IIITV ML Challenge #01

A TWIST with MNIST

Team Name: Enigma

Team Members

Aman Kumar Raj	201951019
Jatin Goyal	201951072

Problem Statement:-

The task for this challenge is to create a model which can accurately identify the value from the images having values between 10 to 19.

Link of kaggle notebook:-

https://www.kaggle.com/jatingoyal123/mnsit-10-20

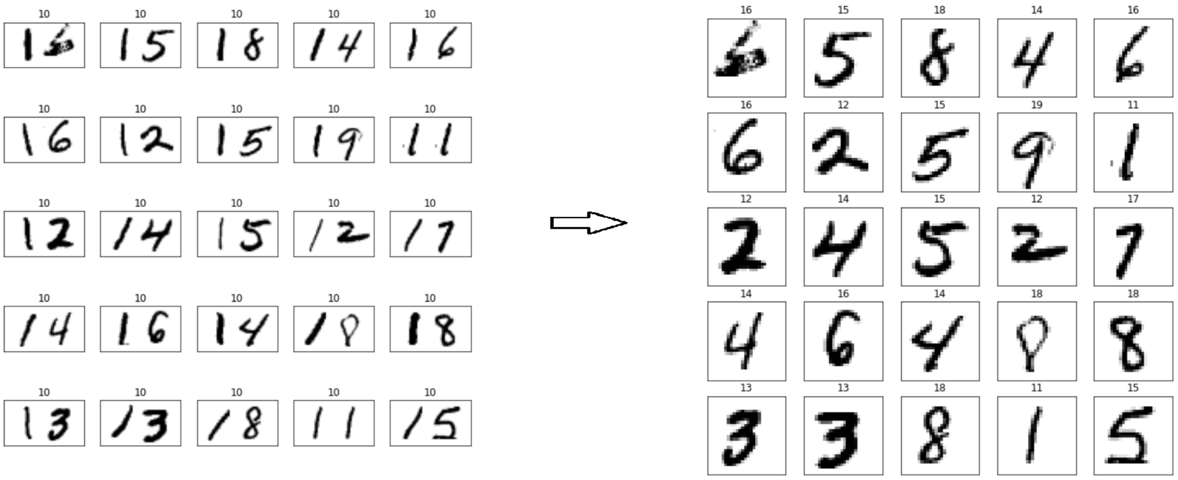
Approach:-

First, we created a baseline for this competition using different TensorFlow and PyTorch models and we achieved an accuracy of .93. Then, we tuned hyperparameters using optuna, which gave us an accuracy of .98.

But after this, we were not able to improve performance much. So, we analyzed the dataset and we found that images have size of 28*50 which suggested that two images of single digits have been joined together to create this dataset. Also, the two digits in each image were quite separate from each other.

