CSE4037 - Deep Learning

J Component Report

Handwritten Letter Recognition in Japanese and Telugu

By

Reg. No : 19MIA1003 Name : Niranjan J
Reg. No : 19MIA1001 Name : Roshan Srinivaas S
Reg. No : 19MIA1093 Name : Vignesh. N

M.Tech Integrated Computer Science Engineering With Business Analytics

Submitted to

Dr. R. Rajalakshmi

School of Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION BY THE CANDIDATE

I hereby declare that the report titled "**Handwritten Letter Recognition in Japanese and Telugu**" submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr. R. Rajalakshmi, Associate Professor, SCOPE, Vellore Institute of Technology, Chennai.**

Signature of the Candidate

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Rajalakshmi,** School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Ganesan, Dean,** School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Head of the Department** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled “**Handwritten Letter Recognition in Japanese and Telugu**” is a bona-fide work of **Niranjan J(19MIA1003), Roshan Srinivaas S(19MIA1001), Vignesh. N (19MIA1093)** carried out the “J”-Project work under my supervision and guidance for CSE4037 – Deep Learning.

Dr. R. Rajalakshmi

SCOPE

TABLE OF CONTENTS

ABSTRACT

The recognition of Japanese handwritten characters has always been a challenge for researchers. A large number of classes, their graphic complexity, and the existence of three different writing systems make this problem particularly difficult compared to Western writing. As for all handwriting recognition problems, handwritten Japanese character recognition is difficult due to the wide variability of writing styles and the confusion between similar characters. Also, for Chinese and Japanese characters, the large number of characters (classes) poses a challenge in efficient classification. Also Japanese characters have more complex understanding since it has more strokes in letters/characters than other western languages. In this work, deep convolutional neural networks are used for recognizing handwritten Japanese characters : hiragana, katakana. This work focuses on the classification of the type of script, character recognition. Similarly for Telugu character recognition we are using telugu vowels dataset where we will be building a CNN model for recognition.

INTRODUCTION

KMNIST (Kuzushiji-MNIST or Cursive hiragana-MNIST) was introduced as an alternative to MNIST. It contains images with the first entries from the 10 main Japanese hiragana character groups.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.

The images are storred in numpy arrays of 60,000 x 28 x 28 and 10,000 x 28 x 28, respectively. The labels are also stored in two numpy arrays, one for train and another for the test set.

The dataset consists all the **Telugu characters** that contains Vowels, Consonants and combine characters such as Othulu (Consonant-Consonant) and Guninthamulu (Consonant-Volwels). The main objective of this dataset to recognize handwritten Telugu characters. There is a significant difference between Indian literature and English literature, i.e., if we see English literature only 26 Characters, but where in Telugu total number of characters are 1,924 (Achulu (Vowels)- 16, Hallulu (consonants)- 36, Othulu – 36 and Guninthamulu – 34*16=544). Hence, problem of recognition of Telugu characters are complex in compare to English. Furthermore, no dataset of Telugu characters that covers all characters in Telugu literature and even the worldwide encoding standard “Unicode” have not covering all Alphabet in Telugu. The objective of this work is to present a Handwritten Telugu character dataset with Telugu Alphabets (vowels), assigning unique label to each character from there assign ‘Unicode’ to each label. The dataset is designed to recognize all short of handwriting styles. Hence, we create dataset from different distinct writers.

LITERATURE SURVEY

Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks :

In this work, deep convolutional neural networks are used for recognizing handwritten Japanese, which consists of three different types of scripts: hiragana, katakana, and kanji. This work focuses on the classification of the type of script, character recognition within each type of script, and character recognition across all three types of scripts. Experiments were ran on the Electrotechnical Laboratory (ETL) Character Database from the National Institute of Advanced Industrial Science and Technology (AIST). In all classification tasks, convolutional neural networks were able to achieve high recognition rates. For character classification, the models presented here in outperform the human equivalent recognition rate of 96.1%.

Convolutional neural networks (CNN) have emerged as a powerful class of models for recognizing handwritten text, especially for handwritten digits and Chinese characters. In this work, CNN’s are used for recognizing handwritten Japanese characters. For both discriminating between the scripts and classifying the characters within each script, CNN’s were able to achieve high accuracies.

Handwritten recognition of Hiragana and Katakana characters based on template matching algorithm :

Japanese has become one of the most popular foreign languages in Indonesia. From a survey conducted by The Japan Foundation in 2017 shows that Indonesia is the first largest student in Southeast Asia, therefore those interested in learning Japanese are quite popular in Indonesia. One of the things learned in Japanese is about writing Hiragana and Katakana characters, each of which has 46 standard characters. The research aims to implement a template matching algorithm for handwritten recognition of Hiragana and Katakana characters, which can later be implemented in application design for learning Hiragana and Katakana character writing. Through the application of template matching algorithm, the accuracy level of handwritten recognitions pattern matching reaches 89.8%, so the algorithm will be very suitable if implemented for Hiragana and Katakana characters writing learning applications.

