

# cs577 Project Report

Ayman FAHSI – A20440820

Mouhammad BAZZI – A20522180

Department of Computer Science

Illinois Institute of Technology

---

## REAL-TIME EMOTION AND GENDER CLASSIFICATION

Ayman was responsible for the training of the emotion classifier while Mouhammad was responsible for the training of the gender classifier. Both of us worked on the main pipeline (the core of our project). And both of us worked on the report, presentation PowerPoint, etc.

### I. PROBLEM STATEMENT

We attempt to generate two models, one to classify gender and one to classify emotion. The models should be sufficiently accurate and computationally efficient, such that they can be applied to a camera feed for real-time classification. As mentioned, there are two difficulties with this task:

- a. **Model Computation Costs**
- b. **Classification Accuracy** (The average success rate of a human for classifying emotion is about 65%)

The main goal of this project is to build a **real-time pipeline** that can take camera feed input and classify both the gender and the emotion of the faces detected.

### II. PROPOSED SOLUTION

We divide this section into three to cover Data Sets, Model Architectures, and the Real-time Pipeline.

#### A. Data Sets (IMDB and FER-2013)

The **IMDB** dataset will be used for the **gender classification**. It is very large (460,723 images) and each image contains a lot of information that will not necessarily be useful for our problem. So, we will download only the “faces” of this IMDB dataset. There is no unique size for all the images (so we will have to reshape the images). The images are in color. And each image is labeled as 0 or 1 or NAN, where 0 means “**Woman**”, 1 means “**Man**” and NAN means that we don’t know.

After downloading locally (or uploading the data on a cloud service) we had to some steps in order to load and pre-process all the images:

1. We first **loaded** the metadata file using the SciPy library in order to get all the labels (0, 1, or NAN) associated with an image name.
2. After that we loop through all the images (by getting their path using the image name) and we **store** an array of tuples of (img, label). We also **filter out** corrupt images and images with a NAN label.
3. It must be noted that we load the images in our array using the OpenCV library and while they are loading we **reshape** each image as 64x64. (64 is a number chosen by the user)
4. Then we **split the data** into a training, validation, and testing set using ratios selected by the user. To do this split we use the *train\_test\_split* function from the Scikit-learn library.
5. And finally, we **normalize** the data (dividing by 255) and we **one-hot encode** all the labels (in order to be able to have a SoftMax activation as the last layer and not sigmoid).

The **FER-2013** dataset will be used for the **emotion classification**. In this dataset, there are 28,709 gray-scale labeled images of 48x48. The labels are integers from 0 to 6. Where 0 means “**Angry**”, 1 means “**Disgust**”, 2 means “**Fear**”, 3 means “**Happy**”, 4 means “**Sad**”, 5 means “**Surprise**” and 6 means “**Neutral**”.

After downloading locally (or uploading on a cloud service) the CSV file, we have to do some steps in order to load and pre-process the images:

1. We **load** the data using the Pandas library and we separate the labels from the image.
2. Then we have to **convert** the sequence of pixels (that represents the image) into an array of float (initially the sequence of pixels is loaded as a string sequence).
3. Then we **reshape** the image into an array of (48,48,1).
4. Then as seen for the IMDB dataset, we **split the data** into training, validation, and testing sets.
5. And we finally **normalize** the images and **one-hot encode** their labels.

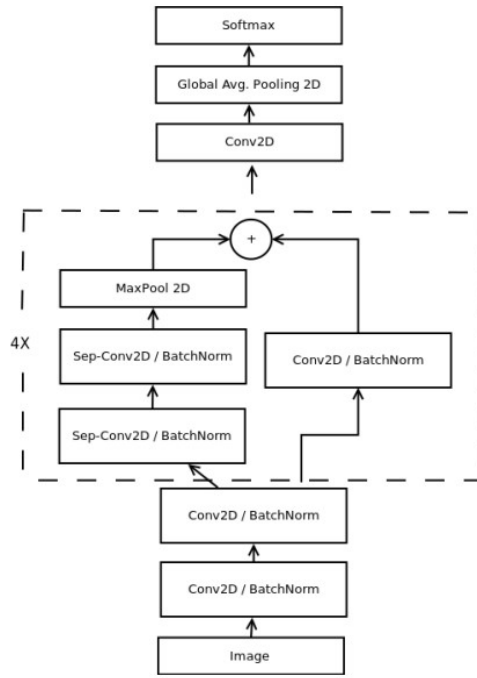
## B. CNN model (Basic and Residual models)

We designed two models, one that will be called the “Basic model” and another one that will be called the “Residual model”. We applied both models to both datasets, resulting in 4 distinct models.

### The Residual model:

We use the design depicted below to build the model and implement the classifier using the functional Model API. The input layer is comprised of standard convolutions, which feed into four residual blocks, each performing standard convolution and batch normalization, alongside depth-wise separable convolution and max pooling. We classify the output layer with standard SoftMax operations.

- We replaced the final fully connected layer with **Global Average Pooling** to reduce the number of parameters and therefore save computation time.
- We use **depth-wise separable convolutions** because they reduce the number of parameters of the model by separating the processes of feature extraction and combination. [6]



**Fig. 1: Description of the Residual model [2]**

### The Basic model:

We design the basic model to resemble the residual model. The model is comprised of four layers, each performing convolution twice (similar to the first input layer of the residual model), followed by Batch Normalization, Average Pooling, and Dropout.

