and process visual data. In simple words, it is a process of extracting information from images or videos. Optical Character Recognition (OCR) is a set of computer vision tasks that convert scanned documents and images into machine-readable text. Modern OCR algorithms can convert any type of text present in the image into digital information. The common applications of OCR are automatic license plate recognition, traffic sign recognition and extracting information from documents like cheque, invoice, passport, bank statement and business cards. The accuracy and success of OCR depend on the specific use cases. Every use case has its own challenges and requires different kinds of pre and post-processing of images. Our use case is to read text from the images of nameplates/steel type plates attached to power supply equipment.

Figure 1.1: A sample steel type plate/nameplate image

Most electrical appliances such as motor come equipped with a nameplate/ steel type plate. The steel type plates that are attached to the motor contain a wealth of important information such as identification numbers, manufacturing year, Horse Power (HP), Revolutions Per Minute (RPM), etc., This information describes their electrical properties needed to maintain and repair these devices. Figure 1.1 shows a sample steel type plate image in the dataset. These steel type plates differ in fonts, size, template and material used. The current practice is that the maintenance operator transcribes this information from the images of steel type plates for maintenance and repair. This manual approach is time-consuming and susceptible to errors but can be solved in real-time by the inference of a deep learning model. Hence we propose an automatic steel type plate recognition system.‘

Moreover, the steel type plate images are of poor contrast and include texts in corroded areas and difficult lighting conditions, such as reflections or shadows. Figure 1.2 shows the images of corroded steel type plates in different lighting conditions. These occlusions become problematic for automatic text extraction of the steel type plate images. Several related approaches exist to extract text from business cards, license plates and documents. But they usually consist of low noise or noise-free and do not need much intensive preprocessing of images for the extraction of texts. These assumptions cannot be guaranteed to be true in our case of weathered and rusted steel

Figure 1.2: Weathered steel type plates under different lighting conditions

type plates since the stained areas contain a significant amount of text information. Hence most of the available OCR engines would fail in reading the steel type plates.

Generative Adversarial Network (GAN) is a generative model developed by Ian Goodfellow in 2014, which consists of two neural networks called generator and discriminator, competing against one another to improve their predictions [Goodfellow et al., 2014]. The image-to-image translation GANs are conditional adversarial networks that involve controlled modification of the image from a source domain to the target domain. Thus the automatic steel type plate recognition system consists of an image-to-image translation GAN for translating the noisy steel type plates into clear images and an OCR engine to extract text from the clear image.

1.2 Aim

The goal of the thesis is to find out if image-to-image translation GAN can be used as an image pre-processing method for steel type plate/nameplate images so that it improves the character recognition of OCR engines. Since the invention of conditional GAN, several works on image-to-image translation have been proposed. We investigate three kinds of image-to-image translation GAN models that were chosen based on their different learning approaches in order to select the best model for optimizing the steel type plate images: i) Isola et al. [2017] proposed pix2pix GAN that works by learning the map-

ping between the source domain and the target domain using paired dataset. ii) Zhu et al. [2017] proposed CycleGAN is an unsupervised way of training image-to-image translation models. It does not require a paired dataset for training instead uses an unordered collection of images from two different domains. iii) Stoller et al. [2019] proposed FactorGAN trains image-to-image translation models when data points are incomplete in semi-supervised fashion. Thus, the proposed framework uses the best generator model out of the above three GANs to translate the noisy image to a clean image and then extract text from these optimized images through an OCR engine.

1.3 Research questions

In order to achieve the aim, this thesis will answer the following questions:

1. Can an image-to-image translation GAN be used to improve the accuracy of character recognition for the images of weathered steel type plates?

2. Can the findings of Zhu et al. and Stoller et al. be used to make up for the few image pairs in the steel type plate dataset?

3. How to evaluate the three different image-to-image translation GAN models under the scope of the thesis and choose the optimal model for the proposed steel type plate recognition system?

4. How good is the proposed GAN engine compared to a commercial OCR tool like Google Vision? How can the performance of the overall framework be measured?

1.4 Contributions

This thesis examines the in-depth analysis of using generative models to improve the text detection and character recognition in the context of images of steel type plates/nameplates attached to the electrical equipment.

A dataset of steel type plate/nameplate images was created with approximately 500 images crawled from google image search using python scripts. Then labels were manually created for those images which are explained briefly in Chapter 4. A unique approach is used to create synthetic data close to real image distribution to mitigate the problem of fewer image pairs in the dataset. The synthetic dataset consists of images with its corresponding output pair. Then the use of three kinds of image-to-image translation GAN was investigated to improve the robustness of the OCR engine against the images with adverse lighting and occlusions. The hyperparameters were tuned to get an optimal combination and the experiments were conducted to select the best GAN generator model for integration with the OCR engine.

Different evaluation metrics for GAN like Structural Similarity Index Measure (SSIM), Frechet Inception Distance (FID) score and Mean Square Error (MSE) were explored. The GAN model is evaluated using both qualitative and quantitative metrics. The proposed OCR engine was built by integrating a CRAFT based text detection model and a text recognition model. In order to evaluate the accuracy of the proposed OCR engine, it is compared with one of the benchmark OCR engines like Google Vision OCR. Finally, both the OCR engines are evaluated with original images as well as GAN translated images. The results were tabulated and analyzed using bar graphs to get a better understanding which is briefly discussed in Chapter 5.

1.5 Thesis outline

The rest of the report is organized as follows: Chapter 2 discusses the literature review and background that is relevant to understand the thesis. Deep learning sets the foundation throughout the thesis and thus a general introduction to neural networks, generative models and OCR models is provided in this chapter. Chapter 3 introduces the proposed framework, where we discuss the overall architecture and training procedure in detail for each of the components; Chapter 4 focuses on the dataset collection and description with

insights. Besides, this chapter discusses briefly the creation of labels and how the synthetic dataset was created. Chapter 5 presents the results obtained from the experiments both quantitatively and qualitatively along with the experimental setup. A brief discussion on the result is provided with several insights and visual samples of GAN generated images. Finally, Chapter 6 concludes the thesis research with remarks and provides possibilities for future research directions.

Chapter 2

Theory and related work

This chapter describes the theoretical concepts that are required to understand this thesis and provides an overview of several related areas of research in the context of improving character recognition of steel type plates. It also provides an outline of current research done in the use of GANs for character recognition.

2.1 Introduction to deep learning

Deep Learning (DL) is a field of AI made up of neural networks inspired by the structure and function of the human brain. AI is growing at a rapid pace and deep learning is one of the major contributors to it. Deep learning is continuously changing the world around us. Its application ranges from speech recognition to driverless cars.

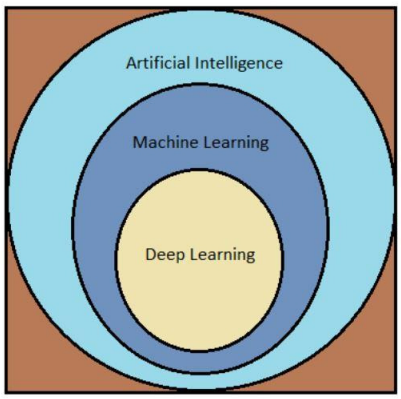


Figure 2.1: DL vs ML vs AI. Source: Adapted from [Roy, 2020]

Figure 2.1 shows the relationship between DL, ML and AI. **Artificial Intelligence** is the branch of com-

puter science that makes the computer mimic human intelligence such as visual perception, voice recognition, translation between languages and decision making.

Machine Learning is the subset of AI that enables the computer to learn to do a task on its own without explicitly telling it how to accomplish the task.

Deep Learning is the subfield of Machine Learning (ML), which deals with algorithms to train models like object detection and speech recognition using a multi-layered neural network with a vast amount of data. According to [LeCun et al., 2015], "Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction".

2.1.1 Neural networks

Figure 2.2 shows the structure of a simple neural network. An Artificial Neural Network (ANN) usually consists of an input layer, n number of hidden layers and an output layer. The input layer receives the input and sends it to the hidden layer. The hidden layer is responsible for processing the input and extracting the features, which is then passed onto the output layer. Each layer contains several neurons, and the neurons in each layer are connected to all the neurons in the next layer.

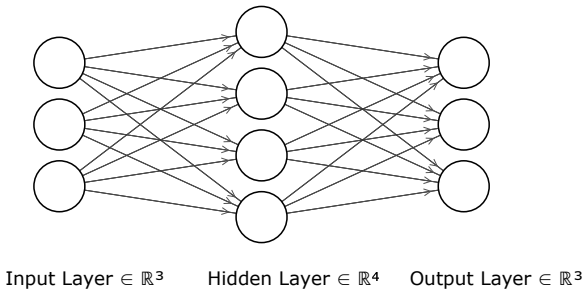


Figure 2.2: A simple neural network. Source: Created using tool [alexlenail, 2018]

Neuron

Neurons, commonly known as nodes, are the basic building blocks of neural networks. A neuron receives input along with weights & bias, and after processing, outputs the result as shown in Figure 2.3. The output of a neuron is either used as the input to neuron in the next layer, or as the final output.

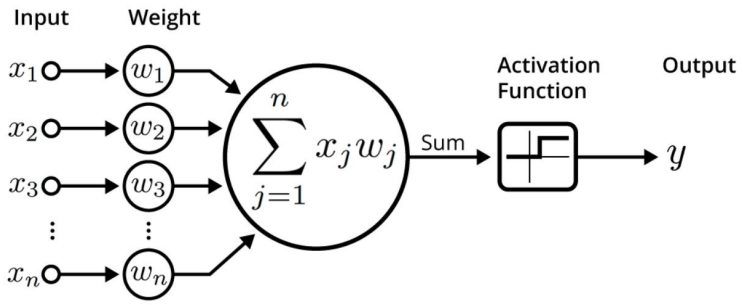


Figure 2.3: Components of a neuron. Source: [Saxena, 2017]

The output of a neuron can be mathematically represented as

$$Y = \sum_{j=1}^{n} w_j * x_j + b$$

where w is the weight and b is the bias

Weights

When the input data is passed into the neuron, a random weight is assigned to each information, and then the input information is multiplied by the corresponding weight. During training, the weight of each information is updated according to the corresponding feedback.

Bias

Bias is used as a constant to adjust the output along with the weighted sum of input in order to best fit the given data. It is added along with the result of the product of inputs and weights in the network.

Activation function

The activation function converts an input signal of the neuron into any desired output signal. It is introduced to bring non-linearity into the network. The most common activation functions are sigmoid, Rectified Linear Unit (ReLU), tanh and softmax which are shown in Figure 2.4.

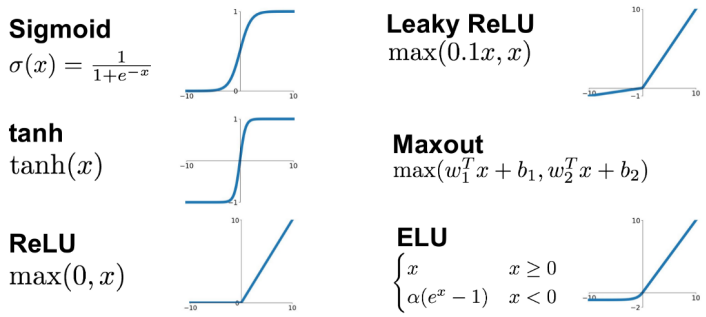


Figure 2.4: Different activation functions. Source: [Jadon, 2018]

The final output of a neuron is given by

$$Y = \phi(\sum_{j=1}^{n} w_j * x_j + b)$$

where ϕ can be any of these activation functions.

2.1.2 Loss function and optimizer

When training a neural network, the goal is to make the prediction as close as possible to the ground truth. The accuracy of the network is measured using the loss function. One of the most commonly used loss function is Mean Square Error (MSE). During training, the network makes use of loss value to adjust the weights so that the error is reduced at each iteration.

$$MSE = \sum_{n}(y - \widehat{y})^2$$

where y is the ground truth and $\widehat{y}$ is the predictions.

Optimizers are used to minimize the error calculated from the loss function. The most common optimization algorithm is gradient descent. Kingma and Ba [2014] proposed Adaptive Moment Estimation (ADAM) optimizer which is the current benchmark optimizer for training deep learning models. It combines the Adagrad and RMSProp algorithm for better computational efficiency and memory requirements.

2.1.3 Backpropagation

The weights and biases in a neural network are learnable parameters. At first, when the neural network is built, they are randomly initialized. At each iteration, the network error is calculated by the forward propagation, and then to minimize the error, the gradients of the loss function are calculated and fed back to the network to update the weights & biases in the network. This is called backpropagation.

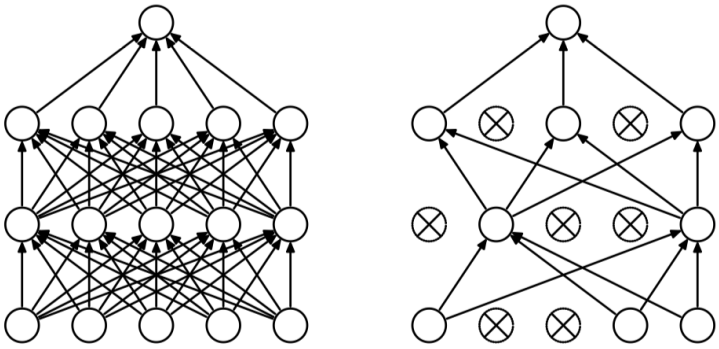
2.1.4 Hyperparameters

Hyperparameters are the parameters that are not learnable during training a neural network. Some of the most common hyperparameters in deep learning are

(i) **Learning rate:** It is denoted by α that determines how fast the algorithm converges. If the learning rate is too high, it will overshoot the minima and if it is very low, the convergence will need more steps. Ideally, the learning rate is set between 0.0001 to 0.01.

(ii) **Regularization:** Regularization is the technique to avoid the overfitting of the model. The regularization parameter is denoted by λ. Dropout is used as a regularizer in neural networks. It refers to the training process of discarding a certain number of neurons present in the hidden layers of the network.

(iii) **Batch size:** While training the neural network, instead of sending the entire input at once, the input is divided into several blocks of equal



(a) Standard Neural Net (b) After applying dropout.

Figure 2.5: Effect of dropout. Source: [Srivastava et al., 2014]

size called batches. In simple words, batch size determines the number of training examples present in a batch.

(iv) **Epoch:** An epoch is when the entire training data is propagated forward and backward through the neural network once.

2.2 Convolution Neural Networks

Convolution Neural Network (CNN) is a type of deep neural network that is most commonly used for image recognition and classification. CNNs have been proven successful in various computer vision tasks like object detection, traffic sign detection, face recognition, scene text recognition, etc. The hidden layers [Goodfellow et al., 2016] in the CNN can be divided into convolutional layers, pooling layers, ReLU layers and fully connected layers as shown in Figure 2.6.

