Machine Learning and Data Analytics Report

Signature Classification using Deep Learning

By: Aneri Gandhi
Roll Number: 17CS01003

ABSTRACT:

As part of the course Machine Learning and Data Analytics , under the given problem statement of classifying and identifying signatures, I have implemented a Convolutional Neural Network based model for signature classification. The model identifies signatures of 5 people distinctly and classifies others based on whether signature is present or absent. I have collected my own dataset and trained a simple model to fit it.

METHODOLOGY:

Step 1: Data collection

The first step to train any supervised algorithm is to create a training and testing dataset. Since I created the dataset on my own, it was a very small dataset. Total 186 images were collected and divided into 164 training and 22 testing for 7 classes. The seven classes are : {'Aneri' , 'Mona', 'Manoj', 'Sudipta', 'Yukta', 'Unrecognized_sign', 'Not_signature'}. The first 5 recognized signatures of 5 individuals and other two classes classifies as not a signature and unrecognized signature. The dataset can be found here.
The images are preprocessed and resized to (256 x 256 x 3) size. To overcome the problem of small dataset, data augmentation has been used while training.

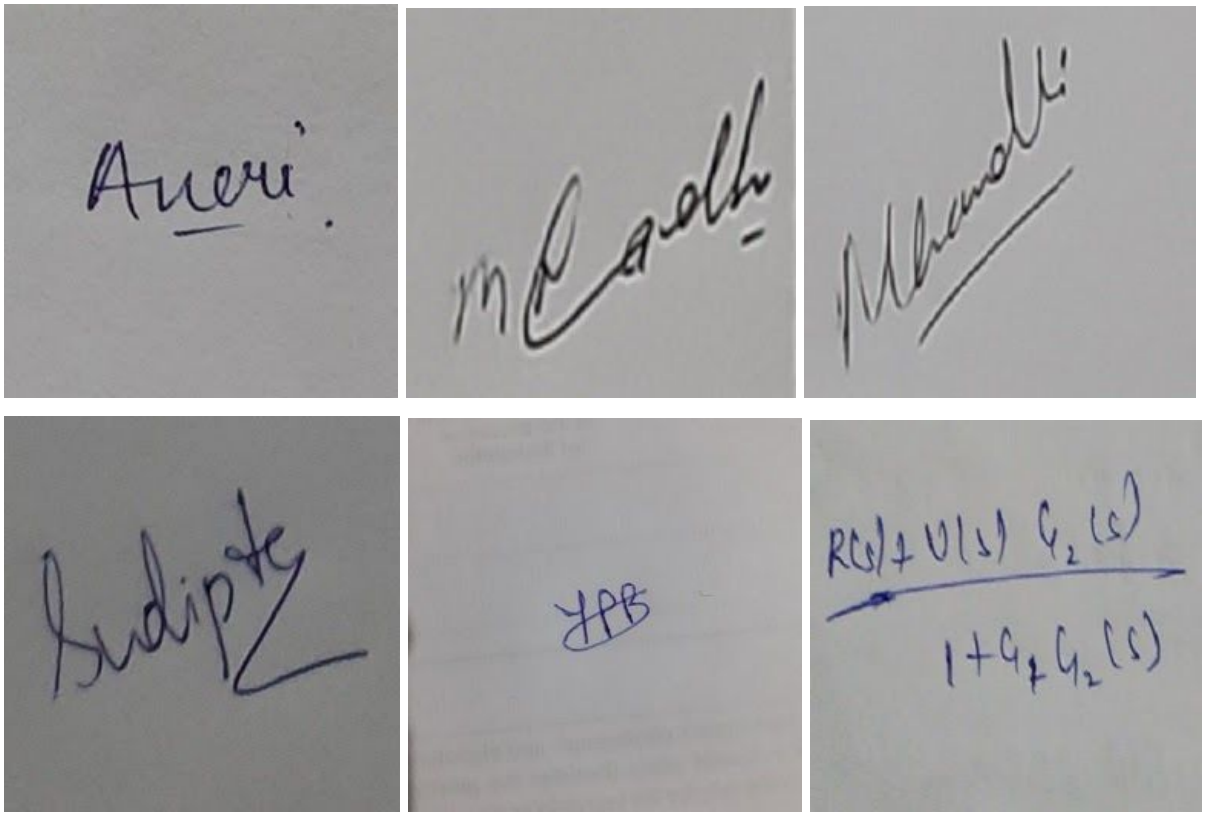

Figure 1: Sample Dataset. (a: Class Aneri, b: Class Manoj, c: Class Mona, d: Class Sudipta ,e: Class Yukta, f: Class Not_Signature)

