HANDWRITING DIGIT RECOGNITION

BSc Hons – Computing Science

ZHIJIAN DAI,2537085

Table of Contents

List of Figures

List of Tables

Abstract

This paper is research on the offline handwritten digit recognition (HDR). HDR is part of the computer vision research, because it is a challenging computational problem mainly due to different handwritten style, image quality or scan quality and other associated with the handwritten pattern problem.

This project is aims to using Recurrent Neural Network combine with different handwritten digit dataset to find out the best dataset for training. To increase the difficulty will be introduce data augmentation. Which increase the data amount for training accuracy.

Also, we will analysis both Concurrent Neural Network and Recurrent Neural Network, with deep review of pros and cons for both neural networks. With explanation of data augmentation and their technique. And review the recent paper, analysis their result and algorithms. Also, will have a brief look at the 'Feature Extraction' technique. Which it will be use in the future improvement.

To achieve the goal of this project, we will use testing set from the dataset to generate the test accuracy, later will be using human handwritten digit and G-Capcha digit for testing the best model, and analysis the bad example and explain the reason why it predicts the wrong result. Also showing the good case example.

At the end will be conclude the experiment result and giving some future improvement detail.

Attestation

I understand the nature of plagiarism, and I am noticing the University's academic misconduct policy.

This dissertation reports is original work by me during university project.

Signature: ZHIJIAN,DAI Date: 16/04/2021

Acknowledgements

I take this opportunity to express sincere gratitude to my supervisor, Dr.Mohamed eelsayed Elawdy, for his support, invaluable advice and detail feedback for my project. Thank you for being a great supervisor and your guidance throughout this project.

1.Introduction

1.1Background and Context

Handwritten character recognition can be divided into offline handwritten character recognition and online handwritten character recognition due to different data collection methods and the purpose of the applications.

- The handwritten text processed by the **online handwritten character recognition** is a text signal obtained by the writer using a physical device (such as a digital pen, a digital handwriting pad, or a touch screen) to write onsite, and the written trajectory is instantly loaded into the computer through timing sampling.
- The handwritten text processed by **offline handwritten text recognition** is a two-dimensional picture of the handwritten text collected by image capture devices such as a scanner or a camera.

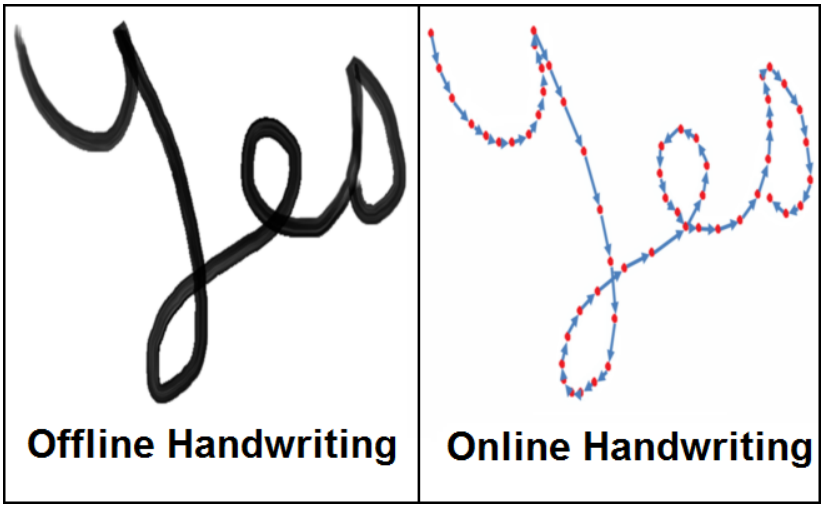


Figure 1 Online/Offline handwriting (Sources: https://www.kaggle.com/c/icdar2013-stroke-recovery-from-offline-data/overview/description)

And there are some challenges for online/offline handwriting recognition. Here a short discussion is given. More technical details will be explained later.

- Due to the different recognition objects, the methods and strategies adopted by these two types of handwriting recognition technologies are also different. The recognition object of the former is a series of sampling point information arranged in chronological order, while the latter is the two-dimensional pixel information that loses the writing stroke order information.

- Due to the fact that there is no stroke order information, and because the camera scanning equipment is in different lighting, resolution, writing under conditions such as paper, digitization will bring certain noise interference. Generally speaking, offline handwritten text recognition is more difficult than online handwritten text recognition.

1.2 What is Handwritten Recognition

This chapter will discuss what is 'Handwritten Recognition', and how different set can affect Machine Learning result. Also, it will provide a depth overview of the recent literature relate to this study. Firstly, the part of this section presents an overview with references to the approaches to Handwriting Recognition and the template matching Machine Learning (ML) techniques. Second part will analysis of the factors which affect the recognition accuracy rate (Train & Test).

So why we need digit/character recognition?

- As it turns out, it always saves time, if program can automatically extract characters and digits from papers, whenever possible. As a result, there are more and more research efforts that focusing on using personal device such as laptop or mobile phone to scan paper handwriting transfer to digital format such as "Word" document.
- However, this leads to the next issue. The poor quality of handwriting is the main problem for low accuracy handwritten recognition.

For example, Bank Cheques, which are still widely used in bank. Different people have different handwriting style, which made people hard to read. "As many countries use cheque truncation systems (CTS) nowadays, much time, effort and money can be saved if this entire process of recognition, verification and data entry is done automatically using images of cheques. An attempt is made in this paper to present the state of the art in automatic processing of handwritten cheque images." (R. Jayadevan · S. R. Kolhe · P. M. Patil · U. Pal, 2011).(Figure 2)

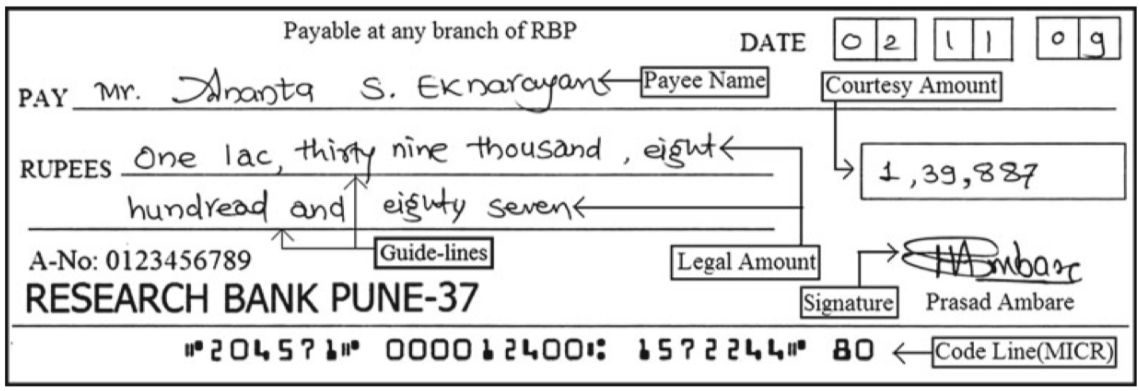


Figure 2 A typical writing example for bank cheques (Sources: Jayadevan, R., Kolhe, S.R., Patil, P.M. and Pal, U., 2012. Automatic processing of handwritten bank cheque images: a survey. International Journal on Document Analysis and Recognition (IJDAR), 15(4), pp.2667-296.)

1.3 General steps for handwritten recognition

As the result, handwritten recognition can be divided to four steps,

1. First step is record handwritten by scanner or camera.
2. Second step is to conduct image segmentation or image cropping to single characters or number.
3. Third, import image to training model getting result.

4. Last step, output result could be class or output long string.

We will revisit the whole process later in methodology chapter.

1.4 Real word applications for handwritten recognition

- Google Handwriting Input (Figure 3)
    - Directly fetch your handwriting on screen and now is able to translate all kind of English writing with the state of art RNN based algorithms with attention models. Although it needs time to learn what human writes but the average waiting the is short, even with the sloppiest scrawl.
    - Working in an online learning style. it allows easy input of ideographic lettering and voice and recognizes emoji-style drawings.

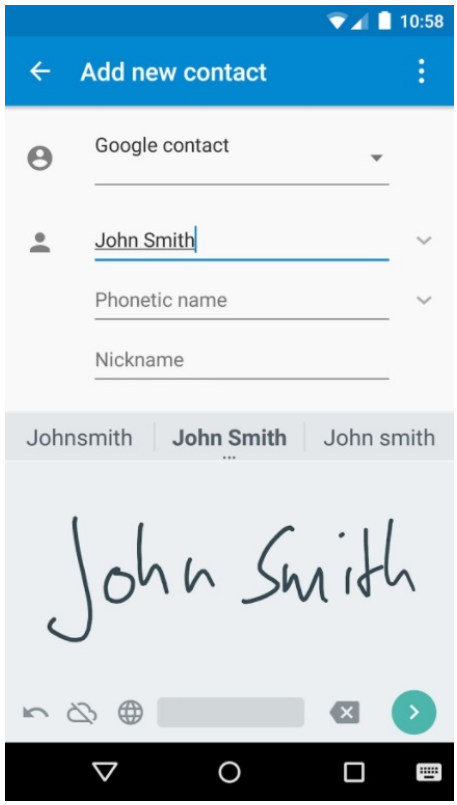


Figure 3 Interface of google Handwriting Input (Source: https://www.businessnewsdaily.com/10467-best-handwriting-recognition-apps.html)

- Healthcare system
    - One good example at this point is the pharmaceuticals, where recognition of prescription is the primary focus. Just consider the thousands of hand-written scripts every day but some of them are not clear enough to be recognized.
    - In this case, one of the effective ways is to ask machine to help instead. In addition, patient enrollment and form digitalization are also critical and can be helped with the handwriting technique.

- Online Libraries

- By now lots of e-book has been generated thanks to the help of database. However, most of them requires human efforts to sort out, which is not efficient.
- With the help of hand recognition, we are able to automatically pick out the book by the handwritten title, which is more time efficient compared with human beings.

1.5 Challenges for handwriting

1. It is always possible to receive task to recognize unclear handwriting, which itself is a challenge
2. Even if for the same person, the writing style may change over time.
3. The quality of document may be affected by the collection method, which may result in unclear scanning or images.
4. Human may not write sentence in a straight line, as opposed to printing machine.
5. Human may even write in variant rotations.
6. Lastly, it is not easy to collect a well labelled dataset, for the reason mentioned above. It turns out to be another challenge.

1.6 Scope and Objects

The projects main goals were:

1. Having a basic recognition system can recognize print digit number
2. Using different dataset for further training
3. Apply Pytorch.torchvision for data augmentation
4. Using Pytorch.torchvision find out the best model

1.7 Report Organization

This report will be divided to four chapter,

1. Chapter 1 explain the challenge of the Handwritten Recognition and the application of Handwritten Recognition.
2. Chapter 2 will analysis the relate work from the dataset that have been used by literature, explain the related algorithms and explain the figure and result about related algorithms. Also conclude the Pros and Cons of related algorithms.
3. In chapter 3 gone talk about the method my project is using,
    a. For example, the Neural Network, Dataset and Data Augmentation.
    b. Also explain the parameter using in this project.
    c. Then giving the quantitative and qualitative result and explain the bad example and good example.
4. The last chapter will be summary of my work and what can we do next.

2. Literature Review and Related Work

2.1 Introduction

By far handwriting recognition receives increasingly attention and becomes one hot topic for researcher to investigate. In recent year, deep learning methods turn out to be the mainstream. In this chapter, we will give a brief review for methodologies applied in the field of handwriting recognition.

2.2 Traditional method

Traditional method prefers time series-based model. One of the typical examples is Hidden Markov Models (HMM). Although support vector machine (SVM) is also preferred in the context. And here is one example for how traditional method works with Hidden Markov Models.

1. Conduct preprocessing for the raw text input.
2. Perform feature extraction to gather critical information. Such as character size, inflection points etc.
3. Train Hidden Markov Models with all given features.