- We manually decrease the kernel size as the number of units increases and hold dropout to 50%.
- We use AveragePooling as opposed to MaxPooling due to the small image sizes.

This model will be used as a reference to evaluate how much the simplification of the Residual model will affect the accuracy.

### Hyperparameters:

As in the paper, we used ReLU activation and Adam optimizer for both models. We also referenced the original research paper for hyperparameters and tuned them according to our own results.

## C. Real-time Pipeline

In order to make our real-time pipeline we do the following steps:

- First, we **load** the two models that we trained before (one for gender and the other one for emotion classifications)
- Then, we use the OpenCV library to **capture video feed** via either webcam or video file.
- We use the *Haar Cascade Classifier* [4] model to **detect** any **faces** within the input.
- We generate a dynamically sized box over the detected face and run **predictions** using our two models.
- Finally, we **display** the two predictions near the sized box.

### III. IMPLEMENTATION DETAILS

We divide this section into four to cover data sets importations issues, model training and hyperparameter tuning issues, real-time pipeline issues, and instructions on how to use notebooks.

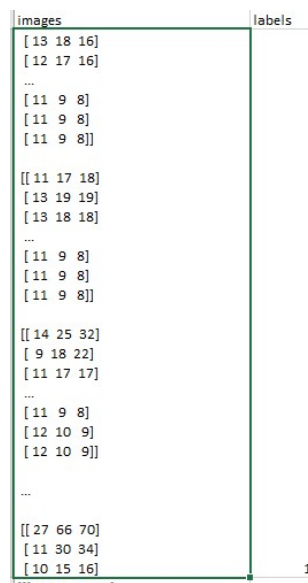
#### A. Data Sets importations issues

We make use of two data sets, FER-2013 to classify emotions, and IMDB to classify gender.

FER-2013 was already prepared as a CSV file and we ran into minimal difficulties during preprocessing

**IMDB**, however, was nearly **impossible to upload**, import, and preprocess locally, as the dataset is comprised of 460,723 images. Doing so via Google Colab proved to be ineffective. The specifics are as follows:

- Only **loading** the data **locally** costed about **1-2 hours**, and both machines experienced difficulties, (Ayman's **laptop overheated** to the point of shutting down, and Mouhammad's **VScode crashed**.)
- We attempted to load the raw data into Google Drive, and **bought extra storage** to do so, but the browser **could not** handle the upload of all the files at one time. Uploading in 100 subsections would have taken roughly **50 hours** to fully upload. (Approximately 30 minutes for each 1% of the dataset)
- We then attempted to load the compressed file to google drive, which took approximately 50 minutes (7GB file). We then **decompressed** the data directly **onto the drive**, the command for which took 2 hours to run. This was all done through Google Colab as **our machines could not handle the strain**.
- After the two-hour mark the decompressed images were stored within the session, but updating the actual drive **took much longer**. After constant checks to the actual drive, we concluded that it took 20 minutes for 1% of the dataset to be reflected in the actual drive. We estimate the total time to have been **33 hours** for all images to be reflected in the drive.
- We then decided to attempt the preprocessing on the session storage, then **download a CSV file (that contains all the data preprocessed)** in an effort to save time and memory. Unfortunately, the data was **not saved correctly** and included “...” symbols as opposed to actual values.



images	labels
[ 13 18 16]	
[ 12 17 16]	
...	
[ 11 9 8]	
[ 11 9 8]	
[ 11 9 8]	
[[ 11 17 18]	
[ 13 19 19]	
[ 13 18 18]	
...	
[ 11 9 8]	
[ 11 9 8]	
[ 11 9 8]	
[[ 14 25 32]	
[ 9 18 22]	
[ 11 17 17]	
...	
[ 11 9 8]	
[ 12 10 9]	
[ 12 10 9]	
...	
[[ 27 66 70]	
[ 11 30 34]	
[ 10 15 16]	1

Fig. 2: Screenshot of how was saved the data in the CSV file

- We concluded that this error was due to the Pandas library, so we opted to save the preprocessed data as a NumPy array and **flatten the data**. Doing so **ended** the session due to **exceeding the allotted RAM**.
- **Purchasing Colab Pro** and repeating ALL previous steps **did not resolve** any of the issues.
- Due to the previously listed issues, we opted to **import a subset** of the dataset via uncompressing within the drive. We waited for approximately **8 hours** and **imported 29%** of all images on our Google Drive (so 133,609 images).

After the images had been imported, we could start preprocessing within Drive. The program was reading the images at a rate of 3,000 images per 15 minutes. To save time, we began to **train the model on smaller subsets of the dataset** (starting at 12,000 images), then read in more images and trained again for the remainder of the dataset.

## B. Model training and hyperparameter tuning

Due to the size of the large datasets, (28,709 images for FER-2013, and many thousands for IMDB), we attempted to **reduce training time** by **purchasing Google Colab Pro**.

Regardless, the training still took a while to do. Due to the difficulties we came across when importing and reading the data, we did not optimize the hyperparameters well. The lack of data used from the IMDB was especially damaging. We attempted to combat this issue through **data augmentation** and a gradual **increase in dataset size**.

## C. Issues with the main program (related to the Jupiter file named *main\_program.ipynb*)

We did not face many issues when creating the main program and instead had several minor issues. For instance, the *Haar Cascade Classifier* (used to recognize faces), was not working as intended and not loading the models correctly. We resolved these kinds of issues through Stack Overflow.