Step 2: Model Creation

A simple convolution network is used to train the model. It has 3 convolution layers, 3 max pooling layers and 2 fully connected layers. For convolution, the 'same' padding is used to retain the size of input. ReLU activation function is used in each layer except the last layer where softmax function is used to categorize the data. The detailed model summary is shown below in Figure 2.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d_3 (MaxPooling2	(None, 128, 128, 32)	0
conv2d_4 (Conv2D)	(None, 128, 128, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 64, 64, 32)	0
conv2d_5 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_5 (MaxPooling2	(None, 32, 32, 64)	0
dropout_1 (Dropout)	(None, 32, 32, 64)	0
flatten_1 (Flatten)	(None, 65536)	0
dense_2 (Dense)	(None, 128)	8388736
dense_3 (Dense)	(None, 7)	903

Total params: 8,418,279
Trainable params: 8,418,279
Non-trainable params: 0

Figure 2: Model Summary

Step 3: Model Training

The model is then trained using loss function as **categorical cross entropy** and **Adam** optimizer. I tried training the model with two values of learning rate : 10^{-6} and 10^{-4} . Both times, the model was trained upto 500 epochs.

The training curves obtained are as follows:

Case 1: Learning rate= 0.0001

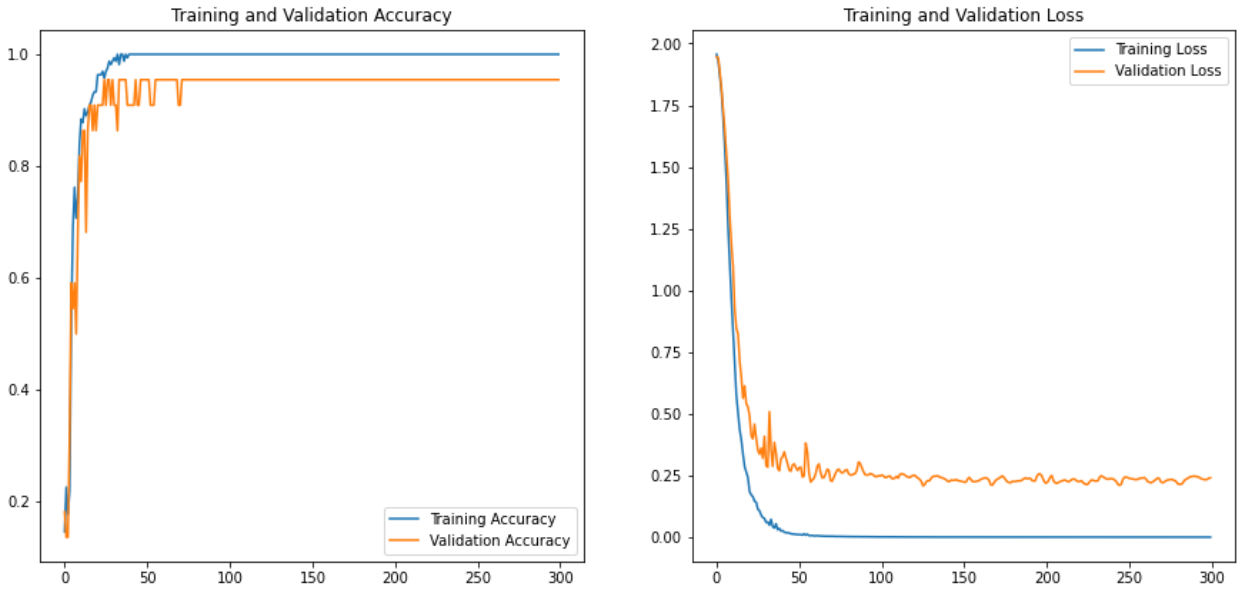

Figure 3: Learning curve with lr=0.0001 . (Note : The model was trained for 500 epochs but it is plotted only till 300 epochs as it saturates by then.)