Online writing-box-free recognition of handwritten Japanese text considering character size variations :

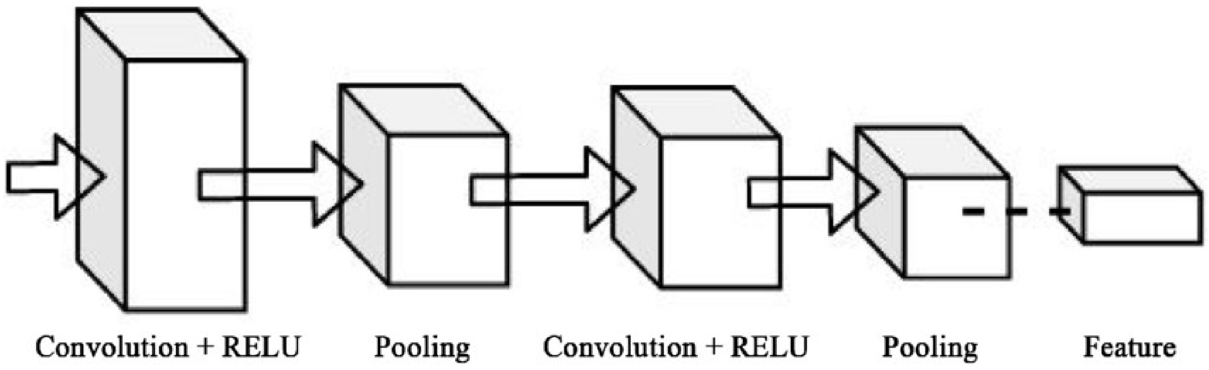
An online writing-box free method for recognizing handwritten Japanese text is proposed. This method is achieved by the following procedure. First, the average character size of input handwritten text is estimated. Second, candidates for character segmentation are detected using geometric features between two adjacent strokes. Finally, a search is performed by dynamic programming for the string that maximizes evaluation score (acceptability as Japanese text). The evaluation score reflects the likelihood of character segmentation, recognition, context and the size of each character. The size is a newly introduced factor since alphabets, numerals, symbols, Japanese phonetic characters, simple Chinese characters and compound Chinese characters are all written in different sizes even in a single line of text. Experimental results show that the incorporation of the character size likelihood increases the character recognition rate for Japanese text.

Study on handwritten Telugu character recognition with attention networks :

Opitcal Character Recognition (OCR) is a wildly popular area of research. This is especially true for the subdomain of handwritten character recognition. However, most of the research done in this field is for the task of recognizing English characters. The goal of this project is to design and implement an accurate character recognizer for handwritten Telugu characters. The problem is not easy because of writer-dependent variability and the complex nature of the Telugu script to begin with. The potential for a system like this is huge, because of the amount of handwritten data in Telugu. Government forms, local hospital forms, medical histories and prescriptions, tax records, and historical text, all have handwritten characters. For the recognizer, the project primarily focuses on deep attention networks and compares their performance on this challenging real-world dataset with other, more traditional approaches such as Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs). Attention networks were chosen because of their record of being able to improve results in several other tasks such as speech recognition, image captioning, and irregular text detection. They are also a more intuitve solution to deciphering the characters as they mimic how humans would approach the problem.

PROPOSED METHODOLOGY

For our project we are using Convolutional neural network(CNN). CNN is a deep learning technique to classify the input automatically,it extracts useful features from the given input automatically making it easy for us. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition. It is a class of deep neural networks that require minimum pre- processing. It inputs the image in the form of small chunks rather than inputting a single pixel at a time, so the network can detect uncertain patterns (edges) in the image more efficiently. CNN contains 3 layers namely, an input layer, an output layer, and multiple hidden layers which include Convolutional layers, Pooling layers(Max and Average pooling), Fully connected layers (FC), and normalization layers. CNN uses a filter (kernel) which is an array of weights to extract features from the input image. CNN employs different activation functions at each layer to add some non-linearity.



1) First we will do a data preprocessing to prepare for the model.

We reshape the numpy arrays for images to associate to each image a (28 x 28 x 1) array, with values normalized.

2) We process both the train_data and the test_data.

3) We further split the train set in train and validation set. The validation set will be 10% from the original train set, therefore the split will be train/validation of 0.9/0.1.

4) We will use a Sequential model :

The Sequential model is a linear stack of layers. It can be first initialized and then we add layers using add method or we can add all layers at init stage. The layers added are as follows :

Conv2D is a 2D Convolutional layer (i.e. spatial convolution over images). The parameters used are :

filters - the number of filters (Kernels) used with this layer; here filters = 32;

kernel_size - the dimmension of the Kernel: (3 x 3);

activation - is the activation function used, in this case relu;

kernel_initializer - the function used for initializing the kernel;

Input_shape - is the shape of the image presented to the CNN: in our case is 28 x 28 The input and output of the Conv2D is a 4D tensor.

Conv2D with the following parameters:

filters: 32;

kernel_size: (3 x 3);

activation : relu;

MaxPooling2D is a Max pooling operation for spatial data. Parameters used here are:

pool_size, in this case (2,2), representing the factors by which to downscale in both

Dropout : Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. The parameter used is:

rate, set here to 0.25.

Conv2D with the following parameters:

filters: 64;

kernel_size : (3 x 3);

activation : relu

MaxPooling2D with parameter:

pool_size : (2,2);

Dropout : with parameter:

rate : 0.25;

Conv2D with the following parameters

filters: 128;

kernel_size : (3 x 3);

activation : relu;

Dropout. with parameter:

rate : 0.4;

Flatten. This layer Flattens the input. Does not affect the batch size. It is used without parameters

Dense. This layer is a regular fully-connected NN layer. It is used without parameters;

units - this is a positive integer, with the meaning: dimensionality of the output space; in this case is: 128;

activation - activation function : relu;

Dropout. with parameter:

rate : 0.3;

Dense. This is the final layer (fully connected). It is used with the parameters:

units: the number of classes (in our case 10);

activation : softmax;

for this final layer it is used softmax activation (standard for multiclass classification)

5) Then we compile the model, specifying as well the following parameters:

loss;

optimizer;

metrics.