The traditional methods can work on small size dataset and converge in tiny time. However, it is not so scalable and thus is not able to extend the similar method on large dataset. In the meantime, the feature extracted turns out to be the key factor for the training of HMM model. In light of all those factors, it may be safe to conclude that we still need to seek for more compact method on larger dataset (for example, MNIST).

2.3 Review for deep learning-based method

Here we provide a comprehensive review for deep learning-based method. By now both convolutional neural network and deep learning based neural network have been successfully applied to Handwritten recognition problem.

Thus, we will give a review in detail for convolutional neural network (CNN) and recurrent neural network (RNN) as the complementary material. Afterwards we will cover review for the state of art methods in handwriting recognition and general training tricks that may involve in training a high-performance handwriting recognition model.

There are drawbacks to RNNs.

- It fails to store information for a longer period of time. At times, a reference to certain information stored quite a long time ago is required to predict the current output. But RNNs are absolutely incapable of handling such “long-term dependencies”.
- There is no finer control over which part of the context needs to be carried forward and how much of the past needs to be ‘forgotten’. Other issues with RNNs are exploding and vanishing gradients (explained later) which occur during the training process of a network through backtracking. [11]

To the other side, LSTM (Long Short-Term Memory) networks are a special type of RNN (Recurrent Neural Network) that is structured to remember and predict based on long-term dependencies that are trained with time-series data. With LSTMs, there is no need to keep a finite number of states from beforehand as required in the hidden Markov model (HMM). LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage.

The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got the only single neural net layer of tanh, LSTMs comprises of three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where '0' means 'reject all' and '1' means 'include all'.

2.3.1 Review for Convolutional Neural Network (CNN)

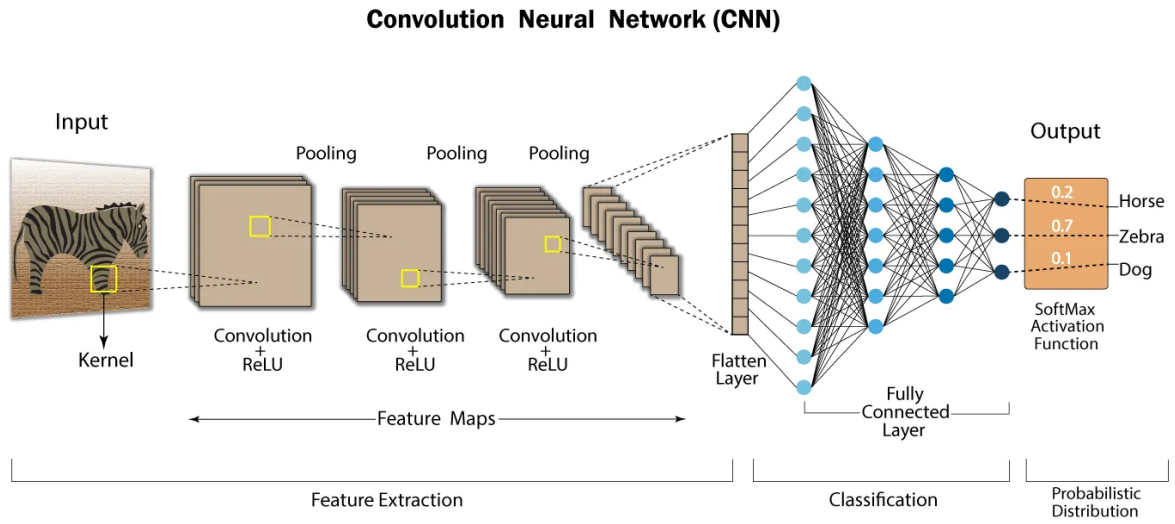


Figure 4 Convolution Neural Network Example (Sources: https://developersbreach.com/convolution-neural-network-deep-learning/)

A standard convolutional neural network composes of a series of convolutional operator. It is similar as the one in dense connection, being make of tons of neural with activation function to add non-linearity. The convolutional layer works with a set of filters to conduct convolution operation. (Figure 4)

So why we need convolutional layer? The reason is due to the property of the image. Pure dense layers are not able to take care of full images as the total parameters are too large to take care by computation units. As it turns out, too many parameters may lead to potentially overfitting issue. [12]

A typical convolutional neural network has following components: convolutional layer, pooling layer and fully connected layer. A combination of those turns out to be a complete structure of the convolutional neural network. Just take a simple convolutional neural network as an example,

- Input layer: A place holder for input image data (can be other sequence that matches input specification), usually in three dimensions.
- Conv layer: conduct dot production for input data and layer weights. Depends on the number of filters, the output channel may change.
- Activation Layer: Activation layer introduce non-linearity to the model. Some of the examples are Tanh, Relu, leaky relu, sigmoid and etc.

- Pooling layer: Perform down sampling to the feature map and gather refined result based on original feature map. The produced result will shrink by stride.
- Fully connected layer: Summarize the feature learned from convolutional layer and compute the class scores.
- Amongst all layers, input, Relu and pooling layer has no parameters. However, conv and fully connected layer has additional parameters.

And here we discuss convolutional layer in detail. Convolutional layer works with a set of learnable parameters with tiny spatially. During the forward computation step, each filter will slide forward by a give stride across the dimension of the feature map (direction of height and width). By doing this, the network is able to learn edge feature, colour in particular regions and some orientation in lower layer, but the complete pattern of the object (such as car) in the higher level. To measure how good a feature can be learned, a receptive field is given to measure how different neural can connect each other and to what extent. Normally this is similar as the size of the filter in height and width dimension. However, the receptive field will always cover the whole channel.

In the meantime, as we move forward kernel, we may shrink the resolution of the feature map, which highly affect spatial arrangement. To this end, zero padding is required to help control the spatial size of the generated feature map. Given the input size W, the receptive field size for the current conv layer F (similar as kernel size) and the stride that you use to slide on the feature map S together with the zero padding you want to use P, we are able to compute the size for the resulting output feature map, which are,

$$Out\_feature\_map = \frac{W - F + 2 * P}{S} + 1$$

Here we use an example to illustrate the idea, given an input feature map of 7*7 and we have a 3*3 filter with a stride 1 and pad 0, we will get a 5*5 output instead.

Lastly, we discuss the difference of the layer pattern between ConvNet and dense layer. The pattern for conv layer is basically a stacking of conv and ReLU layers and sometimes may follow pooling layers to reduce feature map. The common practice is to repeat those blocks twice or three times. Directly invoke fully connected layer will result in a linear activation, which is not desired. In the meantime, it is more preferable to stack multiple blocks of convolution with small kernel size, as compared with one with single large kernel size instead. As those with small kernel size may result in more receptive field as compared with the one that has large kernel size.

2.3.2 Review for Convolutional Neural Network-based model

In LeNet, a seven-layer convolutional neural network is introduced for image classification. A series of convolution, pooling and activation layer is introduced for feature extraction purpose. Two dense layer follows to summarize information flow at the end. (Figure 5)

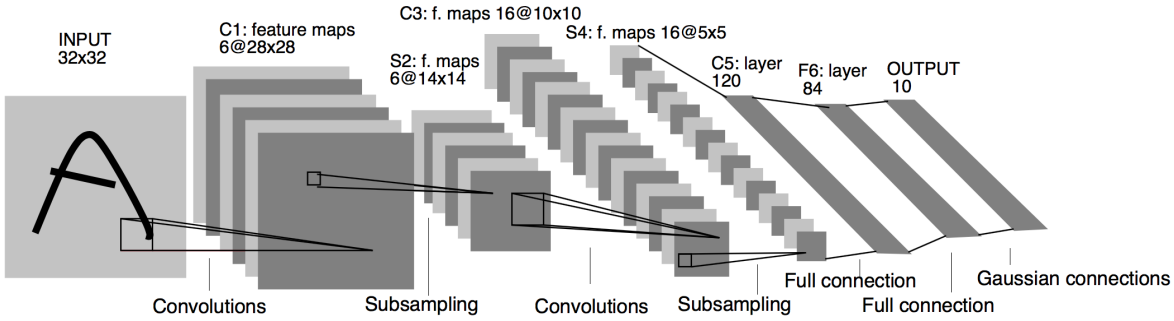


Figure 5 Example of CNN - LeNet (Sources: https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4)

In AlexNet, a stack of convolutional neural model is proposed for image net classification. Relu activation, local layer normalization and dropout operation is introduced for advanced classification performance. Alex net reaches top one with error rate of 37.5%, which beats state of art to a large margin on LSVRC-2010. (Figure 6)

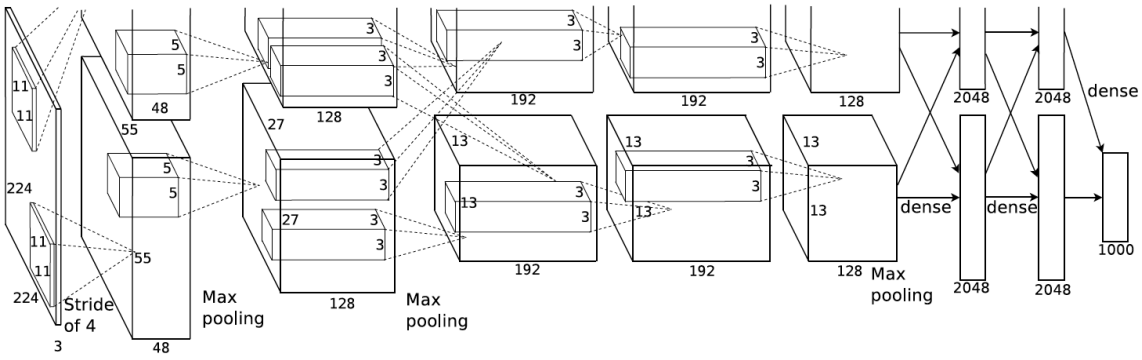


Figure 6 CNN - AlexNet (Sources:https://medium.com/swlh/alexnet-with-tensorflow-46f366559ce8)

In VGG net, a stacking of conv-pooling-activation blocks have been introduced to further boost the performance on image classification. A stacking of four blocks have been introduced to VGG 16 model together with a connection of two dense layers. VGG net proves the possibility for the performance improvement simply by adding more structure to the end of the deep neural network. One thing to note is, the whole model is tiny and clean, make it easier to reproduce and enhance. (Figure 7)

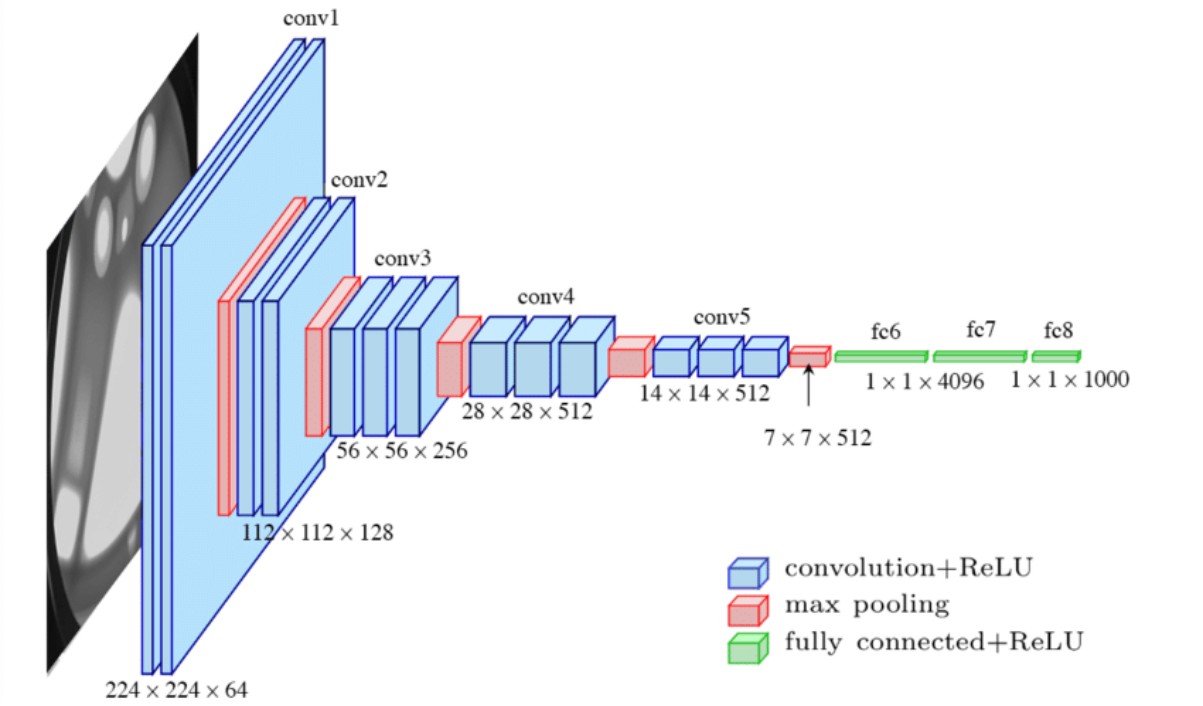


Figure 7 VGG net (Sources: https://www.researchgate.net/figure/Fig-A1-The-standard-VGG-16-network-architecture-as-proposed-in-32-Note-that-only_fig3_322512435)

In Resnet, residual connection is introduced to further enhance the model ability to learn more feature under deep neural network. Identity mapping is introduced to learn residuals, targeting to narrow the gap between ground truth and prediction. By introducing deeper neural network, the Resnet is able to learn features on CIFAR-10 dataset with satisfiable performance.