Case 2 : Learning rate = 0.000001

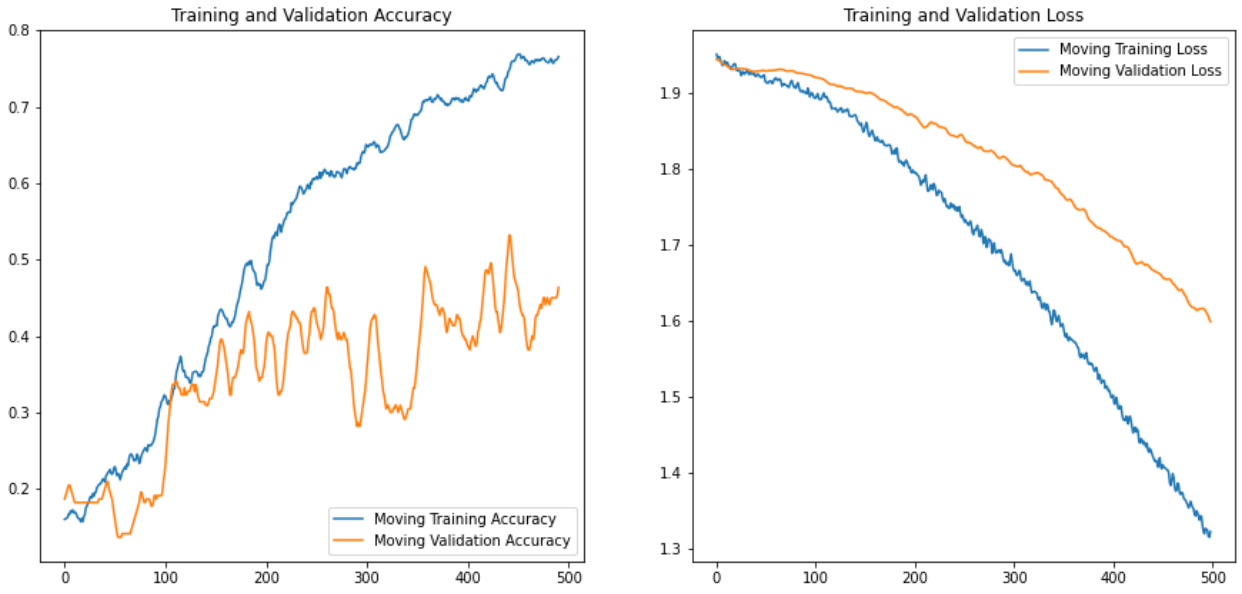

Figure 4: Learning curve with lr= 0.000001

We see with smaller learning rates, the convergence is not reached even after 500 epochs.

The entire code can be found here. The Model 1 has the stored weights of the model with learning rate (0.000001) {the model is trained only to 500 epochs i.e it is not yet converged.} and Model 2 has other model weights. The code used is in MIda_project.ipynb file.

Step 4: Testing Model

The trained models are used to predict the classes of test set used. The results obtained are as follows:

Case 1: Learning rate = 0.0001

	precision	recall	f1-score	support
Aneri (Class 0)	1.00	0.67	0.80	3
Mona (Class 1)	1.00	1.00	1.00	4
Manoj (Class 2)	1.00	1.00	1.00	3
Not_Signature (Class 3)	1.00	1.00	1.00	4
Sudipta (Class 4)	1.00	1.00	1.00	2
Yukta (Class 5)	1.00	1.00	1.00	3
Unrecognized_sign (Class 6)	0.75	1.00	0.86	3
accuracy			0.95	22
macro avg	0.96	0.95	0.95	22
weighted avg	0.97	0.95	0.95	22

Case 2: Learning rate = 0.000001

	precision	recall	f1-score	support
Aneri (Class 0)	0.33	0.67	0.44	3
Mona (Class 1)	0.40	0.50	0.44	4
Manoj (Class 2)	0.00	0.00	0.00	3
Not_Signature (Class 3)	0.75	0.75	0.75	4
Sudipta (Class 4)	0.00	0.00	0.00	2
Yukta (Class 5)	0.60	1.00	0.75	3
Unrecognized_sign (Class 6)	0.00	0.00	0.00	3
accuracy			0.45	22
macro avg	0.30	0.42	0.34	22
weighted avg	0.34	0.45	0.38	22

DISCUSSION and CONCLUSIONS:

We can conclude following things:

1. The learning is slow with lower learning rate. We can notice even after 500 epochs, the model with lower learning rate did not converge and required more epochs whereas the model with higher learning rate converged much faster.
2. We could also set a dynamic learning rate which changes with the epochs. Initially we can have a higher learning rate for faster execution and as it approaches minima, we could decrease it.
3. We can also notice low metrics, precision and recall, when the model is not converged and stopped in the middle. At the same time the model once converged, performs well on the test set.
4. Even though the model with higher learning rate converged by 300 epochs, executing over more epochs did not affect the results in this case due to small dataset. However it could also lead to overfitting. The data augmentation ,early stop and regularization could help prevent overfitting.

LINKS:

Complete project:
https://drive.google.com/drive/folders/1G4r5zWp-YIPP32dr2AAM4b31G10hGX1b?usp=sharing

Database:
https://drive.google.com/drive/folders/1hl3AmwKqho4amWW3O35ZC5-rwNHxaxGU?usp=sharing

Model_1_lr_0.000001 weights/stored model:
https://drive.google.com/drive/folders/15GywHruA_Tm3py5QvYBjYKMafHbTKM0o?usp=sharing

Model_2_lr_0.0001 weights/stored model:
https://drive.google.com/drive/folders/1dASEEwg_F6ILANysTBsj5A0PGsJFBMPq?usp=sharing

Code:
https://drive.google.com/file/d/1RD-8RJHCv_pmZFvW5uzm7NROJguGxDGr/view?usp=sharing