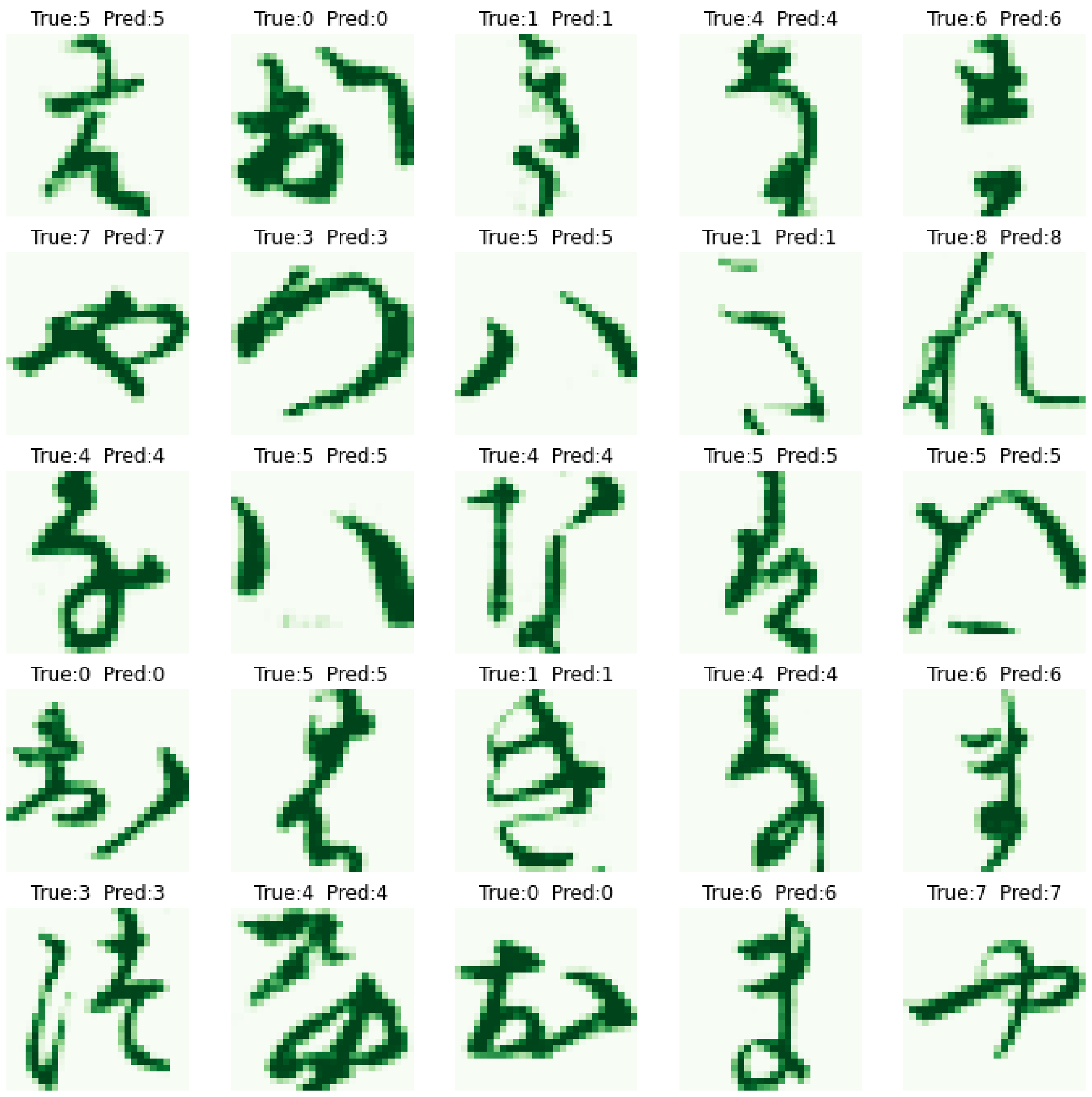
We then compile the model, with the layers and optimized defined.

6) We run the model with the training set. We are also using the validation set (a subset from the orginal training set) for validation.