In addition, we had difficulty with terminating the program. Upon pressing the escape key, the window should close, but instead, the window would freeze and python would crash. We are still unable to resolve this issue and are unable to halt the program without ending it through *task manager*.

## D. Instructions for the use of the Jupyter Notebooks

First, we must ensure that the libraries are all properly loaded (c.f. *requirements.txt* file) (Please note that we did not create a separate environment for this, so the file contains a few unrelated libraries.).

**Each notebook is divided into sections, at the beginning of which is a list of global variables, where the hyperparameters can be manually adjusted. No other section needs modification.**

The *emotion\_classifier.ipynb* and *gender\_classifier.ipynb* Jupyter notebooks should be used to load or build and save an emotion or a gender classifier. You could run only one part of the cells to use these notebooks.

The *main\_program.ipynb* Jupyter notebook is the main notebook that allows us to **analyze video inputs** (camera or file input). To use it you have to run all the cells.

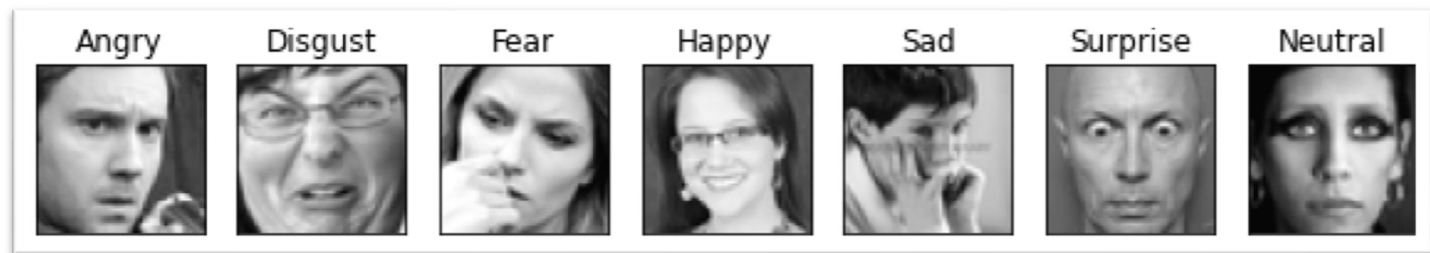
More information can be found in the *README.txt* or directly in the comments of the notebook.

# IV. RESULTS AND DISCUSSION

## A. Emotion Classifier

We first examined the class distribution and some images of the dataset to better understand our input, and better interpret our results.

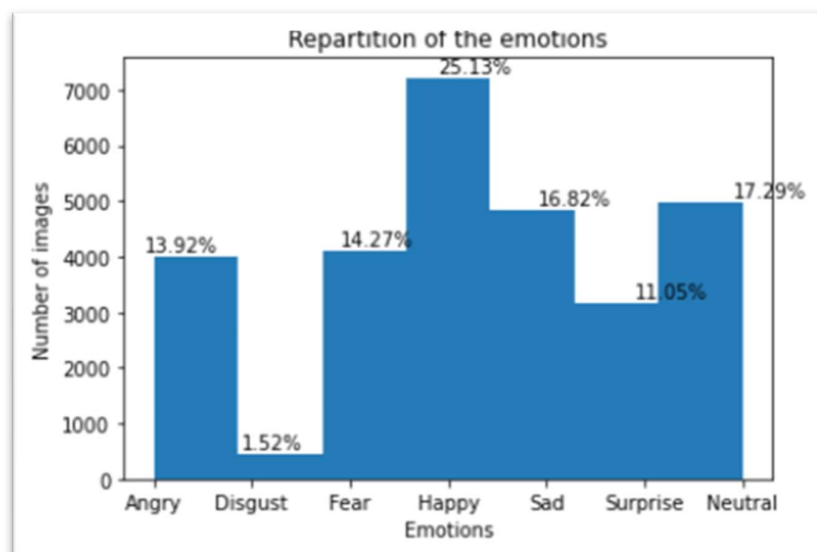
Display some random images for each gender:



We can notice that not all of them are obvious to determine as the “fear” or the “angry” one.

So, it is evident that **in some cases, the labels are difficult for humans to derive based on the image.**

We also output the repartition of the images that belong to each class:



From the distribution, we can see a very **distinct class imbalance**; we can see that the “Happy” class will probably be well classified, and the “Disgust” class will probably have unfavorable performance.

Then we started to train our data using the 2 models explained in the previous sections (the basic one and the residual one).

### Basic Model

We first started by testing our “Basic” model for this classifier.

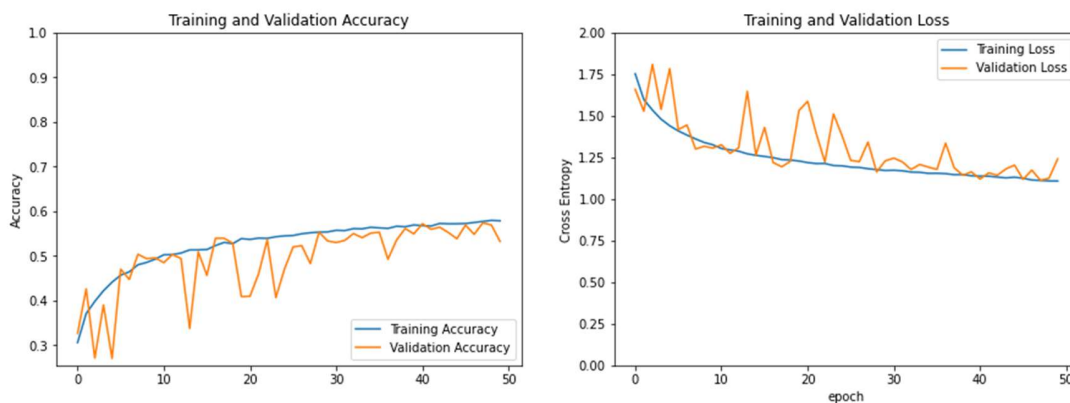
In the interest of time, we selected our hyperparameters to be the same as used in the source paper (like the Adam optimizer, the same number of convolution layers, the ReLU activation, etc.) except the learning rate, the number of epochs, and the hyperparameters of the data generator.

