Optical Character Recognition

A Project Report submitted

by

JYOTI PATEL (19285756017)

in Partial Fulfilment for the award of the degree of

MASTER OF SCIENCE

in

INFORMATICS AND COMMUNICATION

at

INSTITUTE OF INFORMATICS & COMMUNICATION
UNIVERSITY OF DELHI, SOUTH CAMPUS,
BENITO JUAREZ ROAD, NEAR DHAULA KUAN,
DELHI-110021

Certification

This is to certify that the Project report entitled "**Optical Character Recognition**", submitted by **JYOTI PATEL** (Roll No. – 19285756017), in partial fulfilment for the award of "**Master of Science**" in "**INFORMATICS AND COMMUNICATION**" of the **INSTITUTE OF INFORMATICS & COMMUNICATION, UNIVERSITY OF DELHI**. The period of project was from 2nd Feb, 2021 to 1st June, 2021. This has not been submitted to any other University or Institution for the award of any Degree/Diploma/Certificate. The project report has been approved as it satisfies the academic requirements with respect to the project work.

Dr. Sanjeev Singh
(Project Coordinator)

Acknowledgement

This is great opportunity as well as great honour to submit this project. I am thankful to the **Institute of Informatics & Communication** for providing such an excellent infrastructure equipped with ultra-modern facilities which serve as the great source of convenience and information.

I sincerely thank to all those who contributed to the success of this project. First of all, I am thankful to my mentor, **Dr. Sanjeev Singh** for making me able to do this kind of work and giving me a new experience. He made me understand the concepts of Character Recognition and gave sincere thoughtful guidance.

Jyoti Patel

10 June, 2021
Delhi

Contents

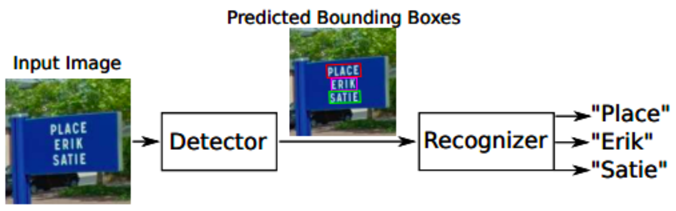
Abstract

Recognizing arbitrary multi-digit numbers and alphabets from view imagery is one of the challenging task which can be tackled with the help of deep learning. To achieve this proposed target, I have used deep convolution neural network with multiple hidden layers that operates directly on image pixels. Having gone through some of the literature, I inferred the knowledge that to achieve such aim, I had to perform per character recognition task. If I wanted to train on the character-level, I had to follow basically three steps: localization, segmentation and recognition.

Chapter 1

Introduction

Text can be found on documents, road signs, billboards, and other objects like cars or telephones. Automatically detecting and reading text from natural scene images is an important part of systems that can be used for several challenging tasks such as image-based machine translation, autonomous cars or image/video indexing. In recent years the task of detecting text and recognizing text in natural scenes has seen much interest from the computer vision and document analysis community. Furthermore recent breakthroughs in other areas of computer vision enabled the creation of even better scene text detection and recognition systems than before.



Although the problem of Optical Character Recognition (OCR) can be seen as solved for printed document texts it is still challenging to detect and recognize text in natural scene images. Images containing natural scenes exhibit large variations of illumination, perspective distortions, image qualities, text fonts, diverse backgrounds, etc.

The majority of existing research works developed end-to-end scene text recognition systems that consist of complex two-step pipelines, where the first step is to detect regions of text in an image and the second step is to recognize the textual content of that identified region. Most of the existing works only concentrate on one of these two steps.