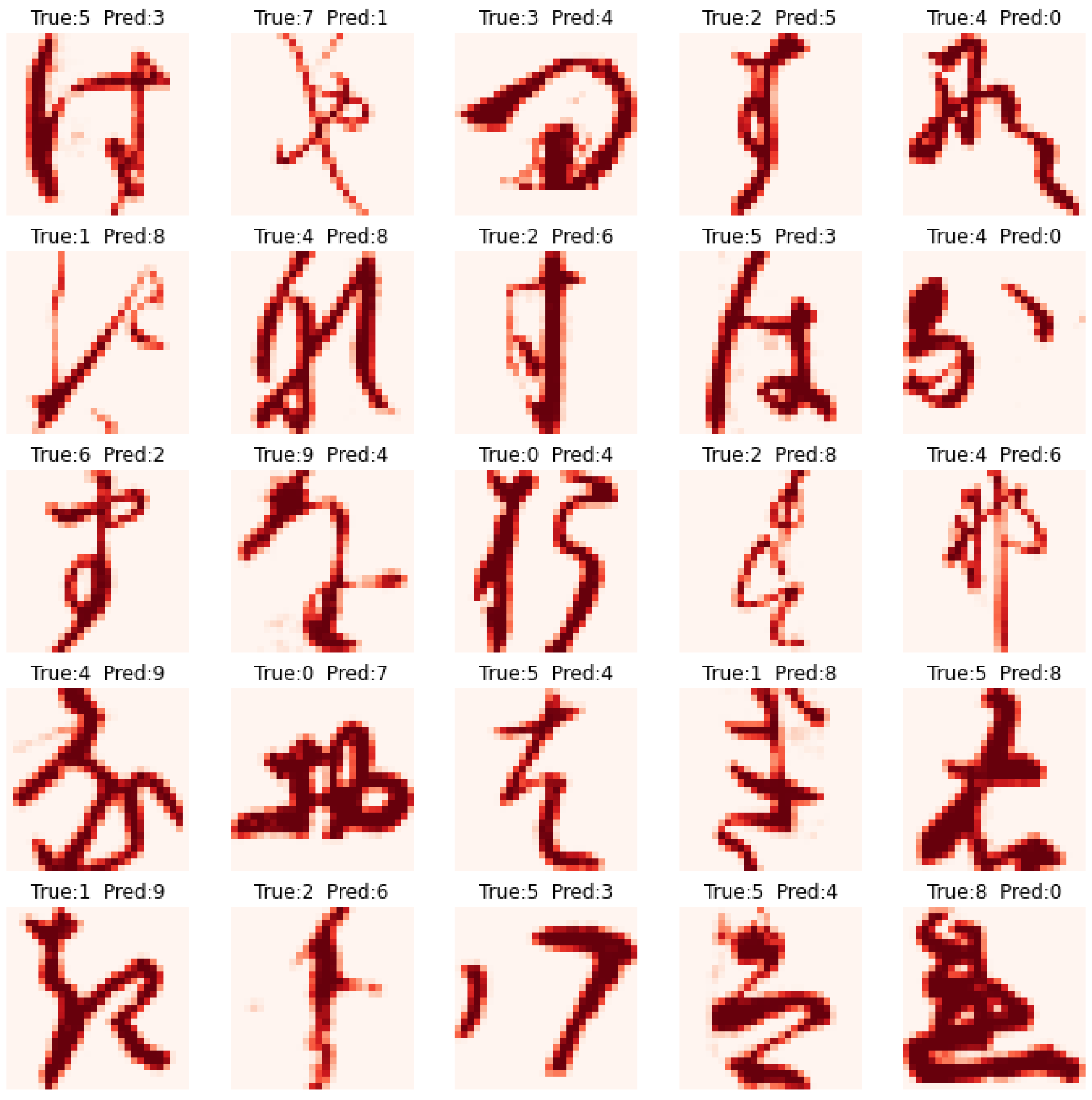
RESULTS AND DISCUSSIONS

1) Validation Prediction (Japanese)

Correctly predicted by the model :