In GoogLenet, an inception modular is introduced for a balanced learning of feature. The idea of feature fusion is introduced for large reception field and balanced feature learning. A 1*1 convolution is introduced to reduce the amount of the parameters.

2.3.3 Review for Recurrent Neural Network (RNN)

Recurrent neural network improves traditional neural network by adding time series related information to the model. Unlike previous model, RNN is capable of considering all the words that processed in the history.

For recurrent neural network, each hidden layer responses to a certain time step with a number of neurons. At each time step, there are two inputs to the hidden layer. Namely, they are the output of the previous layer and the input of the current time step. And the output is a

combination of the two input to form output feature map. To save parameters, the weighting matrix repeats for each weight groups and thus can counter curse of dimensionality.(Figure 8)

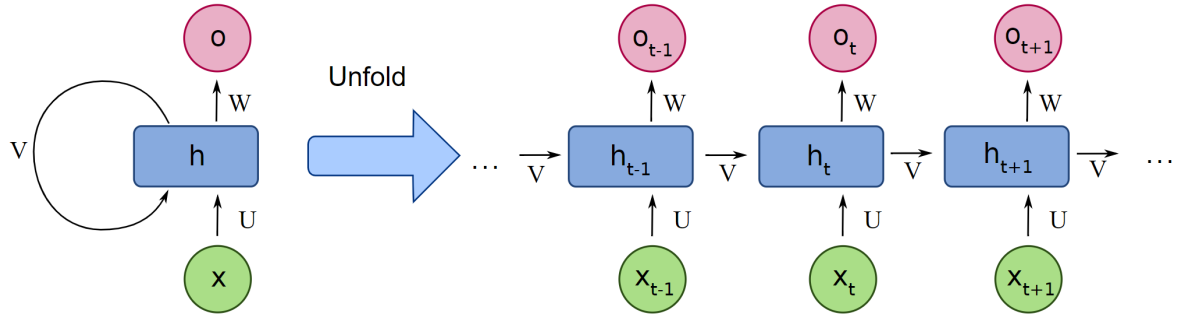


Figure 8 RNN (Sources: https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a24d010)

2.3.4 Review for Recurrent Neural Network based deep learning model

Seq2Seq uses encoder-decoder structure for neural machine translation task. An encoder and a decoder are introduced for feature extraction and language learning. To improve optimization performance, beam search is introduced to the model to speed up filtering process, by picking up best k words with the highest score.

"Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation" [19] introduces RNN based encoder-decoder structure for neural machine translation. Encoder aims to translate input sequence into a fix length vector and decoder learns from the feature and output machine translation result. In the meanwhile, a new unit called "Gating recurrent unit (GRU)" has been proposed for nature language processing. A good performance has been achieved thanks to the newly proposed technique.

"Deep contextualized word representations"[13] illustrates how to use pretrain weights for language model embedding. The paper uses LSTM to learn the context of the sentence bi-directionally. The output of the bi-LSTM can be used to mapping out word vectors or further fine-tune for more complicated task. From the post result analysis, the lower layer provides more useful information for neural machine translation, while the upper layer can help better understand the meaning of the sentence.

2.3.5 Review for deep learning based Handwritten recognition model -

"Handwritten Digit String Recognition (HDSR) using Convolutional Neural Network" - Hongjian Zhan, Shujing Lyu, Yue Lu [1]

This is an relate study to Handwritten digit string recognition, which it uses convolutional neural network (CNN) rather than recurrent neural network (RNN). This network has three parts: feature extraction layers, feature dimension transposition layers and an output layer.

HDSR is using line segmentation, traditional way segmentation is input image cut it to small pieces, then combine the recognition result of these pieces with path-search algorithms to get global optimal results. But this method still limits by different handwriting style, background noise (Blur image, low image contract, etc).

In this paper they propose a new way only using convolutional neural network, It has several dense blocks to extract feature, some dimension transposition layer to conduct dimension adjustment at last a connectionist temporal classification (CTC) to calculate loss in training phase and to produce the recognition result while testing.

The dataset using in this paper which is "CVL HDS: Computer Vision Lab Handwritten Digit String data set include coloured handwritten digit string image collect from 300 writers. The dataset content total 7960 image, 1262 for training and the 6698 images for testing. All the images have clear background with different writing style which make image difficult to handle. Second Dataset is "ORAND-CAR", this dataset content 11719 images from the Courtesy Amount Recognition (CAR) from real bank checks. This dataset is related to the image quality, image noise and handwriting style. The dataset split into two subsets: ORAND-CAR-A and ORAND-CAR-B. CAR-A contain 2009 images for training and 3784 images for testing. CAR-B content of 3000 training image and 2926 testing images. The string length in CAR-A is shorter than CAR-B in average. (Figure 9)

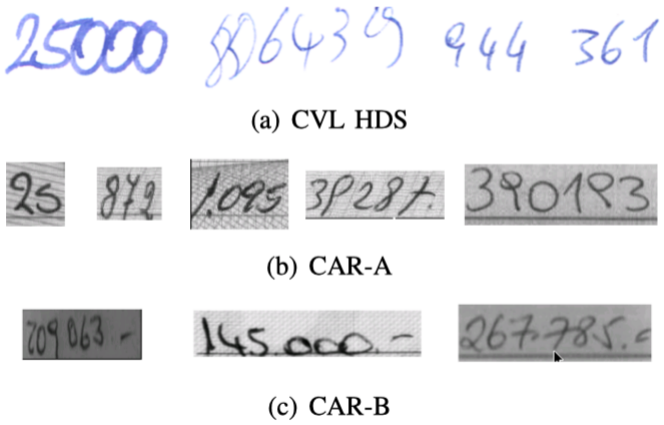


Figure 9 Example of CVL HDS (Source: Zhan, H., Lyu, S. and Lu, Y., 2018, August. Handwritten digit string recognition using convolutional neural network. In 2018 24th International Conference on Pattern Recognition

G-Captcha: Normal dataset string length samples are mostly not longer than 7. But for string recognition longer string mean harder task. So, the team create a dataset by using python package call "Captcha". This package generate image with any length captcha, image will have complex background and different style of digits. This dataset contains 14,000 images, 6,000 images for training and 8,000 for testing.

The team address a problem in Pre-Processing stage, due to different dataset have different image size, there are two method to solve this issue. One is resizing the input images into a fix size, the other way to approach is extend the feature sequence generated by feature extractor to a fixed length. The team choose to resize all the inputs images to uniform size.

In "Wide Residual Networks"[20], a variation of ResNet has been proposed for better network classification performance. The authors believe, it may not actually help network to learn better for the feature in the image, as it is not possible to ensure all the gradients can flow through each Residual block. So, the one that is shallow but wider should aid the classification of the human handwriting. To build a wider network, the author increases the channel by the factor of k. As it turns out, the performance indeed growth.

In “Densely Connected Convolutional Networks”[21], the authors extend ResNet by introducing all the feature map before the current layer to serve as input for the current one. Compared with ResNet, the new network is able to have more bypass to help training, while greatly reduce the required parameters for the network. After modification, the network becomes easier to train and the gradient vanishing issue was solved to some extent.

In “Improved Regularization of Convolutional Neural Networks with Cutout”[22], a new data augmentation method has been proposed. Cutout works as a series of patch level cropping and works in a discrete manner. By working in such manner, it is able to not only let model learn the feature but also can reduce overfitting issue. However, due the fix size of patching, some of the large objects may suffer from the pixel loss. This may hurt the performance of the classification for human handwriting.

In “ColorNet: Investigating the importance of colour spaces for image classification”[23], the author provides a detailed discussion for the possible changes that different colour system may result in. Some of the example are RGB、LAB、YCrCb、HSV、CIE. To fully strength the power of the different classification model, a late fusion is performed to get better feature map given the classification result.

Here is the table to compare the performance of algorithm on MNIST dataset.

Algotithm	Error rate
Cutout	1.30%
Wide Residual Networks	1.54%
DenseNet	1.54%
Wide ResNet	1.70%
DCNN	2.20%

2.3.4 Review for data augmentation technique

2.3.4.1 What is data augmentation and why we need

Data augmentation aims to generate complementary data to overcome the special cases where images with insufficient lighting condition, occlusion, large amount of background pixels and scale. Majority of cases those images lack accuracy. So, the task of data argumentation is to increase the amount of dataset by generating more translation invariance.

Data argumentation provides a way to reduce overfitting and thus can improve the model generalization ability. The dataset after careful selection of augmentation will cover more comprehensive set of data.

Data augmentation works in different styles. There are two sub-categories, namely, data wrapping augmentation and Oversampling augmentation.

- Data wrapping augmentation makes transformations on currently available dataset by reserving the ground true labels. Some of the examples are geometric transformation and colour transformation. In detail, some of the related techniques are random erasing, adversarial training and neural style transformation.
- On the other hand, oversampling targets to create dummy instances and add them to the training set. And here are some of the examples. Such as image mixture, CutMix, colour space segmentation, mosaic augmentation and Generative Adversarial Networks based technique.
    - In some of the application or field study, it is not always possible to seek data due to the cost or the long-time frame. Oversampling technique especially Generative Adversarial Networks can work well in this case to aid data argumentation by providing more generated fake image.
    - In the meantime, for the dataset that suffers from high class imbalance, it is possible to extract frames from low proportion side, where oversampling will help solve the problem.

One note to make is, the two different technique could work together and will not mutually exclusive. For example, it is nature to combine geometric transformation and CutMix together for stronger augmentation result. By doing this, we usually end up with a richer source of data inflation.

Here is an illustration for the taxonomy of data augmentation skills. Please check capture below.

Figure 10 Taxonomy of data augmentation skills (Sources: https://lionbridge.ai/articles/data-augmentation-with-machine-learning-an-overview/)

2.3.4.2 Geometric data augmentation technique

- Image flipping (Figure 11)
    - Image flipping directly flips images horizontally or vertically, depends on the task. Generally, it is more common to flip images horizontally.
    - In the meantime, image flipping is not a label-preserving operation. It means that the ground truth after flipping may not preserve and may need carefully adjustment.

Figure 11 Exnample of Horizontal Flipping (Sources: https://heartbeat.fritz.ai/basics-of-image-classification-with-pytorch-2f8973c51864)

- Image cropping (Figure 12)
    - Image cropping is highly useful in data augmentation. It is able to process image dataset with mixed input size by only cropping a patch from the image. Image cropping can crop randomly, crop in the center or crop by sliding, whereas keeps the completeness of spatial feature. However, labels may not preserve, depends on the way how you crop.

