
VIT®

Vellore Institute of Technology

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

# Corrupt Word Recognition System

## Final Project Report

A project report submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics and Communication Engineering.

Technical Answer To Real-World Problems (ECE 3999)

Under the guidance of

Dr. Anand S

June 2020

Submitted by-:

17BEC0141 Asutosh Mohapatra

17BEC0204 Sangireddy Vikas Reddy

17BEC0240 Parshuram Kumar

## Project Overview

As the name of the course indicates this project to solve some issues faced by normal people in day to day life. All of us have visited doctors in our life and one of the most annoying things about nearly every doctor is their hand-writing. Some times pharmacists also find it difficult to guess the name of the project. In this project, we have tried to overcome this issue with the help of advanced machine learning techniques.

## Abstract-:

Text recognition systems is a quite popular topic these days. So many researchers have tried to create a different system for different purposes. Here in this project, we are trying to guess what is written in a medical prescription with the help of advanced deep learning techniques. Our project field is a new one and quite interesting and will help a lot of people. We have implemented a hybrid LeNet architecture. We have trained our model on a real-world dataset with manual processing to have a good model. Having a test set accuracy of nearly 99% with minimal loss in this project we have achieved our target of accomplishing human-level accuracy.

# Introduction-:

**C**onvolutional Neural Networks are the most used for image classification purposes. But for every different image classification task requires a different model to have the best optimal result. We try to accomplish human-level accuracy in these projects. To get the best model we have tried out 9 different models and trained and tested those on the same dataset. Having a small dataset with different numbers of images for each example is also a big problem to get optimal results in the machine learning field. We have tried to modify the general LeNet architecture available and we have also tried different ways to update the parameters of our network. Creating a disturbance-free dataset and connecting everything properly to get an optimized model was our target and we have successfully achieved it.

## Project Plan-:

1. Preparing raw dataset as per required format(doctor's or similar type of handwriting).
2. Scanning all the images and attaching the correct label to the data.
3. Dividing the prepared dataset it into a test set and training set.
4. Then we will train the dataset using a convolution neural network.
5. Then further testing to verify our recognition system.

# Literature Survey-:

| Reference | Architecture Followed | Accuracy obtained |
|---|---|---|
| End-to-End Text Recognition using Local Ternary Patterns, MSER and Deep Convolutional Nets,IEEE ,2014 | A convolutional neural network trained with different layers and different dropouts for each layer | First kernel of 32 , 2nd and 3rd kernel of 128, fully connected layer of 2048 neurons. For different datasets on an average 87.5% accuracy obtained. |
| Recognition of Handwritten Characters using Deep Convolutional Neural Network,2019 | Image first converted to gray scale and used Convolutional Neural Network with LeNet architecture. | Accuracy of 94% obtained. |
| Text Detection based on MSER and CNN Features,IEEE,2017 | Used CNN's followed by SVM. | A overall accuracy of 85% is obtained. |
| CNN Based Common Approach to Handwritten Character Recognition of Multiple Scripts,2015 | LeNet-5 kind of architecture used. | An overall accuracy of 94% obtained |
| Image character recognition using deep convolutional neural network learned from different languages, 2014 | Shared Hidden Layer CNN architecture used. (Training two different datasets with the same net but the output layers are different. | Improved accuracy of 35% in comparison to conventional methods. |