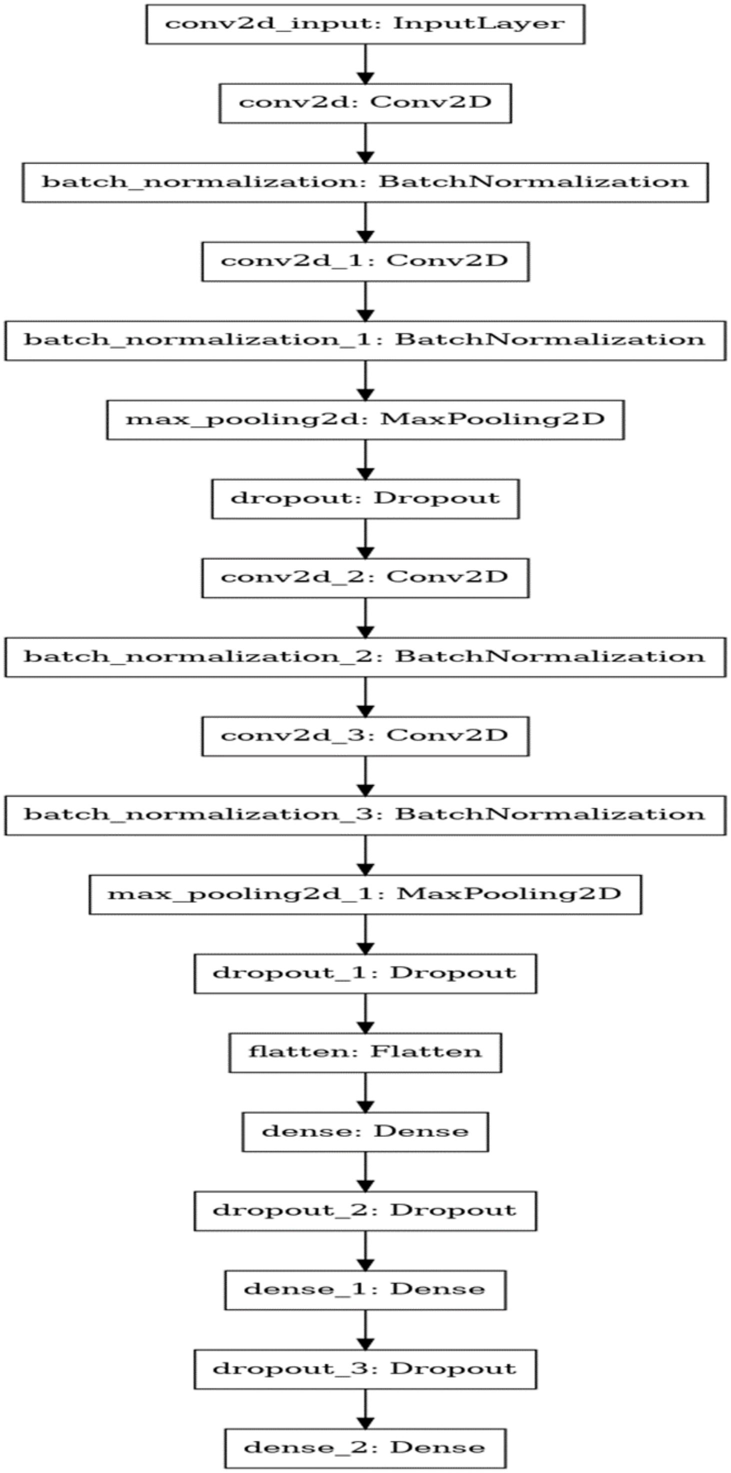
So, we first preprocessed the dataset to remove one number digit from each image. As the first 1400 columns of the dataset are pixel values of flattened images of the handwritten digits. So, we first reshaped it to 28*50. Then, we picked the last 28 columns of each image. Which made out image shape of 28*28. It removed the left side digit i.e one from each image.

Thus, our approach was to reduce the “A TWIST with MNIST” problem to a classical MNIST problem.



Then, we trained our model on this preprocessed data, and this time we had a boost from .980 accuracy to .998. Then after further tuning our model, we achieved an accuracy of 1.

Final Model Architecture:-



conv2d_input: InputLayer
conv2d: Conv2D
batch_normalization: BatchNormalization
conv2d_1: Conv2D
batch_normalization_1: BatchNormalization
max_pooling2d: MaxPooling2D
dropout: Dropout
conv2d_2: Conv2D
batch_normalization_2: BatchNormalization
conv2d_3: Conv2D
batch_normalization_3: BatchNormalization
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout
flatten: Flatten
dense: Dense
dropout_2: Dropout
dense_1: Dense
dropout_3: Dropout
dense_2: Dense

Model Summary:-

We used CNN to make our model. First, we used Conv2D layers with different filters like 512, 256, 128, and 64 with batch normalization in between them.