We suggest you to output the summary of the model to see in more detail how the model is structured (it will take too much place if we screenshot everything in the report). Below is the number of parameters in this model:

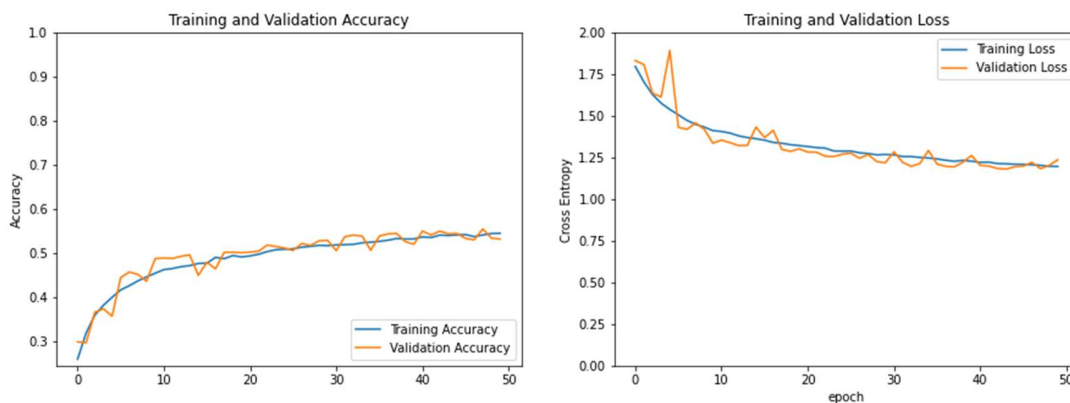
```
=====
Total params: 642,935
Trainable params: 641,463
Non-trainable params: 1,472
=====
```

## 1. Trained with non-augmented data

- a. We first trained with 50 epochs, and a learning rate of 0.001. below are our results; validation accuracy/loss seems very unstable.



- b. So, we decided to reduce the learning rate by dividing it by 10 and we got almost the same converging speed in addition to a stable validation accuracy/loss.



It seems that the model starts converging without huge modifications around 25 epochs. We merged the training and validation data set, and trained our model with the same hyperparameters as before but on 25 epochs. And then evaluated the model obtained on the test set.

**We got an accuracy of 56,18%. It is less than the human's success rate (65%) but the accuracy is acceptable enough for our purposes. It must be noted that the researchers reached an accuracy of 66%, and in comparison, our performance is substandard.**

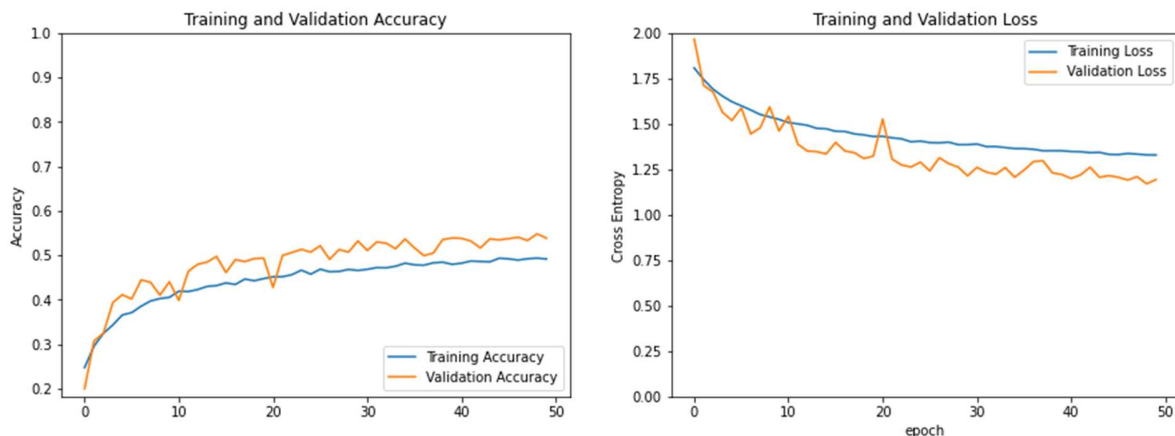
We then outputted a **confusion matrix** in order to understand where our model is struggling (or performing well):



We can notice that as expected, we have a bad performance with the “Fear” emotion and “Angry” emotions and we our “Happy” emotion is outperforming the rest of the classes.