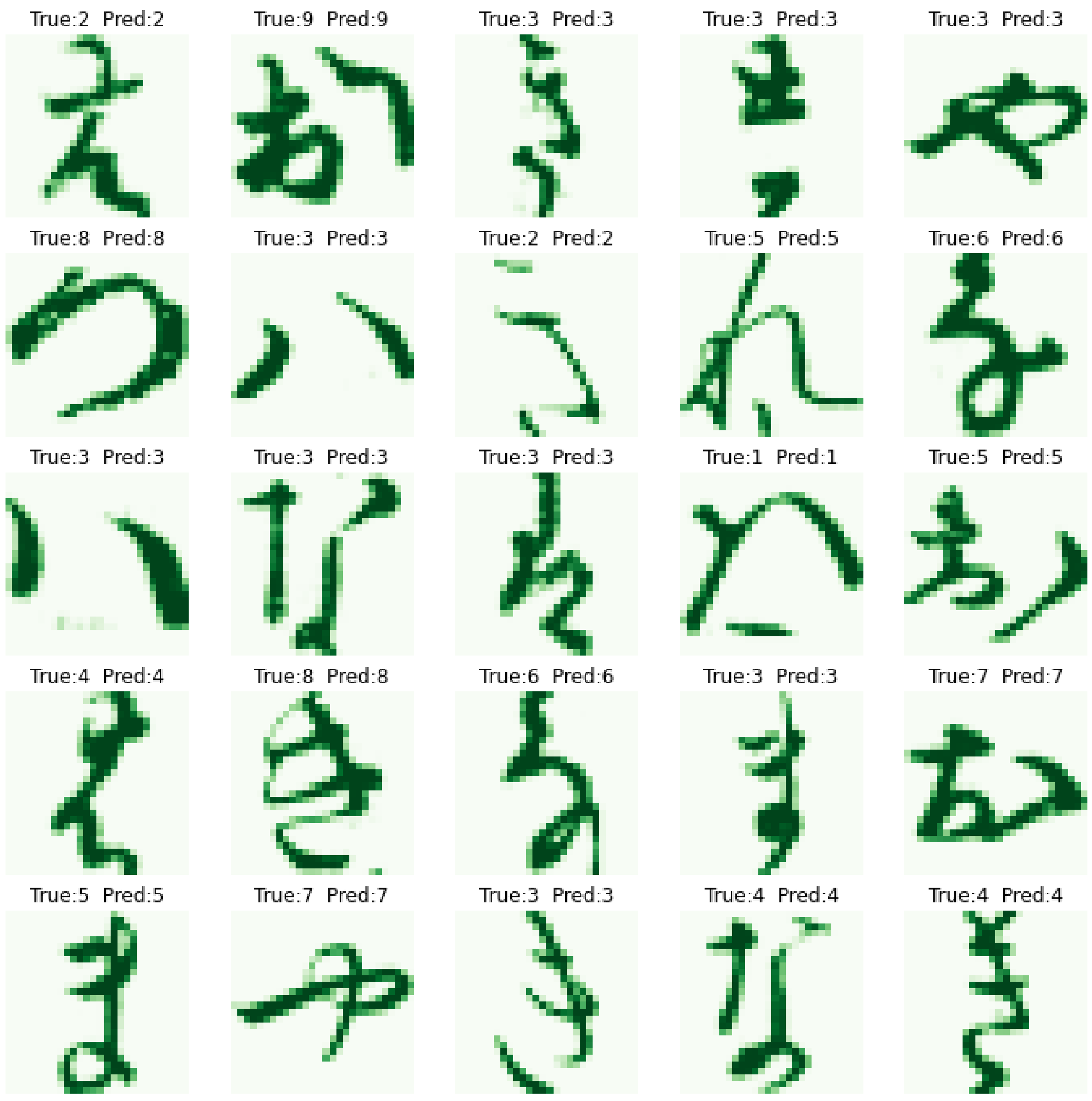


Incorrectly predicted by the model :

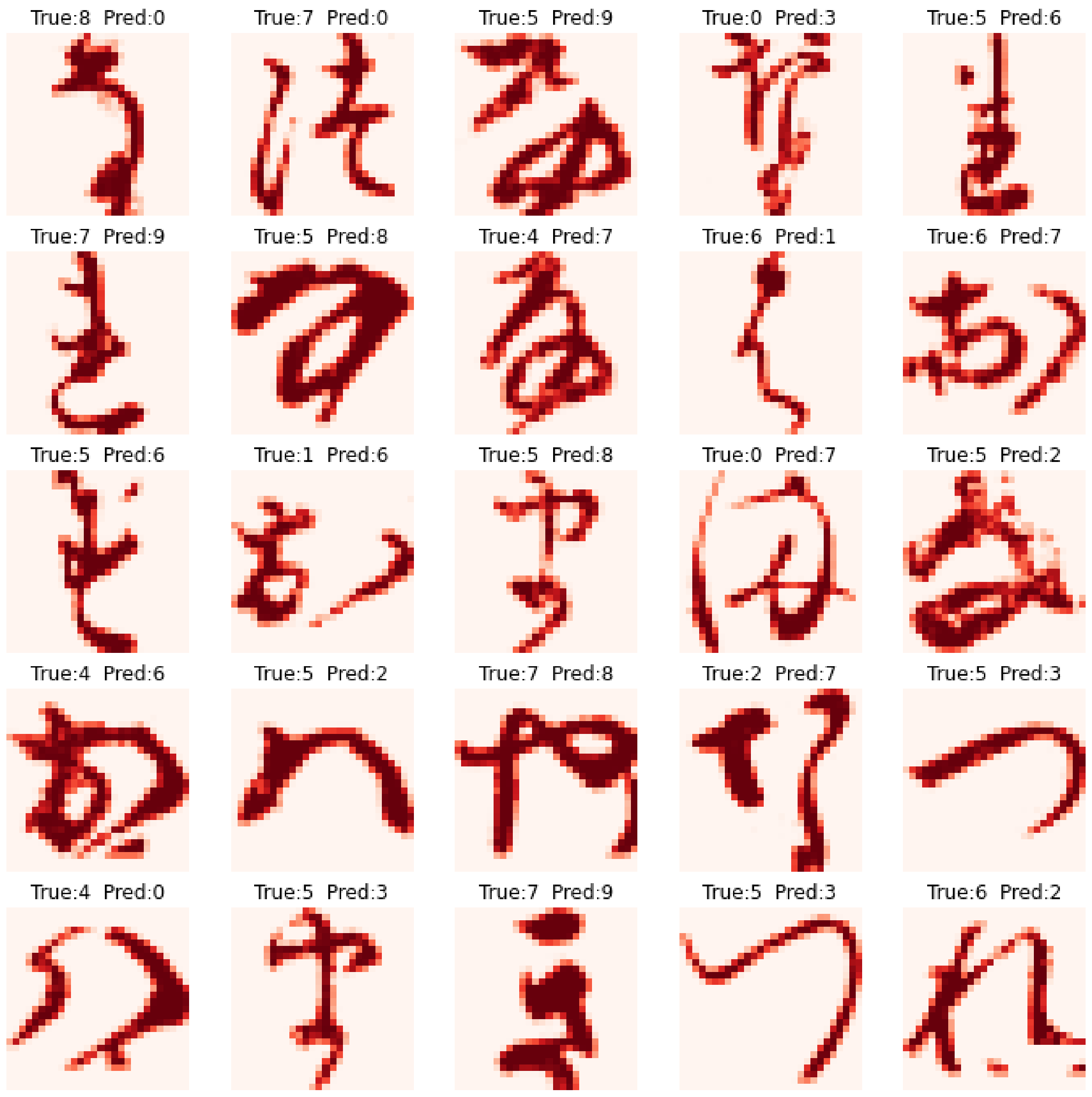


2) Test Predictions (Japanese)