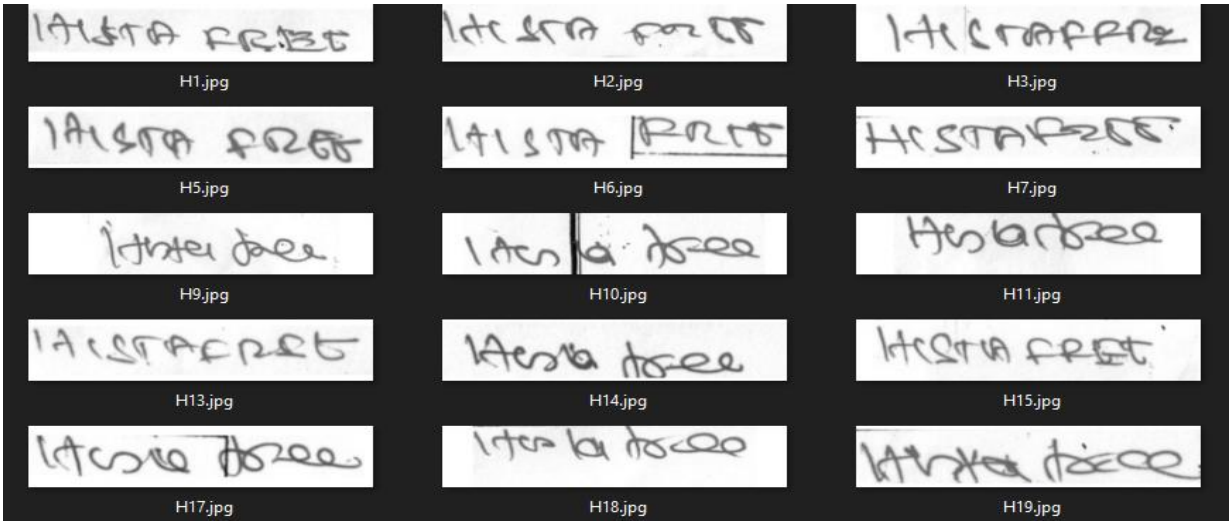
| | | |
|---|---|---|
| A CNN-based Approach to Detecting Text from Images of Whiteboards and Handwritten Notes, 2018 | Used feature pyramid networks architecture with hybridizing ResNet50 type of layers. | There is a 7% improvement observed when compared with it's base paper. |
| Character Recognition via a Compact Convolutional Neural Network, 2017 | Long-short term memory model followed by CNN architecture use. | A improved accuracy of 12% was observed. |
| Malayalam Handwritten Character Recognition Using Convolutional Neural Network,2017 | LeNet-5 architecture used and model trained for several times to get optimum result. | 82% accuracy obtained . |
| TEXT detection based on convolutional neural networks with spatial pyramid pooling,2016 | MSER method used for edge detection followed by CNNs then before fully connected layer one spatial pyramid pooling layer was used. | On different datasets on an average 82% accuracy was obtained. |

# OUR WORK-:

## 1) Generating a Dataset-:

We have used real prescriptions and scanned those and cropped the same medicines written and grouped those.

**Sample dataset-:**


## 2) Pre-Processing of the dataset-:

Before sending the data through a fully connected deep neural network we have extracted important information from it by performing convolution operations. By performing convolution operations we have successfully captured the spatial and temporal dependencies in an image by applying relevant features. Then we have flattened the final image matrix.

Using TensorFlow and Keras python APIs we have preprocessed the data such that an image with the same word written but in a different style like orientation or width or vertically or horizontally flipped manner can also be read by our system.

## 3)Creating A Deep Neural Network Model-:

We have tested the processed data in 9 different models and from those we have chosen the best model that has the best accuracy with minimal loss and computationally efficient. We have used TensorBoard to find the optimal the best model.

## Code-:

```
#IMPORTING ALL THE REQUIRED LIBRARIES

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout,
Activation, Flatten

from tensorflow.keras.layers import Conv2D,
MaxPooling2D

from tensorflow.keras.callbacks import TensorBoard

from keras.preprocessing import image

import matplotlib.pyplot as plt

import time


NAME="project_training_with_convLayers=(32,64)dropout=(
0.2,0.2),denseLayers=(64,128,256,512,1024)dropout=(0.3,
0.4,0.4,0.5,0.5)"

#PREPROCESSING OUR DATA PART 1 CONVOLUTION LAYER

classifier = Sequential()


# Step 1 - Convolution

classifier.add(Conv2D(32, (3, 3), input_shape = (64,
64, 3), activation = 'relu'))

# Step 2 - Pooling

classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

