**Summer Training Report**

**on**

**Street View House Numbers Classifiers**

**A report submitted in partial fulfilment of the requirements for the award of**

**Bachelor of Engineering**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**Adhyan Chawla**

**CO18306**

**Under the supervision of**

**Dr. Sunil K Singh**

**(HOD, Computer Science Department)**



**CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(DEGREE WING)**

Government Institute under (UT) Administration, Affiliated to Panjab University

Chandigarh

Sector 26, Chandigarh PIN 160019

**June, 2020**

**CANDIDATE’S DECLARATION**

I hereby declare that the work presented in this report entitled “Street View House Numbers Classifiers”, in fulfilment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology (Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of Dr. Sunil K Singh. The work reported in this has not been submitted by me for award of any other degree or diploma.

Date: 30 July 2020 Adhyan Chawla

Place: Chandigarh CO18306

**CERTIFICATE**

This is to certify that the Project work entitled “Street View House Numbers Classifiers” submitted by Adhyan Chawla, CO18306 in fulfilment for the requirements of the award of Bachelor of Engineering Degree in Computer Science & Engineering at Chandigarh College of Engineering and Technology (Degree Wing), Chandigarh is an authentic work carried out by him/her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Date: 30 July 2020 Dr. Sunil K Singh

Place: Chandigarh Dept. Of CSE

CCET (Degree Wing)

Chandigarh

**ACKNOWLEDGEMENT**

I would like to express my gratitude to my teacher Dr. Sunil K Singh who gave me the opportunity to do this wonderful project on the topic Street View House Numbers Classifiers. The project helped me learn how to do proper Research and I learned about many new things while doing the project.

**ABSTRACT**

Street View House Numbers Classifiers

Submitted by Adhyan Chawla

Deep Learning is the branch in focus these days. This has been widely used by data science and Machine Learning Engineers in order to make wide range of systems with focus on achieving accuracies in comparison with human level performance. Deep learning is sought after to have made advancements in various fields these days like computer vision, machine vision, speech recognition, natural language processing, audio recognition, and many more.

Deep learning is a broader concept with certain number of branches under it. One such branch is Convolutional Neural Network which deals in solving deep learning related to images. CNN requires minimal amount of pre-processing of data which is the reason why this is preferred over other networks.

The Street View House Numbers Classifiers is completely based on the idea of Convolutional Neural Networks. We classify digits with the help of this branch of deep learning. The structure of these networks has been biologically inspired. ConvNets can automatically learn a wide range of unique set of features optimised for a given task.

In this project, I have used several convolutional layers followed by a number of pooling layers. I have implemented most of the functions in keras framework. After using a different set of layers, I am able to establish a new-state-of the-art of 95.28% accuracy on the SVHN dataset with the help of 1 (input) + 5 (Convolutional + Pooling) +1 (output) layers. In this paper, I analyse the benefits of different methods of building a model using ConvNets and certain features in it. Matplotlib and Seaborn have been used for plotting graphs, NumPy for performing basic operations on arrays and scikit learn in certain areas of the project.

**LIST OF FIGURES**

FIGURE PAGE

Figure 1 2

Figure 2 3

Figure 3 3

Figure 4 4

Figure 5 7

Figure 6 7

Figure 7 9

Figure 8 9

Figure 9 10

Figure 10 11

Figure 11 11

Figure 12 11

Figure 13 12

Figure 14 12

**CONTENTS**

Student’s declaration i

Certificate by the guide ii

[Acknowledgement iii](#_TOC_250001)

[Abstract iv](#_TOC_250000)

List of figures v

Chapter I - Introduction 1

Chapter II – Dataset Pre-Processing 2

Chapter III – CNN Models 5

Chapter IV - Application 10

Chapter IV - Visualization 11

Chapter V - Conclusion 13

Chapter VI – Course Certificates 14

1. INTRODUCTION

The MNIST dataset has been widely in use for classifying numbers but for this project I have used SVHN dataset, which could arguably call as the next popular dataset used very often in solving computer vision problems when the idea of classifying the digits comes into the picture. Although the dataset has been pre-processed but it requires some level of pre-processing before training the model with the help of the-state-of-the-art neural network architectures.

Initially I started with the MNIST dataset, but then I moved to the SVHN dataset just because it offers better results than the MNIST one when it comes to classify the digits. This is a real-world image dataset that offers minimal amount of pre-processing. SVHN is labelled with 6,00,000 images. It also comes from a significantly harder real-world problem of recognising digits and numbers in natural scene images. The images lack any contrast normalisation, contain overlapping digits and distracting features which makes it a much more difficult problem compared to MNIST.

The dataset consists of 73,257 digits for training and 26,032 digits for testing. It also comes with an additional 531,131 somewhat less difficult samples that can be used as extra training data. It is recommended to use the full dataset (630K images) when evaluating algorithms as it is common practice in the majority of the literature but I left the extra-digit dataset for further testing purposes just because of the computational problems and unavailability of good hardware.



Fig 1: Examples of images from the SVHN Dataset

1. DATASET PRE-PROCESSING

As I just mentioned above that this dataset has been pre-processed to some extent but still requires some amount of pre-processing before I use it in the model in order to determine the accuracy of my model.

Pre-Processing is done on the dataset so as to ensure that the data is inside the suitable format and within a consistent scale or range. This is done so as to increase the performance of the models and ensure for comparison of results.

In the first-ever paper on the street view house number classifiers published some years back, Netzer et al used minimal amount of pre-processing in which he just converted the coloured images to greyscale images.

To start with, firstly I transformed the axis of my training and test sets from (height, width, no. of channels, no. of images) to (no. of images, height, width, no. of channels) just for the purpose of plotting the images with the help of Matplotlib library which is very useful for plotting figures and graphs.

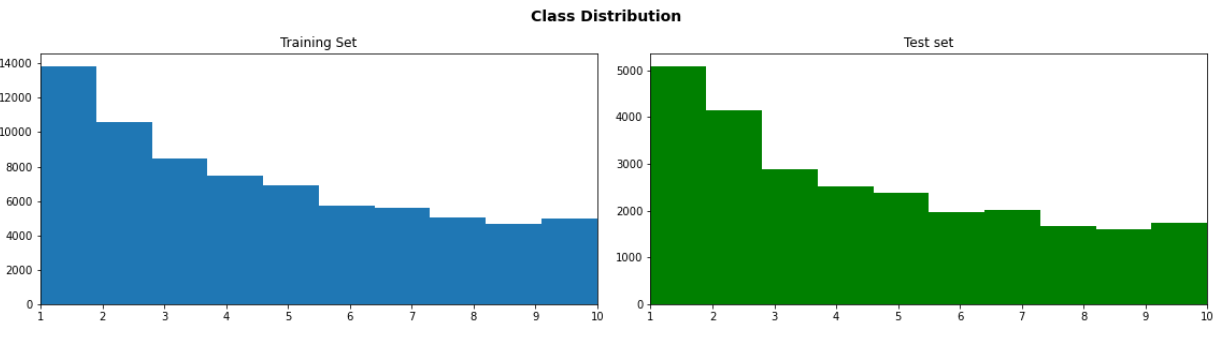


Fig 2: The graph represents the number of images with respect to each label in the training and test sets. The x- axis represents the labels and the y axis represents the number of images for each label. The labels are the digits itself which are to be classified in the images.

Further, I had to change the label 10 to label 0 since we are interested in classifying single digit in my result.

I split the training set into training and validation sets with the help of inbuilt train\_test\_split function, providing randomly 13 percent of my training data to the validation set. This has been done so as to calculate the validation set accuracy and check if our training set data matches with the predicted values and calculate the loss.

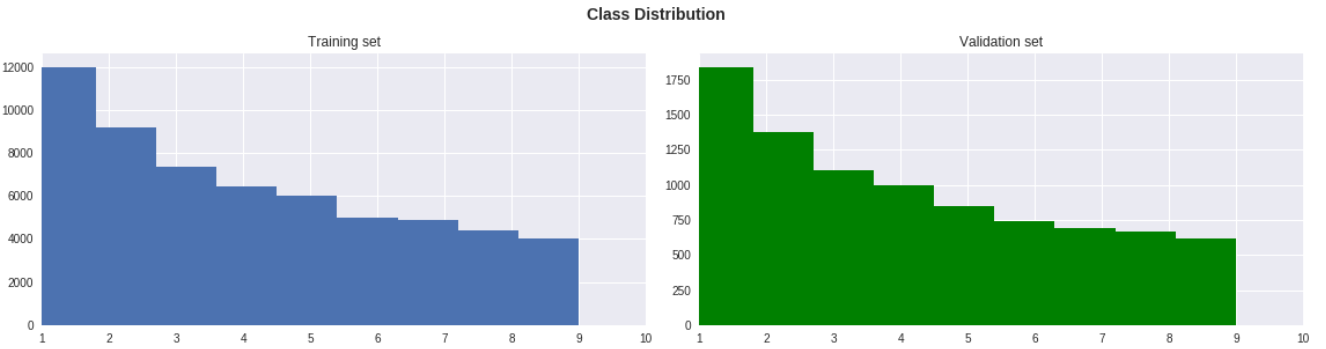


Fig. 3: The above graph shows the distribution of my training set into the training and validation set. The x - axis represents the labels from 0 to 9 while the y – axis represents the number of images corresponding to each label.

Now comes one of the most important parts of pre-processing. Getting started in applied machine learning can be difficult, especially when working with real-world data. This is done when you have categorial data i.e. data in the form of labels which is present in my case. Machine learning algorithms do not operate on categorial data directly. They require the input data as well as the output data to be numeric. So, in order to convert the categorial data into numeric data, we need some sort of application. One such way to do convert the categorial data into the numeric one can be done by one hot encoding. A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value. I have used one-hot encoding in my project. This allows me to make the implementation more efficient.

Another technique that I have used in my project is Data Augmentation. Data Augmentation is a technique that used to increase diversity of training data without adding more data in the training set. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. However, most approaches used in training neural networks only use basic types of augmentation. While neural network architectures have been investigated in depth, less focus has been put into discovering strong types of data augmentation and data augmentation policies that capture data invariances. For data augmentation, I have used keras and imported ImageDataGenerator.

There are a couple of techniques which can also be used in data pre-processing which are Normalising the data, conversion of coloured images into grayscale, etc. I haven’t used these techniques in my project since my dataset was pre-processed and required minimal amount of pre-processing. This did not have a good impact on my results so I didn’t use these techniques for pre-processing.

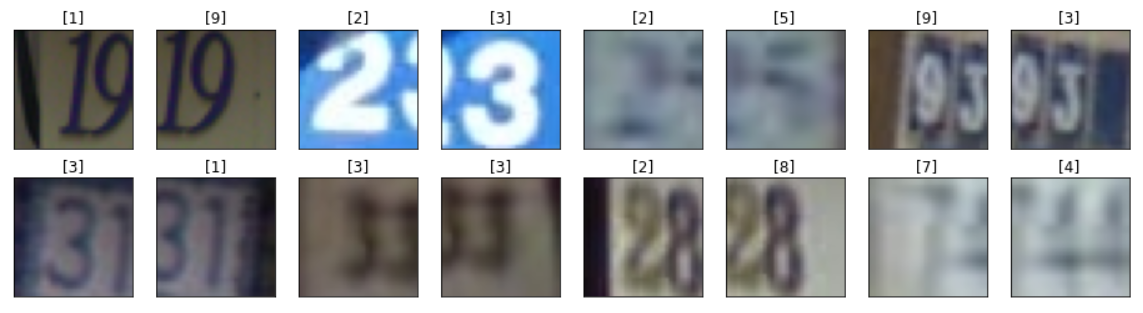


Fig. 4: These are some examples of my training set after pre-processing. These images have been plotted with the help of Matplotlib library.

1. CNN MODELS

This was the most tedious part of the project. Deciding a model in order to achieve good results in accuracy and minimal loss seemed like a daunting task. I used a couple of models in my project which varied from each other in terms of the number of layers and the values of dropout terms.

Firstly, I used an auxiliary model in order to calculate the learning rate for good optimised results. I used AMSGrad (a variant of Adam Optimiser), in which I set a call back in an auxiliary model which will gradually increase the learning rate of the optimizer.

The auxiliary model has 6 CONV layer and 1 Dense layer. It can be showed with a flowchart given below.

INPUT->[CONV2D->RELU->BN->CONV2D->RELU->MAXPOOL2D->DROPOUT]x3->FC->RELU->DROPOUT->SOFTMAX

INPUT: Every image inputted in the model has a size of 32X32X3 pixels. I did not convert the RGB channels to 1 grayscale channel as it did not bring much difference to the results.

CONV2D: I have convolved with the help 32, 3X3 filters, the padding has been kept same with stride equal to 1. Convolutional layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. To extract features from the input image is the primary purpose of Convolution in case of a ConvNet.

RELU: also called as Rectified Linear Unit, is an additional layer of this network. It is applied per pixel and replaces all negative pixel values in the feature map by zero. Convolution is a linear operation and since most of the real-world data is non-linear, RELU function is used to introduce non-linearity in ConvNet. RELU layer will apply an elementwise activation function, such as the max (0, x) thresholding at zero. This leaves the size of the volume unchanged.

POOL: This layer has been used to perform down sampling operation along with spatial dimensions i.e. height and width and increases the number of channels to make it deeper. Max-pooling is used, while building this convolutional neural network. Max Pooling defines a spatial neighbourhood and take the largest element from the rectified feature map within that window. Here, 2x2 window is used and the largest element is selected from that window for the feature map.

FC: represents the fully connected layer, resulting in the volume of size 1x1x10, where each of the 10 numbers correspond to a class score, such as among the 10 labels [0-9] in SVHN dataset. This layer is a traditional multilayer perceptron that uses SoftMax activation function in the output layer. This layer connects the input features to the output and hence, classifies the images into different classes based on the labels.

Dropout has been used in many layers to prevent the overfitting of the data or when the network has a high variance. This means dropping out some units from the layers in the neural network. The Dropout value lies between 0 and 1 which shows the number of units dropped from a particular layer considering the total number of units present in the input layer. After introducing dropout in the layer, the units work without the units that have been dropped. This is very useful in increasing the performance of the model.

I have also used Batch Normalisation at various stages. Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. Again, this has been added to increase our model’s performance.

So once the dataset is fitted into the model, we calculate the learning rate with the help of AMS Grad optimiser. We train this model in batches which is ideal for one’s computation speed. The size of the batches depends on the memory available. In this project, I have taken batch size as 128 which did not create any problems while calculating the learning rate.

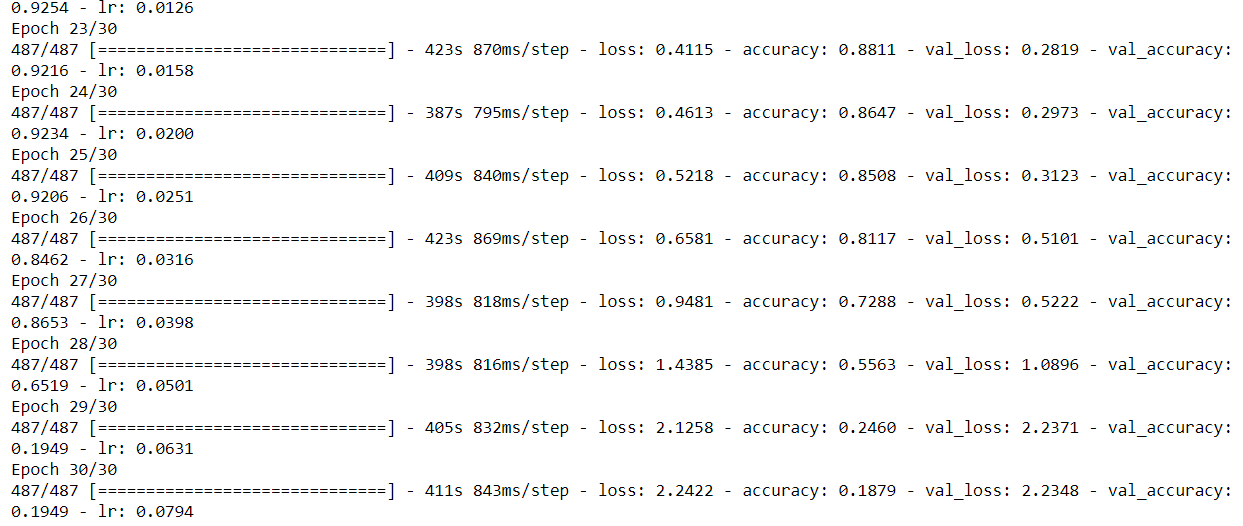


Fig. 5: This shows how the learning rate improved on training in batches.

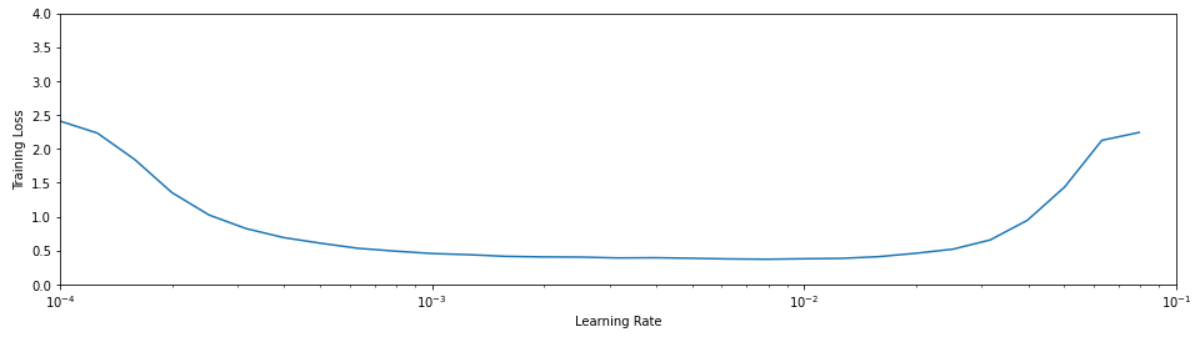


Fig. 6: One can observe that the loss follows a very specific trajectory: a rapid drop followed by a relatively flat line which shoots back up after a certain point. Thus, it is better to choose a learning rate in the region where the loss is stable; a reasonable choice would be **learning rate = 0.001** (or 1e-3).

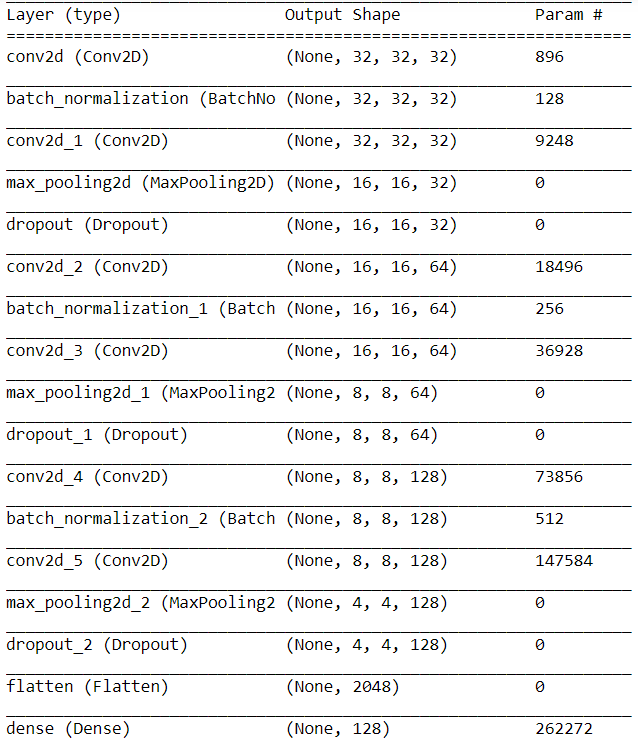
After identifying a suitable learning rate from the auxiliary model, I used the learning rate in the actual model which is trained in a similar fashion to compute the accuracy of the validation set and check how much the predicted values differ from the ground true values.

While training the model, I again divided my training set into different batches of 512 size. I have used Early Stopping while training so as to stop the Epochs because they can lead to overfitting of the data i.e. high variance which will lower down the accuracy. So, by introducing early stopping, the epochs are stopped once the model performance stops improving on a holdout validation dataset.

The model used in order to predict whether the values of validation set differ from training set or not is the same as the auxiliary model which was used to compute the learning rate for the actual model except for the fact that after every layer, I increased the number of filters in the powers of 2 from 32 to 64, to 128 and so on. This is done to capture more complex patterns from the inputs as the network advances. Another reason, especially for convnets, one down sample the image through the layers. Increasing units tries to tolerate this resolutional loss by increasing model capacity.

INPUT->[CONV2D->RELU->BN->CONV2D->RELU->MAXPOOL2D->DROPOUT]x3->FC->RELU->DROPOUT->SOFTMAX

Now let us have a look at the number of parameters used in the model for prediction the validation set.



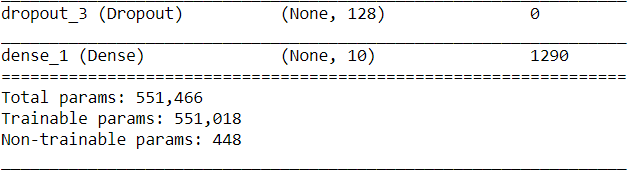


Fig. 7: The above figures shows the number of parameters that have been used in every operation. The total number of parameters is very high but since one is concerned about the accuracy of the model, this number does not matter while learning from these parameters. Also, the trainable parameters are the ones that are updated in backpropagation while the non-trainable ones are those which have been used in batch normalisation i.e. used in computing means and variance or the number of hidden layers itself, thereby the backpropagation do not update these weights. Only the trainable ones are updated.

After training this model, I am able to calculate the accuracy of the validation set as well as the loss. After training with different number of layers, I found out that the model which I have trained with (6 + 1) layers gave me 95.56% of accuracy with 0.1848 of test loss, which is quite an achievement but there is always a room for improvement as there are models which have brought results over 97% and the human level accuracy is considered as 98%.

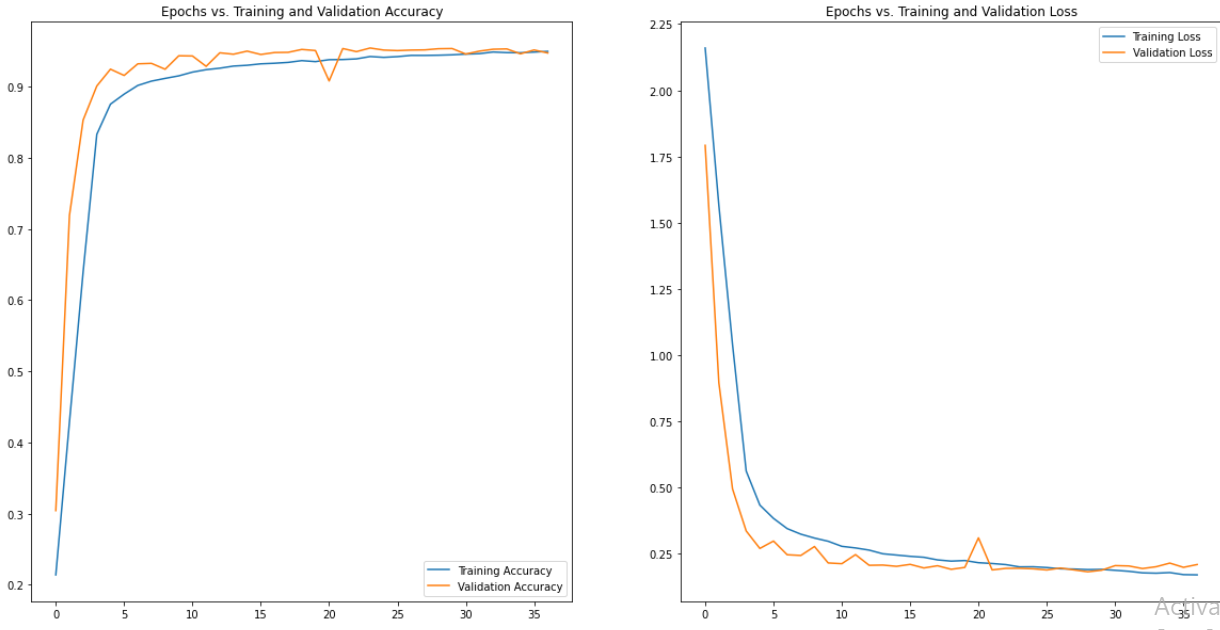


Fig. 8: The x – axis represent the number of epochs and the y – axis represents the accuracy in the left graph whereas in the right one, the x – axis represents the number of epochs and the y – axis represent the loss.

The results are quite good after training this model but can further be improved by:

* Change the way the images are transformed in the augmentation process.
* Change the architecture of our model by adding extra blocks, changing the kernel sizes, making it deeper, etc.
* Train multiple CNNs and make ensemble predictions.
* Use some of the extra data which can be found along with the original dataset.

1. APPLICATIONS

Improving Map Services

In addition to serving millions of users daily with panoramic imagery, Street View images taken worldwide are used to improve the accuracy of maps and address geocoding. House numbers detected in the Street View imagery contributes to that goal in various ways. First, detected house numbers provide us with a view angle in addition to the geocode of the address. This view angle is useful, for instance, when users search for an address in Street View, or when displaying their final destination in driving directions. Without the house number-based view angle the panorama presented will be a default one, which very likely does not point to the desired building. Thus, the user will need to turn the camera by hand in order to achieve the desired view. With the 6-house number-based view angle, the user will be led to the desired address immediately, without any further interaction needed. Second, house number recognition helps improve address geocoding. In most countries, very few addresses have associated geographic coordinates. The geocode of the majority of addresses is therefore usually computed by interpolating the geocodes of neighbouring addresses. Such interpolation introduces errors which may be significant, and may result in poor user experience. With the vehicle’s location, and the house number-based view angle, a better geocode of the building of interest can be computed in a straightforward way.

1. VISUALISATIONS

The below image is an input image and the images below this input image are the feature maps which have been displayed so as to get a sense of how our model learns the features in each convolutional layer.

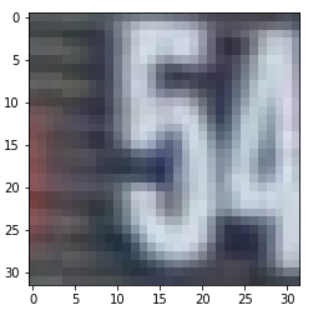


Fig. 9: A random input image

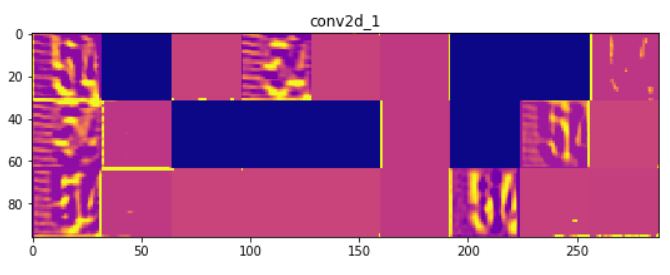


Fig. 10: Convolutional layer 1

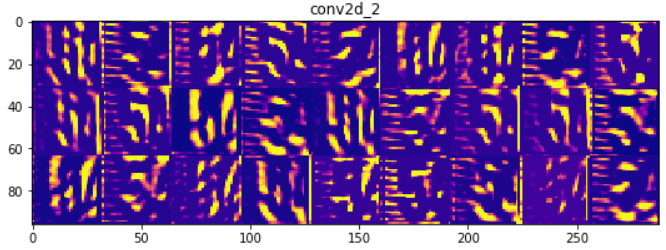


Fig. 11: Convolutional layer 2

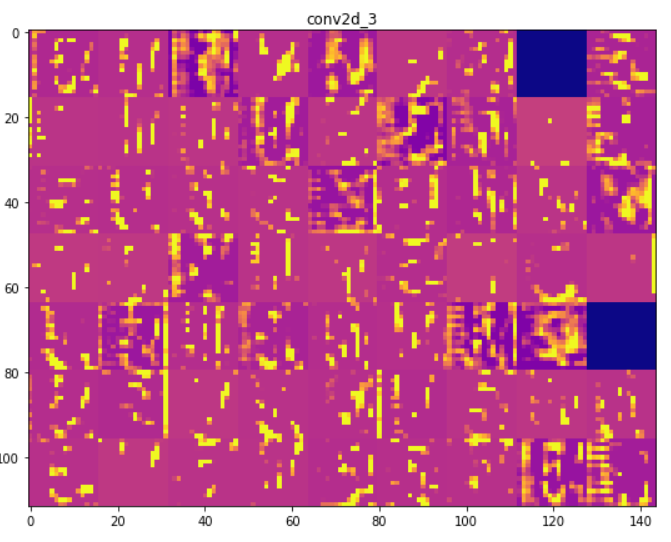


Fig. 12: Convolutional layer 3

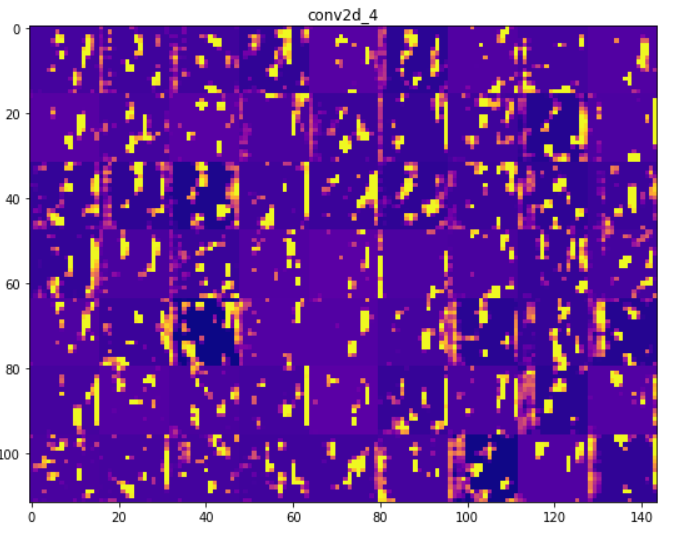


Fig. 13: Convolutional layer 4

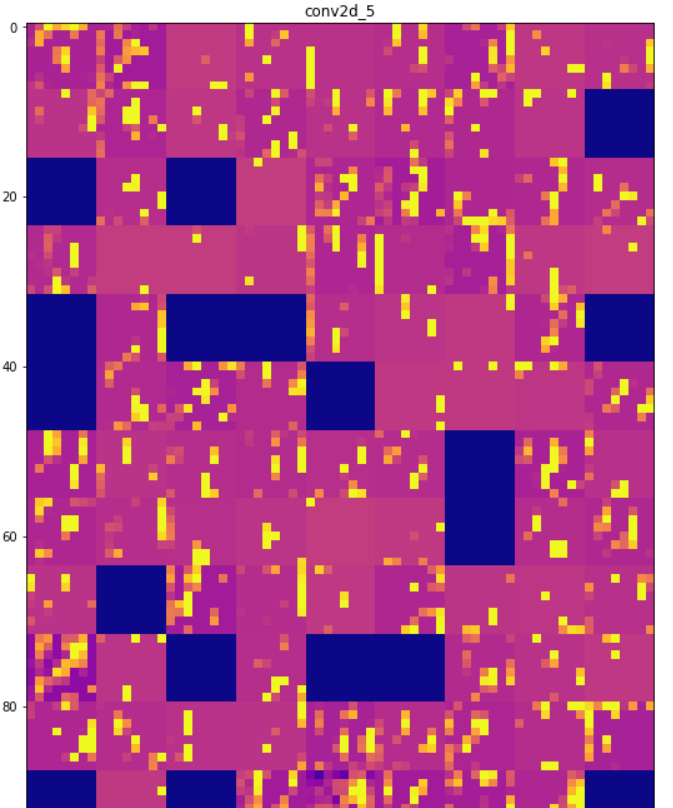


Fig. 14: Convolutional layer 5

In the above, one can see the main idea behind Convolutional Neural Networks, i.e. that as one goes deeper into the layers, higher-level features are learned by the models. At first, it is easy to understand what is going on but gradually, one finds it difficult to keep up with the model's learning process.

1. CONCLUSION

In this project, I have trained a Convolutional Neural Network to recognize the digits in the Street View House Numbers dataset (Format 2). In particular, I have performed some minimal pre-processing of the data, I have augmented the data in various ways, I have created an auxiliary model in order to find which learning rate I should choose for our optimizer and finally, I have trained the final CNN and evaluated it on the test images data. Furthermore, I have provided one useful visualization (feature maps) so as get a sense of how the model actually works and not view it as just a black-box process. Finally, it should be noted that there is quite a bit of room for tuning and different architectures so as to improve the accuracy of the model; nonetheless, the results are pretty good given the simplicity of our approach.

**COURSE CERTIFICATES**