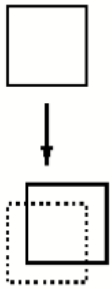
Figure 12 Example of Image Cropping (Source: https://helpx.adobe.com/photoshop/how-to/cropping-photo-basics.html)

- Image rotation (Figure 13)

- Rotations are performed for given image in different direction, from 1 to 360 degrees. Normally, only slight rotation is required on MNIST dataset for better model performance.
    - As the degree of rotation increases, the preservation of ground truth may not preserve.

Figure 13 Example of Image rotation (Sources: https://tympanus.net/codrops/css_reference/rotate/)

- Translation
    - Translation is performed by moving image to the left, right, up or down. As the translation performed, the image may need padding pixels to keep the original resolution. The padding can be filled with 0 or 255 in this case, or even be filled with some random noise. (Figure 14)



Translation

Figure 14 Example of Image translation (Sources: https://www.researchgate.net/figure/Examples-of-all-the-affine-transformations-for-a-simple-geometric-shape_fig1_251509419)

- Noise injection
    - Noise injection will inject a set of random value to images and those random values are usually generated by a Gaussian distribution. It has been tested by one professor from UCI with a dataset from UCI repository. From the result, add noise to input data can help improve the robustness of convolutional neural network.
    - Noise is not only limited to Gaussian distribution.

- Mix image (Figure 15)
    - Mix image is a very counterintuitive mindset for data augmentation. It has been introduced to improve the robustness of the classification model on CIFAR-10 and an image pair are combined with the weights of the pixel on separate image, which are adding up together. It is also possible to merge different image crops together on the same patch.
    - Better results are identified by merging the object from different classes instead of the same dataset. More crops are generated in this manner, which in turn boost the accuracy of the classification model. Similar performance gain has also been observed on the training process of Generative Adversarial Networks.

- One issue for mix image augmentation is, it is hard for human to understand the mixture and thus not possible to explain why this technique works in the majority cases.

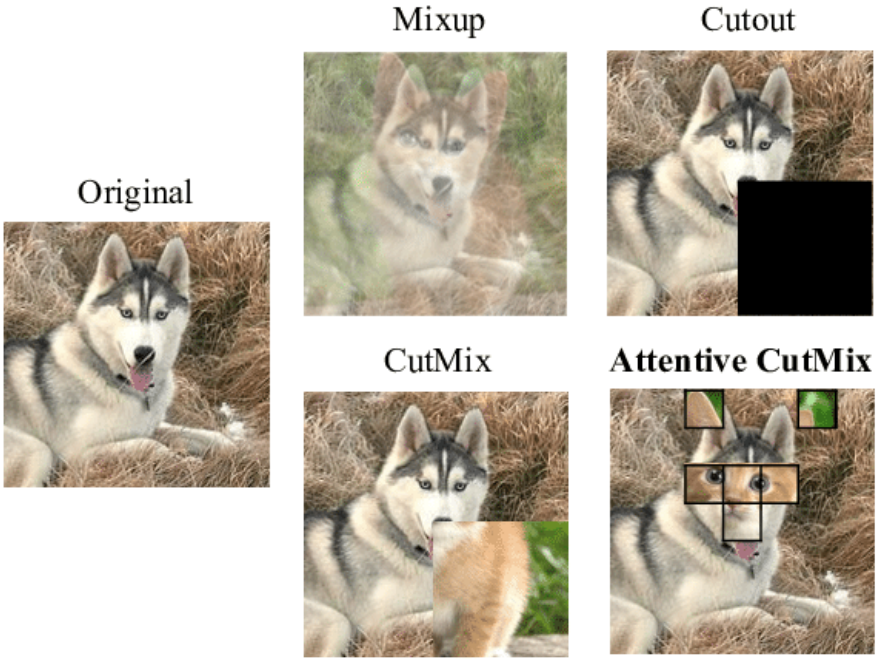


Figure 15 Example of Mix Image (Sources: https://www.researchgate.net/figure/Comparison-of-our-proposed-Attentive-CutMix-with-Mixup-5-Cutout-1-and-CutMix-3_fig1_340296142)

- CutMix [6]
  - CutMix works similar as mix image method. For mix image, we will merge a pair of images together with a weighted sum. However, for CutMix, we will cut out a region from image A and paste it to image B to merge image together.
  - Cutmix has some advantages. It helps filter out non-related pixels and thus improve the training efficiency. As compared with mix-up, it keeps both of the original image without overlap, and thus the generated image will not become unnatured. Those advantages keep CutMix perform better amongst all benchmarks.

- Random erasing
  - Random erasing can be treated as a variation of regularization and is being inspired from dropout regularization. It has been used to counter the effect introduced by occlusion image, where it will force the model to learn features that being occluded. In the meanwhile, even for the complete image, random erasing can also help force the learner to focus on the whole image, instead of the single patch on the feature map.
  - By removing one certain block on the image, random erasing prevent learner from overly focus all other areas, thus, to reduce the probability of overfitting.
  - Random erasing has been combined with Generative Adversarial Networks for image inpainting. It works similar as cutout regularization in this context and the performance is satisfying.
  - However, there is one potential issue when using random erasing. The generated image may not be label reserved. Some of the ground truth may need to be adjusted in order to match the image after done erasing.

2.3.4.3 Deep learning augmentation technique

- Colour space transformation
    - Colour space transformation has one another name "Photometric transformation", where it helps adjust lighting condition by changing the value for over dark or over light pixels.
    - It is also possible to slice out each individual colour channel and place restriction for the min/max pixel values to achieve the purpose of data augmentation.
    - It is also possible to switch the colour channel representation to "Hue, Saturation and Value" (HSV) system (maybe also YUV, CMY) to conduct augmentation in a different dimension.
    - Colour space transformation is good at eliminating colour bias. However, there are still disadvantages in colour space transformation. Such as increasing usage of memory, transformation cost and training time. It is also possible that colour space transformation may result in colour information loss.

- Feature space augmentation
    - Usually, we apply data augmentation on data side. However, when we utilize deep learning technique, it is also possible to collect low dimension feature map after feature extraction via the backbone, which are what called "feature space"
    - To utilize those feature space, a set of statistical methods are proposed. Such as adding noise, conducting interpolation and extrapolation on the feature map to add more variation for collected data.
    - In the meanwhile, it is also possible to use autoencoder as feature extraction tools for feature space augmentation. Here we only use encoder feature, where it extracts the low lever feature of images. To achieve that, a possible cutoff of the output layer is performed. The obtained feature vectors can be used for training machine learning models or as the source for the transfer learning task.
    - As similar with mix image, the output vector of feature space is hard to interpret. It may not be able to explain the internal meaning of the whole feature vectors without special tools or the weights of the whole autoencoder. And thus, data-space method often more preferred.

- Generative modeling
    - Generative modeling refers to the common practice that to create dummy data from a similar dataset that have abundant materials to be learnt from. One of the typical examples for generative modeling is to train a Generative Adversarial Network. [7][8][9][10]
        - Generative Adversarial Network consists of a generator and a discriminator. And one of the best examples to illustrate the difference between those two is the example of counterfeiter and a cop. In such case, counterfeiter is the generator and the cop correspond to discriminator.
            - Counterfeiter will try the best to fool the cop. In turn the cop will catch counterfeiter whenever is possible.
            - The input of the generator is the source data, in terms of images, sheets, data form. And the input for the discriminator is simply from the fake example produced by generator. The higher quality fake sample produced, the higher possibility to fool discriminator.

        - Some useful theory borrowed from other academic field helps guided GAN. For example, from minmax algorithm and Nash equilibrium borrowed from game theory.
    - Vanilla GAN structure introduces multiple dense layers into both generator and discriminator for training acceptable number of images. However due to the extreme large parameter sets introduced by the dense layer, it becomes extremely hard to optimize the model. And thus, it is not able to produce quality results for higher resolution.
    - DCGAN expand GAN based on internal structure of the model. Instead of purely neural network, convolutional layer is introduced for better feature extraction ability. After completion of feature extraction, deconvolutional layer is introduced to project tensor back to an output image. Overall speaking, the structure of encoder and decoder are well balanced.
    - Another GAN variation is called progressively growth GAN. Progressively growth GAN trains a series of neural network with an increasing step of resolution complexity and accept image vectors to train the model. Usually, it can produce amazing results on facial images.
    - CycleGAN provides a new choice for larger resolution as compared to GAN. Comparing with GAN, it brings an additional cycle-consistent loss to help stabilize GAN training. CycleGAN is able to translate from a domain of images to another domain of image thanks to the forward and backward consistency loss. It has a second discriminator to help determine if re-translated image belongs to one class or not.
    - Conditional GAN add a conditional vector to both generator and discriminator for possible assumptions to follow, targeting to solve mode collapse issue.
- Neural style transfer (Figure 16)
    - Neural style transfer targets to utilize the representation of images from convolutional neural network. By manipulate the style of the inference image, the original one can be transferred to another while preserving its original information.
    - Fast style transfer extends the loss function from neural style transfer to generate perceptual loss, which makes it run faster. A feed forward neural network is introduced to stylize images according to it.
    - Neural style transfer can augment dataset by changing lighting variation and adapt image to different styles accordingly. Thus, it helps make decision on which part (or what style) of dataset to sample.
    - We still need to exercise good decision and consider multiple factors such as night-to-day scale, winter-to-summer, or rainy-to-sunny. Neural style transfer can help much with the generation for those datasets. However, if there are not too many variations, the results may not so effective.
    - There are issues with neural style transfer if you are targeting to data augmentation. You need lots of efforts to select styles to transfer image into. So that means another dataset to pick up. If the style set is small, you may not get enough data and thus may overfit the whole dataset, which may result in insufficient data to train.

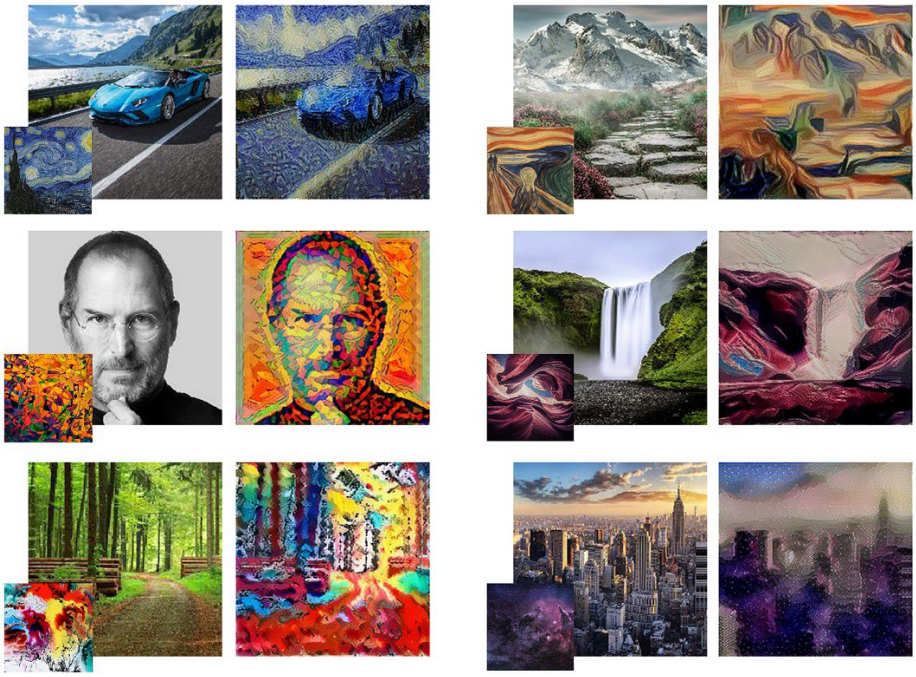


Figure 16 Example of Neural style transfer (Sources: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee)

To sum up, data augmentation is quite useful in majority of the application case study. Some of the common operations, such as image flip, rotation and mosaic augmentation can be stacked together to provide more data that will be available to the whole dataset. And that's why it becomes critical in dataset training.