```
# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# DEEP NEURAL NETWORK MODEL
#Adding a Dense Layer of 512 units and dopout of value
0.5
classifier.add(Dense(units = 512, activation = 'relu'))
classifier.add(Dropout(0.5))

#Adding a second Layer of 256 activation of value of
0.5
classifier.add(Dense(units = 256, activation = 'relu'))
classifier.add(Dropout(0.5))
classifier.add(Dense(units = 5, activation =
'softmax'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss =
'categorical_crossentropy', metrics = ['accuracy'])
```

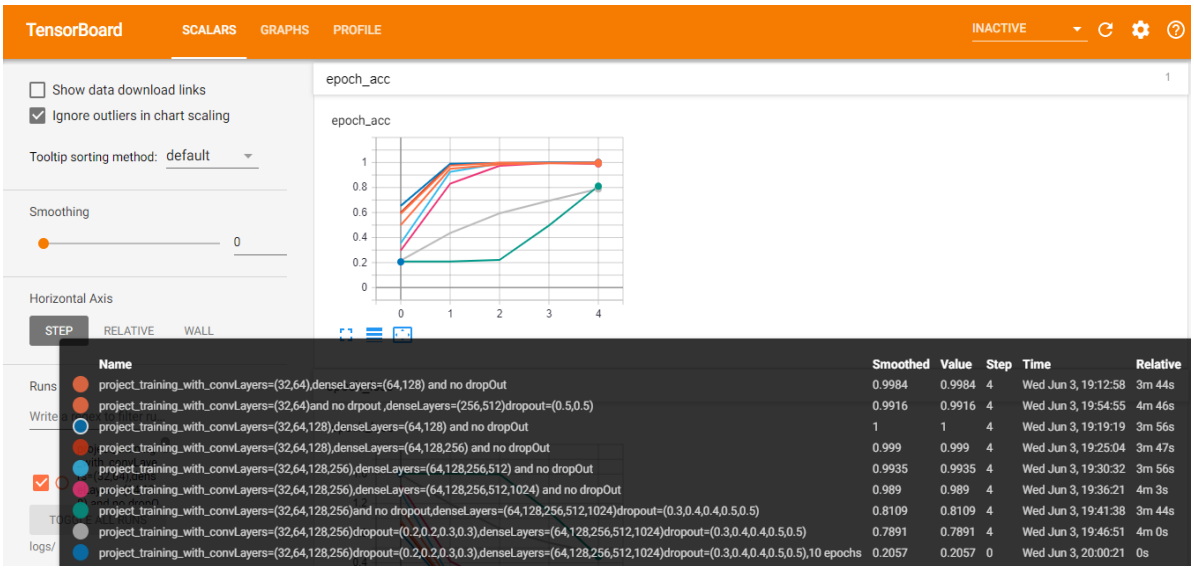
```
# DATA PREPROCESSING PART 2

from keras.preprocessing.image import
ImageDataGenerator


#Processing Train Set

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip =
False)

#Processing Test set

test_datagen = ImageDataGenerator(rescale = 1./255)


#GENERATION OF TRAINING DATA SET

training_set =
train_datagen.flow_from_directory('train',
target_size = (64,64),
batch_size = 32,
class_mode = 'categorical',
subset='training',
shuffle=True)

#GENERATION OF TEST DATA SET

test_set = test_datagen.flow_from_directory('test',

target_size = (64,64),

batch_size = 1,

class_mode = 'categorical',
```

```
shuffle=True)

#Keeping logs of accuracy and losses for every
different runs using tensorboard

tensorboard =
TensorBoard(log_dir="logs\{}".format(NAME))


