Internship Report - First Project

Sunglasses Detection using CNNs

Ismail OUSSAID
with the supervision of: Enrico Marchetto

Summary:

Facing dataset

The idea of the project is to find Deep Learning solutions for Sunglasses Detection provided a load of images web scraped from Google and manually labeled.
First thing is to have a global analysis of the provided data. I have used the face detection json file I was given to capture all the faces from each picture. For example, if in a picture, the face detector had found 5 faces, we can extract 5 new images for our dataset. To do so, I take the coordinates of the bbox for each face in an image to crop it over the present faces and then, I get one cropped image.

Fig.1 Web scraped image (left) and extracted face (right)

After the face extraction, I scrutinize the distributions of widths and heights of the cropped images are as below:

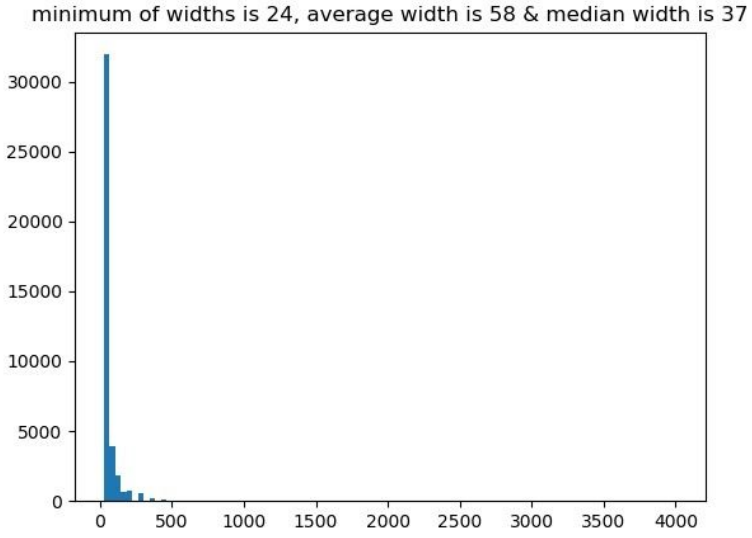

Fig.2 Distribution of widths

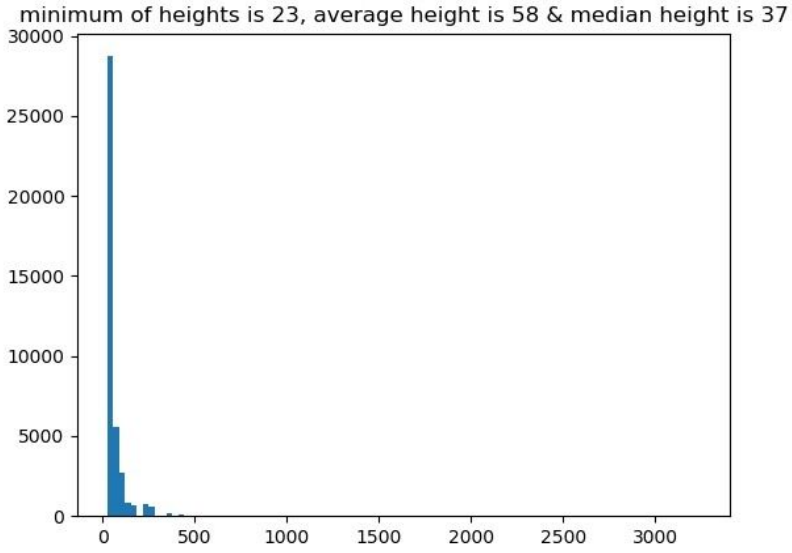

Fig.3 Distribution of heights

So, we have to do some image processing to have our dataset corresponding to the usual surveillance camera (i.e grayscale 24x24) images. Other preprocessings are conceivable but are discussed in the *Appendix*. So, here we see that almost all images are larger than this so the resizing will not make us lose a lot of information. At the end of the processing and scaling, we get a +43000 images dataset.