2.3.5 Review of Feature Extraction

2.3.5.1 Why we need Feature extraction and why we need it

Feature extraction is the single most important factor in achieving high recognition performance. There are few methods of had been researcher in the literature. There are many methods for example: Graph description, Contour profiles, Template matching, Deformable templates, Unitary Image transforms, Project Histograms and Zernike Moments eta.

Feature extraction is the process remove noise from data. Before starting feature extraction need to clarifier what feature need to select. Feature extraction is technique will be used to extract feature from the image most representative information. Minimises the class pattern adaptability and enhance the between class pattern adaptability. Extract feature from each class helps identify from other classes.

2.3.5.2 Feature extraction techniques

There are two kinds of features, statistical features and structural features. Many researchers agree statistical feature can easy use and perform good recognition results especially for closed testing data, also it can easily affect by deformation of symbols, so could not be expanded to more application. Another is structural features, it is more like human mind, structural feature is more reliable to deformation of symbols. But they are usually relying on human summarized

rules for the recognition algorithms. When new classes are introduced into system need more cost to revise the algorithm.

- Statistical Features
  - Decrease pixel density by reducing the dimension of the feature set which provide high speed and low complexity. But does not allow reconstruction of the original image. The major statistical feature will be list below.
  - *Zoning*: The based feature extraction one of the popular methods. It divided image to predefined number of zone, for example image have 100 * 100, Zoning have predefined 30 * 30, Zoning will downscale image to predefined zone size, the image divided into serval overlapping and non-overlapping (Figure 17) [18]

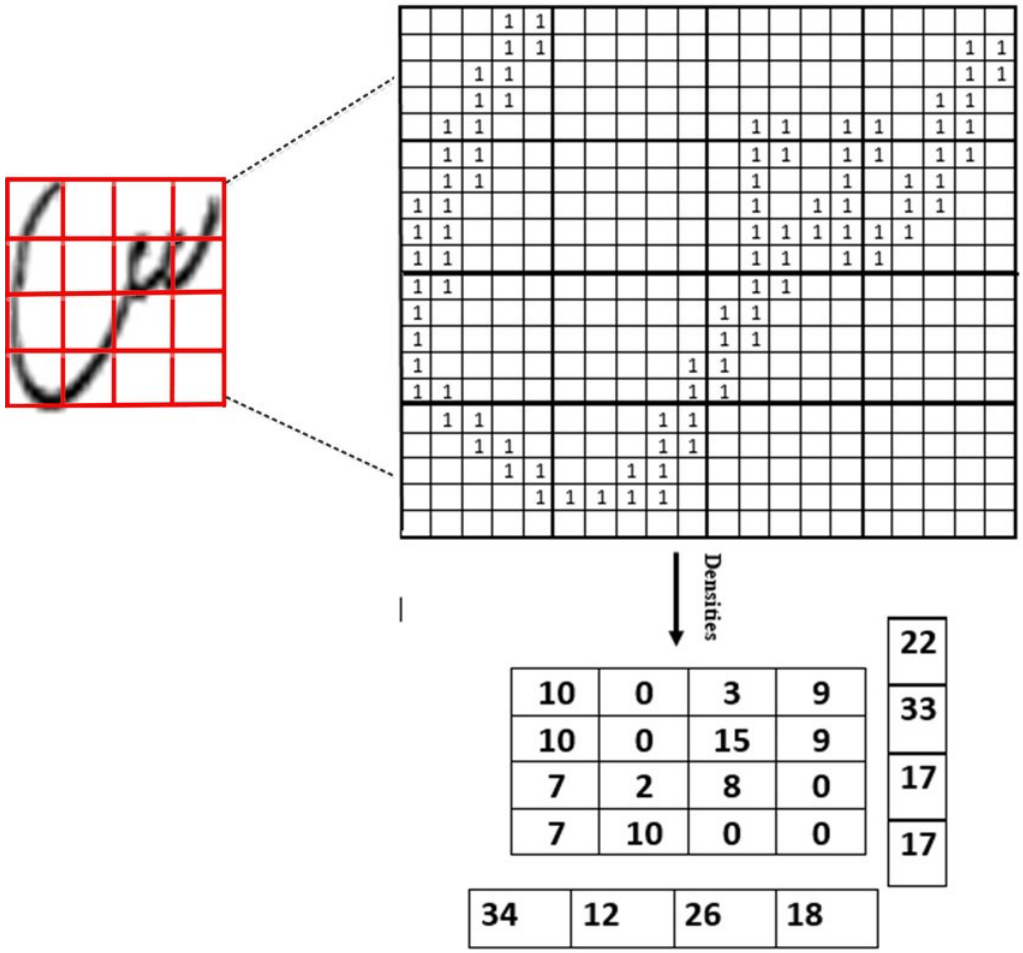


Figure 17 Example of Zoning (Sources:https://www.researchgate.net/figure/Zoning-feature-extraction_fig6_328754005)

- *Structural Features*
  - This method identifies structural features of a character based on geometric properties of the character or digit. Example of structural features are how many numbers of horizontal or vertical lines, numbers of endpoints, number of cross points, horizontal curves at top or bottom. Structural features need some knowledge about the structural of image of character or what sort of components make up the digit and character.

3. Implementation and Results

This chapter will outline and discuss the structure of the research including the method using for this project, giving the parameter during the experiment, data collecting methods, and presenting the quantitative and qualitative of result, explain the bad example.

3.1 Model architecture

In my project I'm choosing Recurrent Neural Network (RNN) for my neural network, and three different dataset, MNIST, Noisy-MNIST and Street View House Number (SVHN).

- **Recurrent Neural Network (RNN)** – It is a neural network takes a fixed size vector as input which result in RNN affected by inputs in different periods, and RNN have the function of storage and memory. This mean RNN learn similarly while training, in addition, they remember things learnt from prior inputs while generating outputs.

3.2 Datasets

- **MNIST** – "Modified National Institute and Technology" is the database for handwritten digit commonly used for training various image processing system. MNIST contain 60,000 images for training and 10,000 images for testing. (Figure 18)

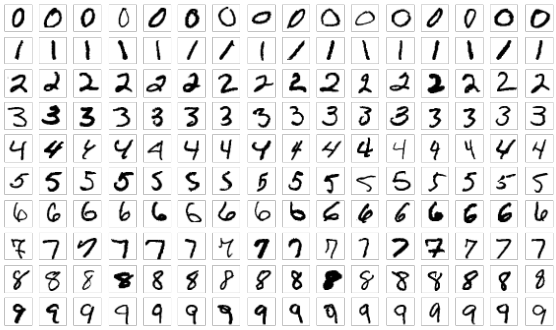


Figure 18 Sample images from MNIST test dataset (Source: https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png

- **Noisy-MNIST (n-MNIST)** – It is based on MNIST handwritten digit dataset by adding
  - Additive white gaussian noise (Figure 19)
  - Motion Blur (Figure 20)
  - A combination of additive white gaussian noise and reduce contrast (Figure 21)

Figure 19 n-MNIST with Additive White Gaussian Noise (AWGN) (Source: https://csc.lsu.edu/~saikat/n-mnist/

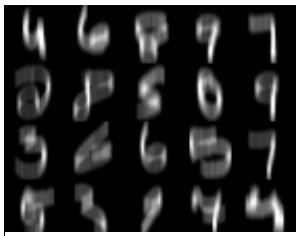


Figure 20 n-MNIST with Motion Blur (Source: https://csc.lsu.edu/~saikat/n-mnist/)

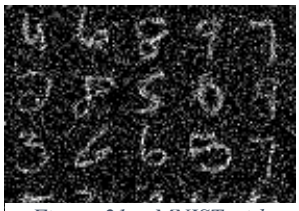


Figure 21 n-MNIST with reduced contrast and AWGN (Source: https://csc.lsu.edu/~saikat/n-mnist/)

- **Street View House Number (SVHN)** – It is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data pre-processing and formatting. It is similar to MNIST, but it is using house numbers in Google Street View images. The has 73257 digits for training, and 26032 digits for testing. (Figure 22)

Figure 22 Sample of The Street View House Number (SVHN) Dataset (Source: http://ufldl.stanford.edu/housenumbers/)

3.3 Implementation Details

- **Hardware setting** – MacBook Pro 16inch, CPU Model- i9-9980HK 2.6GHz, 64GB Ram, 1TB Storage, AMD Radeon Pro 5500M Graphic card
- **Data Augmentation –** To increase difficulty of training and to balance the distribution of the dataset, here we use following three data augmentation technique
    - "Affine"
    - "Random Rotation"
    - "Colour Jitter"
- Parameter setting

The dataset image size is different, the RNN have fixed size 28. So, before importing image to RNN for training will need resize all the input image to uniform size. SVHN and MNIST are using 32*32 image size, Noisy-MNIST is using 28*28 image size. So, downscale all the image to 28*28. This method won't make any image quality lose. The learning rate set to 1Epoch–3 and using "Adam" optimizer and the Batch size set to 64.

For this project we are aimed to achieve the test accuracy above 90% when using test set from three different dataset, later will be giving human written and G-Captcha image for best model testing.

In this project we will MNIST, n-MNIST and SVHN dataset. To find out the best model we introduce Pytorch torchvision transform, we are adding affine, random rotation (15) and colour jitter on the train images. This can increase the difficulty on training also simulate the real work situation where image noise effect the recognition accuracy.

First step we are creating model, the Batch Size = 64, Learning rate = 1e – 3, Number of epochs = 3, and the image size = 28*28 for all training and testing dataset is fixed. Only data augmentation set up will be different. For example, we are using MNIST trainset against MNIST testset, during training will apply only Affine to check the test accuracy compare to MNIST trainset against MNIST testset with Affine and Colour Jitter test accuracy.

Second step, we are using different database trainset against other dataset testset, for example using MNIST trainset for training and SVHN testset for testing the training model. The propose of this is finding the best model for later human handwritten digit and G-Capcha images testing.

3.4 Quantitative Results

TrainSet\TestSet	MNIST	RC	MB	WGN	SVHN
MNIST	95.88%	48.05%	93.16%	59.82%	17.40%
RC	96.63%	94.83%	94.99%	96.15%	18.38%
MB	96.18%	66.12%	97.66%	81.40%	22.46%
WGN	96.63%	95.32%	93.54%	97.15%	17.96%
Combined	98.16%	96.88%	98.26%	98.38%	23.63%

Table 1 Best Model Result RC = Noisy-MNIST with Reduce contrast and Additive White Gaussian Noise, MB = Noisy-MNIST with Motion Blur, WGN = Noisy MNIST with Additive White Gaussian Noise and Combine = Noisy-MNIST RC + MB + WGN.

The Table 1 show the best model result, the column on the left-hand side is the TrainSet use for training, the first row is the testSet use for testing. The best result of test accuracy is highlight by the red colour and green colour is the second-best model result.

As you can see from the Figure 6 the combine trainset has the best result against rest of the testset this is because the Combined trainset have 160,000 images for training. All the Noisy - Combined trainset model is using 10 epochs rather than 3 epochs. Because there are 160,000 images for training, to not overfit each batch size in number of iterations. The best result of combined trainset is against the WGN which it reaches 98.28% test accuracy. This model had applied pytorch transform affine and colour jitter during the training. As you can see from the below Figure 7 the full detail of using Noisy-MNIST Combine as trainset against other testset full experiment detail. There are three model is applying ‘Affine’ during the model training, we can assume the affine can increase the train accuracy as a result increase the recognition test accuracy.