#Assining Parameters to model

classifier.fit_generator(training_set,
                         steps_per_epoch =218,
                         epochs =5,
                         validation_data = test_set,
                         validation_steps =20,
                         callbacks=[tensorboard]
                         )



#Importing some image which are not trained before and
checking how correctly our model predicts those

class_names =
['Histafree','P250','phenzee','rhinoclear','zinova']



test_image = image.load_img('rhinoclear_test.jpg',
target_size = (64,64))
```

```
plt.grid(False)
plt.imshow(test_image)
plt.show()
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
print(result)
print('Actual:  '+class_names[3])
print('Predicted:  '+class_names[np.argmax(result)])
```

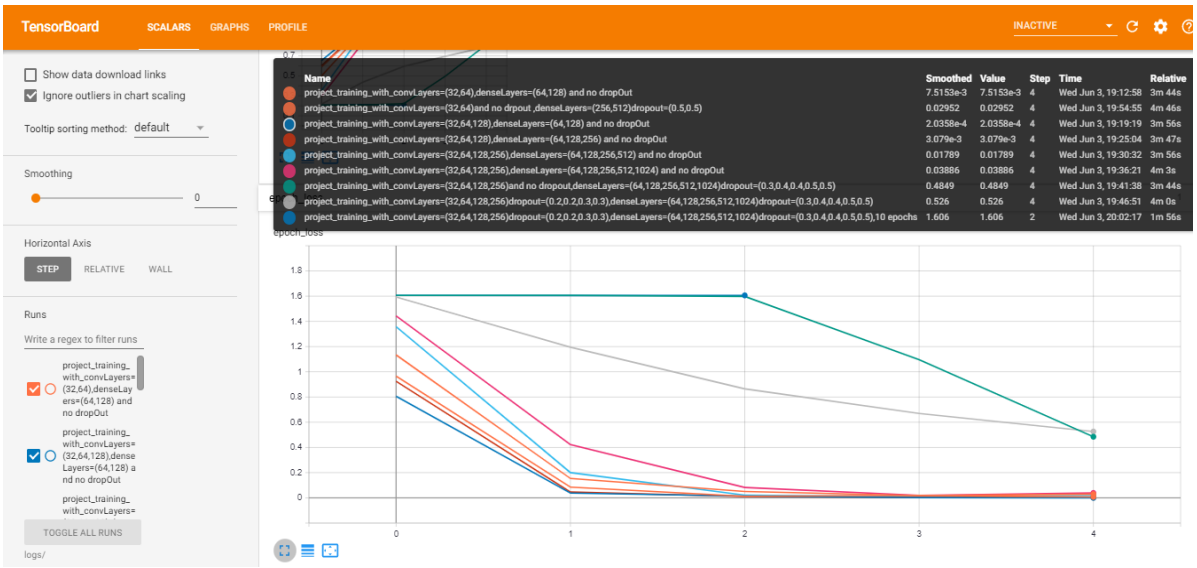
# Training Set Observations-:

## 1) Accuracy




We can observe from here that models with no dropout regularizations have given better accuracy scores.

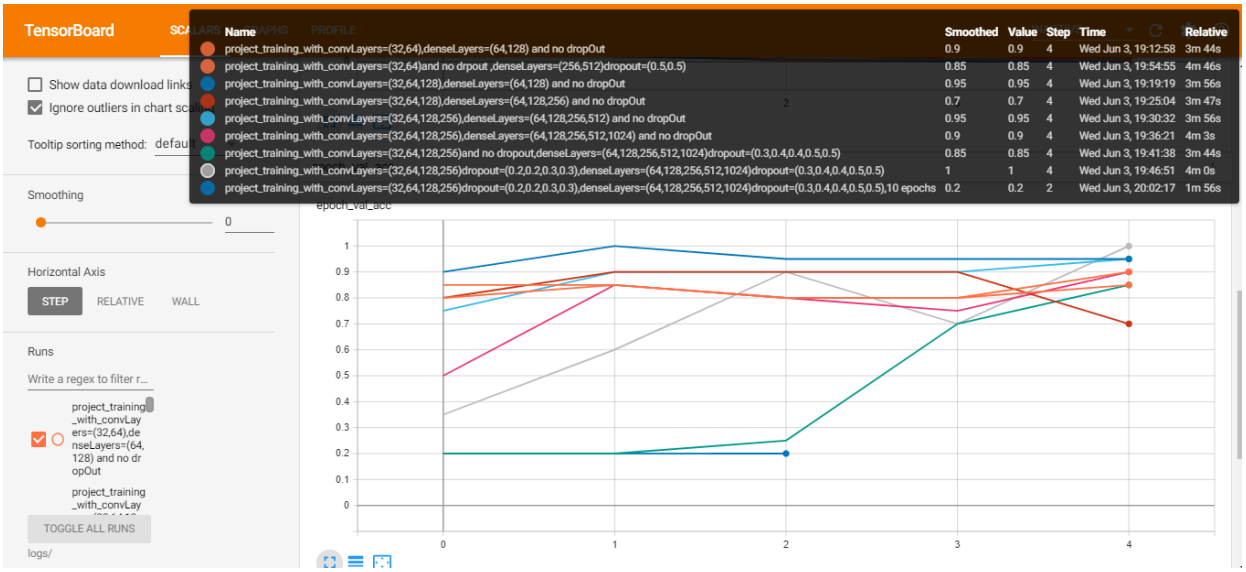
## 2) Loss




Here also we can see that models with no dropout regularizations have less loss.

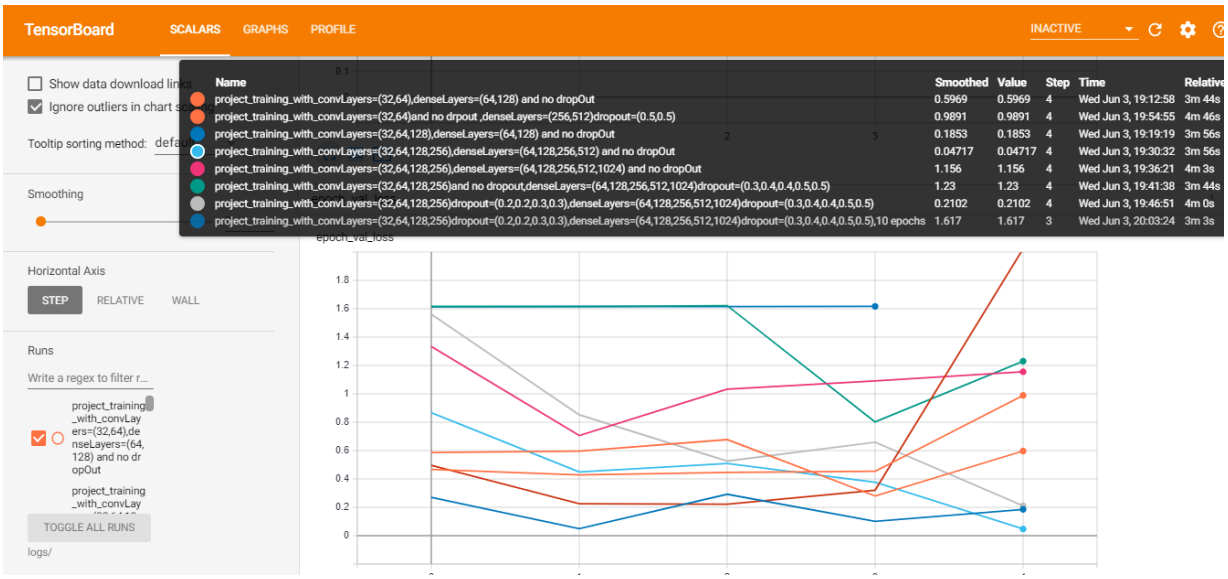
# Test Set Observations-:

## 1) Accuracy-:




Here we can see some interesting changes have happened. The models with no regularization are not the best anymore. Models with regularizations have performed better.

## 2) Loss-:




Here also models with regularizations have performed better.

## Observation-:

After training several different models with different rules of updating parameters we have observed that