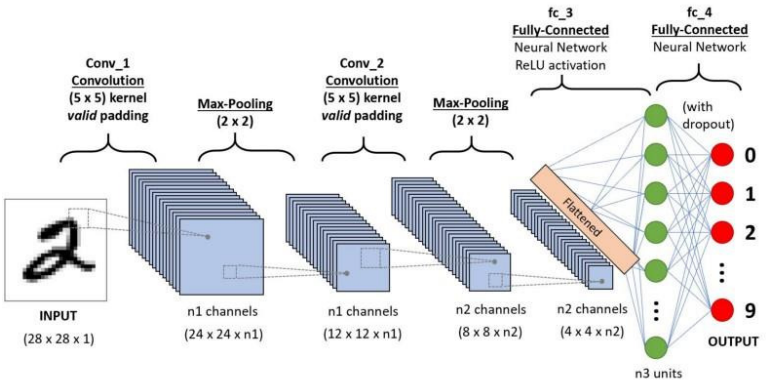


Figure 2.6: A CNN for MNIST handwritten recognition. Source: [Saha, 2018]

Convolution

The operation of convolution layer starts with a kernel filter, which is just a small weight matrix. The kernel slides on the input data, multiplies the elements of the current input, and then summarizes the results into a single output pixel. The kernel repeats this process for each position in the image it slides over, transforming the input matrix into another feature matrix. The kernel filter values are randomly initialized and learnt during training.

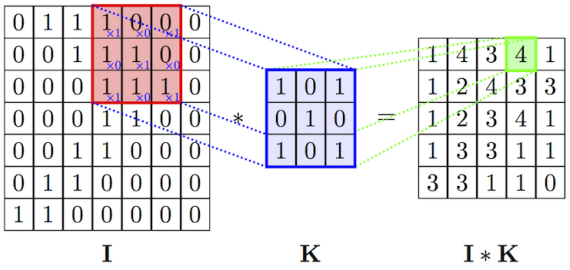


Figure 2.7: Convolution operation. Source: [Vo, 2018]

Generally, an image has a shape of C x H x W where C is the number of channels of an image(for RGB, C=3), H and W are the height and width of the image. The CNN layer consists of a kernel or a filter K with x rows, y

columns and depth d. The result of the convolution [Moutarde, 2018] of an image I with a kernel K with d channels is given by

$$Conv(I,K)_{\text{xy}} = \sum_{i=1}^{h}\sum_{j=1}^{w}\sum_{k=1}^{d} k_{ijk}.I_{x+i-1,y+j-1,k}$$

Pooling

After the filter passes through the image, a feature map is generated for each filter. These functions are then obtained through activation functions, which determine whether a feature exists at a given location in the image. Then we can do a lot of things, such as adding more filter layers and creating more feature maps. As we create deeper CNNs, these maps become more and more abstract. We can also use pooling layers to select the maximum values or average values on the feature map and use them as input for subsequent layers.

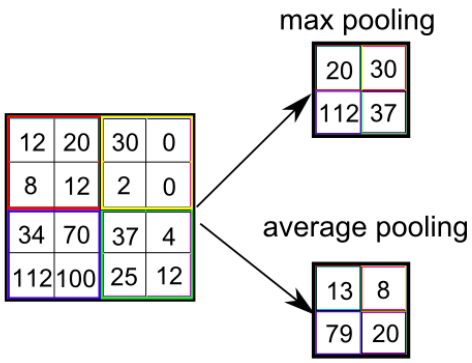


Figure 2.8: Pooling operation. Source: [Saha, 2018]

Fully connected layer

This layer is similar to the conventional neural network, and its role is to fully connect the features obtained through multiple convolutional layers and multiple pooling layers and calculate the final predicted value. As the name

implies, each neuron in this layer will be connected to all the nodes in the previous layer.

Normalization layer

The normalization layer is used to normalize the activations in a neural network to give them a mean of 0 and a variance of 1. This thesis utilizes three normalization techniques i) Batch normalization, ii) Instance normalization and iii) Spectral normalization.

1. **Batch normalization:** Generally the data should be normalized to make its distribution consistent. But during the training process of the deep neural networks, the data is fed into the network as batches and each batch has a different distribution. Hence Ioffe and Szegedy [2015] proposed batch normalization to make the distribution consistent and to reduce internal covariate shift.

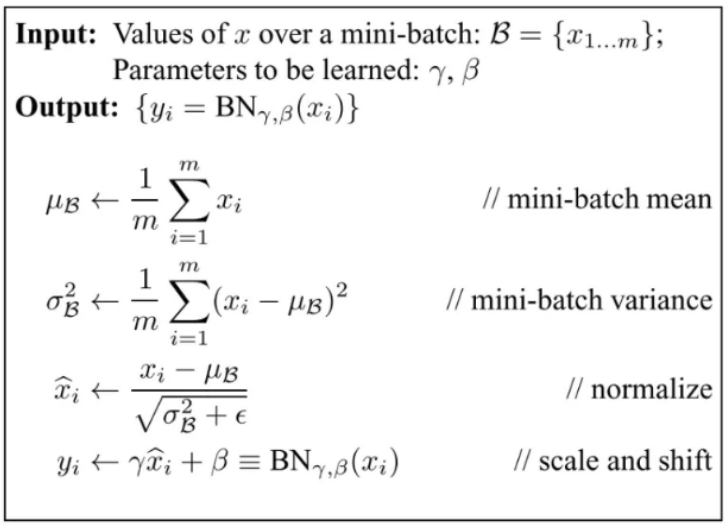


Figure 2.9: Batch Normalization algorithm. Source: [Ioffe and Szegedy, 2015]

2. **Instance normalization:** The batch normalization doesn't work well in distributed training in the case of multiple Graphics Processing Units (GPU). Also, it focuses on normalizing each batch to ensure consistent

data distribution. However, in image stylization, the generation result mainly depends on an image instance, so normalizing the entire batch is not suitable for image stylization. Hence, Ulyanov et al. [2016] proposed instance normalization method normalizes only the height and width of the image. Thus it can accelerate model convergence and maintain independence between each image instance.

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

where x $\in \mathbb{R}^{\mathrm{T}\times\mathrm{C}\times\mathrm{W}\times\mathrm{H}}$ is the input tensor with a batch of T images and $\mathrm{x}_{\mathrm{tijk}}$ is its tijk-th element, where j and k are the dimensions, i is the feature/color channel and t is the image index in the batch [Ulyanov et al., 2016].

3. **Spectral normalization:** To improve the training stability of the discriminators and mitigate the problem of gradient explosion, Miyato et al. [2018] proposed a method called spectral normalization. It normalizes the weight for each layer of the network such that its Lipschitz constant always stays equals to one.

2.3 Generative Adversarial Networks

Goodfellow et al. [2014] proposed GANs which is a type of deep neural network that generates data close to the input data distribution. The GAN network consists of two networks, Generator G and Discriminator D. These two neural networks compete against each other to learn the probability distribution of the dataset. The generator network learns to produce realistic samples and the discriminator learns to distinguish real data and samples produced from the generator.

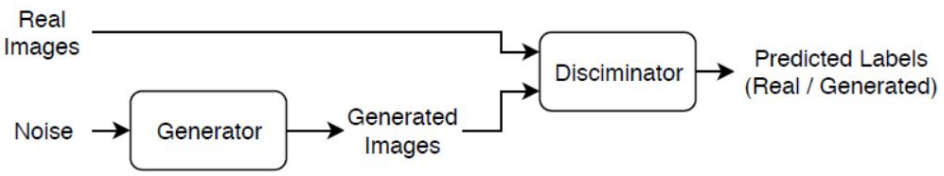


Figure 2.10: Generative Adversarial Network (GAN). Source: [Mathworks, 2020]

In Figure 2.11, the black dotted line is the Gaussian distribution of real data, the green line is the fake distribution learned by the network, the blue line is the probability of judging the network as a real picture, the horizontal line marked x represents the sample following the Gaussian distribution x space, and the horizontal line marked z represents the sampling space subject to uniform distribution z. It can be seen that the mapping relationship from the space of z to the space of x is learned.

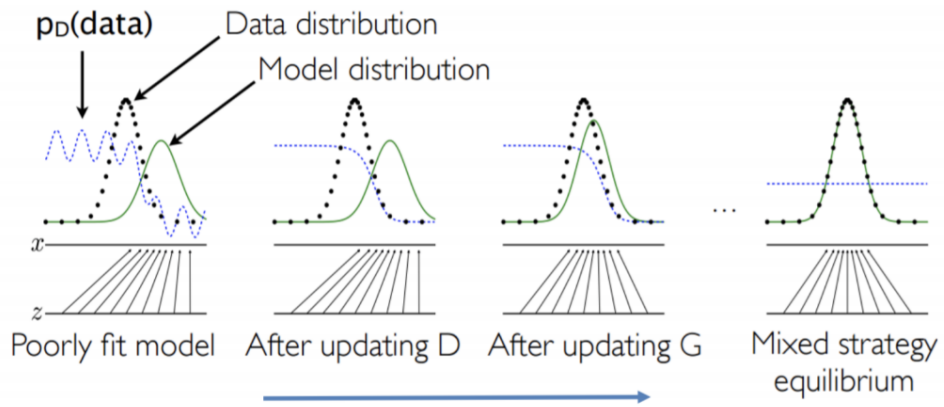


Figure 2.11: Illustration of GAN learning to map uniform noise to the Gaussian distribution. source: [Vallecorsa, 2017]

The training process involves two objectives

- The discriminator tries to maximize the probability of identifying real data from the fake data generated by the generator.
- The generator generates realistic samples to fool the discriminator i.e. the generator tries to minimize the discriminator's probability of clas-

sifying real data from the fake data. Thus the generator and discriminator play a minimax game.

Both networks compete to get better against each other. The objective function V(G,D) of minimax game [Goodfellow et al., 2014] between Discriminator D and Generator G is given by

$$\underset{G}{min}\underset{D}{max}V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z}(z)[log(1 - D(G(z))]$$

where, D(x) is the discriminator's probability estimate that the real instance of the image is real, $\mathbb{E}_x$ is the expected value over instances of real data x, G(z) is the generator's output for the noise z, D(G(z)) is the discriminator's probability estimate that a fake instance of the image is real and $\mathbb{E}_z$ is the expected value over all random noise inputs to the generator.

2.4 Conditional GANs

Conditional GAN is an extension of the original GAN. Both the generator and the discriminator add additional information y as a condition, y can be any information, such as class labels, image, or music data. The conditional GANs are used in various applications such as image-to-image translation, style transfer, text to image synthesis, etc.

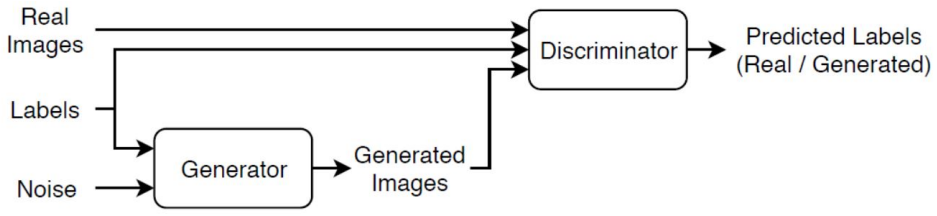


Figure 2.12: Conditional GAN. Source: [Mathworks, 2020]

Figures 2.10 and 2.12 show the difference between the standard GAN and conditional GAN ie. conditional GAN is realized by sending additional information y to the discriminator and the generator as part of the input

layer. Similarly, the objective function of conditional GAN is a two-player minimax game with conditional probability [Isola et al., 2017] which is given by

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim P_z}(z)[log(1-D(G(z|y))]$$

where, $D(x|y)$ is the discriminator's probability estimate that real data instance x is real given the target label y and $G(z|y)$ is the generator's output when given noise z and the target label y.

2.5 OCR

OCR is the computer vision task of localizing and detecting the text in an image. The recent developments in the deep learning approaches revived interests in the OCR problem. Different neural network architectures have been investigated to address the OCR challenge. Before the deep learning era, traditional approaches adopted connected component analysis or sliding window approaches as a solution for OCR.

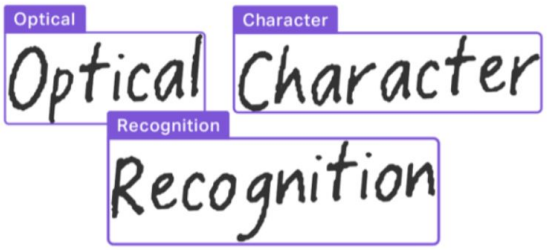


Figure 2.13: Optical Character Recognition. Source: [AICompare, 2020]

The existing methods can be classified into three types:

1. **Text detection:** It detects & localize all the texts in the image. Bounding box coordinates is the output of the text detection.

2. **Text recognition:** It uses the output of text detection and converts the text into a digital format.

3. **End to End approach:** They perform both detection and recognition as a single approach.

2.6 Related works

This section provides an overview of the literature review in several related areas of research. Some of the existing methods for OCR are reviewed under different perspectives which are as follows:

2.6.1 General OCR

Initially, researchers tried object detection models like faster R-CNN for text detection [Nagaoka et al., 2017; Zhong et al., 2019]. But these algorithms are not very accurate and does not perform well for scene text detection.

Tian et al. [2016] proposed Connectionist Text Proposal Network (CTPN) models utilize the idea of anchoring and Recurrent Neural Network (RNN) for sequence labelling. The input is usually passed into the pre-trained models like VGG trained with the ImageNet dataset. The output is then passed to the Bidirectional LSTM. A fully connected layer is used as an output layer which gives the bounding box coordinates for the anchor box, text probability scores and side refinements. The detection is not good for natural scene images, such as curved text and multi-directional text (vertical text).

SegLink is proposed especially for long and multi-oriented text. It is similar to the CTPN idea, at first, all parts of the text are found, and then they are connected to form a complete text line [Shi et al., 2017]. An improved version of the SSD-based network structure is proposed to simultaneously predict two basic elements of text line detection are: segments and links. The segment is actually a bounding box output similar to the Single Shot Detector (SSD) but added with a direction which is expressed as $(x_b, y_b, w_b, h_b, \theta)$ where x_b, y_b represents the center of the segment, w_b, h_b indicates the width and height of the segment and θ represents the rotation angle of the segment.

Links are the connection of segments, i.e, the probability value of whether two boxes are of the same text. One drawback of SegLink is that it cannot detect text lines with large intervals, cannot detect curved text.

Zhou et al. [2017] proposed Efficient and Accurate Scene Text Detector (EAST) uses VGG16/PVANet trained on ImageNet dataset as a feature extractor. The input is passed into the feature extractor and a U-Net architecture is used to merge the feature maps. The output layer consists of probabilities of text in a region and all the bounding box coordinates. The limitation is that the receptive field of EAST is not very large and hence it is not good for detecting lengthy texts.

The existing text detection methods such as bounding box regression is not accurate for the texts in arbitrary shapes such as curved text, and the segmentation method does not work well when the text is close to each other. In order to solve these two problems, Wang et al. [2019a] proposed a new method called a new Progressive Scale Expansion Network (PSENet). It works as a segmentation based detector with numerous predictions with for every individual text instance. Thus, the proposed pixel-based segmentation method can accurately locate texts of any shape and detect two text instances even when they are very close.

Similar to the above approach, Tian et al. [2019] proposed a method called Learning Shape-Aware Embedding for Scene Text Detection. The idea is to map image pixels to embedding in the feature space, pixels belonging to the same text instance will be closer to each other in the space, whereas pixels of different text instances will be further away from each other. ResNet-50 is used as a feature extractor. The network output includes the embedded feature map and the mask map of texts. It performs well on three benchmark scene text detection datasets ICDAR15 [ICDAR15, 2015], MSRA-TD500 [MSRA-TD500, 2012], and CTW1500 [Yuan et al., 2019].

