Convolutional Neural Network on Street View House Numbers Images using Keras

Deepshikha Wadikar  
School of Computing  
Dublin Institute of Technology

Dublin, Ireland  
deepshikha.wadikar@mydit.ie

*Abstract*—This experiment attempts to learn digits from Street View House Number (SVHN) dataset using Convolutional Neural Network model. CNN is trained on train and extra training samples.

Keywords—Convolutional Neural Network, street view house number, image classification, keras.

# Introduction

Image Processing is a method to perform some operations on an image, to get enhanced image or to extract some useful information from it. Deep Learning uses neural networks to learn useful representations of features from data. A Convolutional Neural Network or most commonly known as CNN or ConvNet is an Artificial Neural Network that is popularly used for image classification. In order to test these models, various datasets are available. One particular dataset of interest is Street View House Numbers dataset. This dataset contains images of house numbers. There are two types of datasets, full image view and cropped image view. For this experiment, I will be using cropped images. Every image contains an image of a number and its subsequent label. The cropped images are of .mat file format. The dataset contains 73,257 images in train data set, 531,131 images in extra data set and 26,032 images are present in test data set.

The general architecture of the CNN model is shown below:

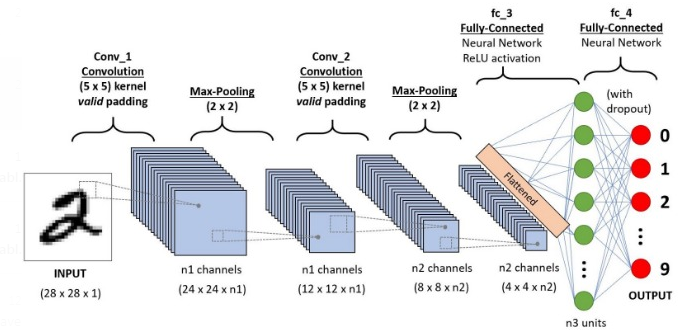


Fig.1. CNN Model Architecture

Here in this experiment for training the model I have used Keras implementation. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, It enables fast experimentation with deep neural networks, it is user-friendly, modular and extensible.

# Model Design

This experiment was designed n order to understand the architecture flow of Convolutional Neural Network model to do image processing. The model was trained on both train and extra dataset by concatenating both the datasets. The model was implemented using tensorflow and keras.

Firstly all the train, test and extra datasets are downloaded in python using *‘urllib’* library. Then the model is built on train data and extra dataset by concatenating the two files. As the data increases the performance of the model to train the model in CNN will be increased. The output variables are having classes from 0-9 as labels. The training and test data set images are normalized. Then one hot encoding is performed to convert the output of each class into binary.

The keras Sequential () model is imported from ‘*keras.model’* library. Keras sequential model is the simplest model defined in the Sequential class which is a linear stack of layers. On the top of the basic Sequential model the Convolutional layer, Max Pooling, Flatten and Dense layers are added by importing the layers using *‘keras.layers’* library. The model summary is displayed in the below figure –

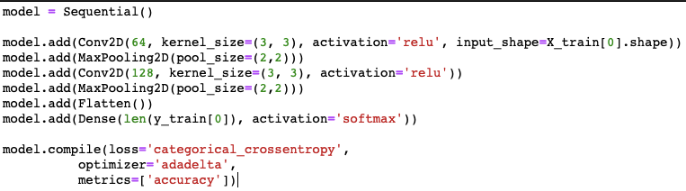


Fig.2. CNN model

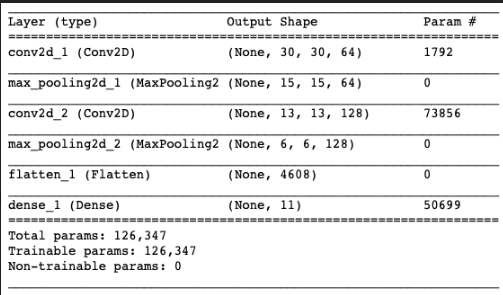


Fig.3. CNN Model Summary

The Convolutional layers are the major building blocks used in CNN model. Conv2D layer is used. For first convolution layer of the model parameters are set like filters = 64, kernel\_size = (3,3), activation function ‘relu’ is used and the shape of the input is set. Next the MaxPooling2D layer of kernel\_size (2,2) is used. Then again Conv2D layer with 128 filters were added, kernel\_size = (3,3), activation function ‘relu’. Again MaxPooling2D layer is applied and then the output is flattened using flatten layer then the flattened output is fed into the Dense layer which is the output layer of the model in which activation function is set as softmax.

Now these layers of Convolutional Neural Network are – Convolutional layer which acts as the filter for the input in this layer we have parameters like filters, kernel\_size and activation\_function, Max pooling layer2D is used which is used for spatial data. Here the pooling size is provided. This layer if used to extract the important information will provide the output. Then the flatten layer is used to flatten the data to convert the 4 dimensional matrix into 2 dimensional. To build the model loss is used as *‘categorical\_crossentropy’*, optimizer is used as *‘adadelta’* and initially performance of the model is based on calculating accuracy.

Adadelta (an adaptive learning rate method) optimizer is a more robust extension of Adagrad that adapts the learning rates based on a moving window of gradient updates.

In *‘model.fit’* command the train data is splitted into train and validation and the validation is set as 10% and train data set into 90%. Then batch size is set as 128 that means for each epoch multiple batches of 128 will be executed. Epochs is set as 5 for this model.

# Results

The result of the Convolutional Neural Network model is displayed below -

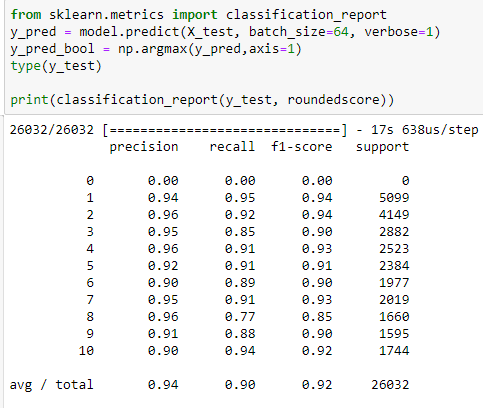


Fig.4. Prediction Results

The*‘sklearn.metrics’* library is used for classification\_report. In classification\_report as shown in the figure the f-score, recall and precision are calculated class-wise.

The accuracy of the model is calculated as 91% and f-score is calculated as 0.92. The accuracy and f-score are calculated using sklearn.metrics library.

The CNN model can be saved in the archive folder by using ‘model.save’ method with .h5 extension format. The model is created as HDF5 file and returns a compiled model. The saved model can be loaded in python using ‘load\_model’ function which will returns a compiled model which will be identical to the model built. The created model is saved in the current directory.

The image predictions will be performed using the test (filename) function. In this function the image of a house number is fed as input into the model and the predicted result will display the house number based on the model precision and accuracy.

The next function is traintest() function which will download the files and build the model in the function then evaluate the result and finally return the F1 score of the model.

Precision – is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved.

Precision = True Positives/(True positives + False positives)

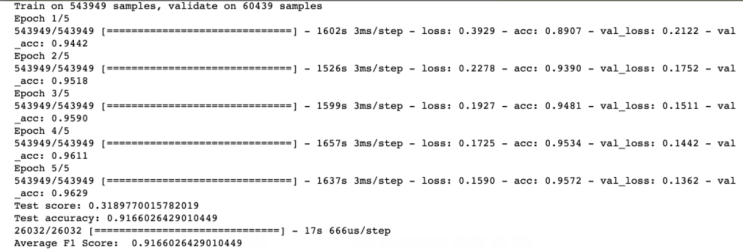
Recall - is the ratio of the number of relevant records retrieved to the total number of relevant records in the database.

Recall = True Positive/(True positive + False Negative)

F-measure - is a measure of test’s accuracy. It contains both precision p and recall r of the test to compute the F-score or F measure of the model.

F-Measure = 2\* (precison\*recall)/(precision+recall)

For the model performance, F1-score can be better measure to find the performance of the model as F1- score is the harmonic mean of precision and recall.



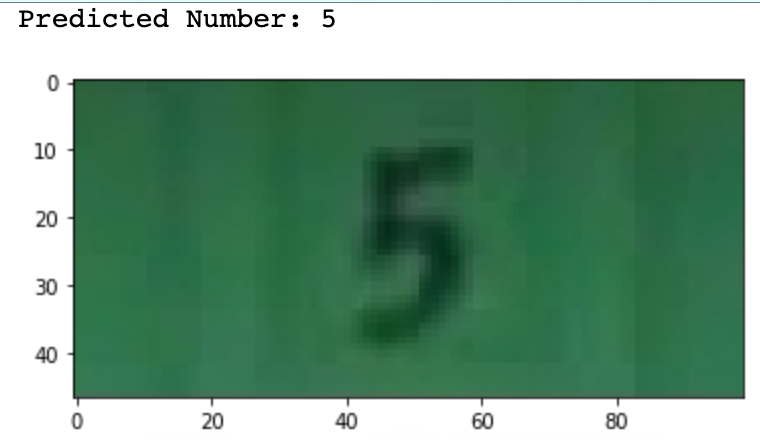


Fig. 5. Results from traintest() and test(filename) functions

# Conclusion

This experiment sheds light on the Convolutional Neural Network model for image processing. The CNN model flow is explained and how the image processing is performed is handled. Many python libraries were used and their utilities were covered. Libraries like – os, matplotlib, h5py, tnsorflow, numpy etc were used to perform the classification and preprocessing of data. Image data analysis was performed on SVHN data set. The image details were captured and learnt like – number of pixels, three channels of image – Red, Green and blue and each colour has its own pixels. The model is built on training data and then the evaluated is performed on test data

##### References

[1] Thomas Almenningen

<https://github.com/thomalm/svhn-multi-digit>

[2] <https://www.kaggle.com/oricou/svhn-avec-cnn>

[3] <https://keras.io/>

[4] Huawu Deng .Gaussian MRF rotation-invariant features for image classification. [IEEE Transactions on Pattern Analysis and Machine Intelligence](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=34),[10.1109/TPAMI.2004.30](https://doi.org/10.1109/TPAMI.2004.30),,pages951-955