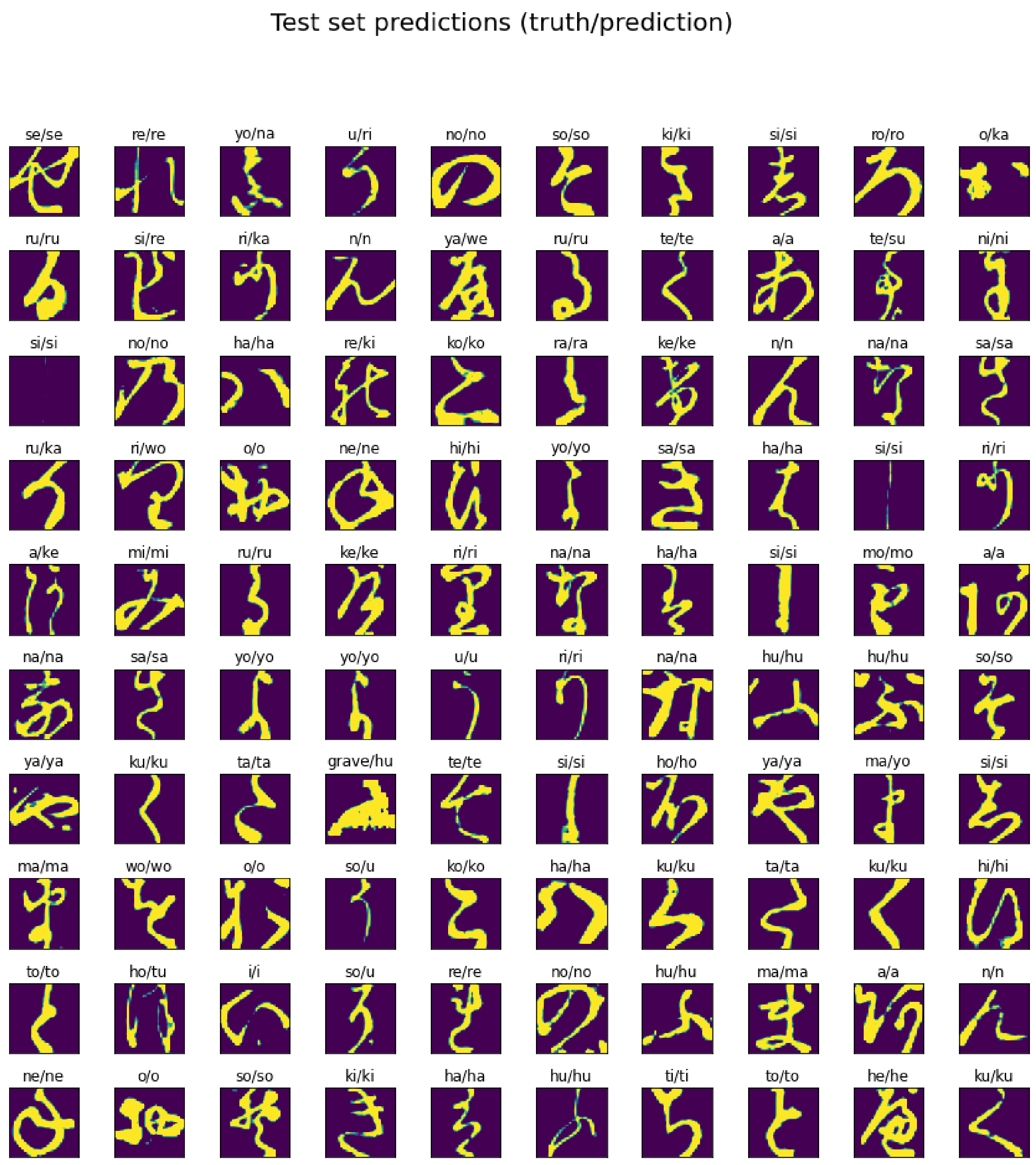
Correctly predicted by the model :



Incorrectly predicted by the model :