## 2. Trained with augmented data

We repeated the same steps but this time on an augmented data set, after tuning the learning rate (we selected 0.0001) we got this accuracy and loss on the validation training data set:

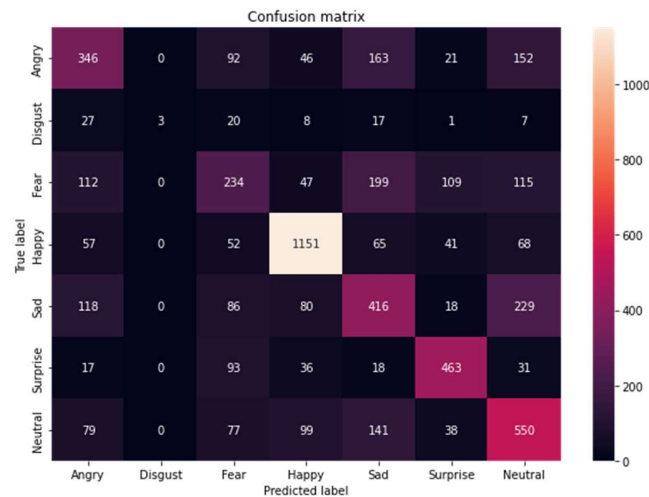


Because of the data augmentation, we expected an increase in instability. We also decided that after 22 epochs we converge.

This being the case, we retrained the model over 22 epochs using the combined training and validation datasets. The resulting **accuracy was 55.08%**.



Below is the confusion matrix associated with the prediction on the test set:



It seems that the data augmentation in this case with this “basic” classifier was useless.

## Residual Model

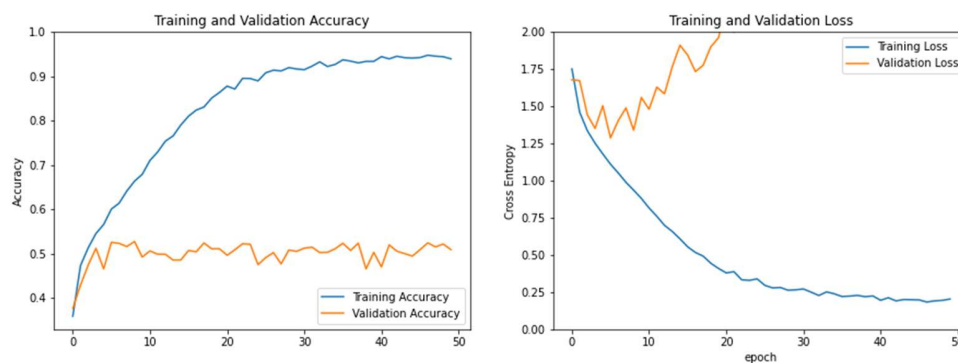
Let’s do the same process but on the residual model. As before we will use almost the same hyperparameters as used by the paper, we will tune only the learning rate, data augmentation hyperparameters, and the number of epochs for the learning. Displayed below are the number of parameters of this model:

```
=====
Total params: 65,603
Trainable params: 63,947
Non-trainable params: 1,656
=====
```

We have a reduction of nearly 90% in the number of parameters compared to the basic model.

### 1. Trained with non-augmented data

As usual, we start by training the model on a large number of epochs (50) and a learning rate of 0.001. Below are our results:



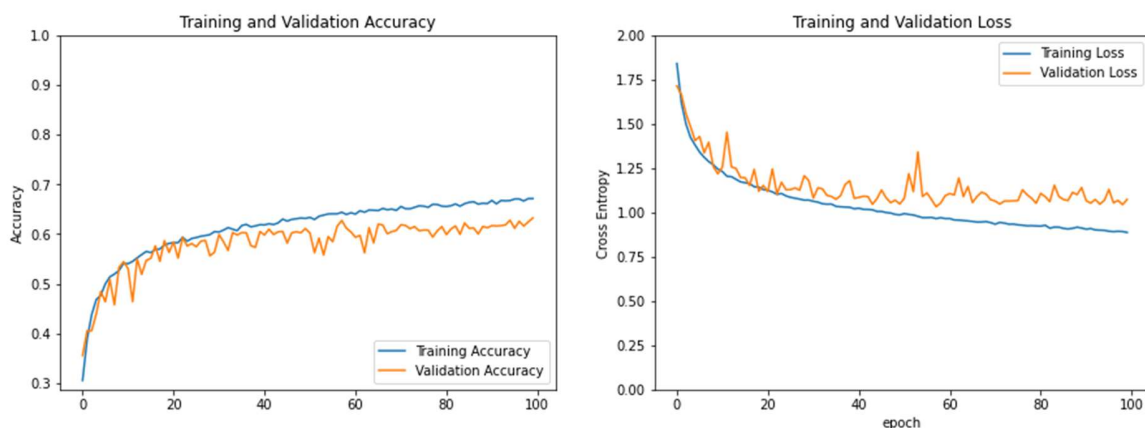
We overfit extremely quickly (after 4 epochs). This is expected because our model is “less complex” due to the reduction of the number of parameters. We then trained our model using the training and the validation data over 4 epochs and evaluated the trained model on the test data. Our test **accuracy was 51%** with the following confusion matrix:



Despite having a far better accuracy in comparison to a human or what they got on their paper, **we can notice that this model classifies Happy, Sad, Surprise, and Neutral well**. While the other classes have a majority correct, we can see that the classification is not as strong as we’d like it to be. Maybe it’s due to the **dataset quality** (few examples and a lot of these examples are “hard” because similar to other emotions of another class).