models with no regularization have performed very well in train set but underperformed in the test set. This must-have happened due to the overfitting of our models that is our model simply memorized train set.

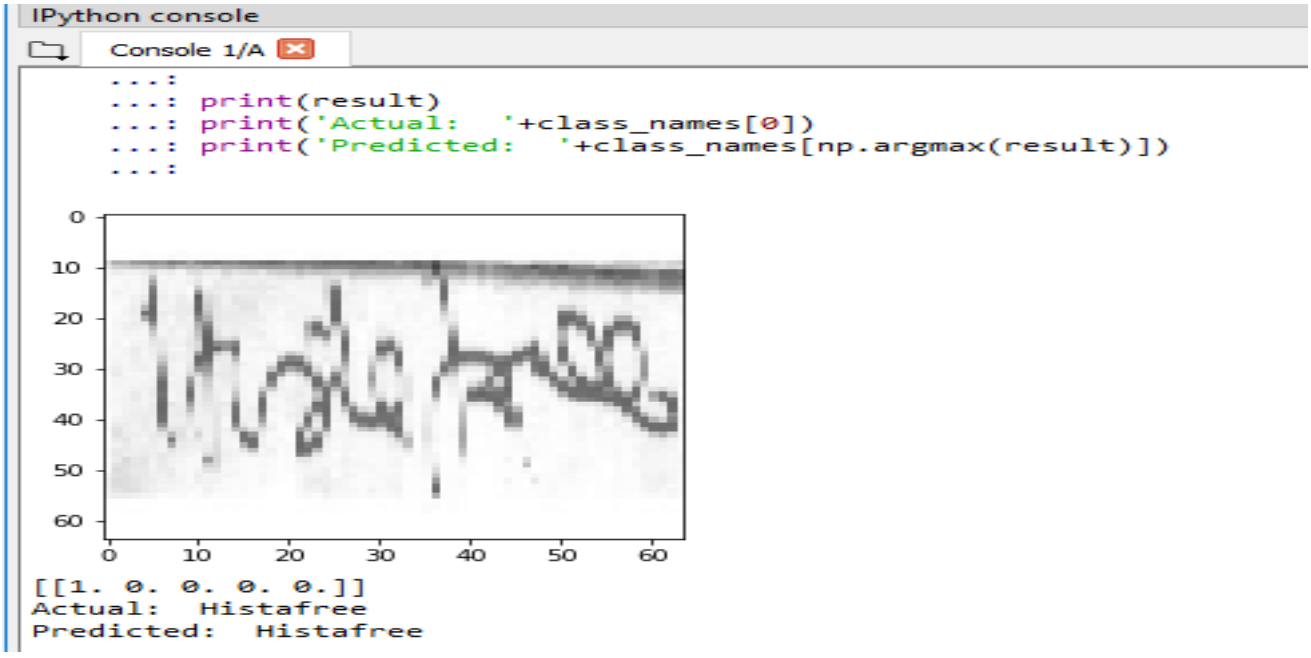
Similarly, models with regularization have performed well in both in both train and test set. Hence we can observe that here there is no overfitting. With more epochs, the results can be further enhanced.

Hence we have chosen a model with regularization, that has performed very well on our train and test set both.

## Evaluation-:

We have tested our model with several different images that it hasn't come across previously to verify the models performance.

1)

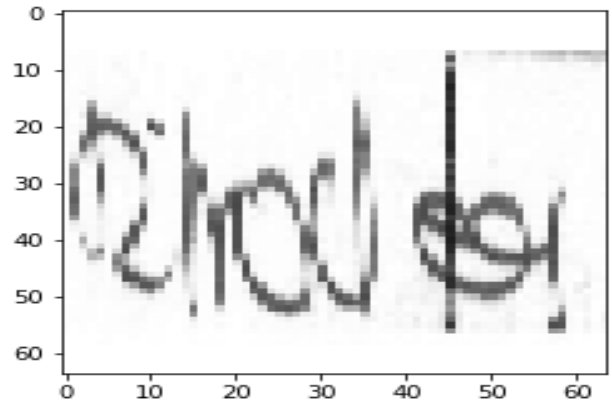


2)

IPython console

Console 1/A

