KANNADA MNIST

-Hemanta Ingle, Siddhesh Gaike, Ms EE, San Jose State University, CA

ABSTRACT

The classification of digits in Kannada language is a replacement to the traditional English digits recognition.in this paper we have tried to implement a CNN to classify the Kannada dataset.

Guided by-

Prof- Birsen Sirkeci, Electrical Engineering Dept, SJSU, CA.

INDEX-

I. INTRODUCTION-

Problem statement-The classical MNIST data set is about classifying 10 classes of digits from English language , now that all the researches have been concentrated on the MNIST data sets it has become to come up with new datasets that perform good not only on a particular language but on an array of datasets available across all the languages , the whole idea is to come up with a model that can detect images, digits from as many languages as possible. Kannada MNIST is an attempt to create such a classifier that works on a totally new language.

Motivation- Bringing up a model on a new dataset that the world hasn't seen is a major motivation. empowering a machine with the ability to read new datasets gives this project a motivation, the data is a purely synthetic data handwritten by the volunteers and hence recognition is a challenge

Challenges-

a) Making a MNIST classification model that is capable to recognize all the glyphs accurately despite of the uncanny resemblances between some of the digits.
b) Training a model on a dataset and then testing it out on the Kannada test dataset and the dig-MNIST dataset
c) Recognition of the partially written digit or if the digit is interfering or overlapping the other.
d) Boosting the auto segmentation algorithms that can classify the dataset and in turn return with a good data set that is more refined and can be free of a flaw from the glyph errors.
e) Achieving a good accuracy without much preprocessing .

Current Methods- there are current methods of classifying the digits but they fail to recognize some of the images as they misinterpret it with other digits.

Possible Solution-we have now successfully identified the problem and also have come up with an idea to use Neural networks for digit classification. Algorithms can be developed that make accurate predictions for digit recognition by training the model on basis of data available.

Goal- achieve a MNIST level accuracy

Evaluation metric –
Number of images categorized correctly or we can say the accuracy of the model. For eg. The categorization accuracy of 0.85 indicates that 15% of the digits of the test dataset were wrongly classified.

II. DATASET DESCRIPTION-

The dataset contains digits in Kannada language and is divided into two sections

1) Kannada MNIST dataset-60k images of 28*28 pixels (grayscale)
2) Dig-MNIST-1024 images of 28*28 pixels

Kannada MNIST is a attempt to replace the conventional MNIST dataset of digits in English language , the author of the original MNIST dataset has also released an extra Dig MNIST dataset for testing.

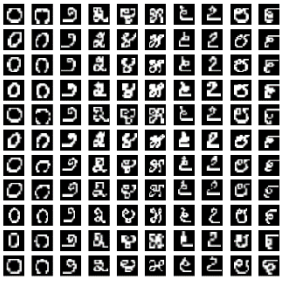


Fig.2.1-Variation of glyphs across the digits.

The fig 2.1 shows the images of the digits in Kannada MNIST that are pre- processed and we can observe the variation of the styles across the patterns of the styles of writing the same digits , this fact poses a additional challenge to our model as it can be thought of as a data augmentation- that helps the model to train well and perform good on the images even if they aren't in right orientation or even if they are anywhere in the images.

Dataset Files available-

Two files are available to us namely test.csv and train.csv, they have images of the digits from Kannada language, images are 28*28 pixel a total of 784 pixels,[3] each pixel is having a value of 0 to 255.the training dataset comprises of 785 columns , the first column is the hand written digit and the other columns have a ,pixel values associated with it where pixel {x}, where x ranges from 0 to 783, we decompose x as x=i*28+j, i,j range from 0 to 27.the row i and column j of the matrix (28*28) indicate the position of the pixel{x}.

The pixels in the image when represented can be shown as

```
000   001   002   003 ... 026   027

028   029   030   031 ... 054   055

056   057   058   059 ... 082   083

 |     |     |     | ...   |     |

728   729   730   731 ... 754   755

756    757   758   759 ... 782   783
```

Fig.2.2- Pixel value representation.