## 2. Trained with augmented data

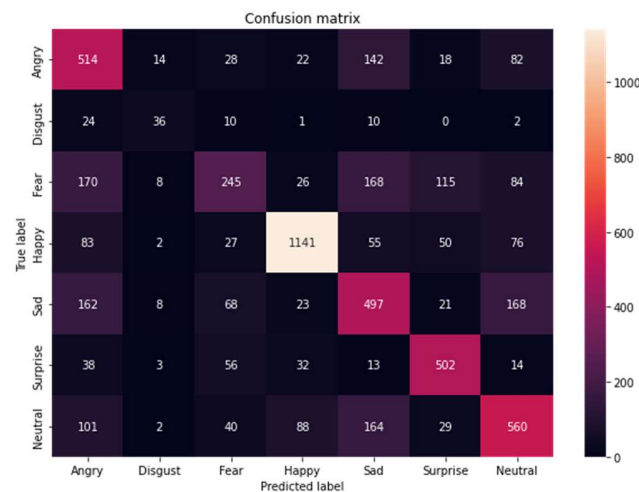
Because data augmentation tends to reduce overfitting drastically, we trained the model for 100 epochs to observe model behavior (learning rate still at 0.001) and we got this result:



We can confirm that the model does not overfit, but also seems to converge near the 55-60% range. We retrain the model on the combination of validation and training sets for 35 epochs, and we evaluate the model on the test set. **We can see a very good improvement with an accuracy of 60,86%.**

**This is by far the best-performing model, as it is closest to the accuracy reached by the paper (66%).**

We also displayed the confusion matrix to see in more detail where the improvement occurred:



We can confirm vast improvements to all classes but **Disgust**, which data-wise, is comprised of about 600 examples, compared to the 28,000 other examples (the sum of the examples of the 6 other classes). **This model classifies emotion with much more certainty than the previous models.**

### Conclusion (about Emotion Classifier):

Our **“Basic Model”** have **very poor results** in comparison with what the researchers achieved in their paper, and maybe this could be explained by the small differences that exist in our architecture (not exactly at 100% the same number of hidden units, etc.).

The **non-data augmented residual model is better** than the basic model because it reaches the same accuracy with nearly a **10x reduction in trainable parameters**. We can conclude that it's better than the basic. But not enough good in comparison to the expected accuracy (something around (66%).

But with the data augmentation, we got very satisfying results with only a **5% difference between our model and the model published in the paper**. Also, we think that in this case data augmentation was more impactful because we initially overfitted too quickly. (The basic model did not benefit from the data augmentation because it was converging without overfitting; the utilization of the augmented data on the basic model was kind of useless.)

## B. Gender Classifier

This set of classifications was not without its difficulties. As previously explained, both importing and reading in the data took much longer than expected, so we opted to train models in gradually increasing amounts of data.

So, in order to show that we used the same model for each sub-dataset (we used the residual model without image data generation, with the same hyperparameters, same learning rate (0.002), etc.) We summarized everything in this table:

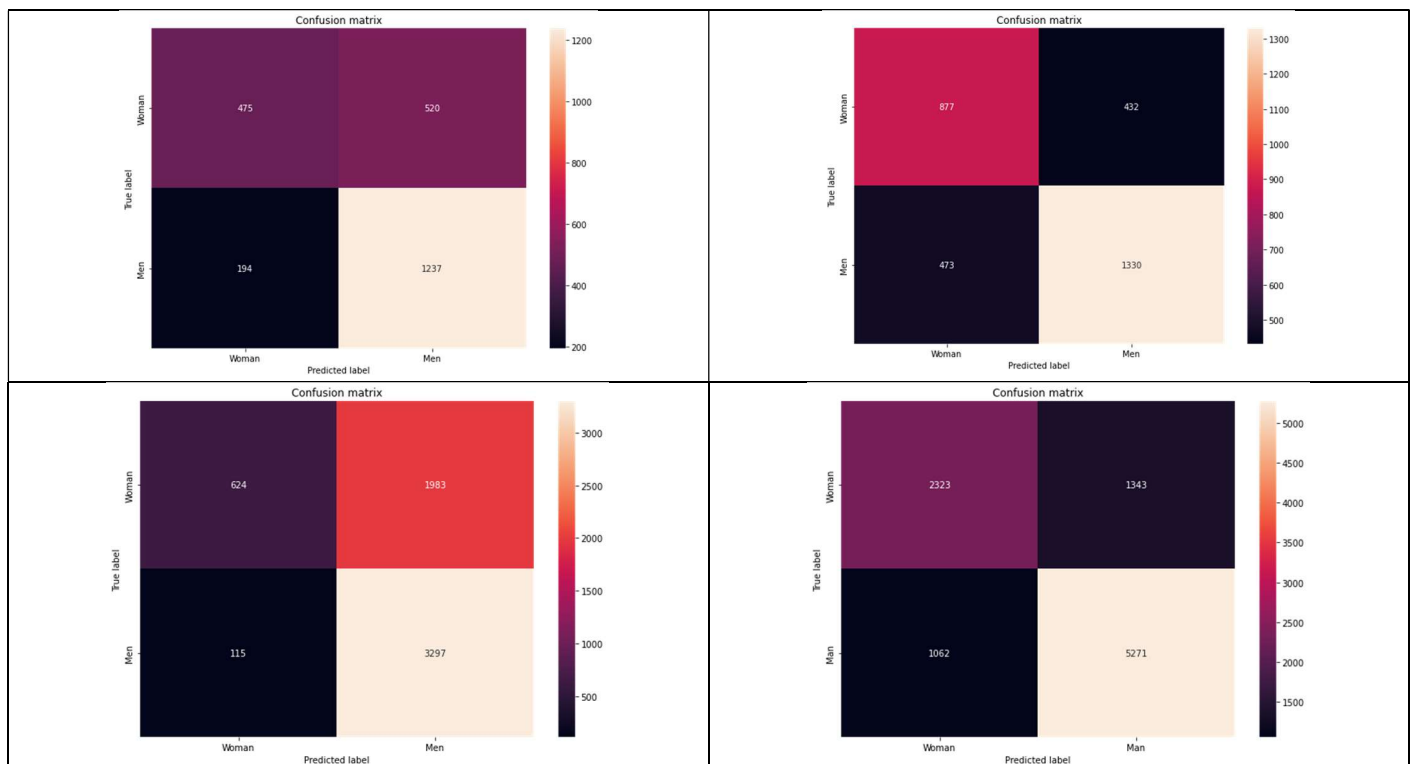
ID	Images Number	Percentage of Men labels	Ideal Epoch before overfit	Accuracy on the test set
1	12,716	59.71%	3	69.56%
2	16,169	58.37%	5	70.91%
3	30,977	57.76%	20	65.14%
4	49,992	64.06%	9	75.94%