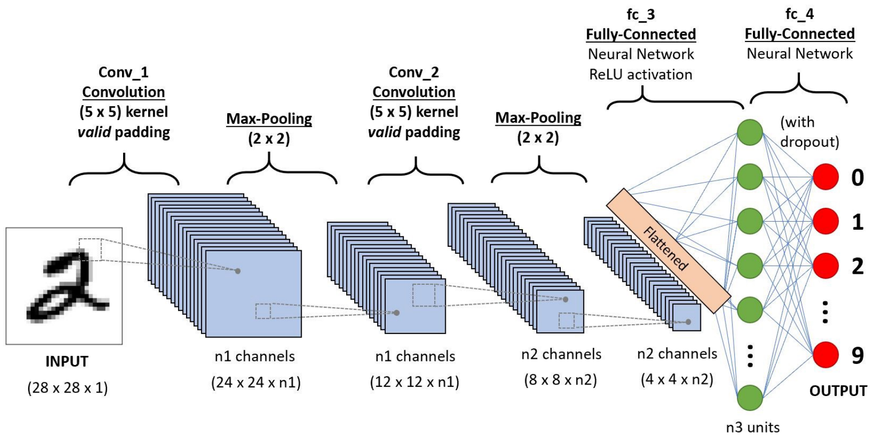
In this report, I have used Convolutional Neural Networks(CNN) to detect text with correct sequence. Here, I have taken those images that contains only digits and alphabets. CNN classify the digits among 32 different classes (1-class for 2, 2- class for 3,....., 8-class for 9, 9-class for A, 10-class for B,.....,32-class for Z). I have used Softmax that generated predicted logits for each digit or alphabet. At last, I have calculated the accuracy by matching predicted logits with actual labels of image digits or alphabets.

The rest of the paper is organized as follows: Section 2 explores past work on deep neural networks and on Photo-OCR. Sections 3 and 4 list the problem denition and describe the proposed method. Section 5 describes the experimental set-up, architecture and results. Conclusion and Future work ideas are discussed in Section 6.

Chapter 2

Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.



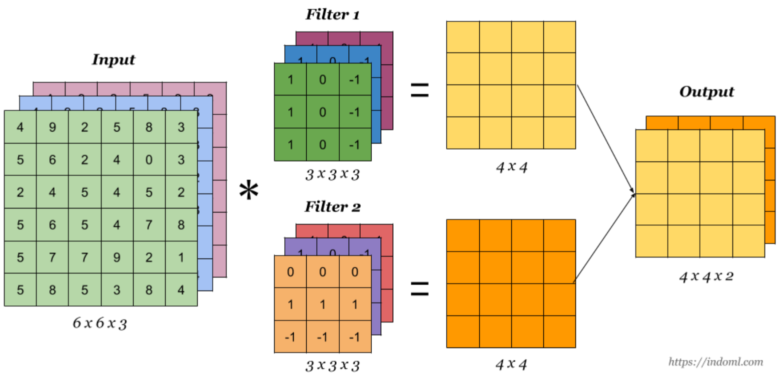
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

2.1 Convolution Layer: The Kernel

The green section resembles our 6x6x3 input image. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x3 matrix.



The Kernel shifts 16 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

2.2 Padding

There are two types of results to the operation - one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

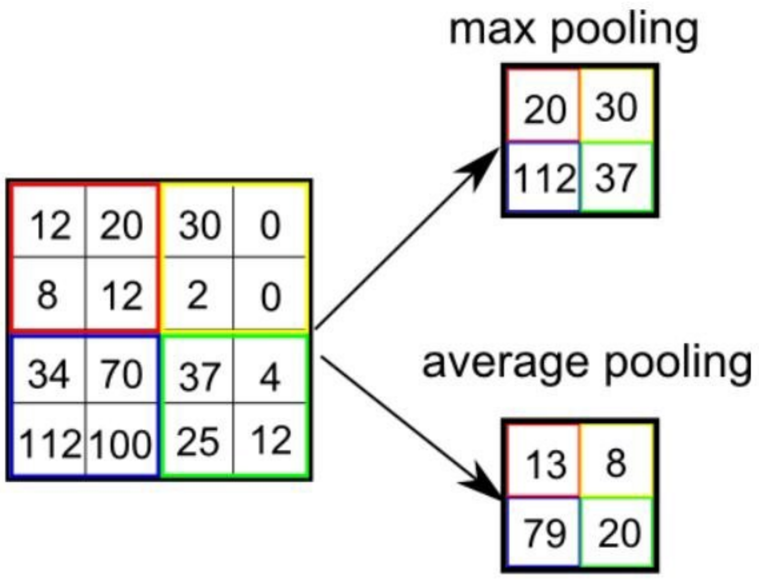
When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name Same Padding.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself Valid Padding.

2.3 Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.