Wang et al. [2019b] introduced a method called Arbitrary Shape Scene

Text Detection with Adaptive Text Region Representation. It involves two-stage text detection. The first stage is similar to Faster R-CNN. Text proposals are obtained through CNN + RNN + Region Of Interest (ROI). The second stage is to refine the text proposals to make the prediction frame more accurate. The network uses SE-VGG16 as a feature extractor. Experiments show that Squeeze and Excitation Network (SE) Block can improve performance. The main advantage of the proposed method is that it uses adaptive points instead of fixed points to represent a text box. The previous methods used fixed points to represent the text box, but the number of points for horizontal text, multidirectional text, and curved text is not the same. So, the points to represent the text box are adapted to detect text of various shapes. It uses the Long Short Term Memory (LSTM) network to learn how many points should be used to represent the text box.

Similar to SegLink, Baek et al. [2019b] proposed Character Region Awareness For Text detection (CRAFT) that detects text area by identifying single character (character region score) and connection between each characters (affinity score). The network structure is similar to the EAST network structure: the feature extraction backbone network part uses VGG-16 with batch normalization; the feature decode module is similar to U-Net, and it also uses a top-down feature aggregation method; the network finally outputs two channel feature map, that is, region score map and affinity score map in the form of the heat map. This thesis uses the CRAFT pre-trained model for text detection and Baek et al. [2019a] proposed model for text recognition.

2.6.2 OCR for steel type plates

Similar to steel type plate recognition, many research works have been published to extract text from the vehicle license plate, forms, and documents [Duan et al., 2005; Zang et al., 2015; Impedovo, 2012; Milewski and Govindaraju, 2006; Laroca et al., 2018]. Several deep learning and computer vision algorithms exist to solve these problems. But the existing approach would fail for our steel type plate images since the complexity of our problem is rel-

atively higher. Before the deep learning era, there have been several classical research works done to improve the accuracy of OCR for steel type plates. Some of the approaches similar to our problem are discussed below:

Novák et al. [2013] used fuzzy logic methods for the recognition of distorted characters printed on metal. YU et al. [2017] proposed a technique that involved image processing with a series of mathematical and morphological treatments for metal surface engraving character detection. Xiang et al. [2018] proposed a technique called Metal Stamping Character (MSC). They combined and fused the images of the steel plates of the same scene from multi-direction and different illuminations to improve the character recognition of metal stamped characters. Raka and Agrawal [2019] proposed an algorithm based on K - Nearest Neighbour (KNN) for reading embossed text from credit/debit cards. After preprocessing steps, they locate the credit/debit card number and split it into individual characters. They created a dataset by manually sorting characters from 0 - 9 and then trained the OCR model with its corresponding label using KNN. These classic approaches are performing basic image manipulation operations like image binarization, thresholding, segmentation and other morphological processes to increase the quality of the original image.

Recent advancements in computer vision have given a set of new possibilities to tackle advanced recognition problems. Patil and Dhanvijay [2015] proposed a computer vision technique to identify texts from the engine and chassis parts. With the success of CNNs in the image classification [Hinton et al., 2012; Simonyan and Zisserman, 2014], Segmentation [Girshick et al., 2014], Object Detection [Russakovsky et al., 2015], it is combined with recurrent networks along with attention mechanism for solving OCR problem.

Bui et al. [2017] proposed an algorithm that automatically selects the preferred pre-processing methods to be applied to a document image to maximize OCR performance for the processed image. They used 21 different such possible combinations to pre-process the image. Their approach achieved a

good improvement in the performance of OCR. But the main drawback is that the possible pre-processing options were limited. Also, it does not suit our requirements.

Similar to our approach, Hu et al. [2019] used CNN for enhancing the character recognition of embossed characters of the shoe soles. Their research focussed only on image enhancement and did not care about the character recognition part.

2.6.3 GANs in OCR

Using standard GANs, Karras et al. [2017] generated highly realistic human faces trained with the celebrity dataset while Jin et al. [2017] generated faces of anime characters. Likewise, Brock et al. [2018] demonstrated the ability of GANs to generate synthetic images of objects that are not differentiable from the real images using the BigGAN technique. Works such as [Zhang et al., 2017; Reed et al., 2016b,a; Dash et al., 2017] used conditional GANs to generate images given its description as text which is popularly known as text-to-image translation.

Large scale annotated datasets are required for the development of robust deep learning models, especially for CNN. Acquiring such datasets is always challenging in cases like medical images. Hence, Frid-Adar et al. [2018] used GANs to generate synthetic medical images as a data augmentation technique to increase the performance of CNN in classifying images of liver lesions. Similarly, Fiore et al. [2019] used GAN for improving the effectiveness of classifying credit card fraud detection. Even though it was a binary classification problem, the dataset was heavily imbalanced i.e. the interested class was represented significantly lesser than the other. Synthetic images were generated for the imbalanced class as an alternative to over-sampling techniques.

For our case, since the goal was to generate a clear image from noisy steel

type plate images in order to maximize the OCR performance, it was realized that one way to go about solving this problem is by using a generative adversarial image. Some of the research works using GANs for improving character recognition are presented below:

Hosangadi et al. [2019] proposed a method that uses DCGAN for enhancing the image quality of documents to improve the OCR performance. Their model uses an encoder trained to learn the latent representation of a low-quality image. The latent representation is then passed as an input to the conditional GAN generator which is trained to produce a high-quality image.

Lat and Jawahar [2018] proposed a super-resolution based solution for improving the OCR accuracy using GAN. They used SRGAN to convert the low resolution (scanned at as low as 75dpi) document images into super-resolved texts and then passed it into the OCR engine.

Similarly, Karimi et al. [2020] developed a model called Handwritten-to-Machine-Print Generative Adversarial Network (HW2MP-GAN) to transform the handwritten text to machine-printed text. Their model utilizes sliced Wasserstein distance and U-Net architecture for better quality image-to-image translation. These approaches work well for printed documents and have shown to improve the accuracy of OCR.

In summary, we list out the papers that especially influenced this work. The proposed overall framework uses a GAN generator model and an OCR engine. For GAN, the best generator model is selected among the three image-to-image translation GANs: i) Isola et al. [2017] proposed a pix2pix GAN , ii) Zhu et al. [2017] proposed CycleGAN and iii) Stoller et al. [2019] proposed FactorGAN. For OCR engine, the text detection uses CRAFT approach [Baek et al., 2019b] and the text recognition uses Baek et al. [2019a] proposed text recognition model.

Chapter 3

Methodology

This chapter describes the key idea, the proposed framework and methodology used in this thesis. A comparison between different image-to-image translation models using GANs is provided. Also, the architecture of text detection and text recognition models used for the OCR engine has been presented. This chapter discusses, in brief, each of the components used in the overall framework and their implementation procedures.

The image-to-image translation is the class of computer vision problems where the goal is to learn the mapping between input and target domain. The applications of image-to-image translation range from style transfer, colorization, sketches to photo, and image super-resolution. In this thesis, we have investigated three different image-to-image translation GANs: Pix2pix, CycleGAN and FactorGAN to improve the readability of texts present in the steel type plate/nameplate images. Then the best generator model from the three GANs is selected and integrated with the pre-trained text detection & text recognition model.

3.1 Pix2pix GAN

Isola et al. [2017] proposed pix2pix (pixel to pixel) Generative Adversarial Network that has shown to produce realistic and detailed synthetic images.

Pix2pix is a supervised image-to-image translation model that uses conditional GANs to learn a function to map input image to output image.

3.1.1 Architecture

Similar to the standard GAN, the pix2pix model consists of two components i) Generator and ii) Discriminator as shown in Figure 3.1. The generator transforms the input image to an image in the target domain while the discriminator measures the similarity of the input image to image passed randomly either generated by generator or the real image and predicts the probability of the image being real. The pix2pix model is a type of conditional generative model with a target image as additional information.

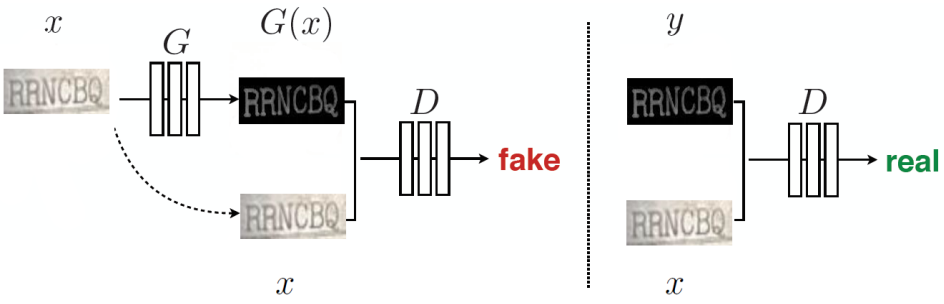


Figure 3.1: Architecture of pix2pix. Source: Adapted from [Isola et al., 2017]

In contrast to the standard conditional GANs, the generator takes the source image with no noise as input whereas both the source image and target image are passed to the discriminator as shown in Figure 3.1. The generator is trained with adversarial loss and L1 loss to produce a meaningful translation of a source image to a target image. Also, the randomness is added to the network in the form of dropout. The architecture of the generator and the discriminator is explained in detail which is as follows:

Generator

For the generator architecture, the ResNet framework proposed by He et al. was used. Residual Network (ResNet) is a CNN architecture consists of a

series of residual blocks with skip connections to address the problem of vanishing gradients. The vanishing gradients problem occurs in deep neural networks when the weights of the initial layer not getting updated properly during back-propagation as the repeated multiplication makes the gradient small. Thus, the performance gets saturated with more layers in the network. Apart from this problem, they showed that adding more layers to the neural network slows down the training and does not improve accuracy. ResNet uses the identity matrix via skip connection that multiplies the gradient by 1 during backpropagation. Thus, it preserves the gradient and avoids information loss [He et al., 2016; Mittal, 2019].

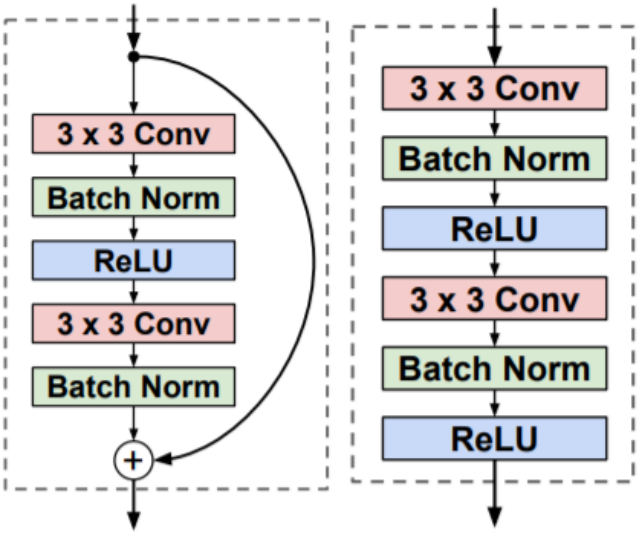


Figure 3.2: Difference between residual block (left) vs ordinary CNN (right) connections. Source: Supplementary material, [Johnson et al., 2016]

The differentiating factor of ResNet from a CNN is their skip connection which is shown in Figure 3.2. The skip connection adds the original input to the output of the convolution block. In traditional CNNs, each layer output is fed as input to the next layer, but with skip connections, each layer output is passed to consecutive layers as well as the non-consecutive layers. This helps in solving the problem of vanishing gradient and the identity matrix helps in maintaining the performance of initial layers with the higher layers [Mittal, 2019]. In our approach, the generator accepts an input image of size

600 x 400 x 3 and generates an output image of the 600 x 400 x 1. The full generator architecture is shown in Table 3.1.

Block	**Filters**	**Filter size**	**Stride**
Conv-BN-ReLU	64	7x7	1
Conv-BN-ReLU	128	3x3	2
Conv-BN-ReLU	256	3x3	2
Residual Block1	256	3x3	1
Residual Block2	256	3x3	1
Residual Block3	256	3x3	1
Residual Block4	256	3x3	1
Residual Block5	256	3x3	1
Residual Block6	256	3x3	1
Residual Block7	256	3x3	1
Residual Block8	256	3x3	1
Residual Block9	256	3x3	1
ConvT-BN-ReLU	128	3x3	2
ConvT-BN-ReLU	64	3x3	2
Conv-tanh	1	7x7	1

Table 3.1: Architecture of pix2pix generator

The generator consists of encoder, 9 ResNet blocks and decoder blocks. Each encoder and decoder block consists of a convolution layers, a batch normalization (BN) layer followed by the ReLU activation function. Each ResNet block has two connections from its input, one connection going through a series of convolutions, batch normalization & activation functions and the other connection skipping over that series as shown in the Figure 3.2.

Discriminator

The PatchGAN architecture proposed by Li and Wand [2016] was used for the discriminator. It compares NxN patches of the image instead of the entire image to classify if it is real or fake. Thus, the PatchGAN is more suitable for producing sharp high-frequency detail and have shown great classification performance in GANs. We used the default 70 x 70 patches as suggested in

the pix2pix paper. The discriminator architecture is shown in Table 3.2. Here LReLU represents the Leaky ReLU activation function.

Block	**Filters**	**Filter size**	**Stride**
Conv-LReLU	64	4x4	2
Block 2 Conv-BN-LReLU	128	4x4	2
Block 3 Conv-BN-LReLU	256	4x4	2
Block 4 Conv-BN-LReLU	512	4x4	1
Block 5 Conv-Sigmoid	1	4x4	1

Table 3.2: Architecture of pix2pix discriminator

It consists of 5 blocks of convolution layers. The Leaky ReLU with 0.2 negative slope is used as an activation function. The first convolutional layer consists of 64 filters and it is upscaled after each block. At the final layer, a sigmoid activation function is used to predict if the 70 x 70 patch image is real or fake.

3.1.2 Loss function

The loss function of the pix2pix is the same as the conditional GAN as discussed in the earlier section. The L_1 loss is added extra to the loss function since the goal of the generator is to generate realistic images such that it fools the discriminator. The L_1 loss calculates the pixel-wise difference between the source image and the target image. Thus forcing the generator to produce images close to the ground truth. The adversarial loss [Isola et al., 2017] is given by

$$L_{\text{adv}}(G, D) = \mathbb{E}_{x,y}[logD(x, y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x, z))]$$

The L_1 loss [Isola et al., 2017] is given by

$$L_{\text{L1}}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

And the final objective function [Isola et al., 2017] is the sum of the adversarial loss and the L_1 loss which is given by

$$G^* = \underset{G}{\underset{D}{arg\,min\,max}} L_{\text{adv}}(G, D) \; + \; \lambda L_{\text{L1}}(G)$$

where λ is the scalar used to determine the importance of the L_1 loss.

3.2 CycleGAN

CycleGAN is an unsupervised method for training image-to-image translation models. They learn the characteristics and styles from an image collection of the source domain and translates it to the target domain in the absence of paired data.

The need for one to one mapping is eliminated by transforming the image from the source domain to the target domain and then back to the source domain just like translating a sentence from English to German and from German to English.

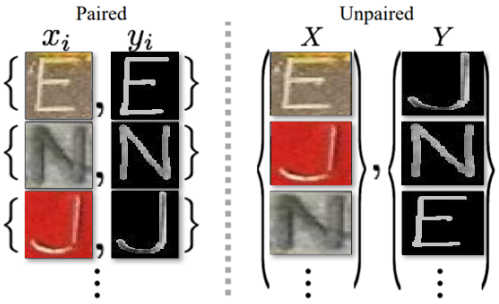


Figure 3.3: Paired & unpaired datasets. Source: Adapted from [Zhu et al., 2017]

The authors conducted experiments on various image datasets such as horse ↔ zebra, apple ↔ orange, photo ↔ monet, etc. Relaxing the one to one mapping makes the CycleGAN a more powerful method to tackle a variety of problems like the translation of noisy steel type plate images to clear images.

