Gender Classification from Eyes Images by using Keras and Resnet50 pretrained model

Noureddin Mohammed Lutfi
0197061
Computer Engineering Department
The University of Jordan
Amman, Jordan
nor0197061@ju.edu.jo,
lutfenour@gmail.com

***Abstract*—Image classification used to be a big deal in the past, but now with modern technologies and advanced deep learning it became much easier, this report contains two different models that can classify human gender from eyes images with different optimizers, models, and parameters. It contains data preparing, model building, testing and many more. The two models used in this project are Keras sequential model and Resnet50 pretrained model.**

Keywords—Image, Classification, Keras, Resnet50

I. Introduction

Gender classification has always been a challenging topic due to human face shape difference and because face contains a lot of details, for example moustache and beard, and so many variables that can affect results such as aging, color and many more. AI and deep learning made this kind of problems much easier and offered a helping hand that can reduce error and effort, in our case of study we need to classify the gender based on eyes images using different techniques to get the best solution [1].

II. Data and Materials

A. Dataset Preview

As earlier mentioned, our dataset contains male and female eyes, RGB coloured and different sized which means we need to do some data preparation. The dataset is not equal since we have different images count per gender, look at Fig 1 which shows a bar plot that describes the dataset. Also, you can find the used dataset in the appendix.

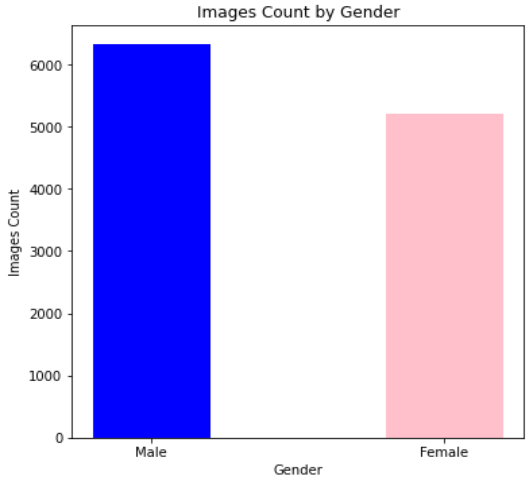

Fig 1. Images Count by Gender

As we can see we have a pretty big dataset, it actually contains 11525 image, 6323 male image and 5205 female image, also we can obviously notice that the dataset is not equal with a quite big difference, which is about 1000 image for male gender, hence we need to consider this while preparing the data for training. For more details a small sample of the dataset before making a simple edit is shown in Fig 2 and Fig 3.

Fig 2. a Sample of Female Eyes

Fig 3. a Sample of Male Eyes

B. Data Preparation

Our main goal is to classify the images into two categories which means it's a binary classification problem. First, we need to resize images to (48,48) pixels which is a convenient size for our dataset, by using PIL image library we can represent these images as 3d array (48,48,3) since we have RGB picture so a new dimension must be added next to the size.

Now we need to somehow save the arrays into another array to get an array of arrays and each single array specifies an image, also save it as npy file so you don't have to repeat this step each time you run the code.

Briefly, Each small array (48,48,3) represents a cell in the big array (11525,48,48,3) that contains the whole images, and each 3d array will be represented as follows : 2d array (48,48) and each cell in the 2d array contains 3 elements that represents 1 pixel with a value between 0 - 255 range which specifies how light or dark the color is (0 for zero light and 255 for maximum light), so we end up with a 2d array of 3 elements which represents the 3 colors (RGB which means Red, Green and blue), so we need 6912 value to represent 1 image and here is a random sample in Fig 4 that shows a male and a female eyes images after resizing to (48,48) pixels.

Fig 4. Random Male and Female Eyes Images

After we get our images converted to array of arrays, we can deal with them since they are numbers now, but we need to add 1 extra column to the big array to represent the labels because we rely on supervised learning technique in our algorithms, it will have either 0 which represents a male or 1 which represents a female.

Moreover, now we have one big array with data and labels which we can split into train set and test set with 0.25 ratio for test set, for validation we can split the train set into train set and validation set with 0.1 ratio for validation set so we get a better accuracy at the training stage.

Finally, one more step is needed as we must scale the data, because the algorithms we will use requires a scaled data to give a good reliable model with high accuracy, given that our data values range is (0-255) so we can simply divide all the sets we made by 255 (just like min-max scaler algorithm) which will give us float values between (0-1).