```
...:
...:
...: print(result)
...: print('Actual:   '+class_names[3])
...: print('Predicted:   '+class_names[np.argmax(result)])
```



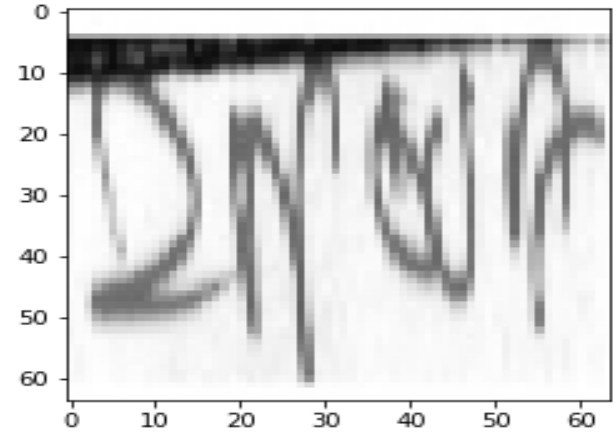

```
[[0. 0. 0. 1. 0.]]
Actual:  rhinoclear
Predicted:  rhinoclear
```

3)

IPython console

Console 1/A

```
...:
...:
...: print(result)
...: print('Actual:   '+class_names[4])
...: print('Predicted:   '+class_names[np.argmax(result)])
```




```
[[0. 0. 0. 0. 1.]]
Actual:  zinova
Predicted:  zinova
```

## Conclusion-:

We have achieved our target of getting a test set accuracy greater than 97% and a loss of 0.1%. Our testing with different images has also given positive results. We have achieved all this with a very small dataset and with less computing power. We can say now that with small datasets also we can develop an optimized model. In our models, we have updated parameters batch-wise and with the help of regularization techniques, we have enhanced the updation of parameters. Python's TensorFlow API was a great help in creating an efficient model. Using TensorBoard we kept a track of our models that is done in industries.

This much accurate results may not be obtained without a noise-free dataset. Cropping every single word from the scanned prescriptions was also a tough job to do. We can say that a good small dataset can be used. Because processing less amount of data doesn't require high computational power hence we need to learn how to "make a model learn patterns" from small datasets. Hence we can say that our results must inspire researchers with small datasets.

## Future Work-:

We need to train the model with more different words by doctors to see how the performance changes. And we also need to have datasets clusters from different places because doctors from one place generally suggest similar medicines. Hence it will help the medicine to detect words accurately.

# Timeline-:

| Sl. No. | Task | FEB | | MARCH | | | APRIL | MAy | JUNE |
|---|---|---|---|---|---|---|---|---|---|
| | | 10-20 | 20-29 | 1-10 | 11-20 | 21-30 | 1-30 | 1-31 | 1-4 |
| 1 | Problem Identification | √ | | | | | | | |
| 2 | Dataset generation | | √ | √ | √ | | | | |
| 3 | Coding | | | | | √ | √ | | |
| 4 | Documentation | | | | | | | √ | √ |

# References-:


1) https://www.youtube.com/watch?v=WvoLTXIjBYU
2) https://www.youtube.com/watch?v=lV09_8432VA&t=1215s
3) https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/
4) https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea