Close or Open Eye Detection

David Flores Diaz A01209414

ITESM Campus Querétaro

1. ABSTRACT

The use of a Deep learning though a Convolutional Neural Network (CNN) to detect in a photo if the person has the eyes opened or closed, trained with 2500 samples in Google Collab in the Keras framework.

2. INTRODUCTION

The eyes are one of the factors taken into account to decide whether someone is paying attention, or is falling asleep or is feeling drowsy, to determine this things is of importance when there are people either driving, operating machinery or other cases were some need to be paying attention to what they are doing.

A Convolutional Neural Network is a class from the Deep Learning area, it involves running an image though a series of filters, to extract key features from said image, and with each pass of new filters or the convolution process the complexity of the features grow, they become more abstract to the point were we can detect objects and faces, this behavior is what we are looking for in this project.

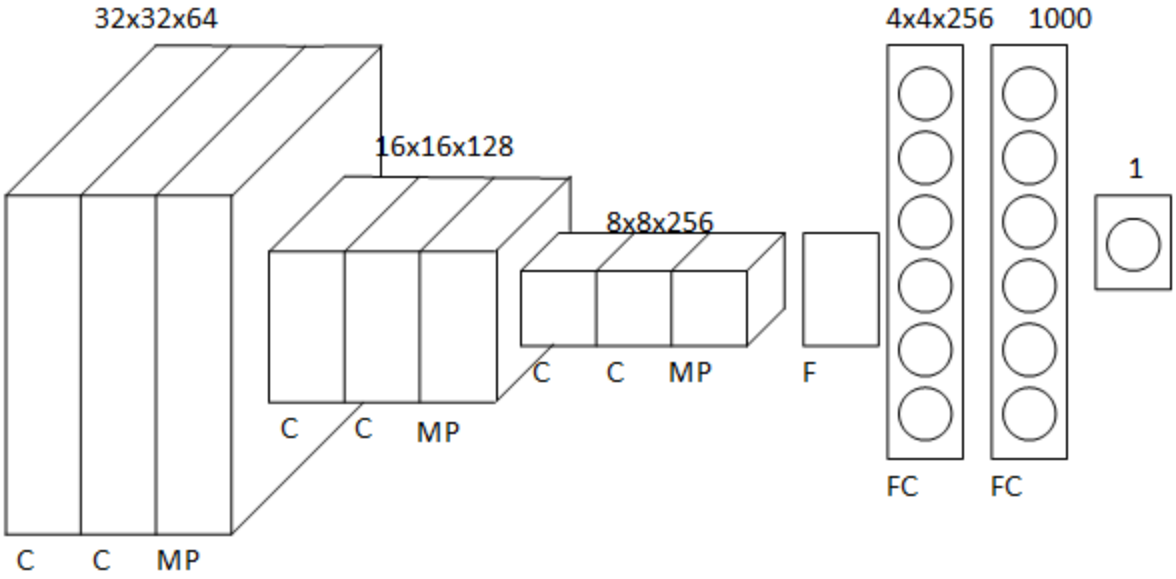
We want to be able to detect if a person has its eyes closed or not, we need to first detect the face, then extract the features like eyes, mouth and others, we will just focus on the eyes, and determine if each eye is open or closed, if both eyes are closed then the person has closed eyes, but just one open eye classifies as open eyes.

3. PROPOSED MODEL

We want to detect the state of the eyes, we first need to get the faces of the subjects and then localize the eyes to pass them through a network.

We can divide this process into 3 steps, the face detection, the eyes detection, and the classification process, the first step can be accomplished by using Multi-Tasked Cascaded Convolutional Networks [1],we can utilize David Sandberg´s implementation in python to predict a box for the faces of the subjects, then we can crop the eyes to feed the into our model.

As the eyes are feed to our model we utilize the next model



Where C are convolution layers, MP are Max Pooling, F is a Flatten layer, and FC are fully connected or Dense layers.

```
Model: "model_50"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_50 (InputLayer)        (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_199 (Conv2D)          (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_200 (Conv2D)          (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_100 (MaxPoolin (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_201 (Conv2D)          (None, 16, 16, 128)       73856
_________________________________________________________________
conv2d_202 (Conv2D)          (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_101 (MaxPoolin (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_203 (Conv2D)          (None, 8, 8, 256)         295168
_________________________________________________________________
conv2d_204 (Conv2D)          (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_102 (MaxPoolin (None, 4, 4, 256)         0
_________________________________________________________________
flatten_34 (Flatten)         (None, 4096)              0
_________________________________________________________________
dense_116 (Dense)            (None, 4096)              16781312
_________________________________________________________________
dense_117 (Dense)            (None, 256)               1048832
_________________________________________________________________
dense_118 (Dense)            (None, 1)                 257
=================================================================
Total params: 18,975,809
Trainable params: 18,975,809
```

```
Non-trainable params: 0
```