III. Buliding and Training Models

After we prepared the data we can try some models, train them with different optimizers, test our model on the test set and observe the results. Furthermore, in our case we will be using Keras model and Resnet50 pretrained model.

A. Keras Model

Keras is a high-level API that runs on top of TensorFlow used for deep learning and building neural networks which we will be using it as our first model.

Building Keras Model

In this model we will apply a sequential model with 4 layers, 3 Dense layers and 1 Flatten layer, the first layer will be the Flatten layer that takes a 3d array as input and converts it to 1d array of 6912 input shape so we have 6912 neurons for the input, the next follows are 2 Dense layers with 450 and 300 output neurons that use relu activation function respectively, this is the number of neurons which we ended up with after many trials that gives better accuracy based on our observation, last layer will be Dense layer too, but with sigmoid activation function and 1 output neuron since our output will be either 0 for male or 1 for female, take a look at Fig 5 that represents the sequential model which we built architecture .

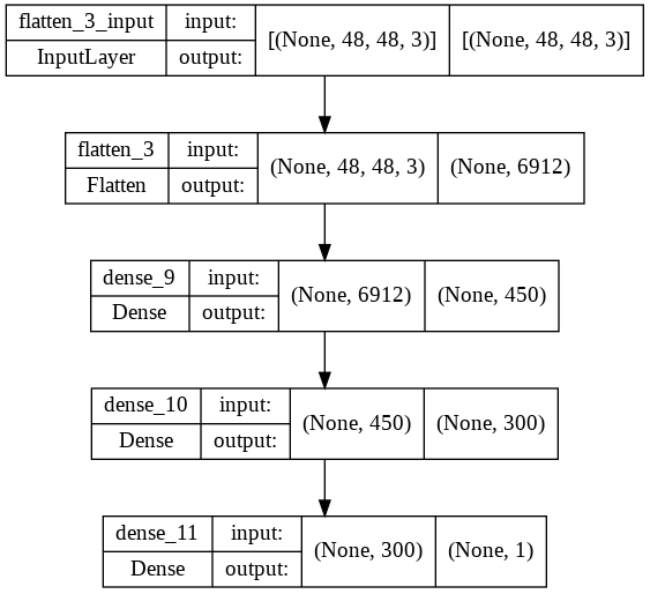

Fig 5. Keras Sequential Model Architecture

Compiling and Training Keras

Now our model is ready for compilation, with 3 parameters loss, accuracy and optimizer. In this stage we will use binary_crossentropy for loss, accuracy for metrics and we will try two different optimizers and observe the optimizations once as adam optimizer and once we will try the sgd optimizer.

a) The first trial will be using the adam optimizer then we will train the model on the prepared dataset and validate it using our validation set, we will use 20 epochs for this model and observe the accuracy as it converges to about 94% on the train set and 88% on the validation set.

b) The second trail will be using the sgd optimizer then we will train the model on the prepared dataset and validate it using our validation set by compiling the same model and only change the optimizer to sgd then observe the accuracy as it converges to about 97% on the train set and 88% on the validation set, which means a slight change in the validation set since 97% on the train set doesn't always mean high accuracy and may cause overfitting which we can check by evaluating on the test set.

look at Fig 6 and Fig 7 which contain useful data about the model scores over 20 epochs for the same model over different optimizers.

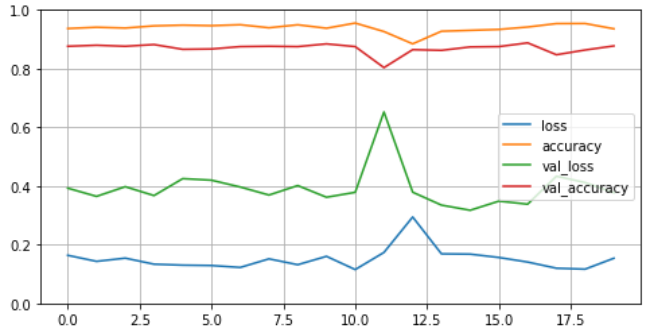

Fig 6. Keras Scores Over 20 Epochs using Adam

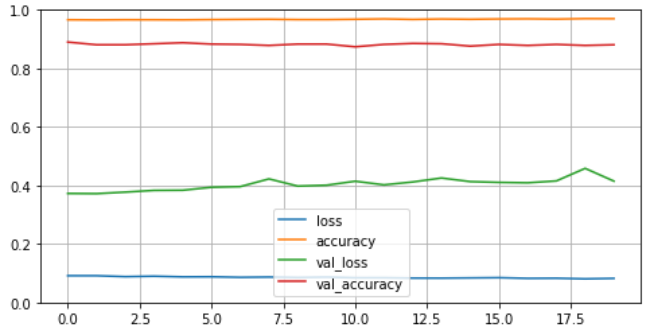

Fig 7. Keras Scores Over 20 Epochs using SGD

We can see that we have a normal output and as expected To be for a sequential model with binary classification problem, but we have a high loss which is about 37% for adam optimizer and 40% for sgd optimizer for validation set, so we need to see the test set results and build out next assumptions based on it.

