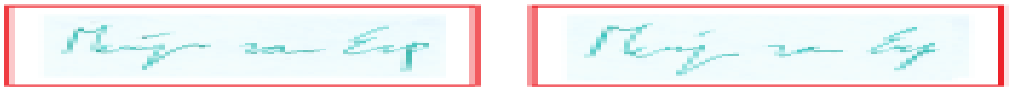
MLDA FIRST SEM PROJECT

SIGNATURE RECOGNITION



By - Kunal Goswami
20CS06014
Mtech. 1st-sem

Mentor - Prof. Lalit Mohan Patnaik
IIT Bhubneswar

Introduction

Signature provides an identification as well as a confirmation. Our model presented is about classification of signature and text data. This classification model can also help in building the Signature Detection model for the document images.

Tools

- Language – Python 3
- Library – matplotlib , keras, tensorflow
- IDE – Jupyter Notebook

Data preparation

I downloaded data from
https://www.kaggle.com/kayademirs/signature
It had two directory Test and Train.
Train contains two class – Class 0 & Class 1.
It will be used to train our model.
Then we have Test folder for checking accuracy.
Class-0 contains our signatures and Class-1 contains our text in real world.

Model used

I have used Convolutional neural network. And done epoch for 50 times.

Training

I have used following python code for training.

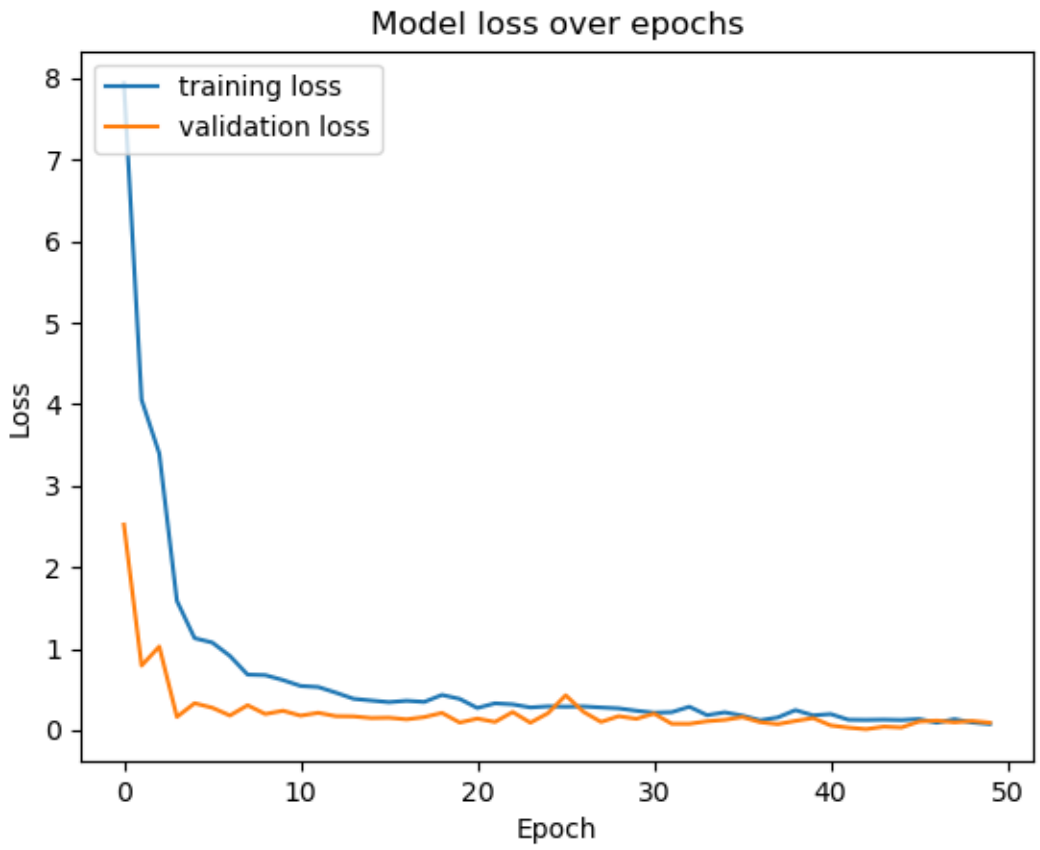
```
rom keras.layers import Lambda, Conv2D, Dropout, Dense, Flatten,
MaxPooling2D
from keras.models import Sequential
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt


imag_shape = (100, 100, 3)      # 3-> r,g,b
nb_epoch = 50                # it reduces error
learning_rate = 1.0e-4


model = Sequential()       # sequential neural network
# 1st Layer
model.add(Conv2D(filters=24, kernel_size=3, activation='relu',
input_shape=imag_shape))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=36, kernel_size=3, activation='relu',
input_shape=imag_shape))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=48, kernel_size=3, activation='relu',
input_shape=imag_shape))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(64, activation='relu'))    #
model.add(Dense(32, activation='relu'))

#Readout Layer
model.add(Dense(2, activation='sigmoid'))

checkpoint = ModelCheckpoint('model-{epoch:03d}-
```

```
{val_loss:03f}.h5',
                                monitor='val_loss',
                                verbose=0,
                                save_best_only='true',
                                mode='auto')                    # It
saves data

model.compile(loss='binary_crossentropy',
optimizer=Adam(lr=learning_rate))  //

datagen = ImageDataGenerator(
       rotation_range=20,
      zoom_range=0.15,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.15,
      horizontal_flip=True,
      fill_mode="nearest")

datagen_v = ImageDataGenerator()

history =
model.fit_generator(datagen.flow_from_directory(directory="./Train
/",target_size=(100,100),color_mode='rgb',batch_size=15,

class_mode="categorical",shuffle=True,seed=42),
steps_per_epoch=15,
                        epochs=nb_epoch,
validation_data=datagen_v.flow_from_directory(directory="./Test/",

target_size=(100,100),color_mode='rgb',batch_size=15,class_mode="c
ategorical",shuffle=True,seed=42),
                        validation_steps=
3,callbacks=[checkpoint],verbose=1)

plt.plot(history.history['loss'],label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.title('Model loss over epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper left')
plt.savefig('model loss.png')
```