**Our results show that more data leads to better predictions.**

The researchers trained their model on 460,370 images and produced an accuracy of 95%. **We were only able to operate on a maximum of 10% of that data and achieved roughly 76%.** Based on these facts, we believe that operating our data on the whole dataset would allow for better generalization, thus leading to better accuracy.

We can also see that the number of epochs to overfit increases with the data augmentation, which is expected as the data augmentation is a kind of regularization.

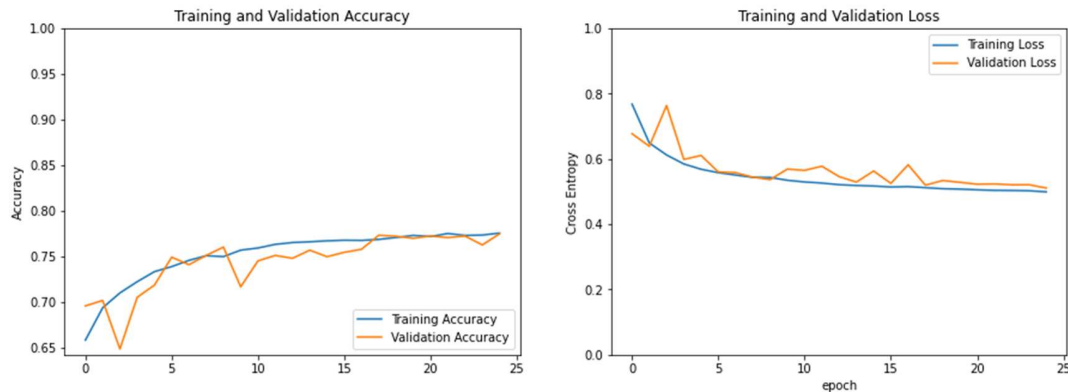
We wanted to examine the models' prediction in greater detail, so we produced a **confusion matrix** for each model. (The top left corresponds to ID 1, the top right to ID 2, the bottom left to ID 3, and the bottom right to ID 4.) Due to the class imbalances, we see that majority of the error falls under the "Woman" label.



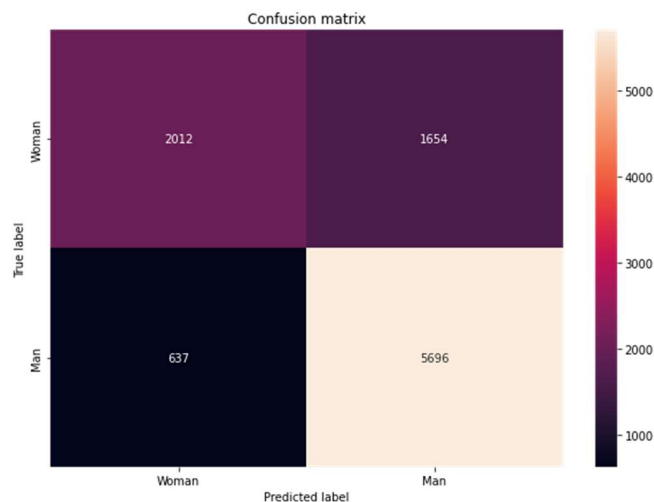
But we can notice that the last (the one with the more examples) is the one **who did the less error on “Woman”**. Probably due to the increase in women’s images that are pre-processed.