Then, comes the first big issue. I check the distribution of the dataset and I get this:

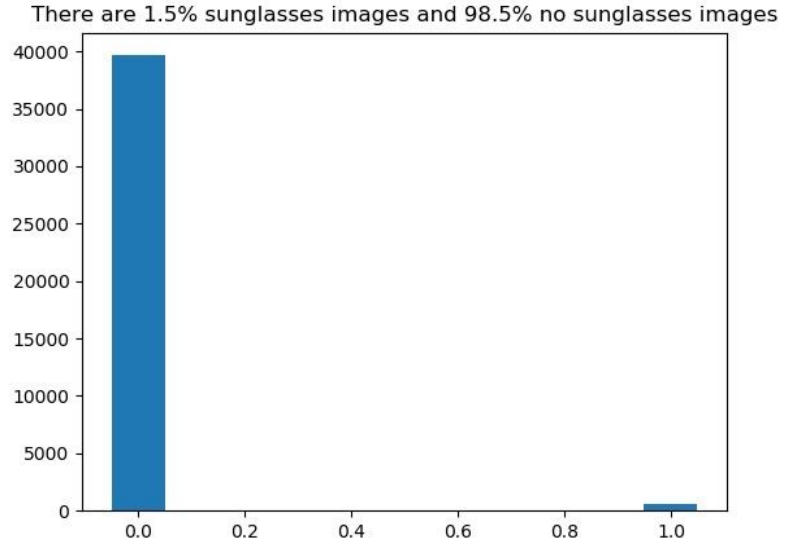

Fig.4 Distributions of labels

There is a very few part of our dataset that corresponds to sunglasses images and this is even worse considering the amount of data (+600 images of people wearing sunglasses) since we want to use deep learning which needs a lot of data to learn. First, to know the main criterias for our model. Paolo wants less FN (we want to ensure that we detect every person with sunglasses) and then, Recall or F1-Score can be interesting to use. Since F1-Score is a harmonic tradeoff between Precision and Recall (diminishes FN and FP) and very adapted for imbalanced datasets, we chose it.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

Fig.5 F1-Score formula

Just to have a basic approach to F1-Score. If F1=0.9, that means: $FP + FN = \frac{2}{9} \cdot TP$ and if TP are few because there are few positive true labels, this will ensure to have very few False predicted labels. Considering that the positive label here is the minority class (sunglasses), F1 Score will help assess a model as it takes into account only TP and not TP+TN, which would make the evaluation biased since a very big part of our data is made of the negative class. Then, it will be our first performance evaluator. Also, we have to know the amount of floating point operations for our model and we will always ensure to have a high F1-Score (+95%) and low number of flop (thousands).

Also, to make the dataset less imbalanced, I made two steps:

- Upsampling by adding more of the minority label images by Data Augmentation that consists in horizontally flipping sunglasses-labeled images
- Downsampling by choosing a maximum of 3k negative labeled images

Then, we have almost 70/30 distributions of 4.2k images, which is still imbalanced. But the majority estimator (estimator that predicts the majority class to every image) has a F1-Score of 70.9%, so we just need to find a DL solution that gives way more than this (the criteria of +95% seems still reasonable).

Deep Learning: Implementation & Results

In this section, we will focus on finding functional convolutional neural networks to detect sunglasses on our dataset. In the *Appendix*, you can find a Transfer Learning approach for this problem. We don't have much time so we have to fix a certain amount of Dense and Conv2D layers. Considering my personal experience, I thought that p convolution layers coupled with max pooling and k dense layers after these

layers and a final dense layer with 2 nodes (2 labels) would be satisfying solutions for our problem with $k \in \{1;2\}$ and $p \in \{1;2\}$ with $p+k \leq 3$.

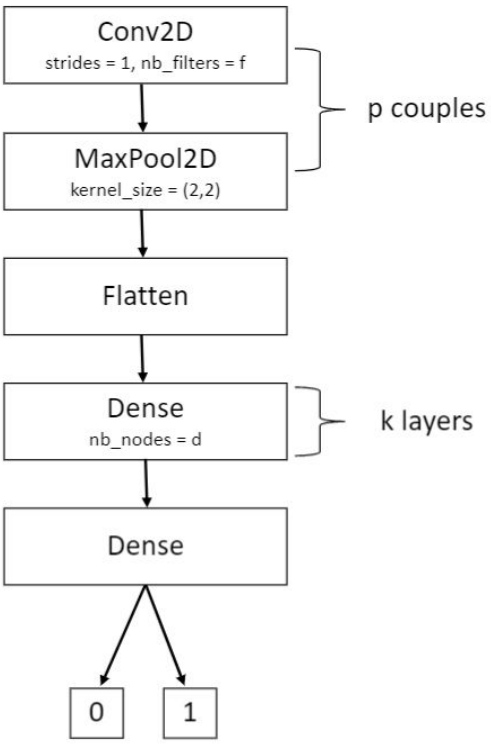

Fig.6 Neural architecture of the sunglasses detector with hyperparameters (f,d)