TEST KERAS MODEL

Finally, after compiling the model and observing the output of the previous trials we can test both on the test set that we created earlier and observe the results. After we have evaluated the adam model on the test set we got 88% accuracy and 34% loss, which is good for the accuracy, but we need a better loss value because it is not a good loss value, while the sgd optimizer had a slightly better accuracy with 89% on the test set but worse loss with 40% which is high loss so we need to try another model, or maybe you can change some parameters to get a better estimations.

As a result, we got no overfitting, but we need to change some parameters and do more research about the model if we want to get better loss values on the same model.

B. *Resnet50 Model*

Resnet50 is a pretrained model with 50 layers which has been trained over millions of images, we will be using this model in our research and check the results that it would give on our dataset.

BUILDING RESNET50 MODEL

In this model we won't build from scratch, this model is already built, and we are just going to reuse it in a way that gives us better accuracy since this model has proved that is very reliable especially for image classification problems and it is based on Rennet basic model [2].

We will use the basic model of Resnet50, look at Fig 8 that represents the Resnet50 pretrained model architecture.

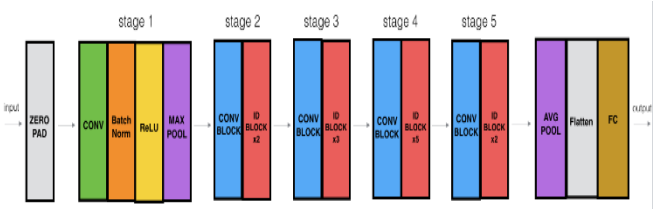

Fig 8. Resnet50 Pretrained Model Architecture [2]

In this model we will use the basic layers with some added features, input shape must be added to the layers as a 3d array (48,48,3),starting from sequential model, next is adding imagenet weights and setting pooling to max. We will also add a Dense layer with 1 neuron and sigmoid activation function for the output and our model will be ready for compilation, check appendix for more information about the code and deep understanding of the build stage.

COMPILING AND TRAINING RESNET50

This model compilation stage is straight forward since our data is ready and all what we have to do is to specify the optimizer which we decided to be sgd, ser loss to binary_crossentropy and metrics to accuracy.

Our next step will be training the model by giving the right parameters for train and validating sets, with 15 epochs which is a good estimation after many trials we tested, but it is hard to include in the research since we have a limitation for the resources we are using, also we decided to give a 100 steps per epoch and it gave a good estimations.

Finally, we will fit the data and wait for the 15 epochs to finish, the results we got on the training and validation set are much better than keras sequential model whether for the adam or sgd optimizer on both train and validation set with 100% score for training set which is huge and we might think its overfitting, but this happened due the complexity of the model not overfitting, also we got only 0.12% loss on the train set which is a huge difference comparing to keras sequential model, but still we need to look at validation and test set since

they are the objective of our optimization. For validation set we have 18% loss which is about half the value of the previous model as well as 95% percent accuracy on the validation set which is a remarkable improvement, look at Fig 9 which shows the change of Resnet50 model over 15 epochs.

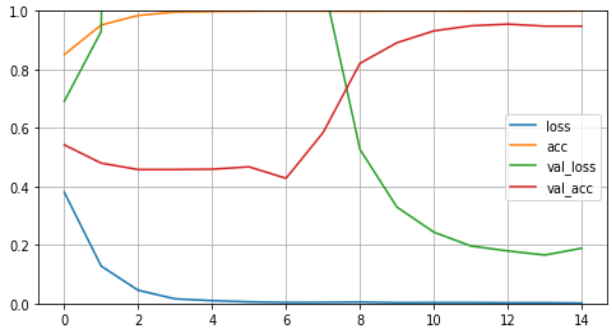

Fig 9. Resnet50 Scores over 15 epochs

We can obviously notice how scoring changes over the epochs, we start with high accuracy for train set and it keeps increasing until it reaches 100% which sounds a bit confusing, but we can simply wait for the test set evaluation to make sure its not overfitting the data, we can also see how the validation accuracy score is a bit lower than train set accuracy which is considerable. Furthermore, we can see how the train and validation loss begin with a high value for the first epochs and start to decrease as long as the model is learning and validating until it reaches the last epoch and here it will get the best values for loss with a better value (lower) for the train and validation loss.