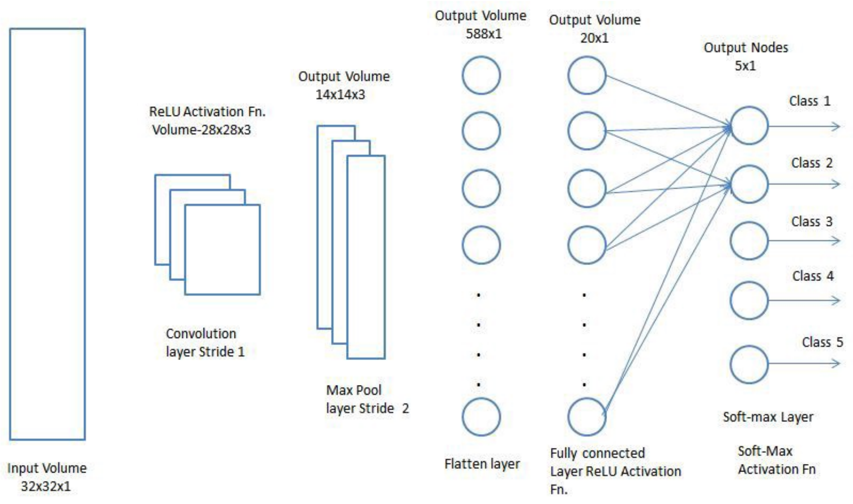
Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

2.4 Classification Fully Connected Layer (FC Layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector.



The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

Chapter 3

Types of OCR

There are more than one meaning for OCR. In its most general meaning, it refers to extracting text from every possible image, be it a standard printed page from a book, or a random image with graffiti in it (in the wild). In between, you may find many other tasks, such as reading license plates, no-robot captchas, street signs etc.

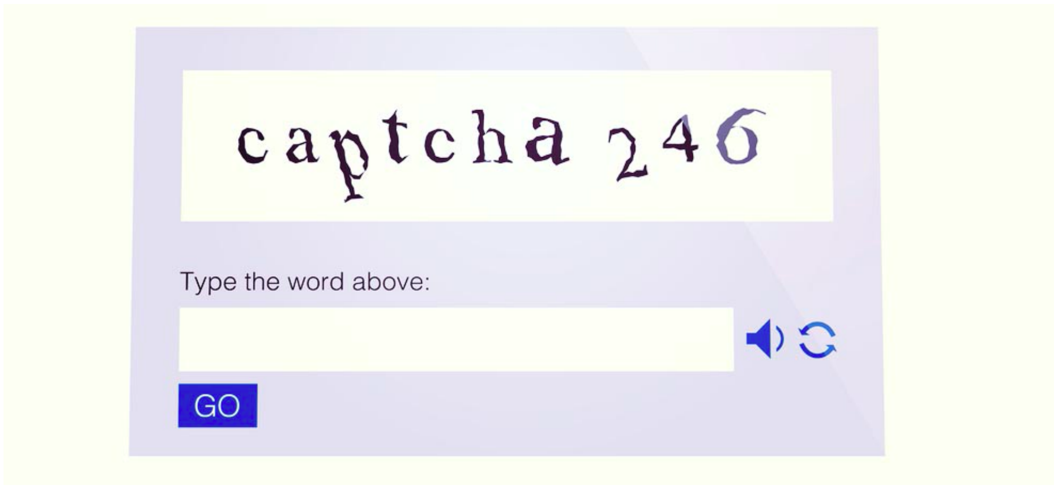
Although each of these options has its own difficulties, clearly. "**in the wild**" task is the hardest.

3.1 License Plates

This task, as most OCR tasks, requires to detect the license plate, and then recognizing its characters. Since the plates shape is relatively constant, some approach use simple reshaping method before actually recognizing the digits. Here are some examples from the web:

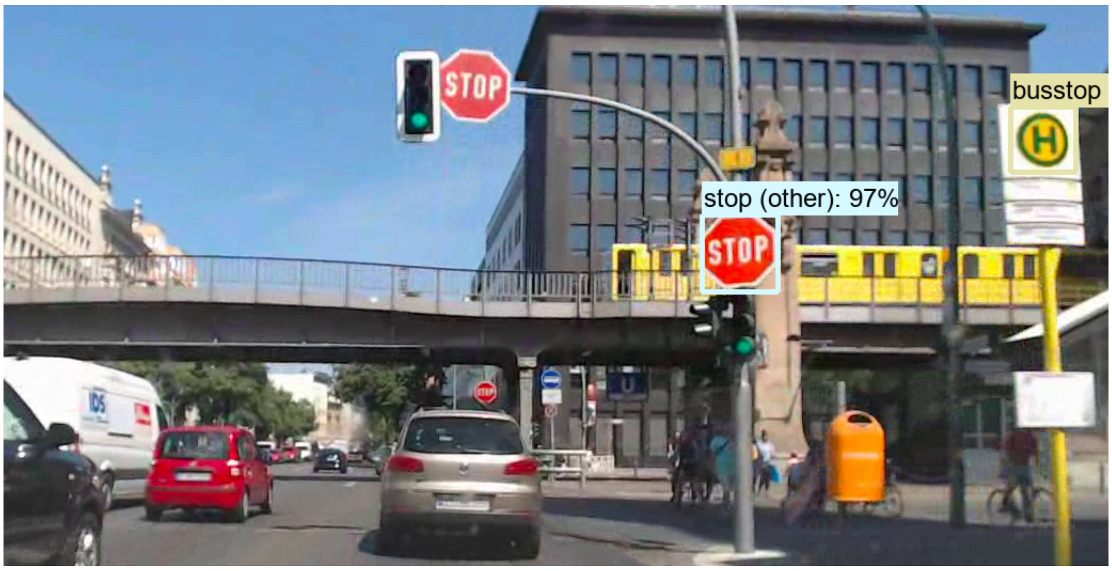
3.2 CAPTCHA

Since the internet is full of robots, a common practice to tell them apart from real humans, are vision tasks, specifically text reading, aka CAPTCHA. Many of these texts are random and distorted, which should make it harder for computer to read. I am not sure whoever developed the CAPTCHA predicted the advances in computer vision, however most of today text CAPTCHAs are not very hard to solve, especially if we do not try to solve all of them at once.



3.3 Traffic Signs

Traffic sign detection is a technology by which a vehicle is able to recognize the different traffic signs located on the road and used to regulate the traffic.



Traffic signs are detected by analyzes color information contained on the images having capability of detection and recognition of traffic signs even with bad visual artifacts those originate from different conditions.

3.4 OCR in the wild

This is the most challenging OCR task, as it introduces all general computer vision challenges such as noise, lighting, and artifacts into OCR.

3.5 PDF OCR

The most common scenario for OCR is the printed/pdf OCR. The structured nature of printed documents make it much easier to parse them. Most OCR tools are mostly intended to address this task, and achieve good result.

Chapter 4

Methods

As I have seen and implied, the text recognition is mostly a two-step task. First, I would like to detect the text(s) appearances in the image, may it be dense (as in printed document) or sparse (As text in the wild).

After detecting the line/word level I can choose once again from a large set of solutions, which generally come from three main approaches:

1. Classic computer vision techniques

2. Specialized deep learning

3. Standard deep learning approach (Detection)

4.1 Classic computer vision techniques

computer vision solves various text recognition problems for a long time.

These has three main tasks.

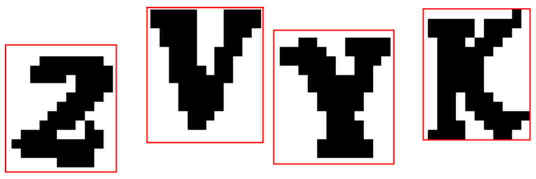
1. Apply filters to make the characters stand out from the background.

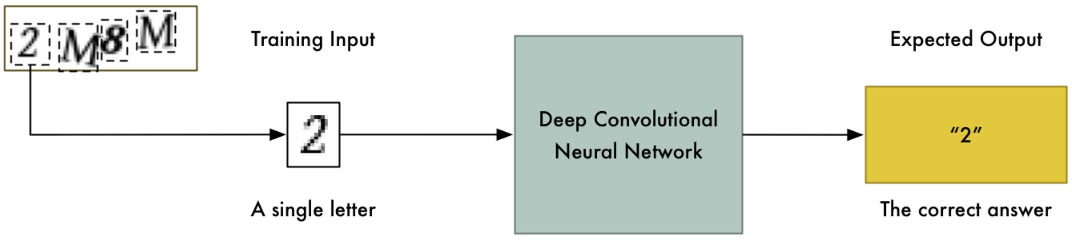
2. Apply contour detection to recognize the characters one by one.

