CNG483 INTRODUCTION TO COMPUTER VISION

PROJECT 3: IRISDEEP

Pınar Dilbaz – 2243392
İbrahim Aydın – 2151835
Muhammed Didin – 2243384

ABSTRACT

In this project, we tried to train our algorithm to recognize the eye with the help of CNN, which we prepared by cropping a given eye around the parameters and shaping the segment we took into the appropriate shape. Thanks to CNN, we were able to implement a very difficult task simply by using Machine Learning. For this, we used 2 of the 4 eye images of 400 different people for the training set, one for the validation set and the other one for the test set. As a result, we achieved up to 35 percent success from such a small set.

***Index Terms*—** CNN; Machine Learning; recognition; keras; RMSprop; softmax

1. INTRODUCTION

In this report, we will explain how we use CNN to recognize eye pictures. In our dataset containing 1600 images, there are 400 different people's eye images, 4 of them are eye images of a human, and we used 2 images for each person for the training set, 1 for the validation set and the last image for the test set. Then, we tried to reach the optimal result by preparing a CNN algorithm with the combinations of convolution, leaky relu, pooling, dropout, flatten, linear dense functions.

2. PREPROCESSING

In the pre-processing part, we first got the parameters. According to the parameters we received, we read the images from the database in grayscale format and cropped them again according to the data in the parameters. Then, we resized it to 128 128 and reshaped it to be a single color channel and adjusted the color parameters to be between 0 and 1. Then, we created the input list by throwing these input pictures into a list and converting the list to float type. While reading the pictures for the output list, we added the person to which those pictures belong to another list as a number (integer) and converted them to float type and completed the output list.

3. CNN – ARCHITECTURE

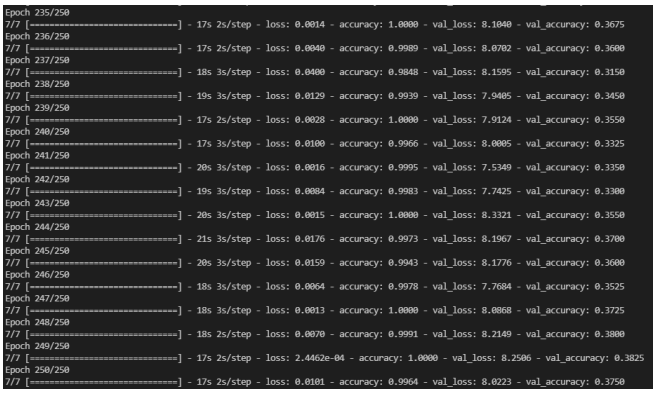
Initially, we created a padded conv2D input layer with 32 neurons, kernel size [3,3] and accepting inputs of (128 128 1). We added the leaky relu layer, which is a more advanced version of the Relu models. Next, we add a padding maxpooling2d layer of size [2,2]. We added a dropout layer to get a more efficient result. Then we repeated these four layers using conv2d layers with 64 and 128 cores, respectively. We converted them into vectors with the flatten layer and added a linear fc dense layer with 128 neurons. Then we added a leaky relu and dropout layer and finished with a softmax output layer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 128, 128, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4194432
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 400)               51600
=================================================================
Total params: 4,338,704
Trainable params: 4,338,704
Non-trainable params: 0
```

Since we always get more accurate results with one hot encoder, we first arranged our output in one hot encoder format and determined our output length as 400. We set our batch size to 128 (it was slower at lower values) so that it would not slow down our program too much later. As a result of our trials, our result reached its maximum value when the epoch was 250. Moreover, our neuron numbers are only in line with the information we have obtained because of trial and error and the research we have done. In addition, adding padding also contributed to the improvement of our result. We have seen that staying close to 0.3 as Hyper parameter is beneficial for us by trial-and-error method. We used the categorical crossentropy loss function and RMSprop optimizer because we were doing the categorization process.

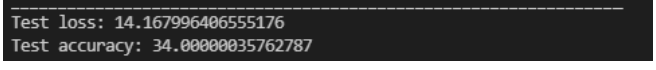
4. TRAINING RESULTS

Parameters:
Epochs = 250
batch size = 128
kernel size = (3,3)
pooling = (2,2)
dropout = 0.3



5. TESTING RESULTS

Our result is between %30-35 band we are getting increase results as we increased our epoch but wait time is not justifiable to corresponding increase in the accuracy. So around 250 epoch we get the ideal result. (accuracy).



6. ADDITIONAL COMMENTS AND REFERENCES

(if there any)