Optimizer	Batch Size	Number Epochs	Learning Rate	trainSet	testSet	Affine	Colour Jitter	Rotation (15)	Test Accuracy	Train Accuracy
Adam	64	10	1.00E-03	NMINST_Combine	MINST	Yes	No	No	98.16%	98.47%
Adam	64	10	1.00E-03	NMINST_Combine	NMINST_RC	Yes	Yes	No	96.88%	98.48%
Adam	64	10	1.00E-03	NMINST_Combine	NMINST_MB	Yes	Yes	No	98.26%	98.40%
Adam	64	10	1.00E-03	NMINST_Combine	NMINST_WGN	No	No	Yes	98.38%	98.64%
Adam	64	10	1.00E-03	NMINST_Combine	SVHN	No	No	No	23.63%	99.48%

Table 2 Full detail on Noisy-MNIST as trainSet against other dataset testSet

3.5 Qualitative Results

In this section, we give a quick analysis for the inference result reported in the project. As we realized that the result cannot be perfectly aligned with the ground truth case by case, some of the wrong samples are still interesting to investigate and may suggest the future improvements. That's the motivation to conduct post-result analysis.

3.5.1 Bad case analysis

And here we list 10 visualization results for the reference purpose.

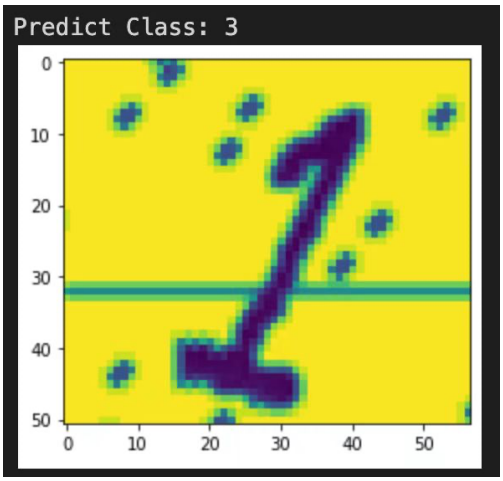


Figure 23 G-capcha Number 1, Predict Result:3

Here is the first plot graph. The input image contains noise, which are a straight line and some dots on the image. We expect to get an inference result of 1 but actually get a prediction of 3, which are against the ground truth. One of the possible reasons for incorrect predict may be due to the writing style of the input image.

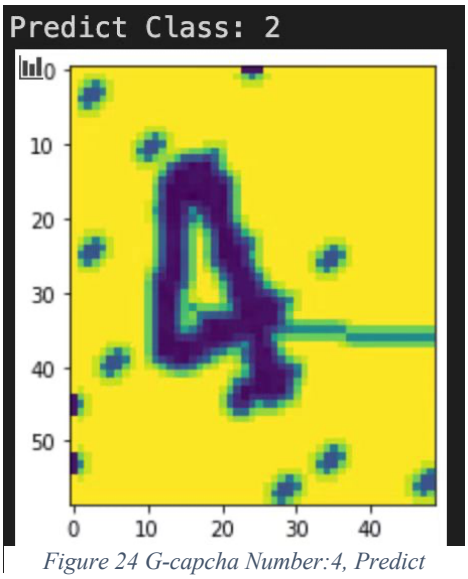


Figure 24 G-capcha Number:4, Predict Result:2

Here is the second plot. As we can be observed from the plot, the input image contains noise, which are a half-straight line and some dots on the image. We expect to get an inference result of 4 but actually get a prediction of 2, which are against the ground truth. One of the possible reasons for incorrect predict may be due to the writing style of the input image.

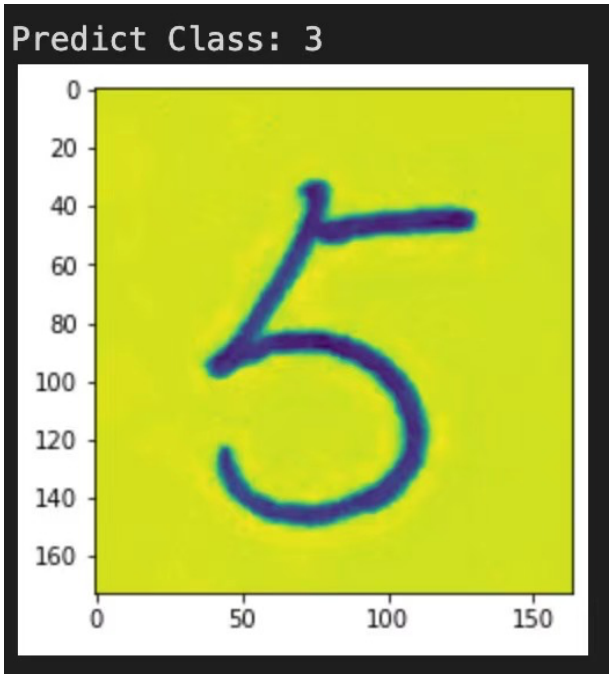


Figure 25 Handwritten Number:5, Predict Result:3

This figure is quite weird as it supposes to have a correct inference result. But it failed. It may be due to the fact that the figure '5' is limited in this dataset and thus we do not have enough samples to train this class. Further investigation may require based on the inference result reported here.

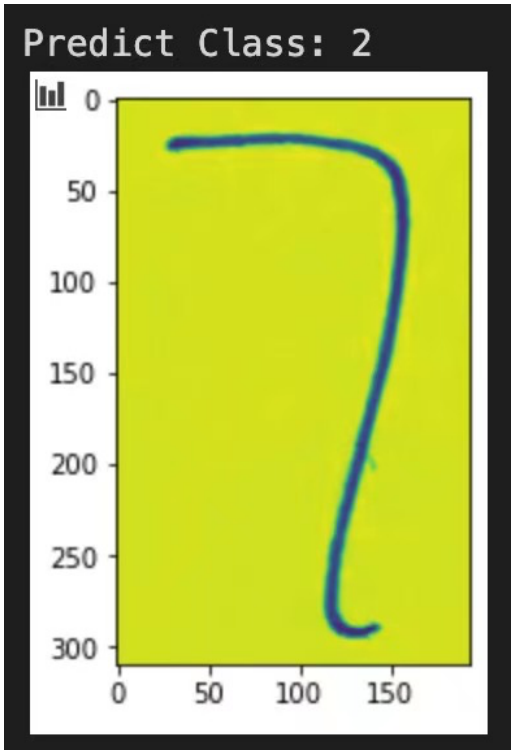


Figure 27 Handwritten Number:7, Predict Result:2

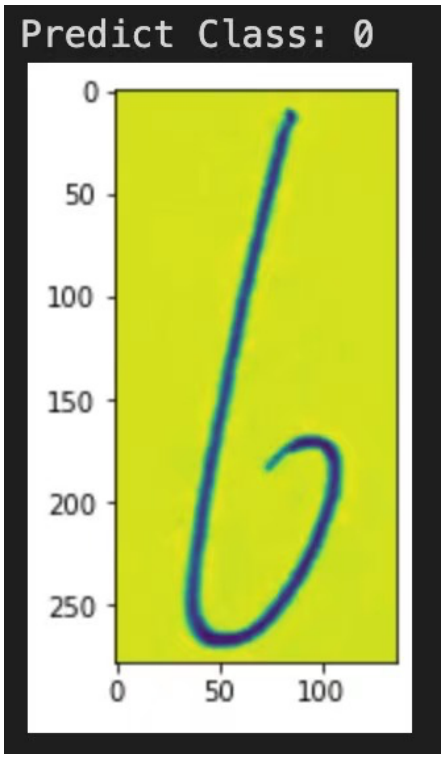


Figure 26 Handwritten Number:6, Predict Result:0

For those two figures, we can directly observe that the incorrect prediction is due to the similarity of the writing for figure pair of (0,6) and (7,2). As those figures are similar, it may even not easy for human to directly tell the difference without a carefully check. Under such condition, it is reasonable that classifier failed to predict correctly.

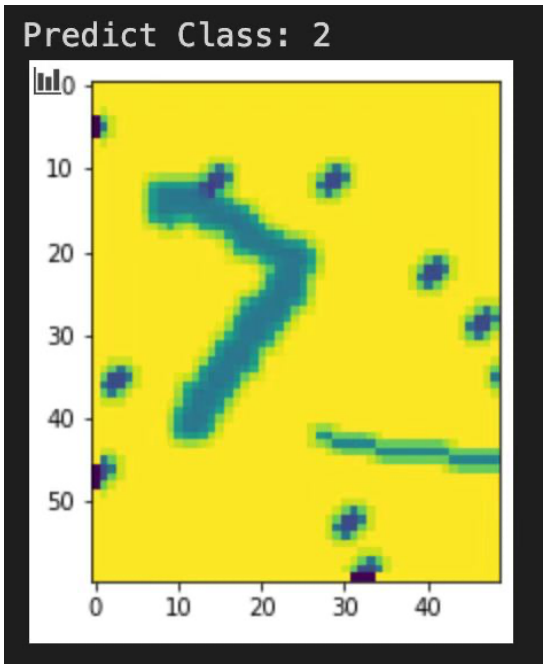


Figure 28 G-capcha Number:7, Predict Result:2

Here is another bad case. The classifier is expected to predict figure “7” instead of “2”, which may be due to the additional line that show up around figure “7”. So, this throws a light on some possible issues that missed in the data preprocessing step. If it is possible to remove the noise (namely the straight line in the plot), it might be possible to correctly predict the figure in the plot.

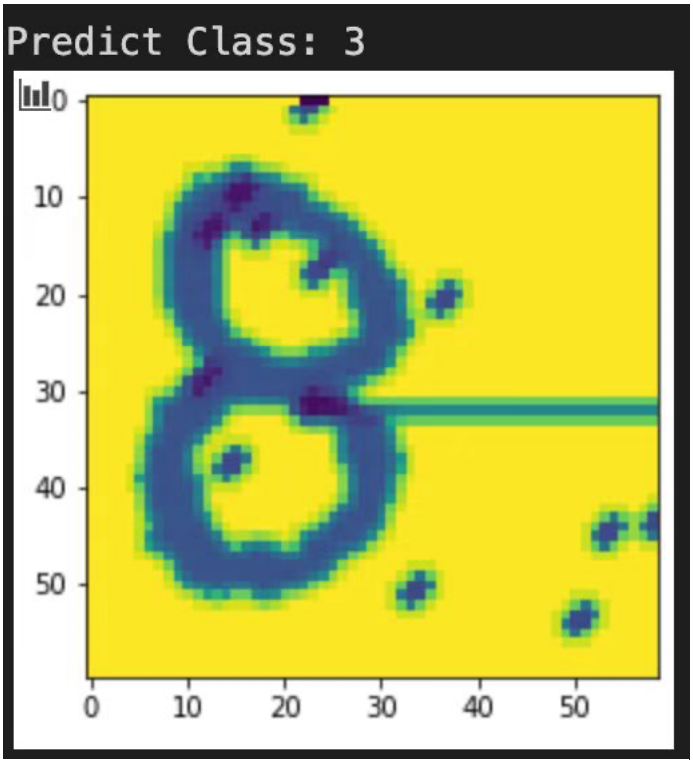


Figure 29 G-capcha Number:8, Predict Result:3

Here is another plot with the similar issues suggested above. It supposes to be 8 but wrongly predicted as 3. Maybe the noise in the plot affects the final prediction, as can be told from the figure. In the meantime, the dot on the figure may also mislead the decision of the classifier. An image smoothing operation may be a good choice in this case.

3.5.2 Good case example

All the result is generated by using the best model.