Test Resnet50 Model

Finally, after compiling, training and validating we can simply evaluate on test set and check the results. After just running the last command in our code we can observe the output of evaluating the test set and the results were promising, we get a higher accuracy which is about 95% for the test set just like the validation set and 19% loss on the test set which is very close to the validation set too and is acceptable loss, but we think there is still more optimizations could be done to reduce the loss and maybe increase the accuracy which isn't as necessary as reducing loss since we have a very good accuracy on the test set and no overfitting.

For more details, look at Fig 10 to that shows an example of predicting the gender of a random sample of two images using Resnet50.

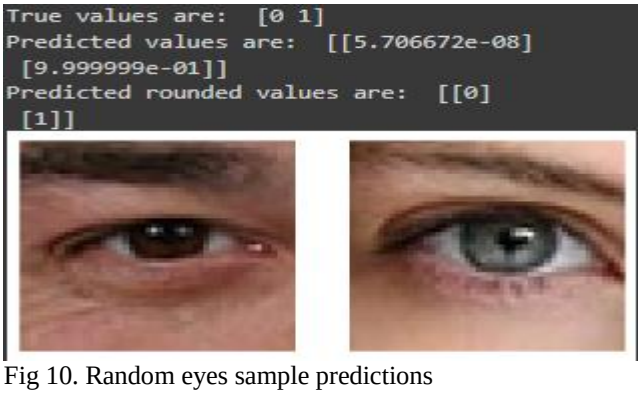

Fig 10. Random eyes sample predictions

In Fig 10 we can see the predicted values of a random eyes sample using Resnet50, knowing that the true values are 0 (male) and 1 (female) for the image 1 and 2 respectively, we can see that the prediction values are so close to the labels, and then we rounded the values to get 0 and 1 which are the same as the labels we have. Also, you can modify the code attached in the appendix section in a way to test for both sgd and adam optimizer using the sequential model and observe the results.

Extra Notes

So far, we have made our own models using different techniques and using a pretrained model, but its important to mention that you might have noticed where the most work and effort is about preparing the data and choosing the right optimizer since it's easy to fit data and try multiple models by only fitting the prepared data to a new model.

It is good to mention some other good models that work well for image classification problems such as VGG-16, Inception and EfficientNet which we didn't have a chance a try because of the usage limit of the resources, you can find all needed material in the Appendix section, also the three earlier mentioned models in addition to Resnet50 model can be found there too.

HONOURABLE MENTION

These two models were tested and coded using google colab notebook which offers you some free resources to use but limited for GPU with some constraints, it is simple, flexible and free, but with some usage limitations .

Conclusion

At Last, we have tested two models so far with different types, one uses sequential keras model which we tried for different parameters and using two different optimizers which didn't give much better performance, so we tried to make another model which we didn't build from scratch, it was Resnet50 pretrained model which proved it is a good model as it gave us much better accuracy with 95% and only 18% loss which is not perfect, but it is a good value and it could be optimized by trying different optimizers, parameters, number of epochs and many more variables that can affect the performance of our models, we can say we have Resnet50 is the best model among the models we have tested with best accuracy and lowest loss.

Refrences

[1] CIMTAY, Yucel, and Gokce Nur YILMAZ. "Gender Classification from Eye Images by Using Pretrained Convolutional Neural Networks." The Eurasia Proceedings of Science Technology Engineering and Mathematics 14 (2021): 39-44

[2] Dwivedi, Priya. "Understanding and Coding a ResNet in Keras." Towardsdatascience, 4 Jan 2019, https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33#:~:text=The%20ResNet%2D50%20model%20consists,over%2023%20million%20trainable%20parameters.

Appendix

- https://www.kaggle.com/code/fatihdeniz/cnn-female-male-eyes
- https://github.com/TroJan2001/Artificial-Intelligence/blob/e6cf731054554cc0e294cb46ace0bd23317ac4f8/Project.ipynb
- https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/