The test data file contains the same kind of data but it doesn't contain the labels. The data files available are as follows

1) Dig-MNIST.csv- label 0 to label 785
2) sample_submission.csv – id and label
3) test.csv- 785 columns- id as a key and rest columns are the pixel values
4) train.csv- 785 columns- label as first column and rest are the pixel values

III. PREPROCESSING-

Converting vectors for convolution-The data needs to be in proper format so that a convolution can be p erformed on it, we load the data one by one from the Kaggle database, here we have 3 files data_train, da ta_test, dig_id . we take the training data and reshape it to a 4d [2]vector required for the convolution , to create the layer we want he the X input in the form of a 4D tensor that corresponds to the image width an d height , the final dimension in the vector corresponds to the number of color channels, here we can obs erve that the component given in the code "X = X.reshape(-1,28,28,1)" as a argument is -1 which indicat es that the total size of the image vector remains constant, the shape [-1] flattens the vector to a single dim ension vector , it is a rule that maximum lone of the component can have a shape -1.

Splitting the data – now we need to split the data in test and training respectively , so we choose the standard way of doing it and splitting it in the ratio of 70:30 ,[5] 70 percent of the data goes to the training and the remaining of the 30 percent of data goes to testing, its worth noting that we also keep the random seed here as 42 which is nothing but a random number generator and just randomly splits the data which avoids the problem of model overfitting or underfitting if there is a bias in the data. sk_learn is a good library to do the splitting of data so we have chosen it. The visualization of the new training data set shows us that it consists of 42000 images of 28*28 pixels

Converting the Train/Test data format- now that we have successfully split the data into test and training and have also visualized the data in the training set it seems that the integer values must be converted to binary format and that's done by importing "categorical" , this is a type of c=scaling we perform to convert the pixel vales to a smaller scale by dividing them to 255, similar operation is done on the test dataset 0f 18000 images.

IV. BASELINE MODEL-

A baseline model is a model where in we start to develop our model and if we find that we are some what near to the expected evaluation metric, we can then try different methods of improving the performance of the model, so for this type of data set and the given problem of identifying a digit form the images we can look out for similar kind of datasets and the problems they have solved. [5]A very good example of such as dataset is the standard MNIST dataset where the CNN models classify the digits with a high accuracy. now we have identified the possible solution of the problem so lets jump into creating a Convolutional Neural Network for the same.

Starting with CNN- Convolutional neural network is used for years and is successful in classifying images, detecting objects, people etc., now the challenge is that will it work well in this case of identifying Kannada digits. So, we start with designing a simple model that will look like the following figure 4.1.

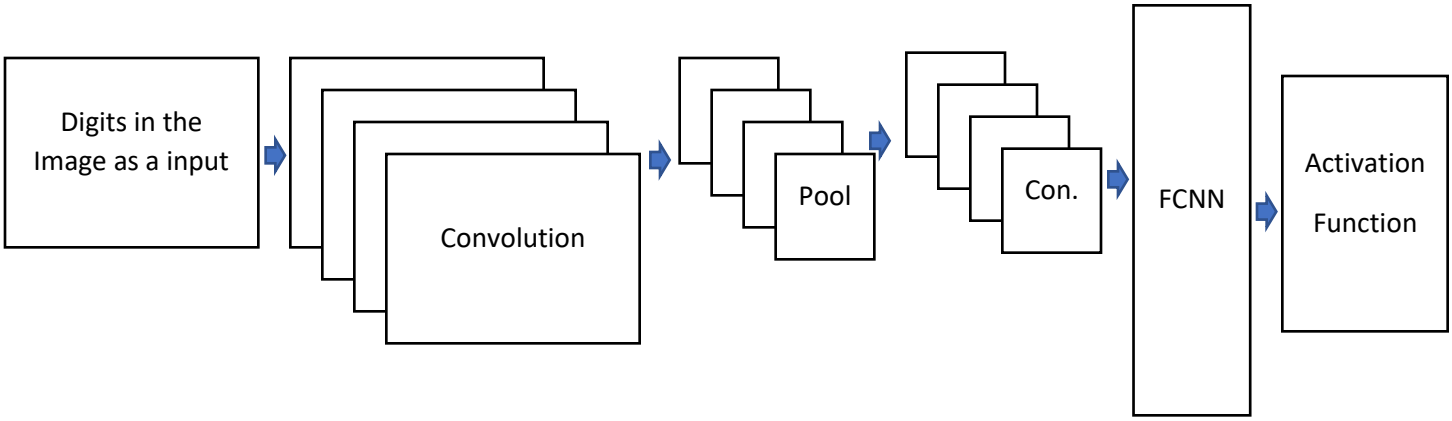


Fig. 4.1 A simple CNN model that we are going to implement. (Activation- RELU for hidden layers and SoftMax for output)

Note- The fig above is not the exact model that we have created. (It is for understanding purpose only)

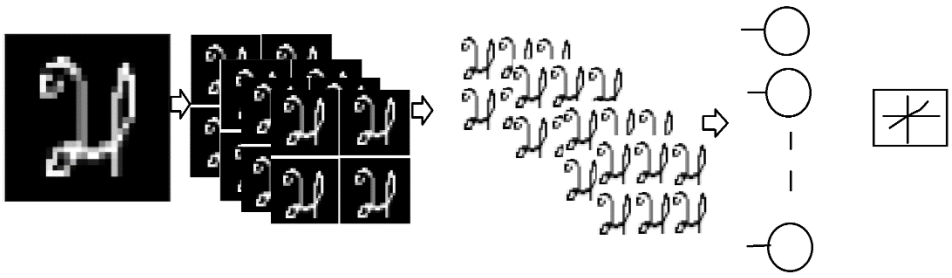


Fig.4.3- sample example of the processing in the CNN

In the above figure we can see that a simple Convolutional neural network model is feeding the Fully [3]Connected Feed Forward Neural Network (FCNN) and then it passes through a activation function - particularly a SoftMax function that gives output in the form of probabilities of the image being from a specific class , we have chosen a CNN because of 2 main reasons

1) Parameter sharing – reduces the load on the computer for training and thus saves valuable computing time and power
2) Feature extraction- the CNN is proven successful in feature extraction and hence it can classify the digit data extracting its vital features.

We now go deeper into the understanding of the various layers embedded into the CNN. Starting with the images as the input the images are of size 28*28 and are provided as a combination of a whole image

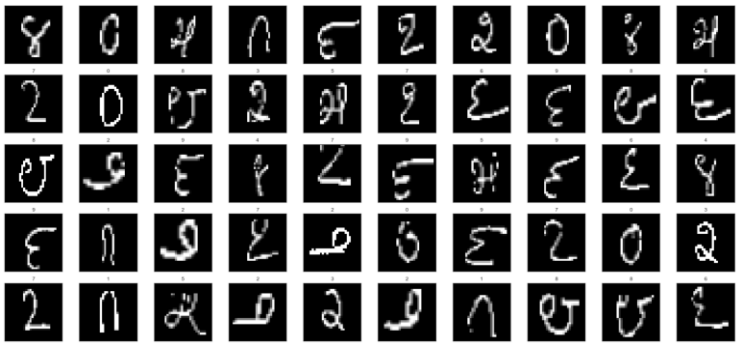


Fig.4.2- the images as inputs.

A kernel then slides over the image and its keeps on sliding on the whole of the image as per the rules defined , here in our example the summary can be shown.

MODEL 1			
Sr. no.	Type of Layer	Output shape	Number of parameters
1	conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
2	max_pooling2d_1 (MaxPooling2	(None, 13, 13, 32)	0
3	conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
4	max_pooling2d_2 (MaxPooling2	(None, 5, 5, 64)	0
5	max_pooling2d_2 (MaxPooling2	(None, 3, 3, 64)	36928
6	flatten_1 (Flatten)	(None, 576)	0
7	dense_1 (Dense)	(None, 64)	36928
8	dense_2 (Dense)	(None, 10)	650
Total	5 CNN+ 3 FCNN +1 Activation		93322

Table 4.1- Detailed description of the convolution neural network

The above table is created with the reference to the CNN model shown in the fig.4.1 , we can see that a 2d convolution is used as a first layer a filter of kernel of 3*3 is sliding over the image of 28*28 pixels creating output images of dimension 26*26 pixels which are called as feature maps , and 32 such feature maps are generated with no padding and a stride length of 1 .the total computed weights are shown in the last column. A sequential model is a fully connected feed forward neural network.

V. MODEL IMPROVEMENTS-

Data Augmentation-data augmentation is performed over the dataset to reduce the memory constrains , this type of ,image pre- processing is done over a large dataset, researches have found out that data augmentation can make a model robust and more accurate so we use the method data generator that is a flexible kind of data generator. The operations performed are.

1) Setting mean to zero over the dataset
2) Each sample mean is set to zero
3) Normalizing the data
4) Randomly zooming the image
5) Applying ZCA whitening
6) Randomly shifting the images horizontally
7) Randomly shifting images vertically
8) Randomly flip images

Using an optimizer in the model- Adam optimizer is used to make the deep learning Neural network work faster by finding out best learning rates for each parameter, it performs good on sparse gradients and not so good on non-convex optimizations of neural networks with Adagrad. we try using RMSProp that solves the problems associated with the Adagrad. Adam is a coalesced version of both RMSprop and the Stochastic gradient descent.

Reduced learning Rate- we use a method that is effective against the problem of skipping the global minimum, in this technique we try to reduce the learning rate as the model doesn't show significant improvement in the learning, it is called as a call back technique.

Adding more Epochs-with the improved epochs and the we are obtaining a accuracy of the base line model we had an accuracy of 0.96 with 10 epochs but as we have increased the epochs to 60 we can observe a improved accuracy of 0.99 at epoch 50 and thus our aim of creating a good model is satisfied.

VI. DISCUSSIONS AND CONCLUSIONS-

Results-

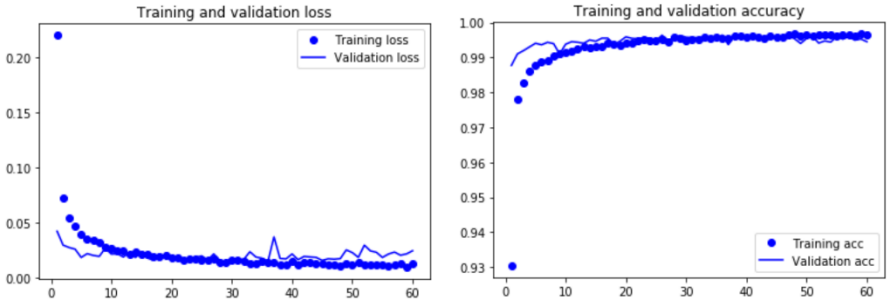


Fig .-5.1 the right left hand side figure shows training and validation loss against the number of epochs , the right hand side image shows the training and validation accuracy against the validation accuracy.

The above left hand side figure shows that the training and the validation loss decreases significantly after 40 epochs and we get the lowest of the loss at epoch number 50, the right hand side image can be thought as a inverted image of the left hand side image as it shows the accuracy of the model with respect to the number of the epochs.

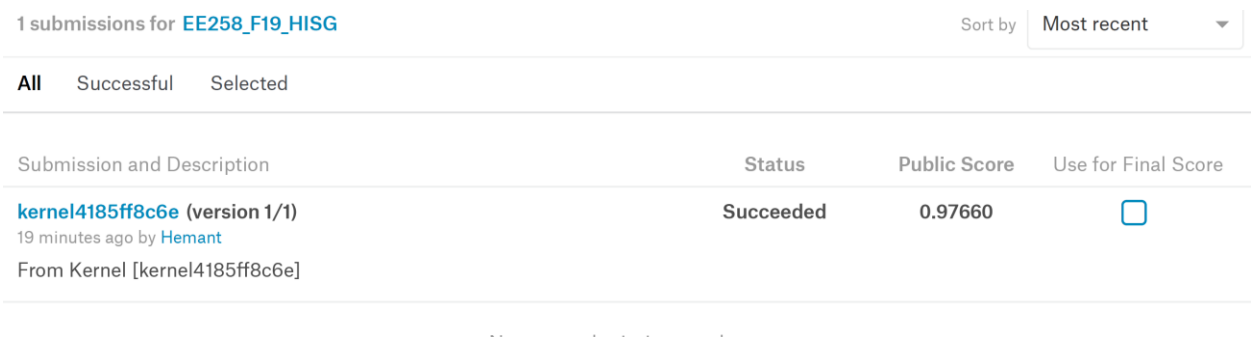
	0	1	2	3	4	5	6	7	8	9
0	1774	7	0	0	0	0	0	0	3	1
1	35	1797	0	0	0	0	0	1	0	1
2	2	0	1831	0	0	0	0	0	0	0
3	1	1	0	1770	0	1	0	12	0	0
4	0	1	0	0	1801	4	0	0	1	2
5	0	0	0	2	0	1767	0	0	1	0
6	0	0	0	0	0	0	1775	5	0	3
7	0	0	0	2	0	0	4	1836	0	0
8	0	0	0	0	0	0	0	0	1775	0
9	1	0	0	0	0	0	8	0	0	1775

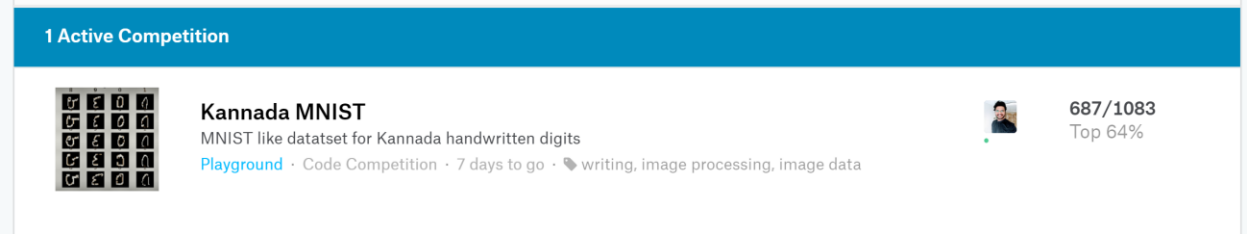
Fig.-5.2 Confusion matrix

The above fig shows the confusion matrix and thus we can find out the number of images that are successfully classified and those aren't. some of the results can be summarized like in case of classification of 0 1774 were correctly classified as 0 while 35 were classified as 1, 2 as 2, 1 as 3 and one as 9.

Final Conclusion-a CNN is a good way to approach a image recognition problem , activation function in the hidden layers should be RELU and a SoftMax at the output , optimization techniques like Adam can be applied to make the training faster and efficient, regularization techniques can be applied to increase the accuracy and reduce the test error (overfitting), data augmentation helps in making the model robust against variations in a completely new dataset.

Results in the competition- a model with the highest accuracy was run, a submission file of predicted outputs was then submitted to Kaggle and a Public score of 0.97 is obtained.





VII. REFERENCES-

[1] Prabhu, V. (2019). *Kannada MNIST | Kaggle*. [online] Kaggle.com. Available at: https://www.kaggle.com/c/Kannada-MNIST [Accessed 11 Dec. 2019].

[2] Prabhu, V. (2019). *A new handwritten digits dataset in ML town: Kannada-MNIST*. [online] Medium. Available at: https://towardsdatascience.com/a-new-handwritten-digits-dataset-in-ml-town-kannada-mnist-69df0f2d1456 [Accessed 11 Dec. 2019].

[3] Prabhu, V. (2019). *vinayprabhu/Kannada_MNIST*. [online] GitHub. Available at: https://github.com/vinayprabhu/Kannada_MNIST [Accessed 11 Dec. 2019].

[4] GitHub. (2019). *kenanajkunic/kmnist-cnn*. [online] Available at: https://github.com/kenanajkunic/kmnist-cnn/blob/master/kmnist-cnn/Understanding%20CNNs%20with%20Kannada-MNIST.ipynb [Accessed 11 Dec. 2019].

[5] Medium. (2019). *Adam—latest trends in deep learning optimization.*. [online] Available at: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c [Accessed 11 Dec. 2019].