To find an interesting architecture, I have chosen the following process:

- Split the dataset into 85 % training set and 15 % testing set
- Train the architecture on Tensorflow & Keras with epochs = 8 for different values of the hyperparameters: f, d, batch_size (Grid Search)
- Test the each model (architecture with fixed hyperparameters) on testing set
- Compute F1-Score for each model

number of Conv2D	number of Dense	f	d	batch size	F1-score
2	1	128	32	32	0,9782869274
2	1	256	4	8	0,9766171679
2	1	128	16	16	0,9665900876
2	1	128	8	16	0,9766156183

Fig.7 Extract of excel tab output of this process for p=2, k=1

I compare the best models of the 3 architectures (p,k)-parameterized.

- For (1,1), $F1-Score_{max} = 97.4\%$,
- For (1,2), $F1-Score_{max} = 97.2\%$,
- For (2,1), $F1-Score_{max} = 98.9\%$.

Now, we will focus on the following architecture:

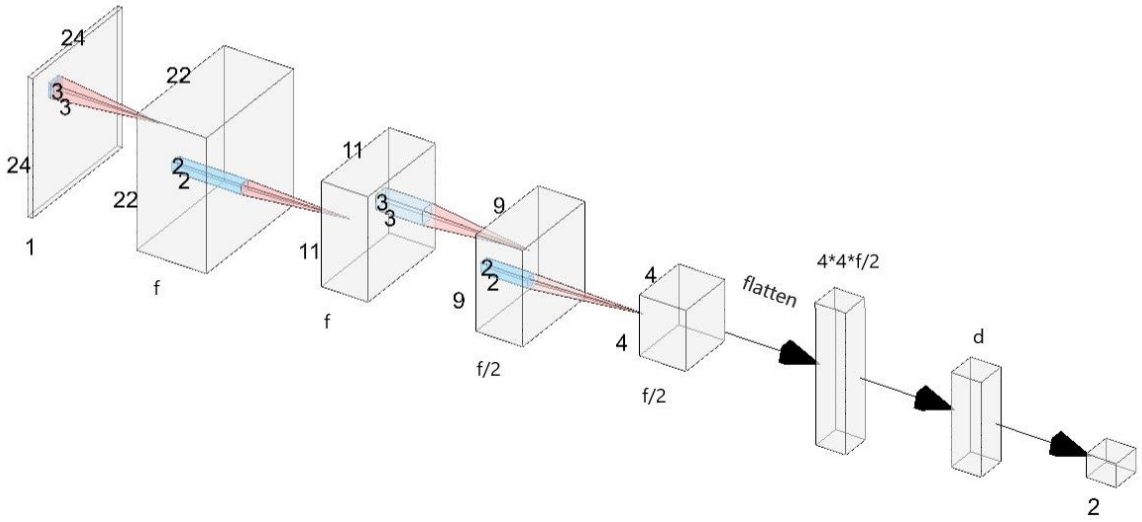

Fig.8 Typical solution architecture with (f,d) parameters

Now, to be rigorous, I have to do a hyperparameters tuning so I do a 5-Fold cross validation for each model for different triplets of f, d & batch_size.
When I did so, the best model was (f,d_batch_size)=(128, 16, 4) with a CV-F1 of 98.65% with a standard deviation of: 0.81% (very reliable). I also studied the effect of some anti-overfitting techniques for 5 days (*see Appendix*)

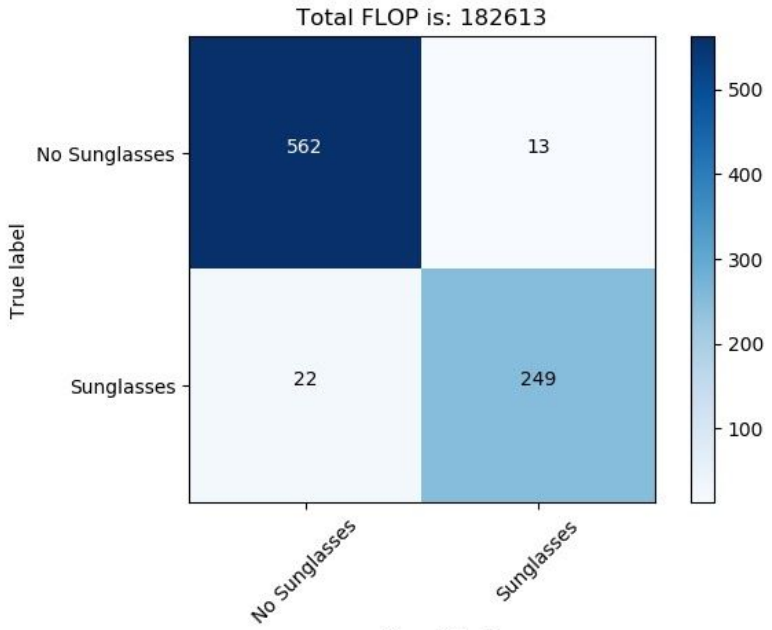

Fig.9 FLOP and Confusion matrix for the best model so far

Considering the confusion matrix, 91.8% of sunglasses are recognized and 97.7% of no-sunglasses are recognized. Now, we have an impressive model but the FLOP are relatively high. Indeed, we rather have a thousand-flop model with a bit lower score.

So, let's check a "thinner" CNN who might reach such performance. So, I put into process a hyperparameters tuning on f, d parameters smaller than 16, with a fixed batch_size of 4 (in my work, I have noticed that a batch size of 4 is the best parameter in these training for every f,d). Finally, we could find an outstanding model.

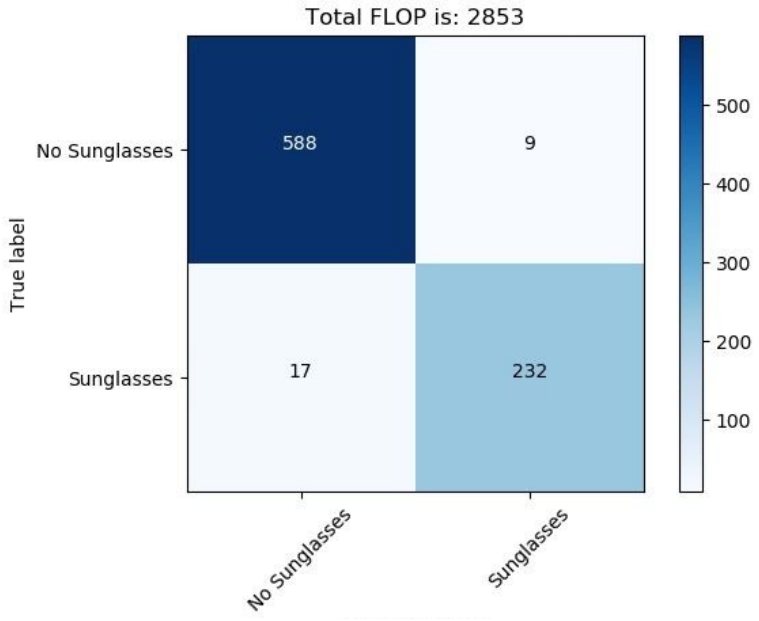

Fig.10 FLOP and Confusion matrix for (f,d,batch_size)=8,16,4

Here, for this 3k-flop model, 93.1% of sunglasses are recognized and 98.5% of no-sunglasses are recognized. This seems to be an outstanding tradeoff.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 22, 22, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 9, 9, 4)           292
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 4)           0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 16)                1040
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 34
=================================================================
Total params: 1,446
Trainable params: 1,446
Non-trainable params: 0
```

Fig.11 Model summary

Furthermore, the model has only 1446 parameters, few layers and on the CV-score is 96.19% with a standard deviation of 3.54%. But it is clearly worth it to lose reliability and lose 2% of score to divide FLOP by 64.

Conclusion

Our most satisfying model is the following:

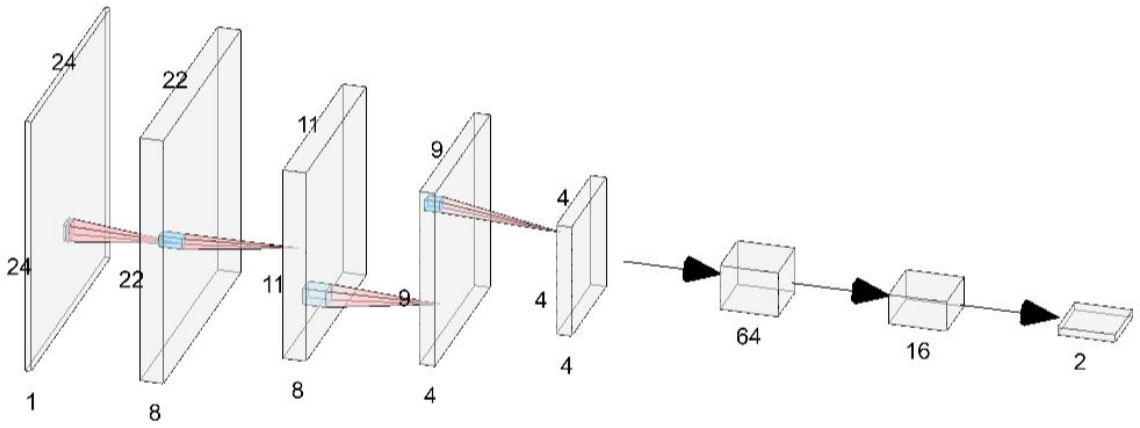

Fig.12 Sunglasses Detector Architecture

It needs to be trained with a batch size of 4 and no need for Batch Normalization or Regularization. Indeed, Early Stopping (https://en.wikipedia.org/wiki/Early_stopping) and ReduceLRonPlateau (https://keras.io/api/callbacks/reduce_lr_on_plateau/) are used for such results. Also, a Data Augmentation by adding Gaussian-noised images could have been a good idea.

The idea of the network is that after the first conv2D layers, 8 feature maps are built (https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2) and on these feature maps is applied max pooling which gives an abstract form of the image by catching sharp and smooth feature (max over each 4 pixels) (https://deepai.org/machine-learning-glossary-and-terms/max-pooling). Next thing is that we build from these new feature maps more “sharp & smooth feature” relevant, 4 feature maps from which we take smooth and sharp features with a max pooling. So, we have 4 feature maps with a 4*4 size and we flatten them into one vector of 64 features (that shall be relevant). Finally, the data passes by a fully connected network in which 16 nodes in a hidden layer captures 16 differentes combinaisons of our precedent 64 features. At the very end, we deduce the likelihood of a class with softmax based on the final 16 new features.

Appendix

i. Effect of anti-overfitting techniques

I study the effect of batch normalization, regularization and Data Augmentation on our models. I have computed average 5-Fold Cross Validation F1-Score (to see performance of the model) and I have computed average Standard Deviation (STD) of 5-Fold Cross Validation F1-Score (to see its consistency). To scrutinize this, I computed such figures for all the models, for 20, 10 and 5 most performing models. Here are the results for the (2,1) architecture without any anti-overfitting techniques.

```
 For all the models, the average F1 is: 93.68% and average STD is: 7.41%
 For the 20 best models, the average F1 is: 97.96% and average STD is: 1.43%
 For the 10 best models, the average F1 is: 98.19% and average STD is: 1.21%
 For the 5 best models, the average F1 is: 98.32% and average STD is: 0.97%
Best model in this case had F1: 98.65%
```

Batch Normalization:

```
 For all the models with Batch Normalization, the average F1 is: 93.03% and average STD is: 9.57%
 For the 20 best models with Batch Normalization, the average F1 is: 97.53% and average STD is: 2.36%
 For the 10 best models with Batch Normalization, the average F1 is: 97.84% and average STD is: 1.85%
 For the 5 best models with Batch Normalization, the average F1 is: 98.11% and average STD is: 1.67%
Best model in this case had F1: 98.23%
```

Regularization with default λ=0,01:

```
 For all the models with Regularization, the average F1 is: 94.36% and average STD is: 6.56%
 For the 20 best models with Regularization, the average F1 is: 97.31% and average STD is: 2.09%
 For the 10 best models with Regularization, the average F1 is: 97.44% and average STD is: 1.97%
 For the 5 best models with Regularization, the average F1 is: 97.55% and average STD is: 1.82%
Best model in this case had F1: 97.89%
```

Batch Normalization & Regularization with default λ=0,01:

```
 For all the models with Batch Normalization, Regularization, the average F1 is: 94.41% and average STD is: 7.08%
 For the 20 best models with Batch Normalization, Regularization, the average F1 is: 97.53% and average STD is: 2.1%
 For the 10 best models with Batch Normalization, Regularization, the average F1 is: 97.87% and average STD is: 1.6%
 For the 5 best models with Batch Normalization, Regularization, the average F1 is: 98.02% and average STD is: 1.36%
Best model in this case had F1: 98.13%
```

Also, I have tried to make a semi-Data Augmentation by adding blurred images to sunglasses-labeled images. The idea is to create images that are more similar to what a surveillance camera can record (blurry or noisy). If I had more time, I would scrutinize blurry AND/OR noisy images in sunglasses-labeled images but also, all the dataset.

Blurred Data Augmentation:

```
 For all the models, the average F1 is: 93.68% and average STD is: 7.41%
 For the 20 best models, the average F1 is: 97.96% and average STD is: 1.43%
 For the 10 best models, the average F1 is: 98.19% and average STD is: 1.21%
 For the 5 best models, the average F1 is: 98.32% and average STD is: 0.97%
Best model in this case had F1: 98.65%
```

```
 For all the models with Blurred images, the average F1 is: 89.83% and average STD is: 11.27%
 For the 20 best models with Blurred images, the average F1 is: 97.6% and average STD is: 1.54%
 For the 10 best models with Blurred images, the average F1 is: 97.96% and average STD is: 1.27%
 For the 5 best models with Blurred images, the average F1 is: 98.24% and average STD is: 1.21%
Best model in this case had F1: 98.47%
```

I need to comment this to specify that I think that even if the F1-Score is smaller (-0.18%) and less consistent (-0.24%) for the Blurred version, it is probably a better one because it has faced harder images and trained on more images.

Blurred Data Augmentation:

```
 For all the models with Blurred images, Batch Normalization, the average F1 is: 92.12% and average STD is: 9.35%
 For the 20 best models with Blurred images, Batch Normalization, the average F1 is: 97.3% and average STD is: 1.86%
 For the 10 best models with Blurred images, Batch Normalization, the average F1 is: 97.68% and average STD is: 1.4%
 For the 5 best models with Blurred images, Batch Normalization, the average F1 is: 97.97% and average STD is: 1.15%
Best model in this case had F1: 98.26%
```

Blurred Regularization:

```
 For all the models with Blurred images, Regularization, the average F1 is: 91.32% and average STD is: 9.0%
 For the 20 best models with Blurred images, Regularization, the average F1 is: 96.92% and average STD is: 2.17%
 For the 10 best models with Blurred images, Regularization, the average F1 is: 97.2% and average STD is: 1.79%
 For the 5 best models with Blurred images, Regularization, the average F1 is: 97.33% and average STD is: 1.63%
Best model in this case had F1: 97.52%
```

Blurred Batch Normalization & Regularization:

```
 For all the models with Blurred images, Batch Normalization, Regularization, the average F1 is: 92.94% and average STD is: 8.38%
 For the 20 best models with Blurred images, Batch Normalization, Regularization, the average F1 is: 97.28% and average STD is: 1.76%
 For the 10 best models with Blurred images, Batch Normalization, Regularization, the average F1 is: 97.59% and average STD is: 1.42%
 For the 5 best models with Blurred images, Batch Normalization, Regularization, the average F1 is: 97.73% and average STD is: 1.31%
Best model in this case had F1: 97.91%
```

Considering all these results, you can understand that I have considered these techniques not very useful for our case.

ii. Filter preprocessing a helpful idea ?

Here you can find my implementation:
https://colab.research.google.com/drive/1kvxBKT-RkyRgQhB9Lz7jT66Qnssz3nV_?usp=sharing

An interesting idea could be to apply filters to our images in preprocessing to make it easier for our models to identify glasses. I couldn't dedicate much time on this part but you can check different filters' effect on our images such as *Adaptive Threshold* or *Gaussian Binary Otsu Threshold*. The issues faced are black people when the light is low, who aren't easily identifiable with these filters.

iii. Transfer Learning as a solution ?

Here you can find my implementation:
https://colab.research.google.com/drive/19EKUKvWAPgWSCdTZCtbTFiLVVeTCeOob

First thing, we have decided to work on MobileNet V2 and it is clearly not an option for our research work as it has half a billion FLOP. Also, it accepts only colored (3 channels) images AND minimum 32*32 images.

I have made two different trainings:

- Train last layer with 2 nodes
- Add a layers with 16 nodes and a final layers with 2 nodes

In both cases, results are very weak: 71% so it is clearly not useful in our case.