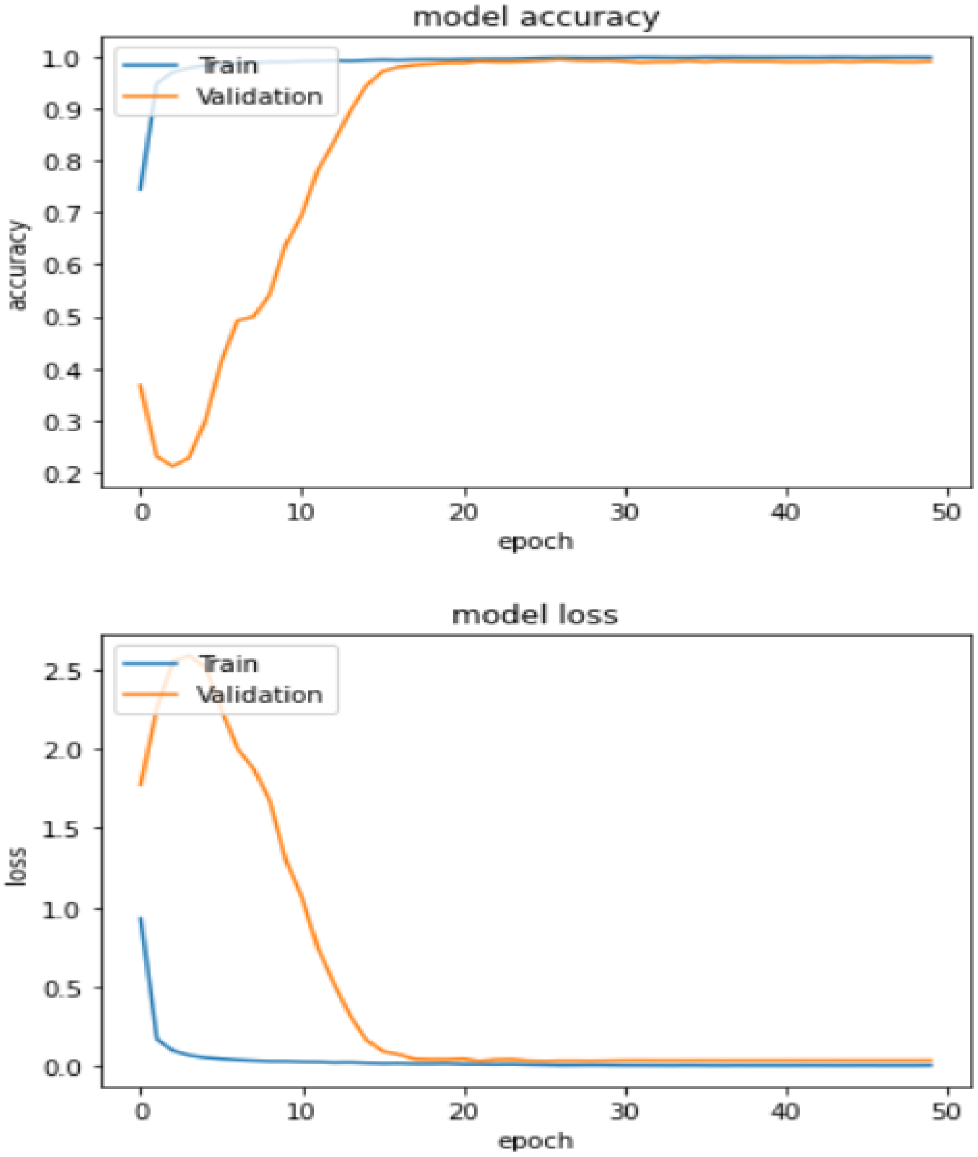
We used batch normalization for stabilizing the learning process which dramatically reduced the number of training epochs required to train deep networks.

We used the relu activation function in layers to help the network learn complex patterns in the data.

We also added dropout layers between them to avoid overfitting. After that we flatten the output and feed it into a dense layer of 512 and 128 neurons with activation function relu.

The output layer of model is a dense layer of 10 neurons(i.e. Number of classes to predict) with softmax activation function to predict probability of each class.

Training Plot:-


model accuracy
accuracy
epoch
Train
Validation
model loss
loss
epoch
Train
Validation