Analytical Data Mining
Project
Image Classification in Real-World Scenarios

For this project, we will be predicting the street numbers from real-world image data using Google's SVHN (Street View House Number) dataset. The dataset contains over 600,000 digit images and comes with an unsolved hard problem on recognition of digits and numbers in their natural state. The dataset contains images from Google's Street View.

Deep learning requires large amounts of training data. The models created are very effective in their tasks like computer vision, speech recognition etc. and have a very high accuracy

The task of computer vision is carried out by the implementation of Convolutional Neural Networks (CNNs). These require a lot of data to be trained that is provided by our dataset.

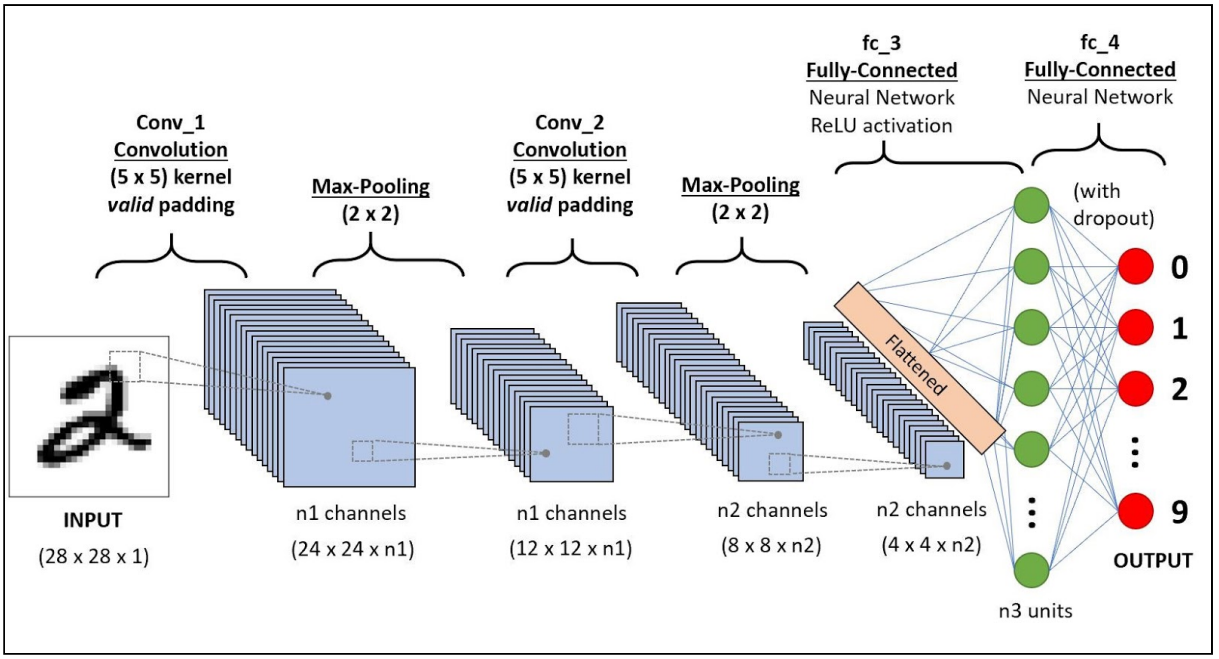


Fig - CNN
Source :
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Common image data parameters are usually the number of images, image height width, pixel count etc.

The Project Architecture :

1. Training Testing Data Preprocessing
2. Model Architecture and its Implementation (Code)
3. Training and Testing the Model
4. Model Implementation on new data

Preprocessing of the training and testing data is an important aspect of the classification model.

Preprocessing of the data is done in order to reduce the complexity of steps and to improve the accuracy of the algorithms used. Since it is not feasible to have a particular algorithm for every data value we use, it is better to convert our data into a general form in order for our algorithm to work.

There are several methods for preprocessing of data :

Converting to Grayscale (Dimensionality Reduction): In some tasks, color of the image is not required for the task to be implemented correctly. Color adds more information to the image and increases complexity and storage.

Standardization: In some algorithms, especially CNNs,all the images of the dataset need to be resized to a uniform size for the algorithm to work efficiently.

Augmentation: Augmentation of our dataset with different versions of images helps in increasing the size of the dataset to train the algorithm with varying versions of the images. Scaling, rotations and other affine

transformations are typical. This is to make sure that our model and identify new unseen data correctly.

Mean and Standard Deviation: It can be useful to check the ‘mean image’ or standard deviated image obtained from our dataset. This could give us insight into our training and testing data on which our model will work.

The implementation of the model itself is has been done using several methods to produce varying accuracies. We focus our task on the preprocessing part of the project. Implementing varying methods of modifying the train/test data and observing the results. Varying methods of preprocessing can help us gain insight into how, in our case, Street View Numbers, a real-world image classification task and other similar datasets could have some optimal data preprocessing.

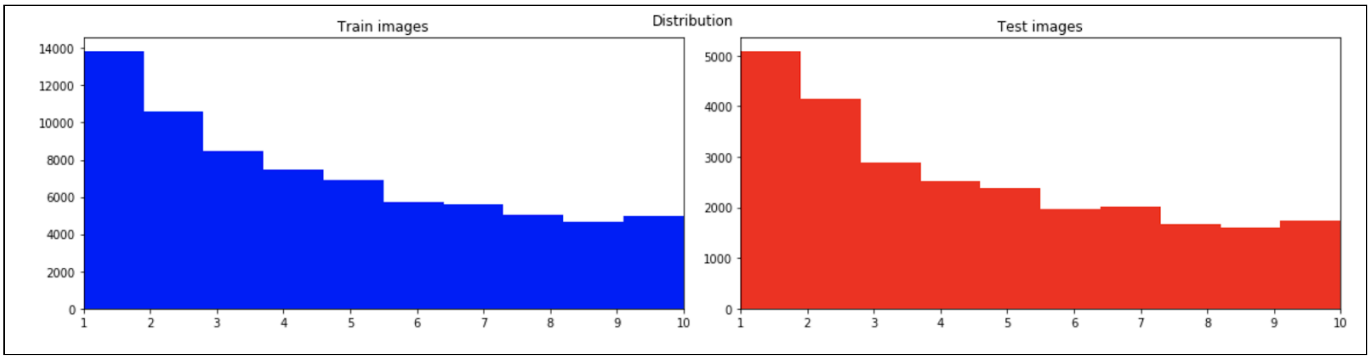
Of course, the selection of appropriate processing techniques is based on our dataset and our task at hand. We try different preprocessing methods and their combinations to gain some insight into the data from the SVN dataset. Such methods could be useful for tasks similar to the project we implement (text detection, etc.)

1. PREPROCESSING OF DATA

For the preprocessing portion, we:

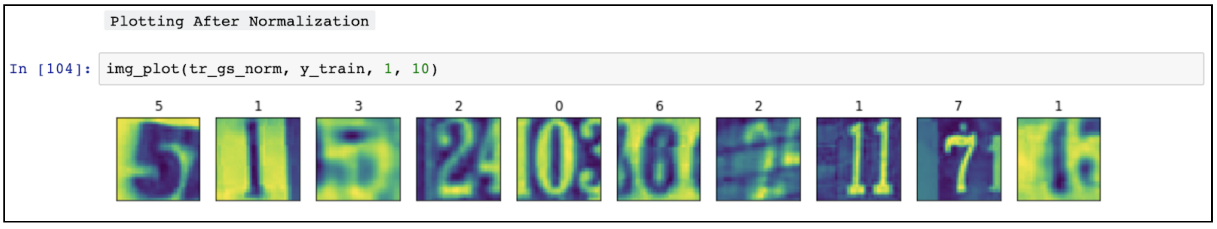
a. Bound our images to the portions we want for training and testing of data
b. Visualize and explore our data
c. Convert our data to black and white from colored
d. Save the processed data to use it on our model

Dataset Distribution of our images:



Datasets Visualized

Images after normalization:

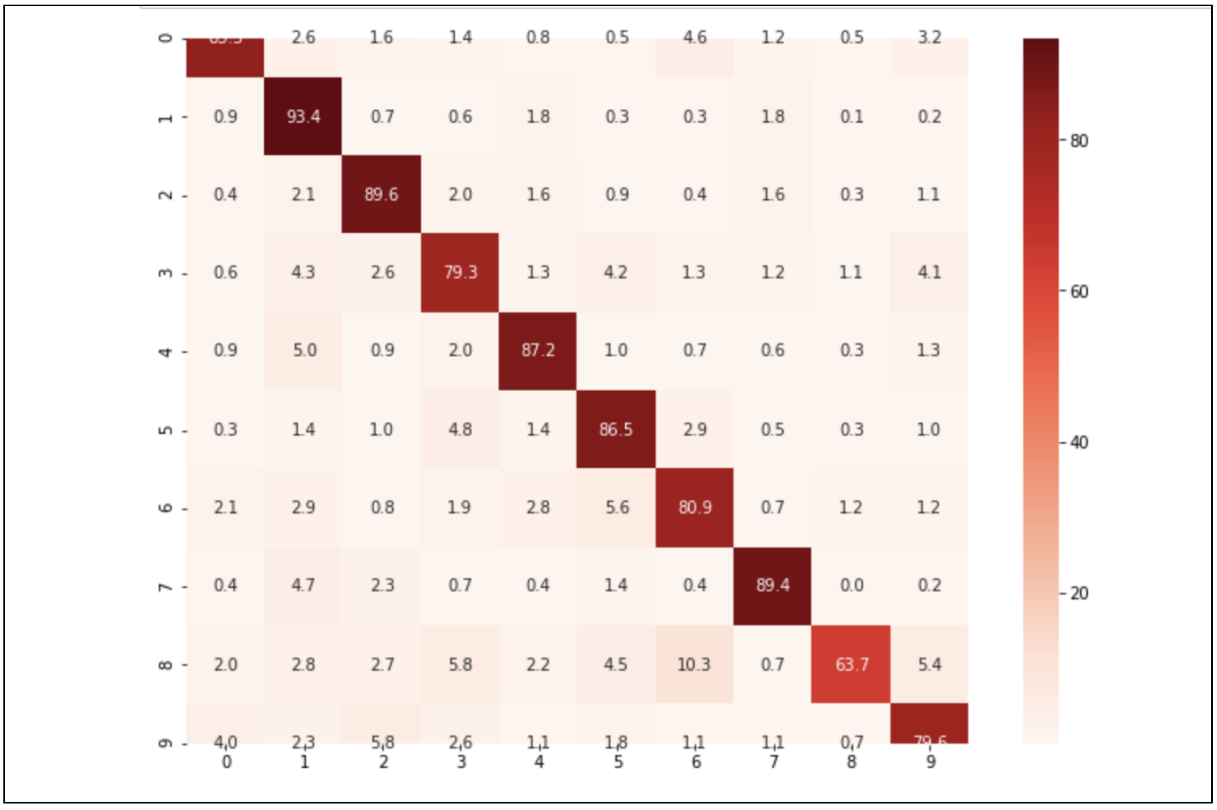


2. CNN MODEL

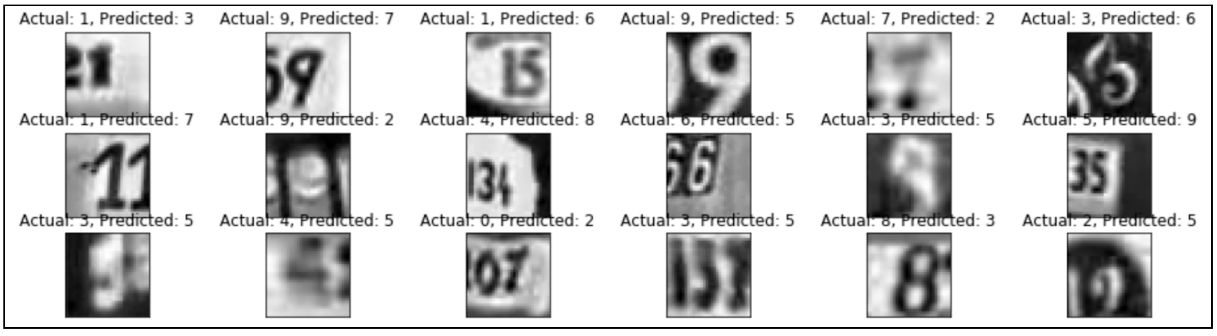
The CNN Model is created usng the Basic Conv Architecture :

INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] -> DROPOUT -> [FC -> RELU] -> FC

Our results and observations are as follows:



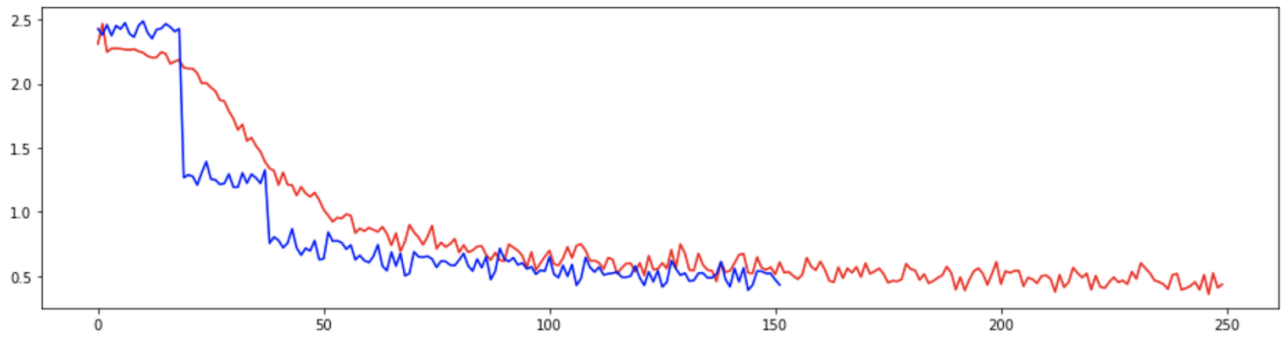
Confusion Matrix from our Model Predictions



Incorrectly predicted Images

```
import matplotlib.pyplot as plt
plt.plot(train_loss ,'r')
plt.plot(valid_loss, 'b')
```

```
[<matplotlib.lines.Line2D at 0x145ac7290>]
```



Our training to validation losses plot

```
Epoch 2 completed out of  2
Test Accuracy : 86.6279963122 %
Time usage: 0:01:42
```

Final Model Accuracy and Time Taken

References:

- Dataset -
  - http://ufldl.stanford.edu/housenumbers/
- Paper -
  - https://arxiv.org/pdf/1811.04256.pdf
- Preprocessing -
  - https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258
  - https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
  - https://medium.com/project-agi/getting-started-with-street-view-house-numbers-svhn-dataset-a96e76962504
  - https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/