3.2.1 Architecture

Every image in a paired dataset is manually mapped from domain X to domain Y. Thus images in both domains share some common features. This

pairing defines a meaningful relationship between images in both domains. In pix2pix, the generator takes input from domain X and translates it to an image close to domain Y. The CycleGAN with an unpaired dataset does not have this luxury since there is no meaningful transformation that it can learn. The difference between the paired and unpaired dataset is shown in Figure 3.3. To establish a meaningful relationship in the unpaired dataset, Zhu et al. introduced the concept of cycle consistency. The cycle consistency refers to the ability of translating an image from domain X to domain Y and then the generated image is translated back again to domain X. The architecture of CycleGAN is shown in Figure 3.4.

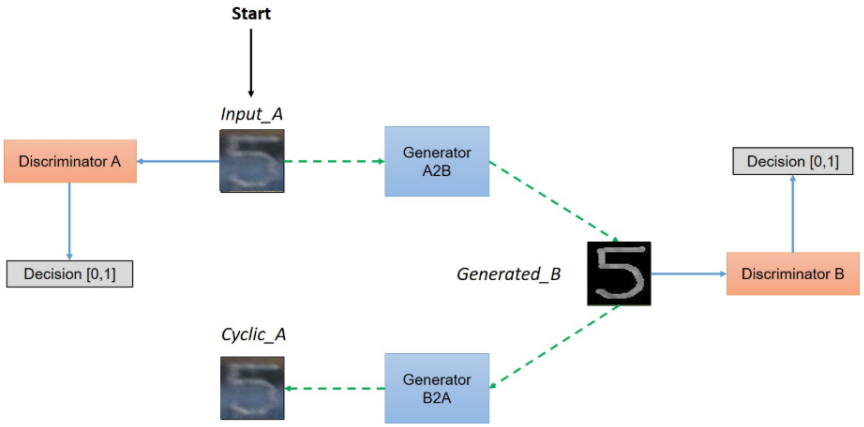


Figure 3.4: Architecture of CycleGAN. Source: Adapted from [Zhu et al., 2017]

A generator will map the input image from a source domain to some image in the target domain with a constraint that the generated image must share some features that can map back to its original image. Thus the CycleGAN architecture consists of two generators G and F. The goal of generator G is to map X domain to Y domain such that G: X→Y and the other generator F will map Y domain to X such that F: Y→X. These generators have their respective discriminators D_x and D_y. Discriminator D_x forces generator G to translate images from domain X into an output that is indistinguishable from the images of domain Y and vice versa for Discriminator D_y.

Generator

The same ResNet-9 blocks discussed in the pix2pix section was used as the architecture for both generators. Since the two generators are used, the images are resized to 300 x 300 to adapt to the memory requirements of the system. The instance normalization is replaced by batch normalization since multiple GPUs are used for training. The full generator architecture is explained in detail in Table 3.3.

Block	**Filters**	**Filter size**	**Stride**
Conv-IN-ReLU	64	7x7	1
Conv-IN-ReLU	128	3x3	2
Conv-IN-ReLU	256	3x3	2
Residual Block1	256	3x3	1
Residual Block2	256	3x3	1
Residual Block3	256	3x3	1
Residual Block4	256	3x3	1
Residual Block5	256	3x3	1
Residual Block6	256	3x3	1
Residual Block7	256	3x3	1
Residual Block8	256	3x3	1
Residual Block9	256	3x3	1
ConvT-IN-ReLU	128	3x3	2
ConvT-IN-ReLU	64	3x3	2
Conv-tanh	1	7x7	1

Table 3.3: Architecture of CycleGAN generator

Discriminator

The PatchGAN discussed in the pix2pix section was used as the architecture for both the discriminators with the default setting 70 x 70. The full discriminator architecture is explained in detail in Table 3.4.

Block	Filters	Filter size	Stride
Conv-LReLU	64	4x4	2
Block 2 Conv-IN-LReLU	128	4x4	2
Block 3 Conv-IN-LReLU	256	4x4	2
Block 4 Conv-IN-LReLU	512	4x4	1
Block 5 Conv-Sigmoid	1	4x4	1

Table 3.4: Architecture of CycleGAN discriminator

3.2.2 Loss function

The loss function of CycleGAN needs to satisfy the following condition - i.e., the generated image should always be mapped back to its original image.

Let the two generators be G, F and two discriminators D_x, D_y. The goal of the generators is to translate an image from domain X to Y and vice versa. Let the training samples in the domain X is defined as $\{x_i\}_{i=1}^{N}$ and training samples in the domain Y is defined as $\{y_j\}_{i=1}^{M}$. The distribution of the data is defined as the $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. The loss function of CycleGAN is defined by two losses: i) Adversarial loss and ii) Cycle consistency loss.

The adversarial loss is used to match the distribution of generated images with the target domain. It is applied to both the generators. The adversarial loss [Zhu et al., 2017] for generator G that translates the image from domain X to domain Y such that G: X→Y is given by

$$L_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_y \sim p_{data}(y)[logD_Y(y)] + \mathbb{E}_x \sim p_{data}(x)[log(1-D_Y(G(x))]$$

Likewise for generator F that translates the image from domain Y to domain X such that F: Y→X is given by,

$$L_{\text{GAN}}(F, D_X, Y, X) = \mathbb{E}_x \sim p_{data}(x)[logD_X(x)] + \mathbb{E}_y \sim p_{data}(y)[log(1-D_X(F(y))]$$

The cycle consistency loss enables both the generators to be stable and consistent. This loss regulates the GAN training process and makes sure that

the generator does not generate any random permutation of images in the target domain [Zhu et al., 2017]. It is given by

$$L_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) \; - \; x\|_1] \; + \; \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1]$$

Combining these two losses,

$$L_{\text{GAN}}\left(G, F, D_X, D_Y\right) = L_{\text{GAN}}\left(G, D_Y, X, Y\right) + L_{\text{GAN}}\left(F, D_X, Y, X\right) + \lambda L_{\text{cyc}}(G, F)$$

where λ controls the importance of cycle consistency loss. So, the final objective function [Zhu et al., 2017] is given by

$$G^*, F^* = \underset{G,F}{argmin} \; \underset{D_X, D_Y}{max} \; L(G, F, D_X, D_Y)$$

3.3 FactorGAN

The last GAN that is investigated in this thesis is Stoller et al. [2019] proposed FactorGAN. Even though the GANs have been producing realistic images with greater success, they rely heavily on the amount of training data. Even though the datasets for the everyday objects are widely available, the datasets for specific industrial use case remains scarce. In the case of pix2pix, the generation of image pairs is time consuming, requires a lot of manual annotation and is expensive. The CycleGAN model does not make use of the paired information in the dataset. Although the methods like [Tripathy et al., 2018; Almahairi et al., 2018] use both the information of paired and unpaired data with additional reconstruction losses, they have to be balanced with many hyperparameters. This leads to computational overheads and thus the convergence of the generator to the desired distribution becomes uncertain [Stoller et al., 2019]. The FactorGAN provides a solution to enable training for generative models with incomplete observations ie when the paired data is less.

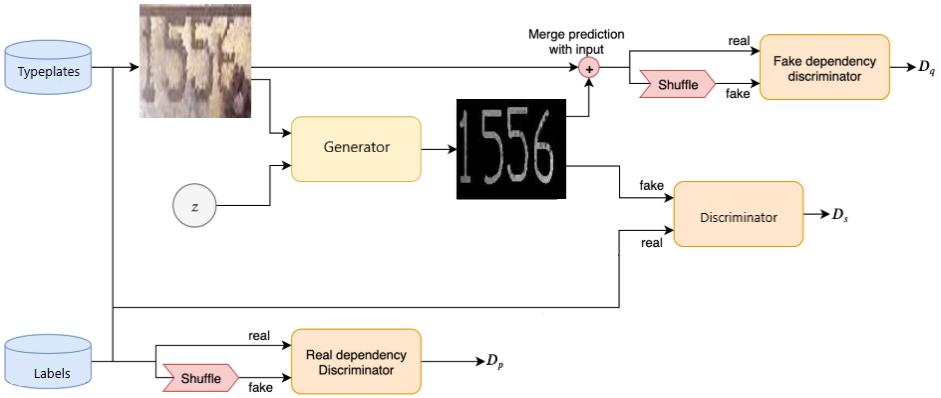


Figure 3.5: Architecture of FactorGAN. Source: Adapted from [Stoller, 2020]

3.3.1 Architecture

The FactorGAN uses a generator and three discriminators as shown in Figure 3.5. The generator generates a clear image of the steel type plate close to the target domain while the discriminator classifies the real and fake image. A real dependency discriminator distinguishes the real steel type plates and its ground truth pairs. A fake dependency discriminator distinguishes the real steel type plates and generated image pairs.

Generator

The U-Net architecture proposed by Ronneberger et al. [2015] was used for the generator model. U-Net is a fully convolutional neural network that can be composed into three parts: shrinkage, bottleneck and expansion. It consists of conv layers, max pooling, ReLU activation, concatenation layers, up and down sampling as in Figure 3.6. The shrinkage part on the left uses convolution and max pooling operations. The expansion part on the right uses several blocks with upsampling and concatenation [Ronneberger et al., 2015]. The full generator architecture is shown in Table 3.5.

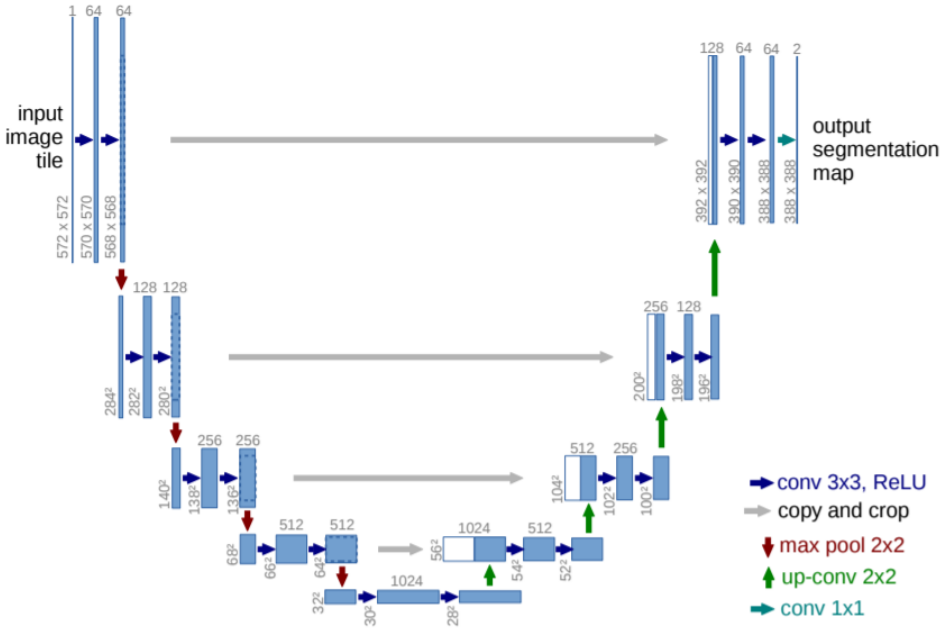


Figure 3.6: Architecture of U-Net. Source: [Ronneberger et al., 2015]

Block	Filters	Filter size	Stride
Conv-BN-ReLU	32	3x3	1
Conv-BN-ReLU	32	3x3	1
Conv-BN-ReLU	64	3x3	1
Conv-BN-ReLU	64	3x3	1
Conv-BN-ReLU	128	3x3	1
Conv-BN-ReLU	128	3x3	1
Conv-BN-ReLU	256	3x3	1
Conv-BN-ReLU	256	3x3	1
Conv-BN-ReLU	256	3x3	1
Conv-BN-ReLU	256	3x3	1
Conv-BN-ReLU	128	3x3	1
Conv-BN-ReLU	128	3x3	1
Conv-BN-ReLU	64	3x3	1
Conv-BN-ReLU	64	3x3	1
Conv-BN-ReLU	32	3x3	1
Conv-BN-ReLU	32	3x3	1
Conv-BN-ReLU	32	3x3	1
Conv-BN-ReLU	32	3x3	1
Conv-sigmoid	1	1x1	1

Table 3.5: Architecture of FactorGAN generator

Discriminator

The architecture of the discriminator is shown in Table 3.6. It consists of convolution layers with spectral normalization and LReLU activation. All the layers except the fully convolution layer have biases. Spectral normalization normalizes the weight such that the Lipschitz constant for each layer as well as the whole network is one.

Block	Filters	Filter size	Stride
Conv-SN	32	4x4	2
Block 2 Conv-SN-LReLU	64	4x4	2
Block 3 Conv-SN-LReLU	128	4x4	2
Block 4 Conv-SN-LReLU	256	4x4	2
Block 5 Conv-SN-LReLU	512	4x4	2
Block 6 Conv-SN-LReLU	1024	4x4	2
Block7 Conv-LReLU	1	1x1	1

Table 3.6: Architecture of FactorGAN discriminator

3.3.2 Objective function

Let the probability distribution of a dataset is p_x over $x \in \mathbb{R}^d$, discriminator D_θ and the generator be G_ϕ that maps an n dimensional input $z \sim p_z$ to a d dimensional sample $g_\phi(z)$ resulting in the generator distribution q_x. Then the loss function [Stoller et al., 2019] for standard GAN is given by

$$\underset{\theta}{argmax}\ \mathbb{E}_{z\sim p_z} log D_\theta(G_\phi(z))$$

To train the GAN with incomplete observation [Stoller et al., 2019], the discriminator D is mapped to a joint density ratio $\frac{p_x(x)}{q_x(x)}$. This joint density ratio is factorized into a product of density ratios.

$$h(\widetilde{D}(x)) = \frac{p_x(x)}{q_x(x)} = \frac{c_P(x)}{c_Q(x)} \prod_{i=1}^{K} \frac{p_x^i(x^i)}{q_x^i(x^i)} \text{ where,}$$

$$c_P(x) = \frac{p_x(x)}{\prod_{i=1}^{K} p_x^i(x^i)} \ \&\ c_Q(x) = \frac{q_x(x)}{\prod_{i=1}^{K} q_x^i(x^i)}$$

For conditional GAN [Stoller et al., 2019], the generator g_ϕ that maps a conditional input x^1 and a noise to an output x^2 resulting in output probability $q_\phi(x^2|x^1)$. In order to eliminate the need for the paired samples, applying the factorization to x^1 and x^2 with K=2 in the above equation results in

$$\frac{p_x(x)}{q_x(x)} = \frac{\frac{p_x(x)}{p_x^1(x^1)p_x^2(x^2)}}{\frac{q_x(x)}{q_x^1(x^1)q_x^2(x^2)}} = \frac{c_P(x)}{c_Q(x)} \frac{p_x^2(x^2)}{q_x^2(x^2)}$$

From the above equation, Stoller et al. justified the use of p and q dependency discriminator to model the input-output relationship and a marginal discriminator.

3.4 Comparison of image-to-image translation GANs under study

To summarize, Table 3.7 presents a brief comparison of the above discussed image-to-image translation GANs. The networks describe the complexity of the model with number of generators and discriminators present. Image size denotes the size of input in pixels used in each GAN. Release date is the given publication date for each related article. The method, backbone, normalization and type represent the loss function, architecture, type of normalization and learning approach used for each GAN respectively.

	Pix2pix	**CycleGAN**	**FactorGAN**
Method	Adversarial loss + L1 loss	Cycle consistency loss	Factorized discriminators
Networks	1 Generator 1 Discriminator	2 Generators 2 Discriminators	1 Generator 1 Marginal Discriminator 2 Dependency Discriminators
Image size	600 x 400	300 x 300	256 x 256
Release date	2017 - 2018	2017 - 2018	2019 - 2020
Normalization	Batch	Instance	Spectral
Backbone	ResNet	ResNet	U-Net
Type	Supervised method that requires paired dataset	Unsupervised method that requires unpaired dataset	Semi-supervised method that handles incomplete data points

Table 3.7: Comparison of image-to-image translation GANs under study

3.5 OCR - Text detection

Baek et al. [2019b] proposed Character Region Awareness For Text detection (CRAFT) is the state of the art for scene text detection. We have used the pre-trained model trained using the CRAFT approach in this thesis for text detection.

When the problem of text detection is applied to real-world scenarios, it can be seen that the texts in the real world are curved, arbitrary or in an irregular shape. The existing bounding box-based method implemented using the anchor box cannot handle the scenarios where the text is excessively deformed or the text is too large. To improve the text detection in such scenarios, the CRAFT framework was proposed.

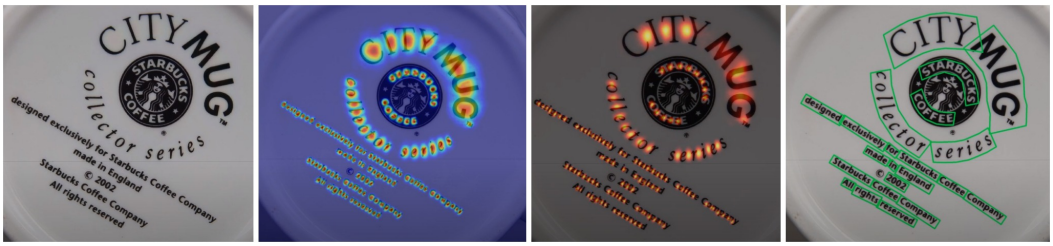


Figure 3.7: The process of CRAFT: (a) Input image (b) Character region map (c) Affinity score map (d) Text detection output. Source: Adapted from [Lee, 2019]

The idea of CRAFT is to detect text by exploring each character region and affinity between the characters. It first locates each character in the word using the region score. Then predict whether each character belongs to the same text using the affinity score and concatenate. Thus this model mainly outputs two things:

(i) **Region score** gives the probability of the detected location being a character.

(ii) **Affinity score** gives the probability of whether the character at this position needs to be concatenated into a word.

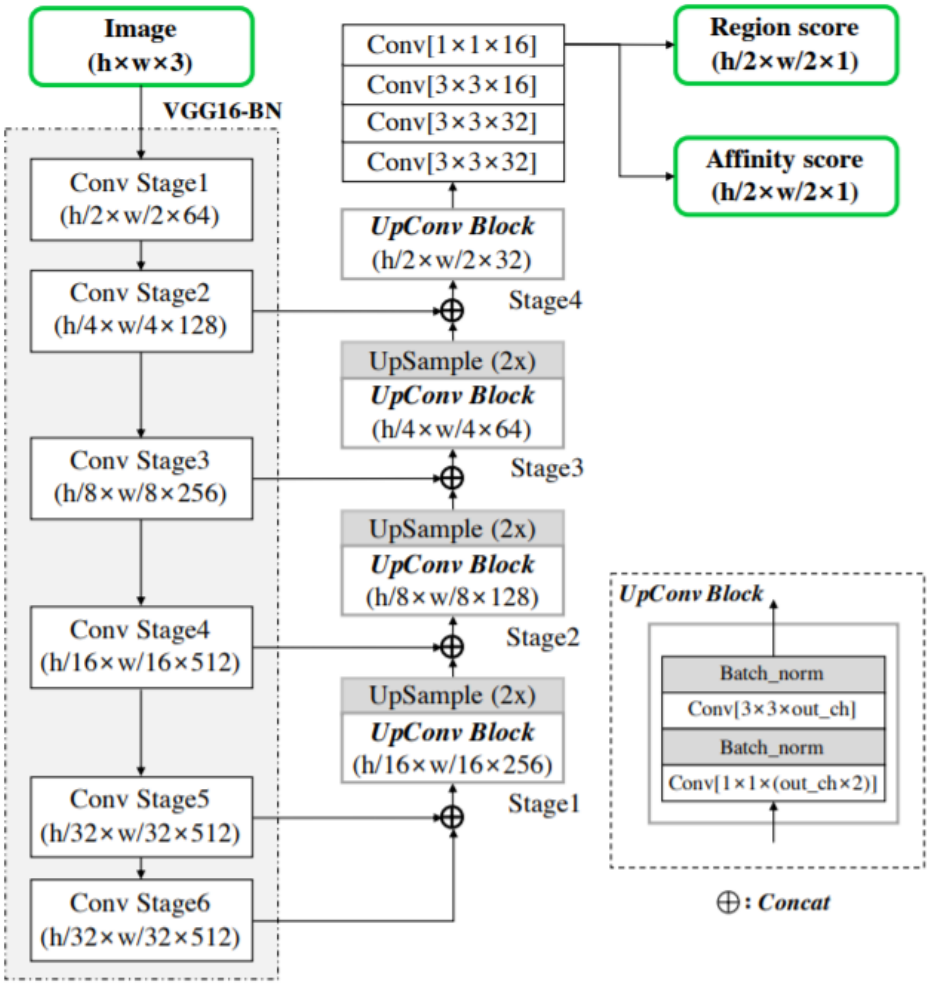


Figure 3.8: The CRAFT architecture. Source: [Baek et al., 2019b]

The network structure of the CRAFT is shown in Figure 3.8. The backbone network is VGG-16, followed by the feature fusion of Feature Pyramid Network. The final output of the model generates a character region prediction (region score) and character association prediction (affinity score). Since the existing datasets for training the text detection models are based on word-level annotations and there is no character level labelling, Baek et al. used synthetic character dataset for training. The pre-trained model used in this thesis is trained in SynthText [Gupta et al., 2016], ICDAR13, ICDAR17 datasets.

3.6 OCR - Text recognition

Several different methods have been proposed in many research works on the scene text recognition. However, each paper that was presented claimed to be the state of the art, there was no fair comparison. Baek et al. [2019a] in their work presented the fair comparison of different existing methods for scene text recognition. We have utilized their proposed framework for text recognition in this thesis.

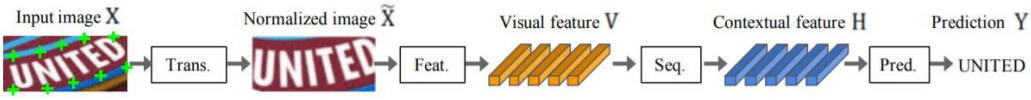


Figure 3.9: Text recognition framework. Source: [Baek et al., 2019a]

An integrated framework that consists of four stages is proposed for scene text recognition as shown in Figure 3.9. It is as follows:

1. **Transformation:** The input image in natural scenes comes in arbitrary shapes like curved, vertical, irregular and diverse shapes. Thus the transformation stage uses the Thin Plate Spline transformation (TPS), a variant of Spatial Transformation Network (STN) to normalize the input image and transform it into a predefined rectangle.

2. **Feature extraction:** The feature extraction uses CNNs to estimate each character in the receptive field. Their framework has the option to select between ResNet, VGG, and RCNN architecture for feature extraction. We have used ResNet for feature extraction in our OCR engine.

3. **Sequence modelling:** Sequence modelling is used to model the contextual information within the sequence of characters using Bidirectional LSTM (BiLSTM) for robust predictions.

4. **Prediction:** Attention (Attn) or Connectionist Temporal Classification (CTC) is used to predict a sequence of characters for an input image. We have used Attn as a prediction module for our pre-trained

model in this thesis. In the case of CTC, it enables the prediction of a non-fixed number of sequences despite the fixed number of features. The core of CTC seems to be that words can be combined by recognizing letters in each input column and removing repeated letters or blanks. On the other hand, Attn is said to automatically capture the flow of information in the input sequence to predict the output sequence. So, Attn helps the model learn a character-level language modelling that represents class dependencies in the output.

3.7 Google Vision OCR

Google provides an Application Programming Interface (API) that allows users to integrate many computer vision cloud services such as face detection, logo detection, object detection, text detection, etc to their framework [Google, 2016]. One such commercial service is the cloud-based OCR that can recognize image files in multiple formats and convert them to text. The API must be enabled first to access the OCR engine and a key for the API needs to be generated. Then the text can be extracted from any number of images that are uploaded. In the work of [Han and Hickman, 2019], they compared different OCR tools available in the market and showed that Google Vision OCR is one of the best tools for optical recognition. We have conducted experiments with the google vision OCR in addition to the proposed OCR to compare and evaluate how good is the proposed OCR engine.

3.8 The overall framework

The proposed framework includes the best generator model from the above-discussed GANs to preprocess the weathered steel type plates. Then the generated image is passed into OCR engine to extract the text out of it.

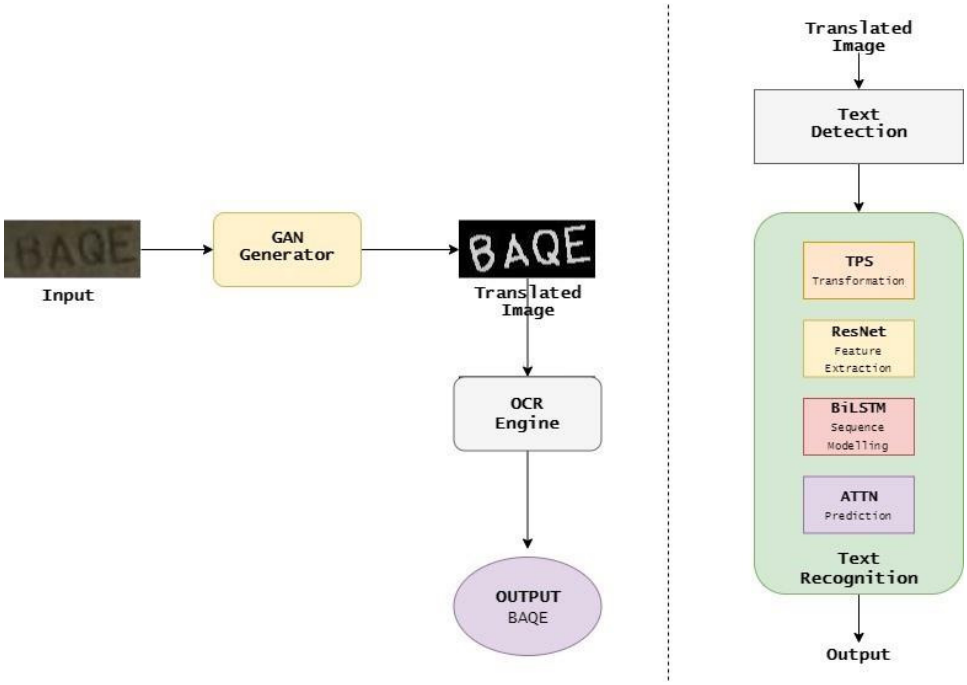


Figure 3.10: (a) The overall framework (b) The proposed OCR

In this thesis, we have studied two different OCR engines 1) Google Vision OCR and 2) The proposed CRAFT based OCR engine with integrated text detection and recognition models. The overall framework is shown in the left of Figure 3.10.

The proposed OCR engine is shown in the right of Figure 3.10. It consists of the CRAFT text detection model followed by the text recognition model. The text recognition model uses TPS for transformation, ResNet as feature extractor, BiLSTM for sequence modelling and Attn for prediction.

Chapter 4

Experiment

This chapter describes in detail the implementation and experimental setup of the overall framework used in this thesis. In addition, Section 4.3 discusses in depth about the dataset of steel type plate images, the process of creating image pairs and synthetic data elaborately. Finally, Section 4.4 describes the evaluation metrics used to measure the performances of all the GANs and OCR engines under study.

4.1 Implementation

In this section, we list out the best hyperparameters for each method from the different settings that were tried out and their implementation in detail. For all the three GANs that were studied, we used the ADAM optimizer with the same values β_1 0.5 and β_2 0.999 for both generator and discriminator. We have used the PatchGAN discriminator for both pix2pix and CycleGAN experiments. We have performed experiments with different combinations for hyperparameters such as

i) Normalization

ii) Regularization using dropout

iii) Batch size with respect to multiple GPUs

iv) Size of the input image

v) Data augmentation using flip

vi) Learning rate
vii) Use of synthetic data
viii) Architecture for generator and discriminator

The random search with qualitative evaluation was used for hyperparameter optimization. Many hyperparameters such as batch size, epoch, etc had little impact on the results of the training. For all the three GANs, the combination of parameters used in the original paper produced the best results. These settings are listed in the following table:

	Pix2pix	**CycleGAN**	**FactorGAN**
Batch Size	4	2	25
Learning Rate	0.002	0.002	0.001
Optimizer	Adam	Adam	Adam
beta1	0.5	0.5	0.5
beta2	0.999	0.999	0.999
Epoch	200	200	200
Dropout	No	Yes	No
Normalization	Batch	Instance	Spectral
Generator	ResNet	ResNet	U-Net
Discriminator	Patch discriminator	Patch discriminator	Convolutional discriminator

Table 4.1: Best parameter setting used in the experiments for each GAN

The original PyTorch implementation of GAN and OCR models that have been used for training can be found in the following repositories :

1. **Pix2pix & CycleGAN** - `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`
2. **FactorGAN** - `https://github.com/f90/FactorGAN`
3. **Text detection** - `https://github.com/clovaai/CRAFT-pytorch`
4. **Text detection pretrained model (craft-mlt-25k.pth)** - `https:`

//drive.google.com/file/d/1Jk4eGD7crsqCCg9C9VjCLkMN3ze8kutZ/view

5. **Text recognition** - https://github.com/clovaai/deep-text-recognition-benchmark

6. **TPS-ResNet-BiLSTM-Attn.pth** - The pretrained model for text recognition can be found in https://drive.google.com/drive/folders/15WPsuPJDCzhp2SvYZLRj8mAlT3zmoAMW

All the codes used in the thesis, latex files and the codes that are referenced from public repositories along with its respective licenses are included in the university GitLab repository. It can be cloned using the following URL: https://www.uni-hildesheim.de/gitlab/panneers/thesis_steel_type_plate_recognition.git.

4.2 Experimental setup

For our experiments, all codes are written in python and implemented in PyTorch. We have used other python libraries such as Numpy, Scipy, Pillow, Scikit-Learn. For training, we have used machines with a configuration having AMD Ryzen TR 2920X processor with 64 cores, 2 x 8 GB RTX 2070 Super X GPU, 128 GB RAM.

4.3 Dataset

This section provides a brief description of the dataset used in this thesis. Generally, the datasets for everyday objects are widely available. But for specific industrial use cases like in our case, it remains scarce. So, the data needs to be created from scratch.

4.3.1 Data collection

Here, we introduce our strategies to dataset collection and processing. The following steps were performed to create the dataset:

1. A python script is written such that it crawls the steel type plate images from the google image search to create the dataset.

2. Then the resulting images of web scraping were manually refined to remove the unnecessary images from the dataset.

3. The images were cropped to remove the background information and to keep only the steel type plates part in the image.

4. In the middle of the experiment, few duplicate images were found in the dataset. Then these duplicate images were removed using a python script.

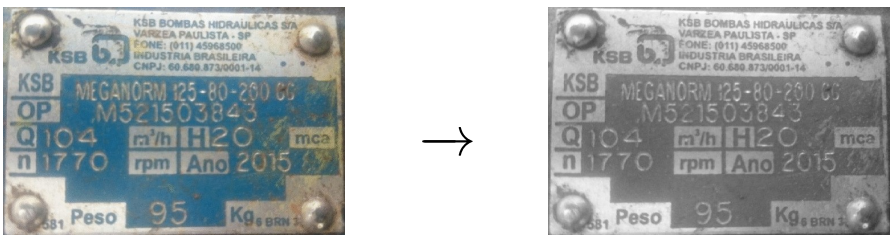
5. All the images in the dataset were resized to a uniform resolution of 600 x 400.

6. The final dataset consists of train, test and validation split with 366 in the train set, 30 in the validation set and 100 images for the testing set.

4.3.2 Label creation

Since we use the image-to-image translation GAN models, the target labels corresponding to each input image needed to be created for training. The labels for every image is created in the following manner :

- Initially the RGB images in the dataset are converted to grayscale images.



(a) RGB image (b) Grayscale image

Figure 4.1: RGB to Grayscale conversion

- Then, we used the *Photo Sketch* app in the iPad and highlighted the text part on top of each grayscale image manually using Apple pencil as shown in Figure 4.2.

Figure 4.2: Annotated image

- Now, the difference is calculated between the annotated image and the grayscale image using *ImageOps* function from the Pillow library. The resulting image is shown in Figure 4.3a.

(a) Difference image

(b) Binary image

Figure 4.3: The resulting image with better constrast between text and background

- Finally the binary image is produced using an *Adaptive Thresholding* function from the python OpenCV library. But thresholding step is skipped during training since it seemed to produce more noise in the translation results.

Figure 4.4: A steel type plate image with its input and output pair

The final image pair sample used for training is shown in Figure 4.4.

4.3.3 Synthetic dataset

A framework to create a synthetic dataset is introduced in this thesis to generate images close to real steel type plates. The synthetic data supplement the original dataset since it has less than 400 image pairs and helps in reducing the manual effort of annotating more images.

The Figure 4.5 shows a steel type plate template. More such templates are available for selection in the framework. The dictionary pool consists of around 750 image patches that contain different shapes of alphanumeric characters as well as word patches randomly cropped from the original images of steel type plates with its corresponding labels. Figure 4.6 shows some of the samples of cropped image patches present in the dictionary with its output pair.

Figure 4.5: A sample template image

Figure 4.6: Sample image pairs in the dictionary pool

The framework is built using python script and OpenCV libraries. At first, it randomly chooses a template of the steel plate. Once the template is selected, the framework loads the objects of the selected template class.

Figure 4.7: A synthetic image with its input and output pair

It contains information such as the location of the templates, number of cropped areas and corresponding bounding box coordinates. Then, the framework picks a random item from the dictionary and fits these into each white boxes of the template. The generated synthetic image is shown in Figure 4.7. The framework has the option to vary the illumination and brightness of the generated image. Furthermore, it also has the functionality to add noises like gaussian, speckle, salt & pepper, etc.

4.4 Evaluation metrics

The evaluation of the generative models is still an open research problem [Salimans et al., 2016]. The GAN generates realistically synthetic images and is often difficult to compare the performance of different GAN models with one another. Borji [2019] in their work, conducted a detailed study on

the evaluation of GAN. According to their research, several methods have been introduced to evaluate GAN yet there is no clear winner on which method captures the strengths and limitations of the model. The evaluation methods for GAN can be generally classified into two types: i) Quantitative evaluation and ii) Qualitative evaluation

4.4.1 Quantitative evaluation

Yang et al. [2018] used metrics such as Mean Absolute Error (MAE), Peak Signal to Noise Ratio (PSNR) to evaluate their work of image-to-image translation model. Similarly in this thesis, we have used MSE, SSIM and FID score to quantitatively measure the performance of the different GAN models. The synthetic images can be quantitatively evaluated with metrics such as MSE that calculate per-pixel-errors. This method calculates the average pixel difference between the ground truth images and the generated images. The MSE is not a great measure for performance indicators since it does not reveal anything about the quality or structural similarity of the image. Yet, higher MSE means the generated image is very different from the ground truth image. The MSE of two images x and y is given by

$$MSE(x,y) = \frac{1}{m*n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[x(m,n) - y(m,n)]^2$$

SSIM proposed by Wang et al. [2004] can be used to measure the similarity and quality of generated images with respect to the ground truth. It compares the pixels of two images using three quantities such as luminance (I), contrast (C) and structure (S). For images x and y , it is given by,

$$I(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad C(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad S(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where, μ_x, μ_y, σ_x *and* σ_y denote the mean and standard deviation of pixel intensities between two images. C_1, C_2 *and* C_3 are constants added for numerical stability. The final SSIM score [Wang et al., 2004] is given by

$$SSIM(x, y) = I(x, y)^{\alpha} C(x, y)^{\beta} S(x, y)^{\gamma}$$

where α, β *and* γ are the parameters to adjust the relative importance of I, C and S. The score ranges between 0 (Low similarity) and 1 (High similarity).

Frechet Inception Distance (FID)

The most commonly used method to evaluate the quality of generated images is to use classifiers pre-trained on real image distribution. Inception score is a measure that classifies if the synthetic images are realistic with the help of a classification network trained on real images. Heusel et al. proposed FID which is an improvised version of the Inception score. FID compares the Gaussian distribution of ground truth with generated images to quantify its quality. The FID score between real and generated images [Heusel et al., 2017] is given by

$$FID(r, g) = \|\mu_r - \mu_g\|_2^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}})$$

It is shown that the FID score is more reliable, consistent and robust than the Inception score. Lower the FID score, better the quality of generated images.

4.4.2 Qualitative evaluation

The manual image inspection is a good starting point for qualitative analysis although visual comparison of real and generated images is difficult when the changes are very small. The qualitative evaluation is not sufficient to determine the performance of the GAN. Therefore it is used together with the other metrics to evaluate the synthetic images such that it complements the quantitative evaluation. In this thesis, the synthetic image generated by three different GANs under study is visually distinguishable.

4.4.3 Evaluation of OCR

For evaluating the OCR engines, we have conducted experiments with the test images before translation as well as the corresponding GAN translated images. These images are passed separately to both the proposed OCR engine and the commercial OCR tool like Google Vision. Finally, we have two outcomes for each OCR engine: i) character recognition of images without GAN translation and ii) character recognition of images using GAN translation. Then the output is compared with the annotations and the results are tabulated. We take into measures like the number of characters present in the text, the number of characters identified and the recognition rate in the name of OCR score. The OCR score is given by the number of characters identified correctly divided by the total number of texts present in the image. The following formula is used to calculate the OCR score:

$$OCR\ score = \frac{Total\ number\ of\ characters\ identified\ correctly}{Total\ number\ of\ characters} * 100$$

Chapter 5

Results & Discussion

The results and discussion for all tests regarding both the GAN and OCR are presented in this chapter. Both the qualitative and quantitative results are provided for the evaluation of GAN models as discussed in the previous chapter. The quantitative results of GAN are discussed using the bar plots with metrics such as SSIM, FID and MSE while the qualitative results are discussed using visual results of test images from the output of the GAN generator. Then character recognition is evaluated using both the google vision API and the proposed integrated OCR engine. This chapter is concluded with the discussion of the effectiveness of GAN in terms of improvement in the character detection and recognition of the steel type plate images.

5.1 Results of GAN

Both the quantitative and qualitative results regarding the steel type plate images translated by the GAN models are presented below.

5.1.1 Quantitative results

As discussed earlier in the previous chapter, the GAN models are evaluated using three different metrics: i) FID, ii) SSIM and iii) MSE. The bar chart (Figure 5.1) below gives information about how the test images are evaluated using the FID metric. Lower FID indicates the better quality of

generated images. The results obtained from the FID evaluation showed that the pix2pix model does well among the other models. The results of pix2pix have the lowest FID score of 15.09 and outperform other models by a significant margin. The use of synthetic data produces similar kinds of results for the pix2pix model. The CycleGAN performs the worst with a score of 51.76. This can be explained by the fact that the model is trained in an unsupervised manner with fewer data points. The FactorGAN produced slightly better results compared to the CycleGAN with a score of 32.35, but not as good as pix2pix.

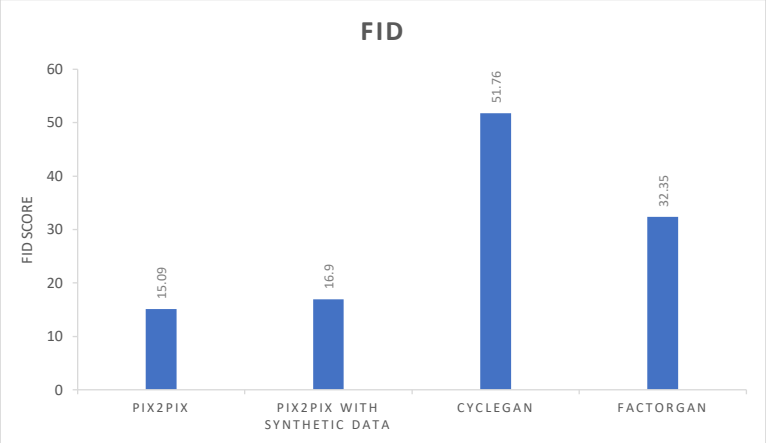


Figure 5.1: Experimental results for FID

In a search for a still more accurate result, the pix2pix model is trained with the synthetic dataset along with the real dataset. This also produced similar results as that of the pix2pix model trained only with the original image dataset. The difference between both the results were very minute as seen in the quantitative results and could not be differentiated during the qualitative evaluation. The CycleGAN and FactorGAN architecture have 4 networks each in total as discussed in Chapter 3. The time taken for training both the models is very high in general. Also, they produced the worst results compared to the pix2pix while using the original dataset. Therefore the experiments with the synthetic dataset were not conducted for both CycleGAN and FactorGAN.

The SSIM score gives the similarity between ground truth data and the output of the generator for the test images. Higher SSIM indicates better quality of generated images. The chart shown in the Figure 5.2 shows the SSIM scores evaluated for different GAN models. The results of SSIM is exactly similar to the FID evaluation. The model that has the best SSIM score is pix2pix with a similarity score of 0.8. The model trained with synthetic dataset + pix2pix produces similar results that of pix2pix with a score of 0.79. The SSIM score for FactorGAN and CycleGAN is very low compared to other models with a score of 0.74 and 0.69 respectively.

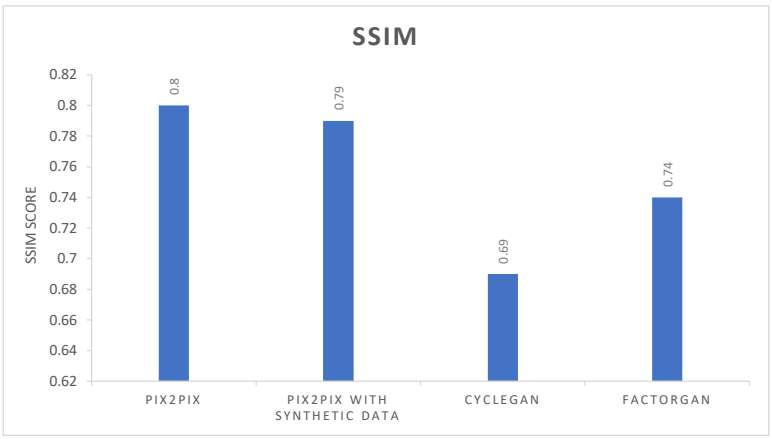


Figure 5.2: Experimental results for SSIM

The MSE is calculated by squaring the absolute difference between the pixels of ground truth and the generated image. The result of this experiment is shown in Figure 5.3. It is seen that the model trained with synthetic data with pix2pix GAN has the least error whereas the CycleGAN model with a MSE of 5739.08 performs the worst. The MSE for pix2pix does not differ much from the best model with the value of 3377.56 while the MSE of FactorGAN is 3664.83.

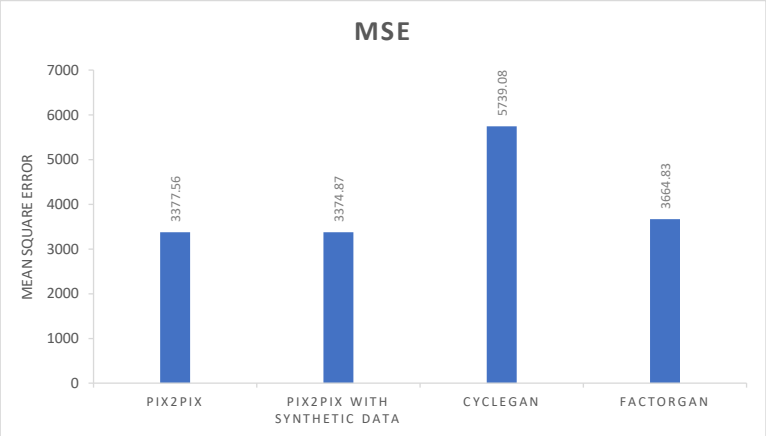


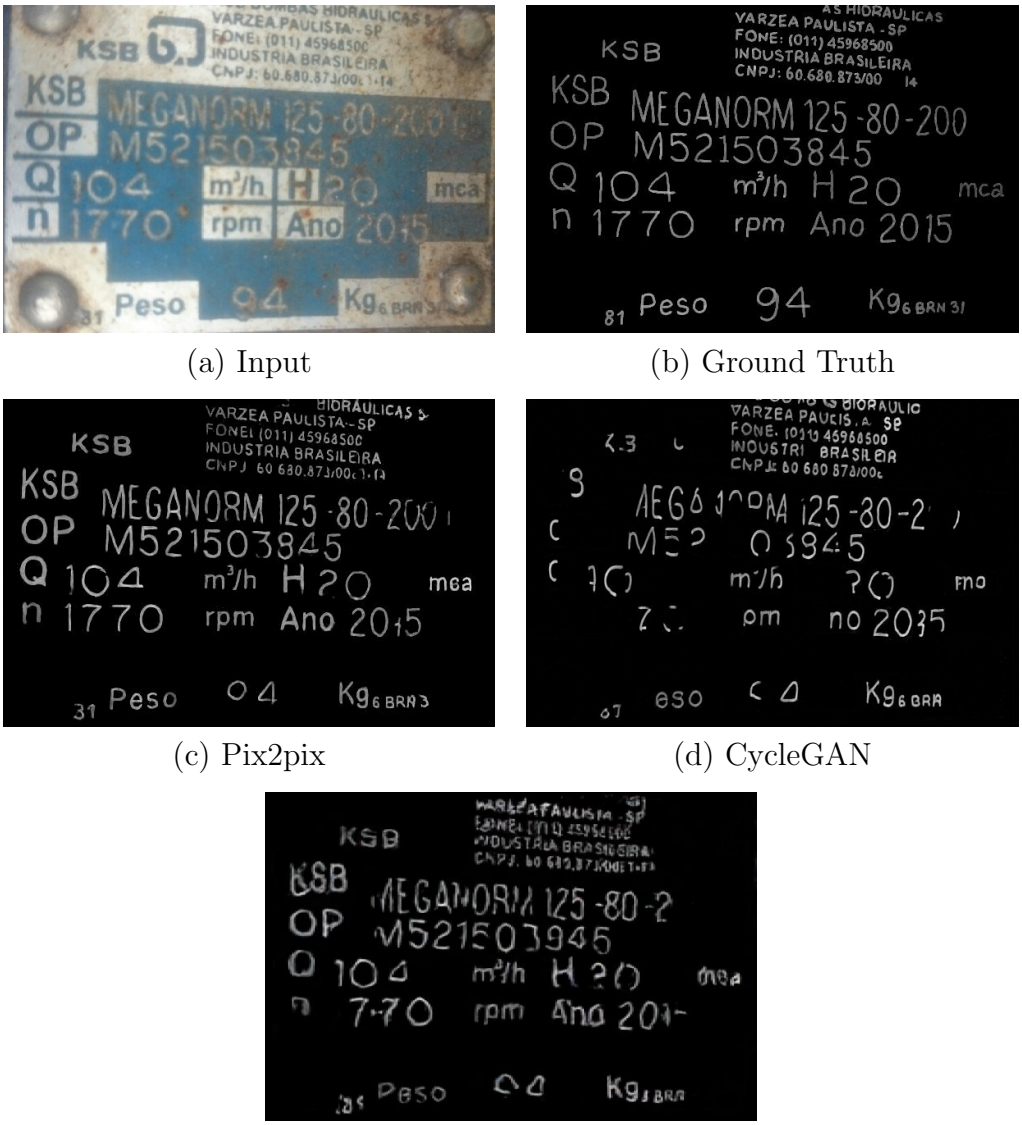
Figure 5.3: Experimental results for MSE

5.1.2 Qualitative results

The qualitative results are evaluated by manual visual inspection of translated test images for all three different GANs. Figures 5.4 - 5.6 show how different models perform when translating the image from unclear to a clearer image. Since it is not possible to display all the images, we have shown the test results of three sample images with their ground truth and corresponding translation by three different GANs.