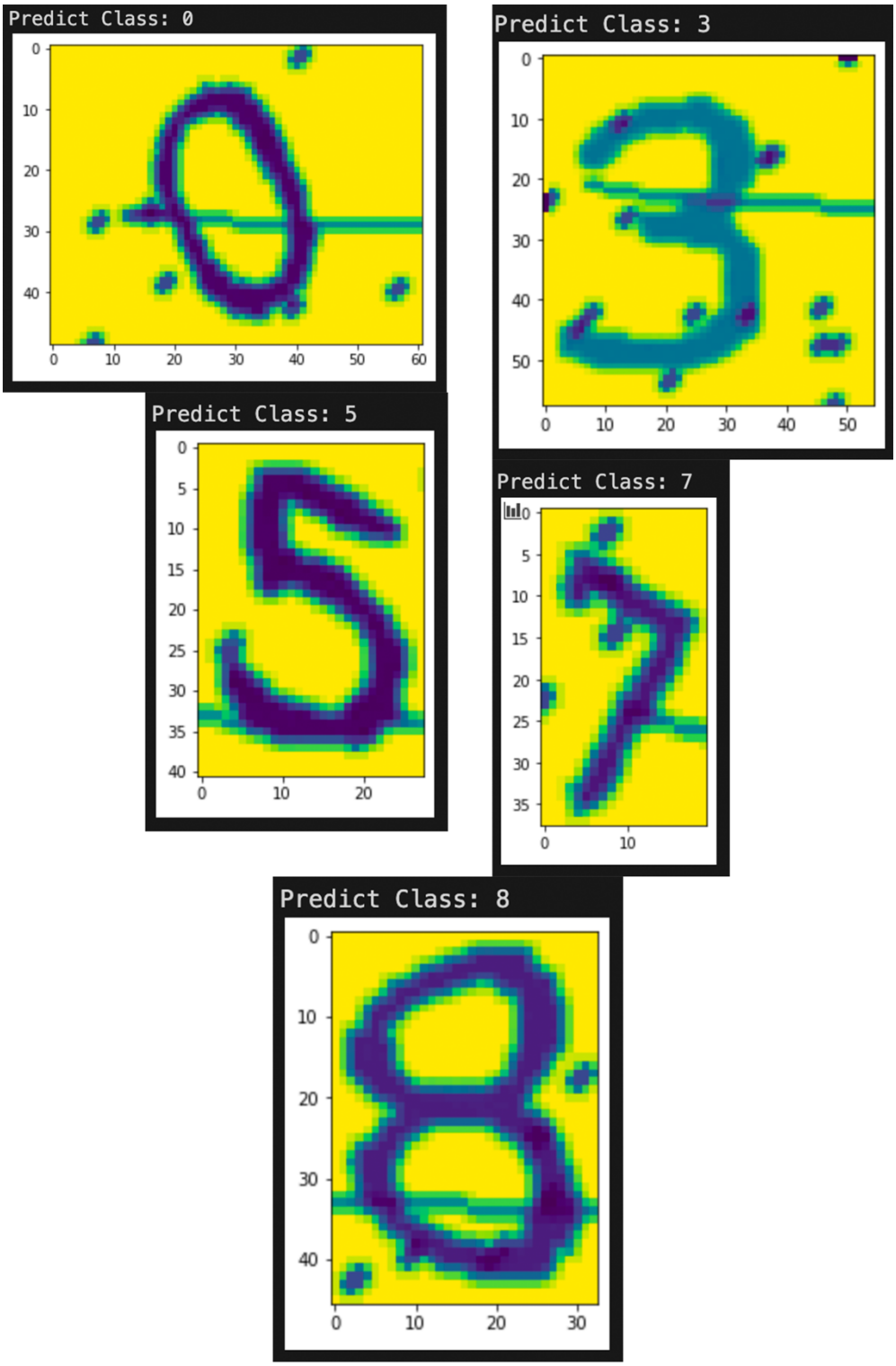


Figure 30 G-Capcha Test Result

As you can see from Figure 30 all the image generate by G-capcha, there are lot of background noise inside the photo, but with using best model, it still able to recognise correct class.

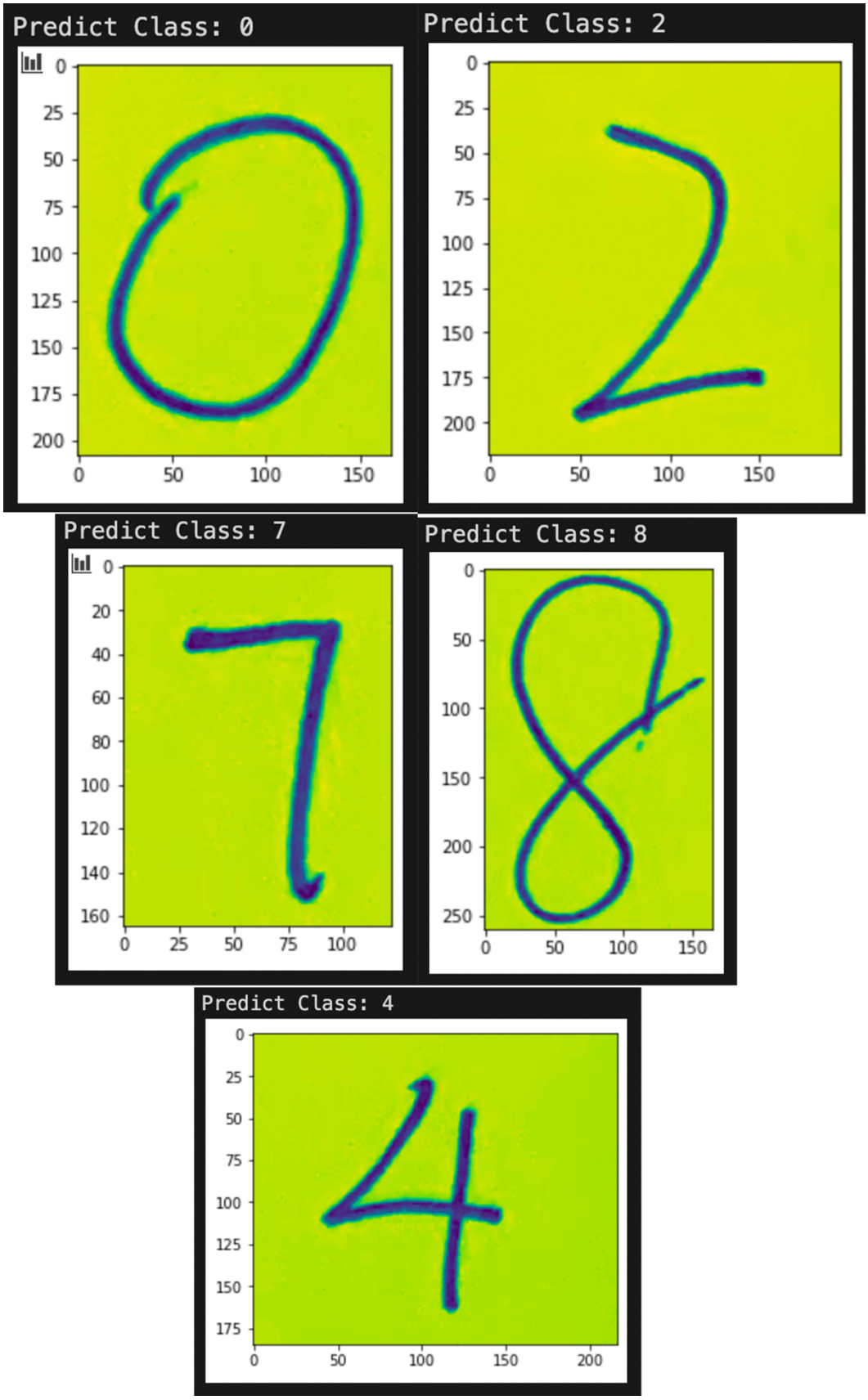


Figure 31 Human handwritten digit

The Figure 31 is using human handwritten digit for the testing, all the number are recognized to the correct class. Even for the very similar number [2,7].

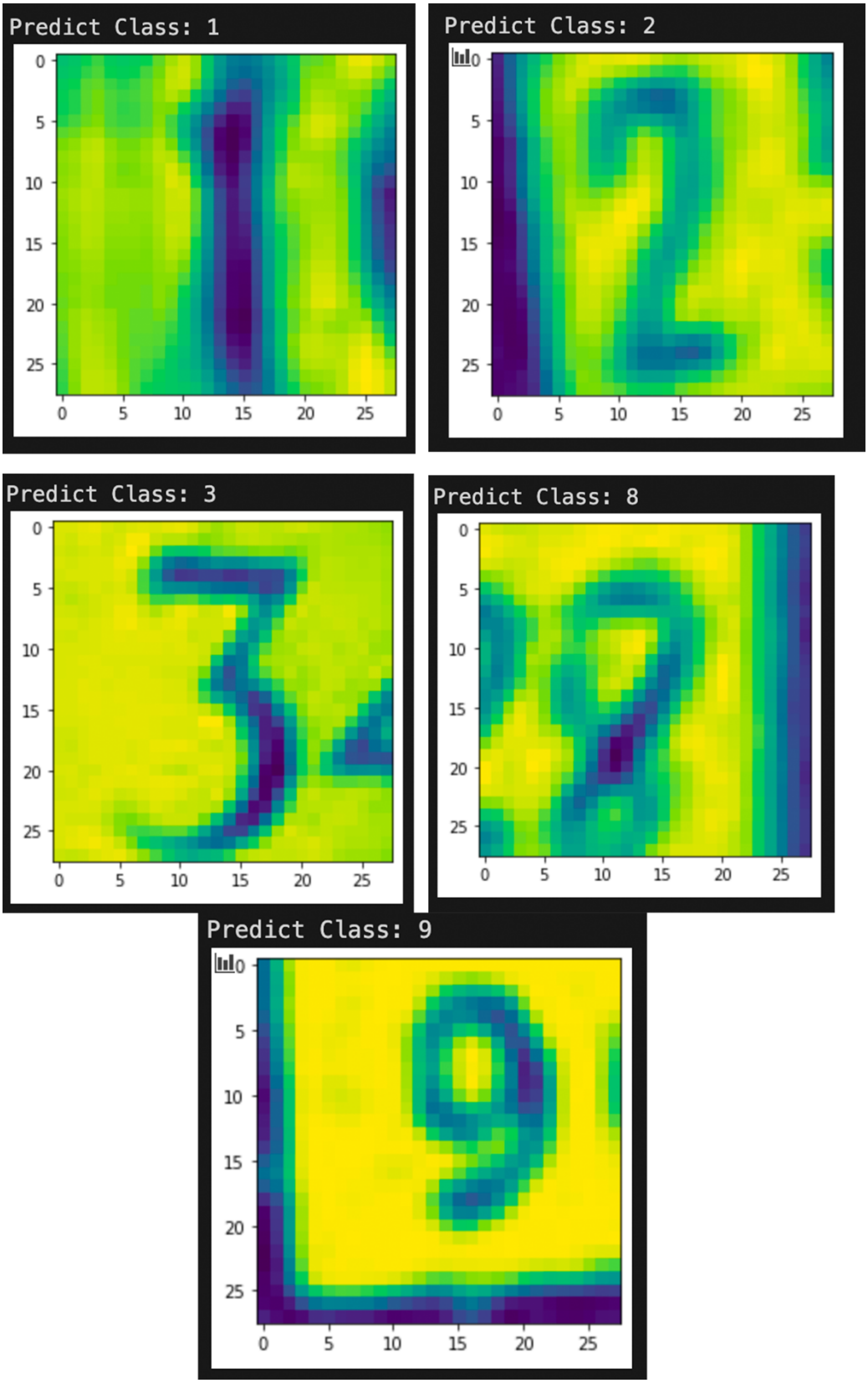


Figure 32 Best Model by using SVHN testset

The Figure 32 is using the photo from SVHN test set, as you can see there are lot of image noise inside which still able to handle and produce a very good result.