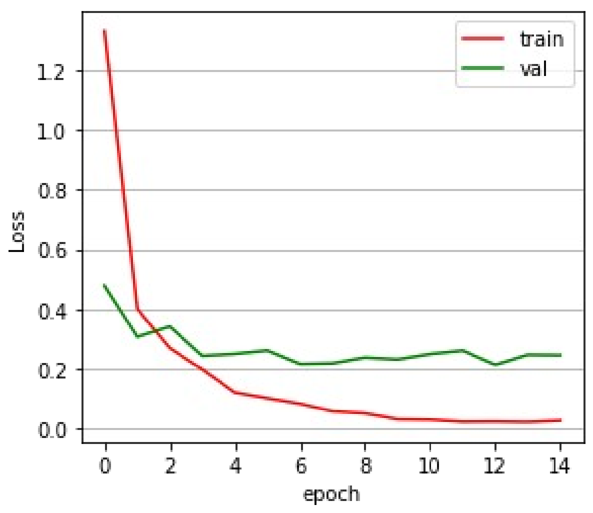


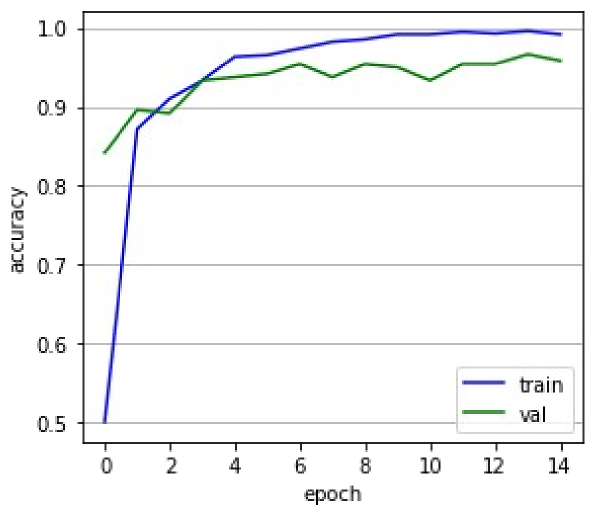
Test Set Predictions : (Truth/Predictions)



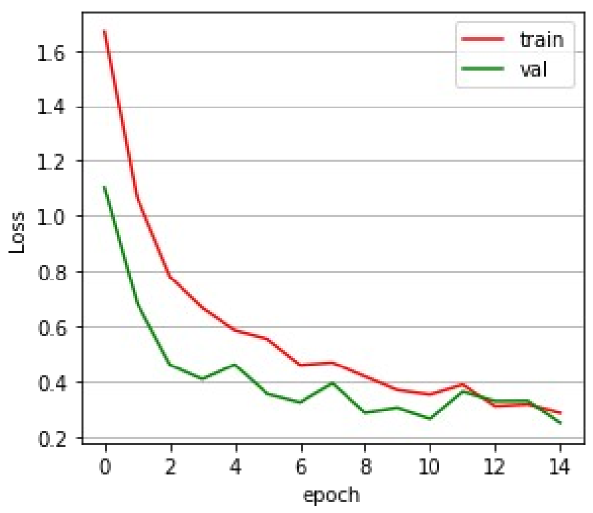
Telugu Character Predictions :

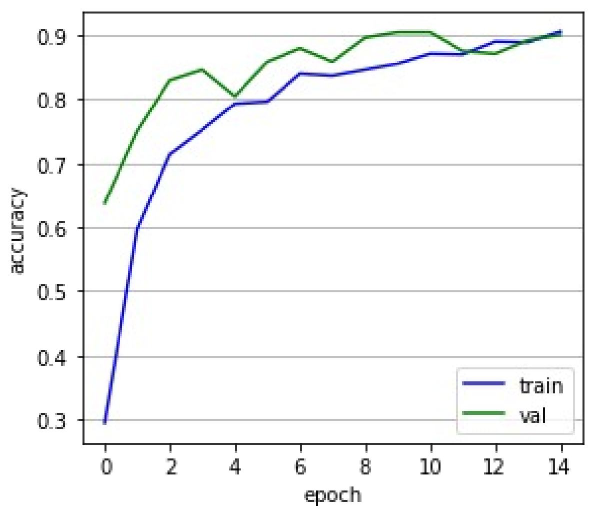
Training With Dropout Maxpool





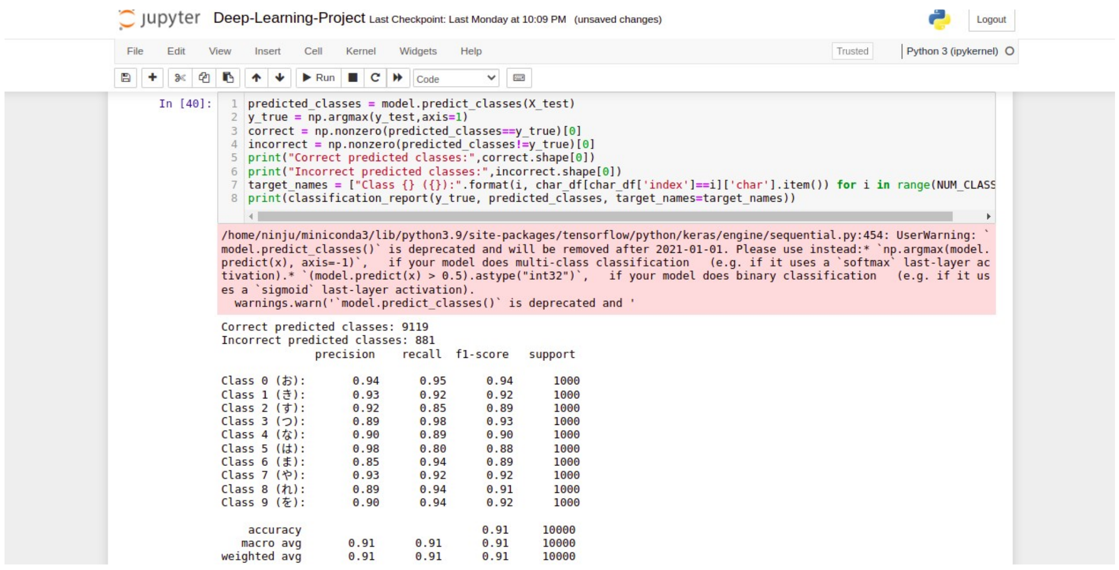
Training With Maxpool Data Augmentation





CONCLUSION

For the Japanese Dataset, we have got accuracy of 0.91.