Both the quantitative and qualitative results of the experiment can be found in the GitLab link shared in Section 4.1. It can also be accessed using the shared google drive `https://drive.google.com/drive/folders/1lMbMObjMx1LoIyi8cyV7Nz1Gy5yUisXB?usp=sharing`. The qualitative results support the findings of the quantitative results. From the results, it can be seen that the pix2pix model successfully generate images very close to ground truth images. The use of ResNet as a feature extractor combined with paired dataset and techniques such as random cropping of the input image to the size of 256 x 256 for each epoch during training contributed to the success of the pix2pix model. This complements the experiments for quantitative evaluation. The misclassified characters for pix2pix are mostly repetitive. In rare cases, the texts were incorrectly transformed or missed.

For example, the characters such as 'D', '/', 'G' in the steel plates are misclassified as '0', '7', '6' respectively.



(a) Input (b) Ground Truth

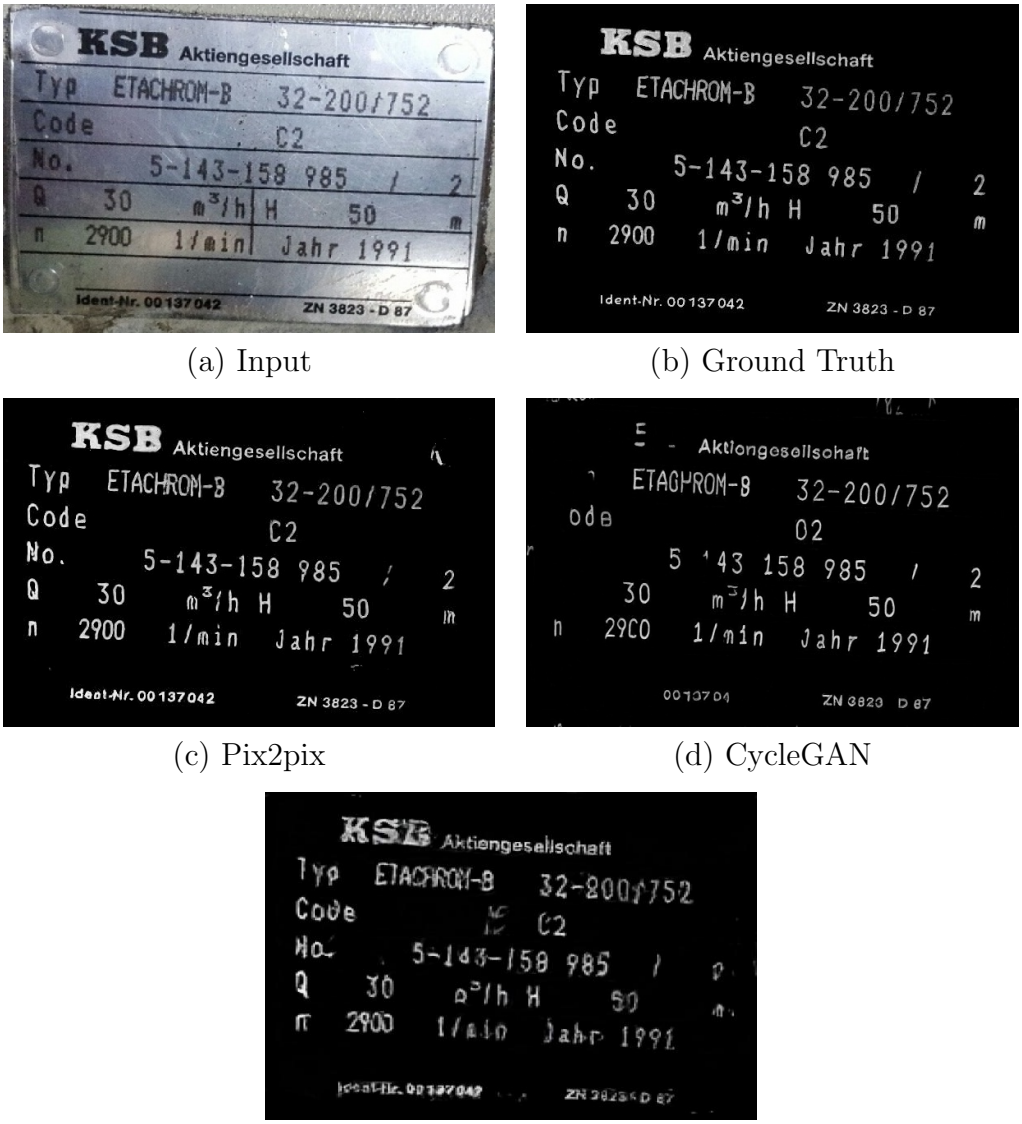
(c) Pix2pix (d) CycleGAN

(e) FactorGAN

Figure 5.4: Test result: Sample 1

The CycleGAN produces the worst results though it uses ResNet as a feature extractor. From Figures 5.4 - 5.6, it can be seen that CycleGAN misses a lot of text areas compared to the annotated images. It gives decent output

only when the text area is clear in the original image. It fails to segment the characters when the corresponding area in the input image has even a little bit of noise such as reflection or under different lighting conditions. Thus CycleGAN skips a lot of characters and could not translate the complete structure of the input image. Thus it results in huge MSE loss, worst SSIM



(a) Input (b) Ground Truth

(c) Pix2pix (d) CycleGAN

(e) FactorGAN

Figure 5.5: Test result: Sample 2

and FID score as found during the quantitative evaluation. The use of un-

supervised learning, unpaired dataset and insufficient data points are the probable reasons for the failure of CycleGAN.

(a) Input (b) Ground Truth

(c) Pix2pix (d) CycleGAN

(e) FactorGAN

Figure 5.6: Test result: Sample 3

The much expected FactorGAN did not live up to its expectations either. Even though Stoller et al.'s hypothesis of improved segmentation accuracy over CycleGAN still holds true for our experiments with steel type plates

dataset, the pix2pix GAN produces results which are far better from the results of FactorGAN. This may be because our dataset is very different from the cityscapes dataset [Cordts et al., 2016] that was used in the Stoller et al. experiments for training FactorGAN. The classes of cityscapes dataset fall under seven groups like human, vehicle, construction, etc,. Their classes have a broader boundary and include large objects whereas our steel type plate dataset mainly consists of two classes: characters and background which have a finer class boundary. The results of FactorGAN show that it captures all the necessary character segmentation areas like pix2pix but the translations are very blurry and cannot be used for character recognition. Thus the FactorGAN has only the second-best results during the quantitative evaluation.

In general, the translation becomes problematic when the font size of the text present in the images is very small or very big. Also, it was noted in the results that embossed texts were not correctly transformed in many test cases. This can be due to the fact that the images in our dataset had more samples of engraved texts than embossed texts.

From both the quantitative and qualitative evaluation, the pix2pix model stands out as the best model. So the final proposed framework consists of a pix2pix model for translating the noisy steel plates to clear images.

5.2 Results of OCR

For evaluating the efficiency of the use of GAN for character recognition, we randomly cropped the 30 test images to the size 256 x 256 and translated it using pix2pix GAN. Both the original image and its GAN translated image is passed into the OCR engine for character recognition. In this thesis we experimented with two different OCR engines: i) The commercial OCR engine using Google Vision API ii) The proposed CRAFT based OCR engine. The results of the experiments are tabulated in Tables 5.1 - 5.3.

5.2.1 Evaluation using Google Vision OCR

Table 5.1 compares the character recognition before and after the use of the GAN translated image using Google Vision OCR. The table presents the total number of characters present in the test images, the number of characters recognized by the Google Vision OCR engine and the average OCR Score. The number of characters recognized is further split into characters correctly identified, incorrectly identified and missed.

	Total number of characters	Characters recognized	Correctly identified	Incorrectly identified	Characters missed	OCR Score
Character recognition without GAN	1303	1054	914	140	249	67.06
Character recognition with GAN	1303	1287	1104	183	16	84.09

Table 5.1: Character recognition with and without GAN using Google Vision OCR engine

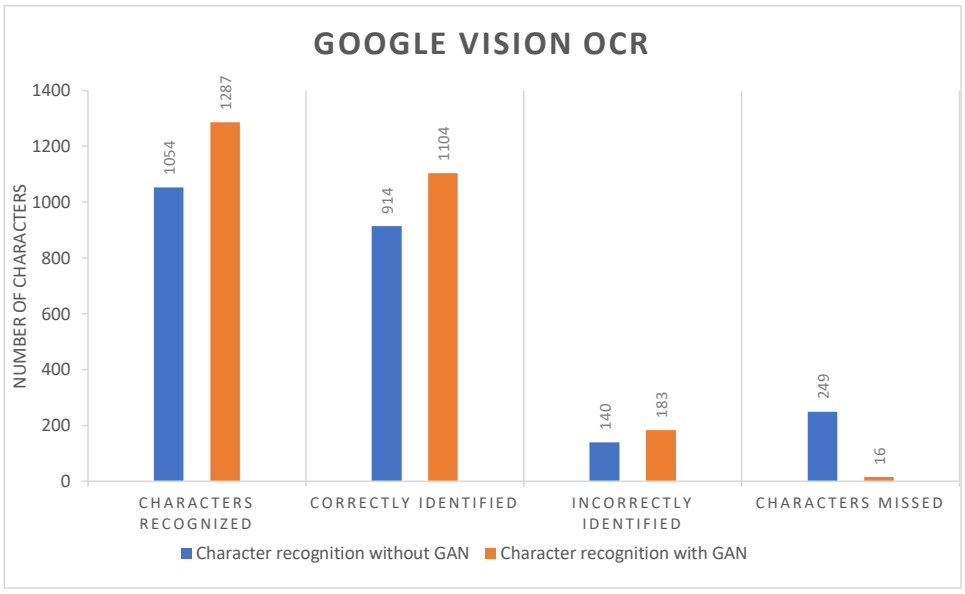


Figure 5.7: Evaluation of character recognition using Google Vision

The total number of characters present in the test images is 1303. The total number of characters recognized for original images is 1054 whereas the

total number of characters recognized significantly increased to 1287 when using their corresponding GAN translated images. Also, the characters missed are drastically reduced from 249 to 16 when using GAN translated images. The results in the above table are plotted using a bar chart in Figure 5.7.

5.2.2 Evaluation using the proposed OCR

The same experiment is repeated for the proposed OCR engine with the same test images for evaluating the efficiency of the use of GAN in character recognition. The results are tabulated in Table 5.2. It compares the character recognition before and after the use of GAN translated image using the proposed OCR. Similar to Table 5.1, this table presents the total number of characters present in the test images, the number of characters recognized by the proposed OCR engine and its average OCR score.

	Total number of characters	**Characters recognized**	**Correctly identified**	**Incorrectly identified**	**Characters missed**	**OCR score**
Character recognition without GAN	1303	1036	830	206	267	61.78
Character recognition with GAN	1303	1267	1094	173	36	84.37

Table 5.2: Character recognition with and without GAN using the proposed OCR engine

In case of proposed OCR, the character recognition is increased from 1036 to 1267 when using GAN translated images. Also, the characters missed are reduced from 267 to 36 when using GAN translated images. The results in the above table are plotted using a bar chart in Figure 5.7.

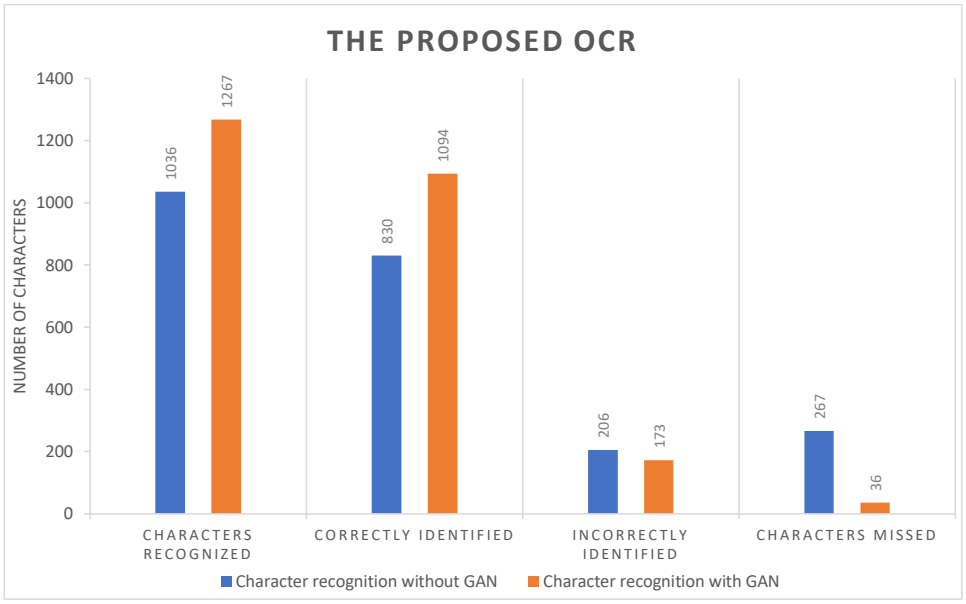


Figure 5.8: Evaluation of character recognition using the proposed OCR engine

Tables 5.1 and 5.2 show that the average OCR score for Google Vision OCR is increased from 67.06 % to 84.09 % while the average OCR score is massively increased from 61.78 % to 84.37 % when using the GAN translated image.