Working

After describing the model architecture, we start giving model the data. The data before going into model is augmented, this is the

pivotal point for my project today. I tried manually augmenting data but the module ImageDataGenerator helped do it automatically albeit except a few missing things. The manual data augmentation also had adding random shadows and random brightness to the photo but I can work with them not present. Image augmentations like rotation, rescaling and zooming are present. I tried making model with maxpool and without it as well, the difference in loss and accuracy in both of the models is not very significant so I considered discarding maxpool in order to shave off some computational time. The activation function used is ReLU and the final layer activation function is Sigmoid. A dropout layer is added with dropout probability 0.5. The model is compiled with evaluation metric loss as binary_crossentropy and optimizer as adam optimizer. As for saving the model, what I did was. I used validation loss as metric for saving the model. The model with lowest validation loss will be saved. Also below is the graph of test and train loss over the epochs. I found it to be satisfactory value of loss and called it a day. Final loss was around 0.028



Next up is testing the model with my own signature, when provided an image the model gives output in form of 2 values, values with probability for both classes, a simple if else to decide which class it belongs to.

Testing

I used following python code for testing one of our test data.

```
from keras.models import load_model
import cv2
import numpy as np

model = load_model('model-042-0.030619.h5')

image = cv2.imread('t2.png')
image = np.asarray(image)
image = cv2.resize(image, (100, 100))
```

```
image = np.array([image])

print(model.summary())

x = model.predict(image, batch_size=1)

t=list(x[0])
print(t)
if(t[0]>t[1]):
    print("It is a signature")
else:
    print("It is not a signature")
```

OUTPUT

```
[0.84580004, 0.06019926]
It is a signature
```

Acknowledgement

I will also thanks to Smannay, Abhisek, Homi who has inspired and help in collaborating to understand the dataset.

Reference

- *https://towardsdatascience.com/classification-of-signature-and-text-images-using-cnn-and-deploying-the-model-on-google-cloud-ml-30bf6f4e3207*
- *https://www.kaggle.com/kayademirs/signature*
- *http://www.iapr-tc11.org/mediawiki/index.php?title=MSRA_Text_Detection_500_Database_(MSRA-TD500)*
- *https://www.youtube.com/watch?v=umGJ30-15_A*