After each convolution we apply a ReLu activation function, since it has proven to be one of the best functions when it comes to CNN implementations because they accentuate the most remarkable features of the layers. we apply Max Pool layers to be able to make assumptions of the most significant values from previous layers, and at the end the fully connected layers to handle classification, this ones as well had a ReLu activation function, except the final FC layer of size one, were we applied a sigmoid activation since we want a binary output.

4. DATASET

The dataset is composed of 2423 samples divided in the 2 classes

- 1192 Faces with Closed Eyes
- 1231 Faces with Open Eyes

All images are RGB and 100x100 pixels since they have been previously cropped, although there is present the original photos, the dataset only contains the Closed Faces Originals

The images were read once and the saved into binary files, so they are easier to load in next training sessions

The images were run though the MTCNN to crop the eyes and be saves to another binary file, these cropped images were eye centered.

The images were then divided into 3 subsets, training, validation and testing being 70%, 20% and 10% respectively from the original database, then they were fed to data generators to get more samples.

5. TRAINING

We compiled the model using as a loss function the binary cross entropy function, because we are dealing with a problem that only has to classes, this is a variation of the cross entropy function, here t1 is the actual class of the sample and s1 is the predicted sample

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1)$$

We used an optimizer called Adam (Adaptive moment estimation), the optimizer is the result of the acquisition of the advantages of two other optimizers, AdaGrad, that maintains a per-parameter learning rate and RMSProp that also maintains a per-parameter learning rate that adapts depending on how quickly the weights are changing.

Adam implements both techniques, it updates parameters depending how quickly are changing but it also takes in account the previous values or moment to have a better prediction for the gradient descent.

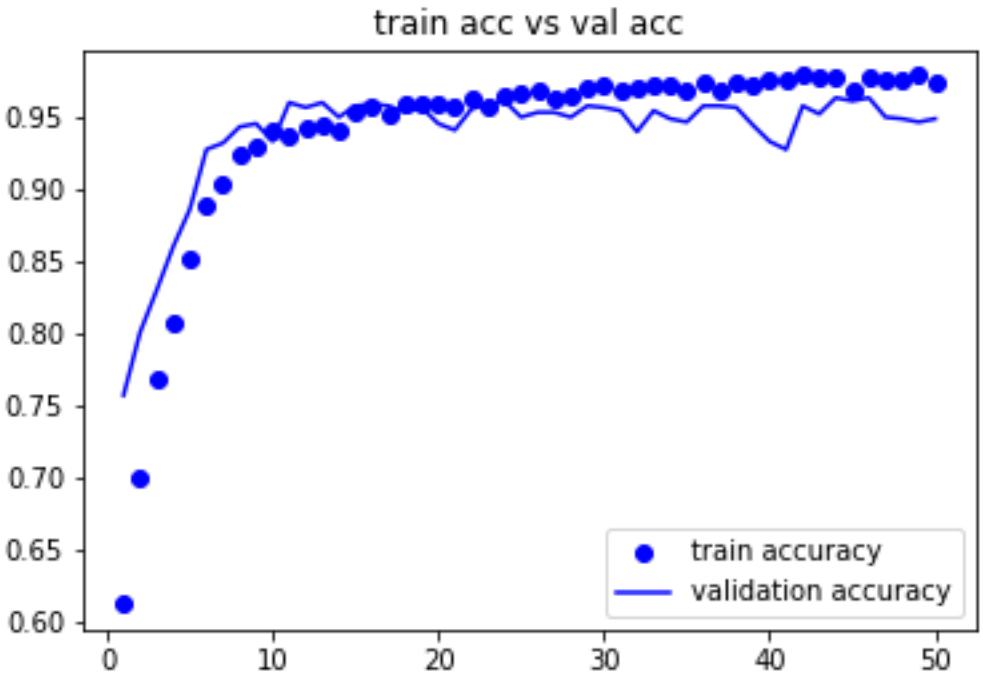
Adam is an optimizer that has been recently used instead of SGD since this technique has a single learning rate that doesn't change through training.