3. Apply image classification to identify the characters.

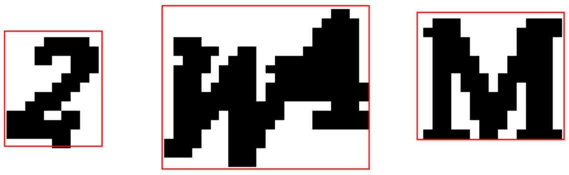
Clearly, if part two is done well, part three is easy either with pattern matching or machine learning.

However, the contour detection is quite challenging for generalization. It requires a lot of manual fine tuning, therefore becomes infeasible in most of the problem. Lets apply a simple computer vision script from here on some images from Captcha data-set. At first attempt we may achieve very good results:





But when characters are closer to each other, things start to break:

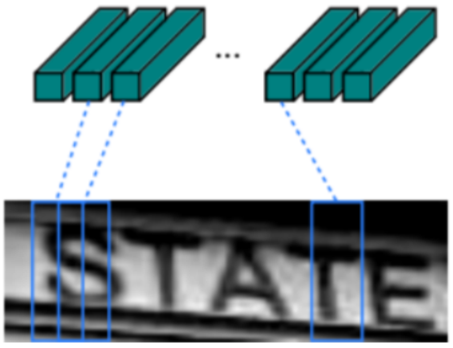


I have found out the hard way, that when I start messing around with the parameters, you may reduce such errors, but unfortunately cause others. In other words, if our task is not straightforward, these methods are not the way to go.

4.2 Specialized deep learning

Convolutional recurrent neural network a hybrid end to end architecture, that is intended to capture words, in a three step approach.

The idea goes as follows: the first level is a standard fully convolutional network. The last layer of the net is defined as feature layer, and divided into feature columns. Every such feature column is intended to represent a certain section in the text.



Second level, the feature columns are fed into a deep-bidirectional LSTM which outputs a sequence, and is intended for finding relations between the characters.

Finally, the third part is a transcription layer. Its goal is to take the messy character sequence, in which some characters are redundant and others are blank, and use probabilistic method to unify and make sense out of it.

4.3 Standard deep learning approach (Detection)

After detecting the words this method apply standard deep learning detection approaches, such as Single Shot Detector(SSD), You only Look Once(YOLO) and Region Convolutional Neural Network(RCNN).

However, SSD and other detection models are challenged when it comes to dense. I find it a bit ironic since in fact, deep learning models find it much more difficult to recognize digits and letters than to recognize much more challenging and elaborate objects such as dogs, cats or humans. They tend no to reach the desired accuracy, and therefore, specialized approaches thrive.

Chapter 5

Experiments

In this section I evaluate my presented network architecture on scene text detection/recognition data-set. I present the results of experiments for dataset, where the difficulty of the task at hand increases for dataset.

5.1 Experimental Setup

I have taken the help of publicly available captcha dataset. I have downloaded the dataset from the following url: https://wordpress.org/plugins/contact-form-7-image-captcha/. I have used Python to implement our code. I have used TensorFlow, numpy, Keras and any other libraries which were required during the implementation time.

OpenCV is a popular framework for computer vision and image processing. I have used OpenCV to process the CAPTCHA images. I used OpenCV findContours() function to detect the separate parts of the image that contain continuous blobs of pixels of the same color. It has a Python API so I can use it directly from Python.

Keras is a deep learning framework written in Python. It makes it easy to define, train and use deep neural networks.

TensorFlow is Google library for machine learning. I have been code in Keras, but Keras does not actually implement the neural network logic itself. Instead, it uses Googles TensorFlow library behind the scenes to do the heavy lifting.

I also use sklearn for labeling the dataset and generate one hot encoder and label encoder.