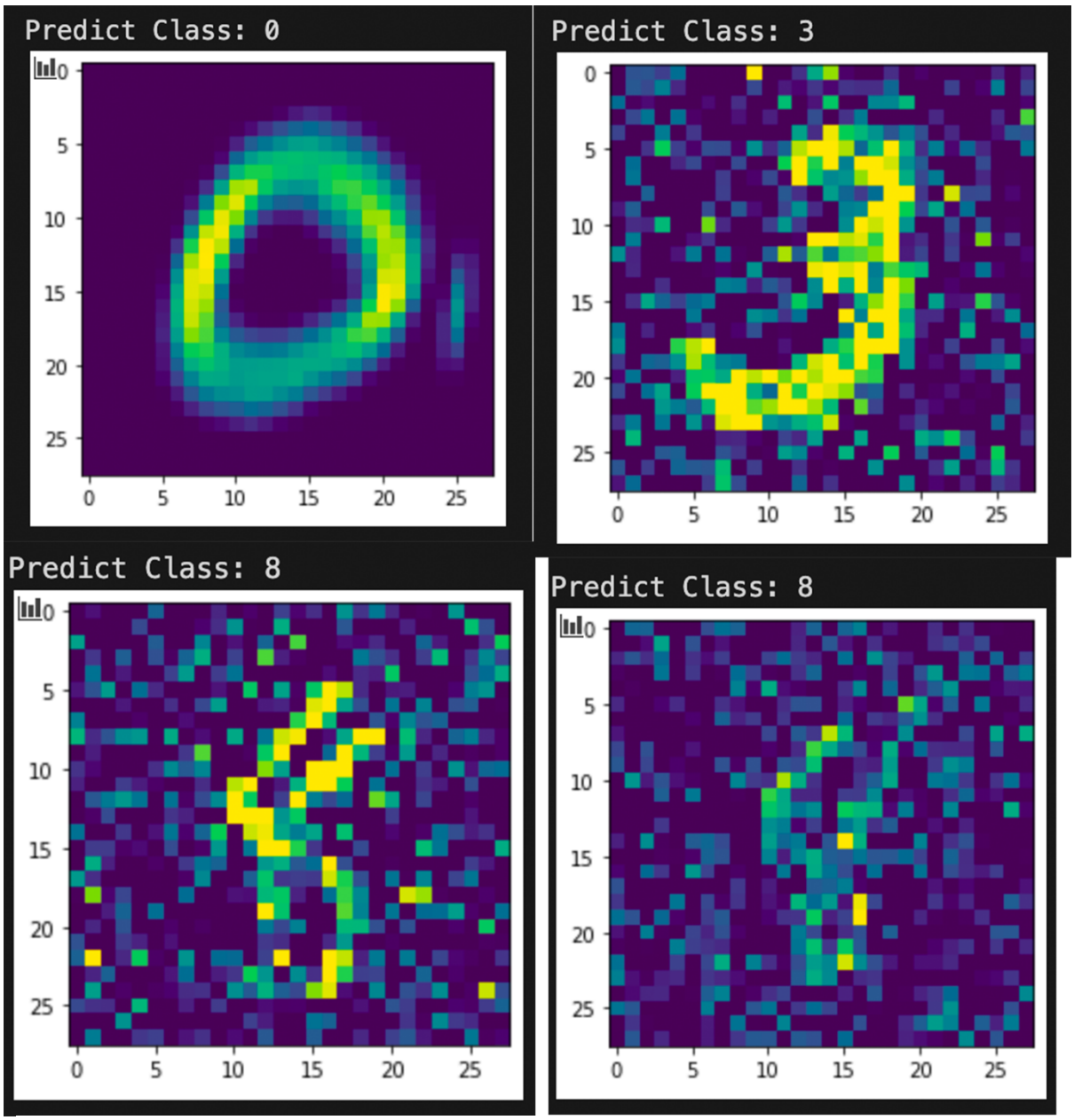


Figure 33 Best Model testing Noisy-MNIST testset

The Figure 33 is using the image from Noisy-MNIST testset, the image are very blurry, low contact and contain lot of noise. So, we can assume when using the best model, it can provide a very good recognition accuracy.

Chapter 4 Conclusion and future work

4.1 Conclusion

In this thesis, a detailed analysis for recognition of the human handwriting has been investigated and fully discussed. The background of the research, the possible application, theorical background, proposed method to solve the human handwriting recognition problem and quantitative result as well as post result analysis have been discussed in each chapter.

On average, the quantitative result shows nearly 98% accuracy achieve on majority of the dataset with detailed study for the possible factors that could affect the performance. Some drawbacks of the experiment design have been illustrated through bad cases analysis. The future improvements can be derived from the discussion above.

In the meanwhile, during the literature review, some identified academic research questions intuited us to balance the tradeoff between precision and the speed of the application. It is the best interest for my thesis to add the speed consideration in our future project accordingly.

4.2 How to improve future work

In this paper, we have presented the Recurrent neural network with pytorch transform for handwritten digit recognition with a high-test accuracy either using testset from the MNIST, n-MNIST and SVHN, and real-world example like human written digit and G-Capcha images. We achieved great improvement on all three datasets with the accuracy of ~98 on average with only RNN based deep learning technique.

In the future work, we plan to introduce more functionality, which are

- line segmentation
- Image segmentation feature (Figure 34)

- Increase digit string length
    - So, it will crop the digit from the images rather done by human and able to recognize more than one digit string in the images.
- Improve the runtime of the current work.

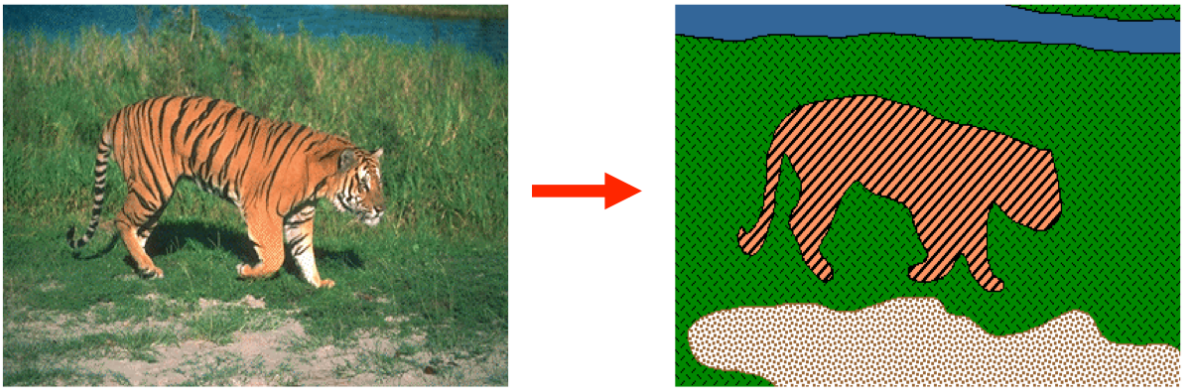


Figure 34 Example of Image Segmentation (Source:https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html)

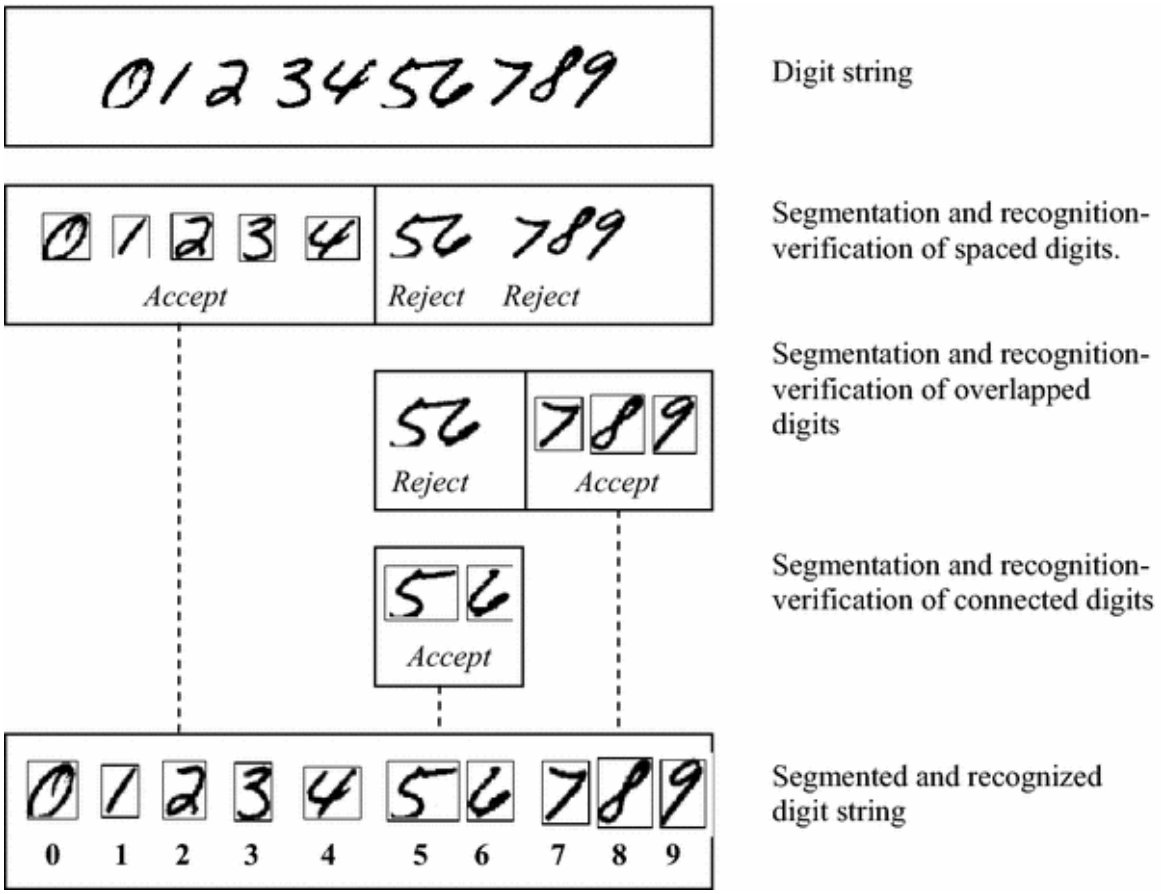


Figure 35 Digit string length segmentation (Sources: https://link.springer.com/article/10.1007/s10044-017-0607-x)

Reference:

[1] Zhan, H., Lyu, S. and Lu, Y., 2018, August. Handwritten digit string recognition using convolutional neural network. In 2018 24th International Conference on Pattern Recognition (ICPR)(pp. 3729-3734). IEEE.

[2] LeCun, Y., 1998. The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/.

[3] Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, Ramakrishna Nemani, Learning Sparse Feature Representations using Probabilistic Quadtrees and Deep Belief Nets, European Symposium on Artificial Neural Networks, ESANN 2015.

[4] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011. (PDF)

[5] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[6] “CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features”, Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, Youngjoon Yoo; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6023-6032

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, BingXu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks.arXiv e-prints, page arXiv:1406.2661, June 2014

[8] Alec Radford, Luke Metz, and Soumith Chintala.Unsu-pervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv e-prints, pagearXiv:1511.06434, Nov. 2015

[9] Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.

[10] Mehdi Mirza and Simon Osindero. Conditional GenerativeAdversarial Nets.arXiv e-prints, page arXiv:1411.1784, Nov.2014

[11] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[12] Karen Simonyan and Andrew Zisserman. Very Deep Convo-lutional Networks for Large-Scale Image Recognition.arXive-prints, page arXiv:1409.1556, Sept. 2014.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[14] C. Szegedy *et al*., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[15] Imagenet classification with deep convolutional neural networks, A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems 25, 1097-1105

[16] Long Short-Term Memory (Sepp Hochreiter and Jürgen Schmidhuber), In Neural Computation, volume 9, 1997.

[17] Rani M., Meena Y.K. (2011) An Efficient Feature Extraction Method for Handwritten Character Recognition. In: Panigrahi B.K., Suganthan P.N., Das S., Satapathy S.C. (eds) Swarm, Evolutionary, and Memetic Computing. SEMCCO 2011. Lecture Notes in Computer Science, vol 7077. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27242-4_35

[18] Vithlani, P. and Kumbharana, C.K., 2015. Structural and statistical feature extraction methods for character and digit recognition. International Journal of Computer Applications, 120(24), pp.0975-8887.

[19] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[20] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[21] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[22]] DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

[23] Gowda, S. N., & Yuan, C. (2018, December). ColorNet: Investigating the importance of color spaces for image classification. In Asian Conference on Computer Vision (pp. 581-596). Springer, Cham.