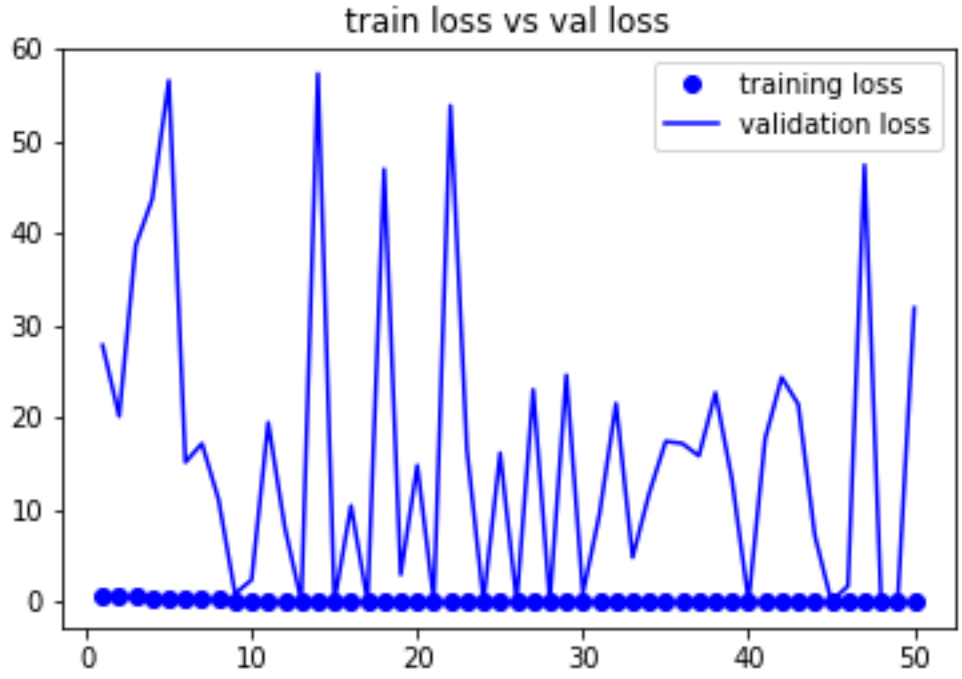
We used as a metric the accuracy of the model, the number of correct guesses over all guesses to get the percentage of correct guesses.

We trained the model for 50 epochs and noticed that the validation loss behaved in a very fluctuating way, perhaps because of the batch size, so we implemented a Checkpoint class to

monitor the val_loss parameter while training, and to save the weights of the model after each epoch, so in the end we just loaded the weights that presented the best validation accuracy with the lowest validation loss.

6. **RESULT**





We see bot accuracies growing at almost the same rate which means that there is no overfitting , the epoch were we obtained the best results was at epoch 49 with a Train accuracy of 97 and a

testing accuracy of 95 with 0 validation loss, which made it a pretty good model for eye detection. The fluctuations of the validation loss were product of the change in the weights to find the correct way of descending.

7. REFERENCES

1. Raul Gomez, (May,23,2018), “Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names” , Available at: https://gombru.github.io/2018/05/23/cross_entropy_loss/
2. F.Song, X.Tan, X.Liu and S.Chen, Eyes Closeness Detection from Still Images with Multi-scale Histograms of Principal Oriented Gradients, Pattern Recognition, 2014. Available at: http://parnec.nuaa.edu.cn/xtan/data/ClosedEyeDatabases.html
3. Jason Brownlee, “How to Perform Face Detection with Deep Learning”,(June,3,2019), Available at: https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/
4. Jbrownlee,” MTCNN face detection implementation for TensorFlow, as a PIP package. “,(Sep,10,218) Available at: https://github.com/jbrownlee/mtcnn
5. Kim, Ki & Hong, Hyung & Nam, Gi & Park, Kang. (2017). A Study of Deep CNN-Based Classification of Open and Closed Eyes Using a Visible Light Camera Sensor. Sensors (Basel, Switzerland). 17. 10.3390/s17071534.
6. Jason Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning”, (July 3,2017), Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
7. Chengwei,” Quick Notes on How to choose Optimizer In Keras”,(February 2018), Avaiable at: https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/
8. Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10):1499–1503.
9. Noor D. Al-shakarchy, Israa Hadi Ali,(April 15,2019),” Open and Closed Eyes Classification in Different Lighting Conditions Using New Convolution Neural Networks Architecture”, Available at: http://www.jatit.org/volumes/Vol97No7/12Vol97No7.pdf