5.2 Final Architecture

Exploration of different neural networks and observation:
I have used Convolutional Neural Network due to their ability to work on the raw pixels of the images. The input to the network is the image where text shall be localized and characters and digits recognize.
I have used Adam Optimizer with learning-rate = 0.001, beta-1 = 0.9 and beta-2 = 0.999
Batch size: 2000
Epochs = 10
INPUT: [20x20x1]

CONV1 $\Longrightarrow$ 3×3×64 $\Longrightarrow$ SAME $\Longrightarrow$ RELU $\Longrightarrow$ [20×20×64]
MAXPOOL-2 $\Longrightarrow$ STRIDE-2 $\Longrightarrow$ [10×10×64]
CONV2 $\Longrightarrow$ 3×3×128 $\Longrightarrow$ SAME $\Longrightarrow$ RELU $\Longrightarrow$ [10×10×128]
MAXPOOL-2 $\Longrightarrow$ STRIDE-2 $\Longrightarrow$ [5×5×128]
CONV3 $\Longrightarrow$ 3×3×512 $\Longrightarrow$ SAME $\Longrightarrow$ RELU $\Longrightarrow$ [5×5×512]
MAXPOOL-2 $\Longrightarrow$ STRIDE-2 $\Longrightarrow$ [2×2×512]
FLATTERN $\Longrightarrow$ [2048]
DENSE $\Longrightarrow$ RELU $\Longrightarrow$ [100]
DENSE $\Longrightarrow$ SOFTMAX $\Longrightarrow$ [32]

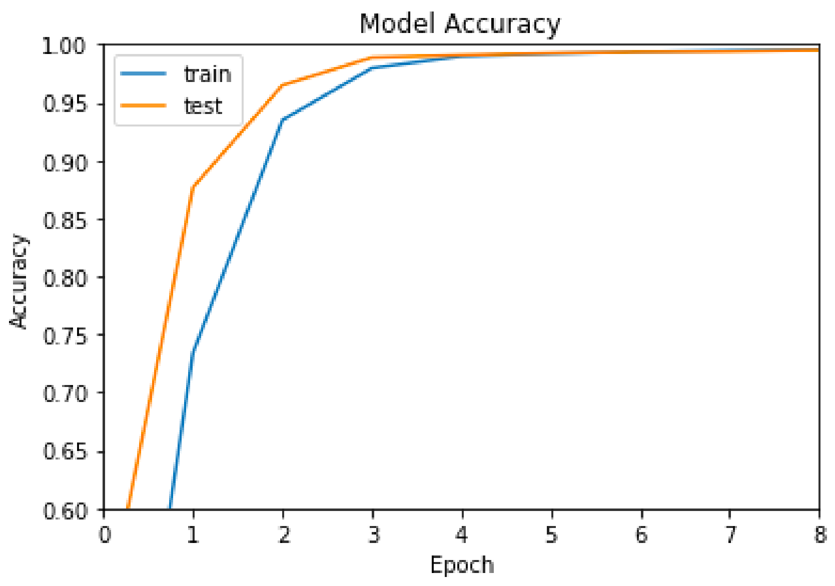
S.No.	Layer (type)	Output Shape	Parameters
1	conv2d 1 (Conv2D)	(20, 20, 64)	640
2	max pooling2d 1 (MaxPooling2)	(10, 10, 64)	0
3	conv2d 2 (Conv2D)	(10, 10, 128)	73856
4	max pooling2d 2 (MaxPooling2)	(5, 5, 128)	0
5	conv2d 3 (Conv2D)	(5, 5, 512)	590336
6	max pooling2d 3 (MaxPooling2)	(2, 2, 512)	0
7	flatten 1 (Flatten)	(2048)	0
8	dense 1 (Dense)	(100)	204900
9	dense 2 (Dense)	(32)	3232

Total parameters: 872,964
Trainable parameters: 872,964
Non-trainable parameters: 0

5.3 Results and Analysis

I have used Convolutional Neural Networks(CNN) to detect image with correct sequence. Here, I have taken those images that contains only digits and alphabets.
CNN classify the digits among 32 different classes (1-class for 2, 2- class for 3,....., 8-class for 9, 9-class for A, 10-class for B,.....,32-class for Z). The given image are like the ones the neural network is trained on. That text on a white background. It fails to recognize some of the characters if the letters look different or if the background is of somewhat different colour. Also note that the data-set on which neural network is trained on does not have some characters like 'L' or '1' or '0' etc so it will make wrong predictions on those. Also note that this has been trained on capital letters of English alphabet so it cannot detect small letters from the English alphabet.
I have used Softmax that generated predicted logits for each digit or alphabet. At last, I have calculated the accuracy and losses by matching predicted logits with actual labels of image digits or alphabets.