5.2.3 Comparison of OCR engines

This section compares the impact of image-to-image translation GAN on both the OCR engine in terms of improvement of character recognition. Table 5.3 presents the comparison of both the OCR engines in terms of improvement in character recognition when using GAN.

As discussed above in the tables 5.1 & 5.2, when using GAN translated image, the character recognition i.e. the detection of the number of characters in the steel type plates is improved by 20.09 % for the proposed OCR. Even for the commercial OCR tool like Google Vision, the character recognition of steel type plates is improved by 20.16 %.

OCR engine	Improvement of character recognition using GAN translated Image	Percentage increase in OCR Score
The proposed OCR	20.09	36.57
Google Vision	20.16	25.39

Table 5.3: Comparison of impact of GAN on both OCR engines

Similarly, the OCR score for Google Vision is improved by 25.39 % while for the proposed OCR, it has been massively increased by 36.57 %. This is graphically represented using a bar chart in Figure 5.9.

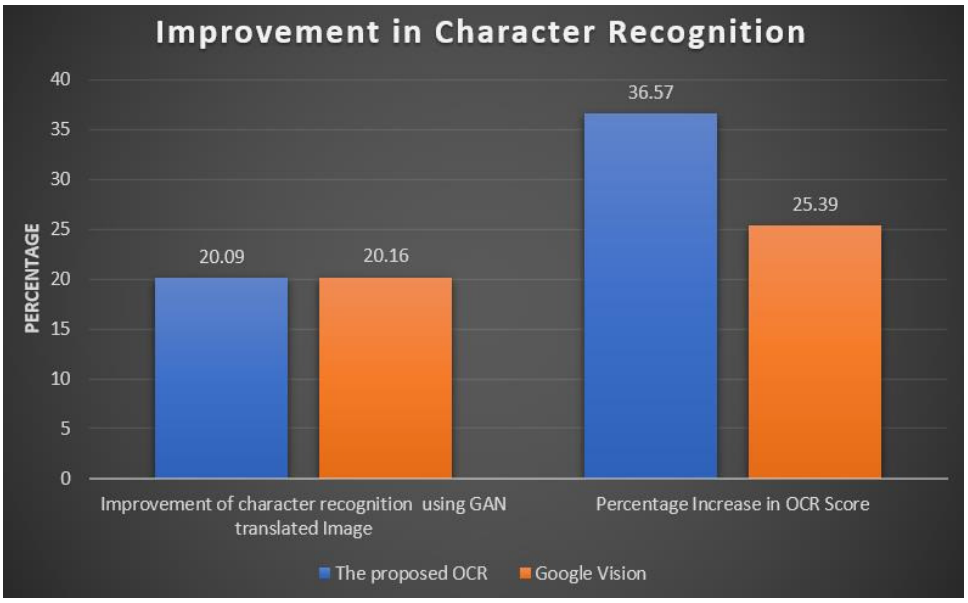


Figure 5.9: Comparison of impact of GAN on both OCR engines

Chapter 6

Conclusion

This chapter concludes the thesis with the contributions of the work with final comments on the results from the previous chapter. Also, we have addressed the limitations and the scope for future work.

6.1 Summary

The goal of the thesis is to investigate the use of the image-to-image translation GANs in improving the character recognition accuracy of steel type plate images. A framework has been introduced in this thesis to create a synthetic dataset of steel type plates along with its label pair to supplement the original dataset. We have studied three different image-to-image translation GAN approaches for translating the images of unclear steel type plates into clear images. These GAN models are evaluated both using the quantitative and qualitative metrics. After evaluation, it is found that the pix2pix generator works best. Thus the overall framework consists of pix2pix generator integrated with the OCR engine. The following are the answers to the research questions stated in this thesis:

1. *Can an image-to-image translation GAN be used to improve the accuracy of character recognition for the images of weathered steel type plates?*

Yes. From the experimental results of the thesis, it has been proved that image-to-image translation GAN models can be used for improving the accuracy of the OCR engines. For even the benchmark commercial OCR engine like Google Vision, our model has improved the character recognition of steel type plates by 25.39 % measured in terms of OCR score. Similarly, for the proposed OCR engine, the score has been massively increased by almost 37 % after using GAN.

2. *Can the findings of Zhu et al. and Stoller et al. be used to make up for the few image pairs in the steel type plate dataset?*

   No, the experimental results show that the techniques of Zhu et al.'s unpaired dataset and Stoller et al. training incomplete data points did not work well for our steel type plate dataset. Generally, the success of any deep neural networks depends heavily on the size of the dataset used for training. Our training dataset has a size of 366 image pairs. To tackle this problem of the dataset with fewer data points, various techniques such as data augmentation, random cropping of the input to a smaller size (256 x 256) than the load size (600 x 400) for each epoch during training are employed. Besides, a framework to generate a synthetic dataset has been introduced to supplement the original dataset. Even though the number of image pairs present in the training set was sufficient enough to train the image-to-image GAN models successfully, more image pairs can still produce a more robust generator model. Notably, the pix2pix model without any data augmentation techniques produced splendid results just with approximately 150 - 200 image pairs.

3. *How to evaluate the three different image-to-image translation GAN models under the scope of the thesis and choose the optimal model for the proposed steel type plate recognition system?*

   We have chosen the best GAN using both quantitative and qualitative evaluations. Metrics such as FID, SSIM and MSE are used for the quantitative measure while a manual inspection is used for qualitative

measure. From the experimental results, the optimal GAN model was found out to be pix2pix.

4. *How good is the proposed OCR engine compared to a commercial OCR engine like Google Vision? How can the performance of the overall framework be measured?*

   We have compared the proposed OCR engine with one of the benchmark OCR engines like Google Vision. We have introduced a metric called OCR score for measuring the performance of the OCR engine. It is found that character recognition is improved by 20 % for both OCR engines when using the translated images. Likewise, after using GAN, the OCR score for Google vision OCR has been increased from 67 % to 84 % and for the proposed OCR engine, it is increased from 62 % to 84 %.

6.2 Limitations

The main limitation of training any image-to-image translation model is the availability of data pairs. In our thesis, we used a small dataset of less than 400 image pairs. The current practice of manually annotating the images and creating a label pair is more time consuming and costly. Even the framework for generating synthetic steel type plate images is limited to the dictionary pool of manually cropped patches. Also, manually generated labels lack uniformity and have a high human bias.

6.3 Future works

Till now, there has been only a little work done in using GANs for OCR. The use of GANs in OCR has great potential and has scope for several promising applications like our steel type plate recognition. As future work, the overall system can be modified such that the model learns the loss function based on the output of the OCR engine directly as a combined loss instead of training image-to-image translation and OCR engine separately.

Bibliography

AICompare (2020). Optical Character Recognition (OCR) : Which solution to choose? `https://medium.com/@aicompare/optical-character-recognition-ocr-which-solution-to-choose-cd4f829c4e5`. "Online; accessed 07-June-2020".

alexlenail (2018). Publication-ready nn-architecture schematics. `http://alexlenail.me/NN-SVG/index.html`. "Online; accessed 09-July-2020".

Almahairi, A., Rajeswar, S., Sordoni, A., Bachman, P., and Courville, A. (2018). Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*.

Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S. J., and Lee, H. (2019a). What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723.

Baek, Y., Lee, B., Han, D., Yun, S., and Lee, H. (2019b). Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374.

Borji, A. (2019). Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Bui, Q. A., Mollard, D., and Tabbone, S. (2017). Selecting automatically preprocessing methods to improve ocr performances. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 169–174. IEEE.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223.

Dash, A., Gamboa, J. C. B., Ahmed, S., Liwicki, M., and Afzal, M. Z. (2017). Tac-gan-text conditioned auxiliary classifier generative adversarial network. *arXiv preprint arXiv:1703.06412*.

Duan, T. D., Du, T. H., Phuoc, T. V., and Hoang, N. V. (2005). Building an automatic vehicle license plate recognition system. In *Proc. Int. Conf. Comput. Sci. RIVF*, volume 1, pages 59–63. Citeseer.

Fiore, U., De Santis, A., Perla, F., Zanetti, P., and Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479:448–455.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

nets. In *Advances in neural information processing systems*, pages 2672–2680.

Google (2016). Cloud Vision documentation. `https://cloud.google.com/vision/docs`. "Online; accessed 09-July-2020".

Gupta, A., Vedaldi, A., and Zisserman, A. (2016). Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Han, T. and Hickman, A. (2019). Our Search for the Best OCR Tool, and What We Found. `https://source.opennews.org/articles/so-many-ocr-options/`. "Online; accessed 09-July-2020".

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hosangadi, R., Adiga, D., and Vyeth, V. (2019). Ocr-friendly image synthesis using generative adversarial networks. In *Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition*, pages 226–234.

Hu, W., Su, Y., and Li, J. (2019). Embossed characters enhancement based on convolutional neural network. In *Proceedings of the 2019 International Conference on Artificial Intelligence and Advanced Manufacturing*, pages 1–5.

ICDAR15 (2015). ICDAR 2015 dataset. https://rrc.cvc.uab.es/. "Online; accessed 15-June-2020".

Impedovo, S. (2012). *Fundamentals in handwriting recognition*, volume 124. Springer Science & Business Media.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jadon, S. (2018). Introduction to Different Activation Functions for Deep Learning. https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092. "Online; accessed 07-June-2020".

Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., and Fang, Z. (2017). Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer.

Karimi, M., Veni, G., and Yu, Y.-Y. (2020). Illegible text to readable text: An image-to-image transformation using conditional sliced wasserstein adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 552–553.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., and Menotti, D. (2018). A robust real-time automatic license plate recognition based on the yolo detector. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE.

Lat, A. and Jawahar, C. (2018). Enhancing ocr accuracy with super resolution. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3162–3167. IEEE.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, H. (2019). The CRAFT process. `https://www.youtube.com/watch?v=HI8MzpY8KMI`. "Online; accessed 07-June-2020".

Li, C. and Wand, M. (2016). Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer.

Mathworks (2020). Train Conditional Generative Adversarial Network (CGAN). `https://de.mathworks.com/help/deeplearning/ug/train-conditional-generative-adversarial-network.html`. "Online; accessed 07-June-2020".

Milewski, R. and Govindaraju, V. (2006). Extraction of handwritten text from carbon copy medical form images. In *International Workshop on Document Analysis Systems*, pages 106–116. Springer.

Mittal, A. (2019). Introduction to u-net and res-net for image segmentation. `https://towardsdatascience.com/introduction-to-u-net-and-res-net-for-image-segmentation-9afcb432ee2f`. "Online; accessed 09-June-2020".

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Moutarde, F. (2018). convnet-notebook.html. `http://perso.mines-paristech.fr/fabien.moutarde/ES_MachineLearning/TP_convNets/convnet-notebook.html`. "Online; accessed 07-June-2020".

MSRA-TD500 (2012). MSRA-Text Detection 500 dataset. `http://www.iapr-tc11.org/mediawiki/index.php/MSRA_Text_Detection_500_Database_(MSRA-TD500)`. "Online; accessed 15-June-2020".

Nagaoka, Y., Miyazaki, T., Sugaya, Y., and Omachi, S. (2017). Text detection by faster r-cnn with multiple region proposal networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 6, pages 15–20. IEEE.

Novák, V., Hurtík, P., and Habiballa, H. (2013). Recognition of distorted characters printed on metal using fuzzy logic methods. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pages 733–738. IEEE.

Patil, A. V. and Dhanvijay, M. M. (2015). Engraved character recognition using computer vision to recognize engine and chassis numbers: Computer vision technique to identify engraved numbers. In *2015 International Conference on Information Processing (ICIP)*, pages 151–154. IEEE.

Raka, P. and Agrawal, S. P. (2019). Ocr to read embossed text from credit/debit card.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016a). Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Reed, S. E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., and Lee, H. (2016b). Learning what and where to draw. In *Advances in neural information processing systems*, pages 217–225.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Roy, R. (2020). AI, ML, and DL: How not to get them mixed! `https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c`. "Online; accessed 19-July-2020".

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`. "Online; accessed 07-June-2020".

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Saxena, S. (2017). Artificial Neuron Networks(Basics) — Introduction to Neural Networks. `http://becominghuman.ai/artificial-neuron-networks-basics-introduction-to-neural-networks-3082f1dcca8c`. "Online; accessed 07-June-2020".

Shi, B., Bai, X., and Belongie, S. (2017). Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2550–2558.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stoller, D. (2020). Architecture of Factor GAN. `https://github.com/f90/FactorGAN`. "Online; accessed 07-June-2020".

Stoller, D., Ewert, S., and Dixon, S. (2019). Training generative adversarial networks from incomplete observations using factorised discriminators. *arXiv preprint arXiv:1905.12660*.

Tian, Z., Huang, W., He, T., He, P., and Qiao, Y. (2016). Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer.

Tian, Z., Shu, M., Lyu, P., Li, R., Zhou, C., Shen, X., and Jia, J. (2019). Learning shape-aware embedding for scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4234–4243.

Tripathy, S., Kannala, J., and Rahtu, E. (2018). Learning image-to-image translation using paired and unpaired training samples. In *Asian Conference on Computer Vision*, pages 51–66. Springer.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Vallecorsa, S. (2017). Detector Simulation and Machine Learning . `https://indico.cern.ch/event/659928/contributions/2691911/attachments/1508729/2351970/DetSim_MachineLearning.pdf`. "Slide no 13; accessed 07-June-2020".

Vo, A. (2018). Deep Learning – Computer Vision and Convolutional Neural Networks. `https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/`. "Online; accessed 07-June-2020".

Wang, W., Xie, E., Li, X., Hou, W., Lu, T., Yu, G., and Shao, S. (2019a). Shape robust text detection with progressive scale expansion network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9336–9345.

Wang, X., Jiang, Y., Luo, Z., Liu, C.-L., Choi, H., and Kim, S. (2019b). Arbitrary shape scene text detection with adaptive text region representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6449–6458.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Xiang, Z., You, Z., Qian, M., Zhang, J., and Hu, X. (2018). Metal stamping character recognition algorithm based on multi-directional illumination image fusion enhancement technology. *EURASIP Journal on Image and Video Processing*, 2018(1):80.

Yang, Q., Li, N., Zhao, Z., Fan, X., Chang, E. I., Xu, Y., et al. (2018). Mri cross-modality neuroimage-to-neuroimage translation. *arXiv preprint arXiv:1801.06940*.

YU, W., WU, Z., LIU, H., CHEN, Q., DUAN, X., MO, J., TONG, J., LIAO, F., and LIN, Q. (2017). An engraving character recognition system based on machine vision. *DEStech Transactions on Computer Science and Engineering*, (aiea).

Yuan, T., Zhu, Z., Xu, K., Li, C., Mu, T., and Hu, S. (2019). A large chinese text dataset in the wild. *Journal of Computer Science and Technology*, 34(3):509–521.

Zang, D., Chai, Z., Zhang, J., Zhang, D., and Cheng, J. (2015). Vehicle license plate recognition using visual attention model and deep learning. *Journal of Electronic Imaging*, 24(3):033001.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915.

Zhong, Z., Sun, L., and Huo, Q. (2019). An anchor-free region proposal network for faster r-cnn-based text detection approaches. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(3):315–327.

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., and Liang, J. (2017). East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.