CNG483 INTRODUCTION TO COMPUTER VISION

PROJECT 3: IRISDEEP

Berkant Tuğberk Demirtaş – 2315232
Sabahattin Yiğit Günaştı - 2315281

ABSTRACT

Identity recognition task on iris recognition with CNN

***Index Terms*—** Identity; Recognition; Classification; CNN; Neural Network; Iris Biometric; Python; Tensorflow;

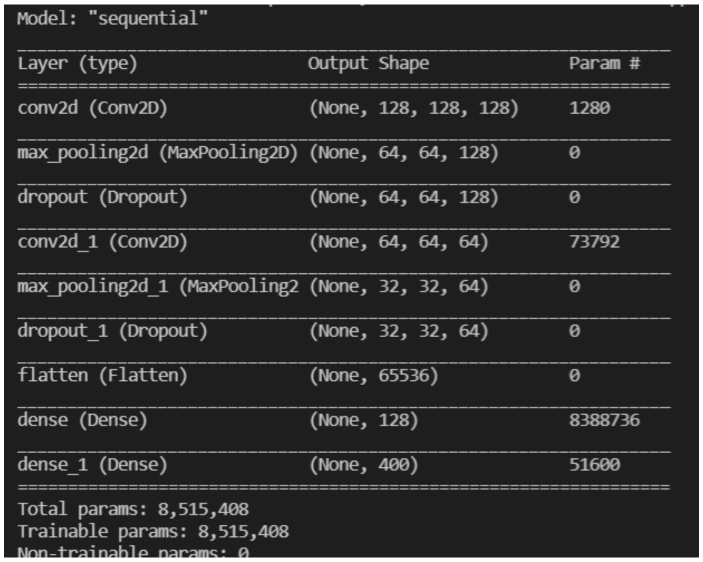
1. INTRODUCTION

This report will briefly explain the project, that is about recognition with CNN using Iris Biometric dataset, details. From pre-processing to results , everything is clearly explained and reported.

2. PREPROCESSING

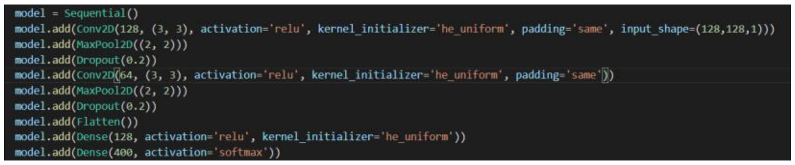
Before talking about pre-processing, we need to understand our dataset. In the folder there are 1600 Iris Biometric photos and a txt file includes the Iris coordinates. Initially, pre-processing process began with the reading dataset and getting iris part from ta photo based on txt file. After getting iris data, the labelling process began. All data was labelled based on their names. Back to dataset, there are 4 photo for left and right eye for a person. They should be considered as 2 people since left and right may be different. Labelling was done based on this info and names. When getting labels and iris crops, splitting data into train-test-validation process came. The data divided into test-train-validation based on the given info in the project description(2 train, 1 test, 1 validation). Apart from splitting rate, 'stratify' parameter was used for being able to split data through labels. Finally, we have the data with pre-processed. Last thing to do, before giving data to the model, we arranged the sizes ; and we applied the one hot encoding to the labels.

3. CNN – ARCHITECTURE



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 128)     1280
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 64)        73792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 128)               8388736
_________________________________________________________________
dense_1 (Dense)              (None, 400)               51600
=================================================================
Total params: 8,515,408
Trainable params: 8,515,408
```

We design our CNN Architecture with 2 CNN layers, 2 dense layer and 2 dropout layer. We used 2D kernels of 3x3 kernel size in both layers. Number of neurons is decreasing like a pyramidal shape. Last dense layer had 400 neurons because we have 400 different classes. The input shape of the first layer is 128x128 since we resized the image in pre-processing . We also have 2D pooling with 2x2 size

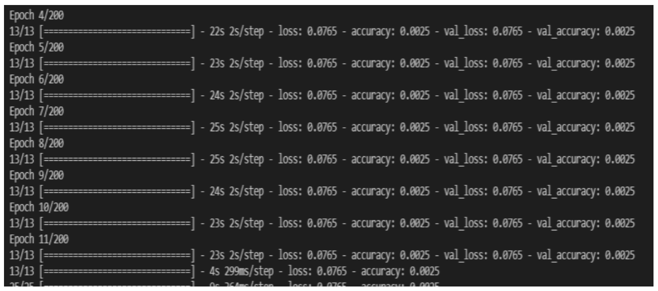


```
model = Sequential()
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(128,128,1)))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(400, activation='softmax'))
```

The reason for choosing this parameters is about trying to optimise the model with trying multiple models. For building this model, we searched and analyzed diffrenet CNN models for different tasks, nearly all of them were using similar default parameters for the tasks. So, we decided to build our model based on this default values, and with the experiences from Project 2 , we tried to optimese model with small changes. We used early_stopping method for determining epoch size dynamically. We deteremined the patience value from our last project. For the batch size, again we tried multiple different sizes and determined with the optimal one. We used RMSprop optimizer.

4. TRAINING RESULTS

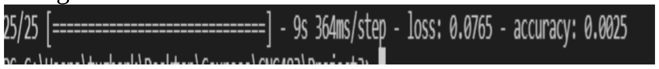
Unfortunately, our model can not learn anything. We couldn’t train the model with different parameters. We thought that it can be from the dataset. It is not sufficient enough for the labelling 400 class correctly. Apart from that, we tried different models also, but our validation accuracy is nothing but less than %1.



```
Epoch 4/200
13/13 [==============================] - 22s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 5/200
13/13 [==============================] - 23s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 6/200
13/13 [==============================] - 24s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 7/200
13/13 [==============================] - 25s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 8/200
13/13 [==============================] - 25s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 9/200
13/13 [==============================] - 24s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 10/200
13/13 [==============================] - 23s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
Epoch 11/200
13/13 [==============================] - 23s 2s/step - loss: 0.0765 - accuracy: 0.0025 - val_loss: 0.0765 - val_accuracy: 0.0025
13/13 [==============================] - 4s 299ms/step - loss: 0.0765 - accuracy: 0.0025
```

5. TESTING RESULTS

Since we couldn’t train the model, our test results are also a disaster. We are getting about %0.25 rate which means our computer learn nothing. We know the problem is in train process, but we couldn’t do anything about that. We tried different parameters with different models but nothing changed.



```
25/25 [==============================] - 9s 364ms/step - loss: 0.0765 - accuracy: 0.0025
```

6. ADDITIONAL COMMENTS AND REFERENCES

Our model can’t be trained. We tried to figure it out but we failed. This is possibly because of model architecture, but even if we succeed to design it correctly, the result can not be high. 2 training image can not be enough for labelling 400 class. There are too much classes.