jupyter Deep-Learning-Project Last Checkpoint: Last Monday at 10:09 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [40]:
predicted_classes = model.predict_classes(X_test)
y_true = np.argmax(y_test,axis=1)
correct = np.nonzero(predicted_classes==y_true)[0]
incorrect = np.nonzero(predicted_classes!=y_true)[0]
print("Correct predicted classes:",correct.shape[0])
print("Incorrect predicted classes:",incorrect.shape[0])
target_names = ["Class {} ({}):".format(i, char_df[char_df['index']==i]['char'].item()) for i in range(NUM_CLASS
print(classification_report(y_true, predicted_classes, target_names=target_names))
```

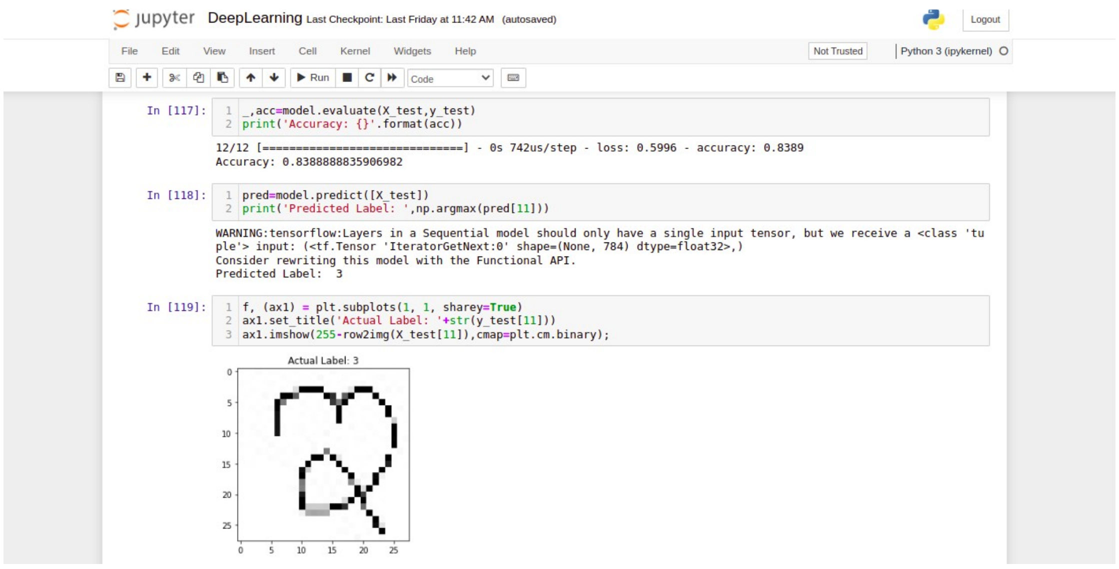
```
/home/ninju/miniconda3/lib/python3.9/site-packages/tensorflow/python/keras/engine/sequential.py:454: UserWarning: `
model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.
predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer ac
tivation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it us
es a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

```
Correct predicted classes: 9119
Incorrect predicted classes: 881
              precision    recall  f1-score   support

Class 0 (お):       0.94      0.95      0.94      1000
Class 1 (き):       0.93      0.92      0.92      1000
Class 2 (す):       0.92      0.85      0.89      1000
Class 3 (つ):       0.89      0.98      0.93      1000
Class 4 (な):       0.90      0.89      0.90      1000
Class 5 (は):       0.98      0.80      0.88      1000
Class 6 (ま):       0.85      0.94      0.89      1000
Class 7 (や):       0.93      0.92      0.92      1000
Class 8 (れ):       0.89      0.94      0.91      1000
Class 9 (を):       0.90      0.94      0.92      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

For the Telugu vowel dataset we have got accuracy of 0.83


jupyter DeepLearning Last Checkpoint: Last Friday at 11:42 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Actual Label: 3
0
5
10
15
20
25

```
In [117]:
_,acc=model.evaluate(X_test,y_test)
print('Accuracy: {}'.format(acc))
```

```
12/12 [==============================] - 0s 742us/step - loss: 0.5996 - accuracy: 0.8389
Accuracy: 0.8388888835906982
```

```
In [118]:
pred=model.predict([X_test])
print('Predicted Label: ',np.argmax(pred[11]))
```

```
WARNING:tensorflow:Layers in a Sequential model should only have a single input tensor, but we receive a <class 'tu
ple'> input: (<tf.Tensor 'IteratorGetNext:0' shape=(None, 784) dtype=float32>,)
Consider rewriting this model with the Functional API.
Predicted Label:  3
```

```
In [119]:
f, (ax1) = plt.subplots(1, 1, sharey=True)
ax1.set_title('Actual Label: '+str(y_test[11]))
ax1.imshow(255-row2img(X_test[11]),cmap=plt.cm.binary);
```

REFERENCES

http://cs231n.stanford.edu/reports/2016/pdfs/262_Report.pdf

https://www.intechopen.com/chapters/40720

https://arxiv.org/pdf/2106.12614.pdf

https://www.worldscientific.com/doi/10.1142/S0219427902000571

https://www.ijarcst.com/doc/vol4issue1/ahmed.pdf