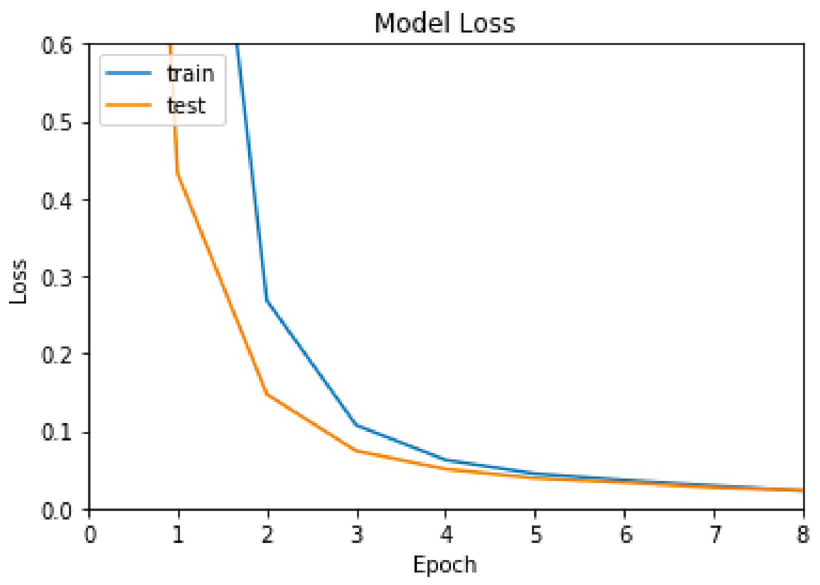
Here is an outcome.



I used categorical crossentropy for finding the loss. Categorical crossentropy is a loss function that is used for single label categorization. This is only one category is applicable for each data point. In other words, an example can belong to one class only.

$$L(Y,\widehat{Y}) = -\sum_{i=0}^{M}\sum_{j=0}^{N}(y_{ij} * log(\hat{y_{ij}}))$$

Where $\widehat{Y}$ is the predicted value.



Using this approach I was able to achieve a reasonably good character recognition accuracy on the test set, but not get good word accuracy.

S.No.	input	Output
1	SUMIT	5ZYL7
2	ANJANI	A5J5NL
3	ANSHUL	ANS5GZ
4	SHUBHAM	S5ZH5AM
5	N9238	N7338
6	OGC3	2GC3
7	ADITYA	WGX3YA
8	C9234	C2234

The discrepancy in character recognition rate and word recognition rate is caused by the fact that the model I trained for this task uses independent softmax classifiers for each character in a word. Having a character recognition accuracy means that there is a high probability that at least one classifier makes a mistake and thus increases the sequence error.

Chapter 6

Conclusion and Future works

I believe with this model I have solved OCR for short sequences. On my particular task, I believe that now the biggest gain I could easily get is to increase the quality of the training set itself as well as increasing its size for general OCR transcription.

One caveat to our results with this architecture is that they rest heavily on the assumption that the sequence is of bounded length, with a reasonably small maximum length N. For unbounded N, our method is not directly applicable, and for large N our method is unlikely to scale well. Each separate digit classifier requires its own separate weight matrix. For long sequences this could incur too high of a memory cost. Another problem with long sequences is the cost function itself. Its also possible that, due to longer sequences having more digit probabilities multiplied together, a model of longer sequences could have trouble with systematic underestimation of the sequence length.

One possible solution could be to train a model that outputs one word (N character sequence) at a time and then slide it over the entire image followed by a simple decoding. Some early experiments in this direction have been promising.

Perhaps our most interesting finding is that neural networks can learn to perform complicated tasks such as simultaneous localization and segmentation of ordered sequences of objects. This approach of using a single neural network as an entire end-to-end system could be applicable to other problems, such as general text transcription.

Bibliography

[1] Sermanet, P. & Chintala, S. & LeCun, Y. (2012) Convolutional Natural Networks Applied to House Number Digit Classification.

[2] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet (2014). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.

[3] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016

[4] A. Ng Machine learning, https://www.coursera.org/learn/machine- learning/.

[5] https://wordpress.org/plugins/contact-form-7-image-captcha/(dataset link)