**We attempt to reduce this imbalance through the following methods:**

1. The first to perform **data augmentation** on the data with the learning rate still held at 0.002. Below are the results:

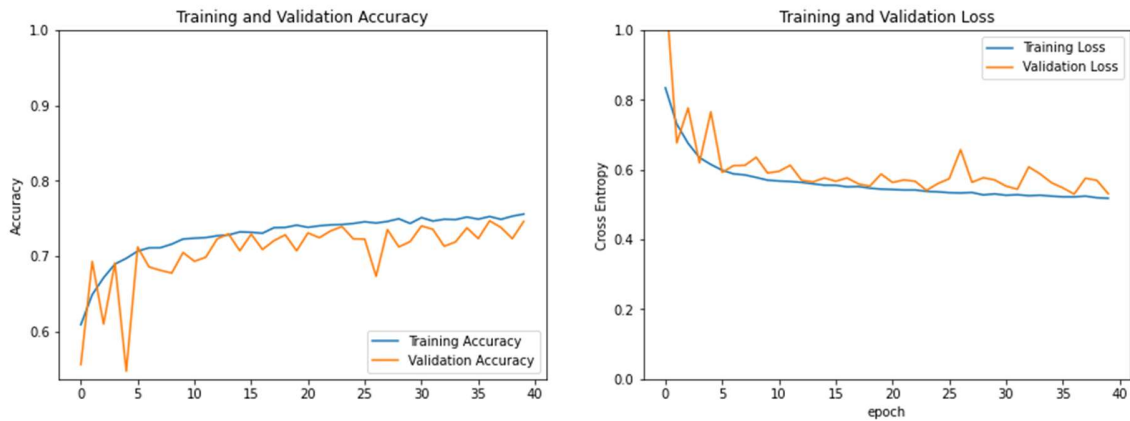


We converge around 20 epochs. As such, we will train the model on 20 epochs using the combination of the training and validation sets. **Evaluating the model on the test set got us an accuracy of 77.08%.**



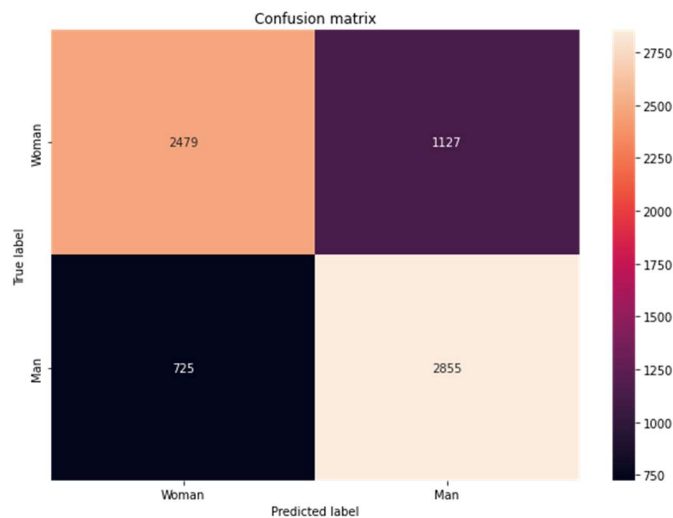
As shown by the figure above, we increase the global accuracy but also increased the error rate on the woman label, which we did not expect. **We can confirm, however, that data augmentation allows us to reach better accuracies.**

2. We attempted to reduce the class imbalance through **down sampling**, and decided to remove randomly some man images (originally 64.06% of the dataset) until we got an even distribution of man and woman labels. **We lost around 15,000 men images**, so we decided to compensate for this lack of images by doing Image Generations.



We converge after 20 epochs. So as usual we merged the training and validation sets in order to fit the model on 20 epochs and then evaluate the test set. **And we got an accuracy of 74.22%.**

And this is the confusion matrix outputted:



This model gave us a fairer error rate between the two labels, and still produced good accuracy for a model that passed from 50,000 images to 35,000 images.

**Important:** We also did training on the **basic model** with Image Data Generation and without on the data set of 50,000 images. **After training, we got an accuracy of 76.02 for the model without data generation and 75.7% for the model with data generation.** (And they did most of the errors were on the women label).

## Conclusion:

We tried to show empirically, **that the more data we have, the better our accuracy will be.**

We also had almost the same accuracy using the Basic model as using the Residual model (the residual model with data augmentation is even better). We interpret this positively because that means that with approximately **90% fewer parameters** (and therefore fewer computations), we achieved the same or better accuracy compared to a classic CNN network. This validates the purpose of our residual models.

We also successfully increase our accuracy on “Woman” labels by **down sampling** the man label. Moreover, this model utilizes even fewer images, leading us to believe that more data would result in a classifier capable of reaching the accuracy around the 95% goal presented in the paper.

## C. Pipeline Results

After determining the best-performing gender and emotion classification models, we applied them to our pipeline. Comparing the basic and residual model variations, our average residual model speed was approximately 20ms, while our average basic model speed was around 30ms. Our runtime results were similar to those of the researchers'; **we attained an acceleration of 1.5x speed.**

We were not, however, able to achieve similar average predictions. We believe this to be due to a **difference in hardware**. Despite that difference, **we still attain a similar acceleration.**

While testing the models we produced, **we realized that the highest accuracy was not the best metric gauge for real-world performance.** For instance, our balanced augmented gender model performed worse than the unbalanced variation. We believe this to be the result of losing 15,000 images. This also leads us to believe that having a class imbalance may be more favorable than a fully even class distribution. More images mean better feature recognition, and because this is a binary classification task, the model could learn the specific features and identify them solely based on them.

Despite these alterations, we still had trouble detecting the "Woman" label. We concluded that **the face detection box was too small** and found that altering the bounds of the detection box changed the models' predictions. For instance, the box would crop out hair in some cases and would classify the woman as a man. **Increasing the default size by 10% reduced the prediction errors very well.**

**In conclusion, while we were able to identify and rectify several key issues, we were not able to achieve the same results as found in the paper. As a reminder the final accuracy of our models and the researchers are presented below:**

**GENDER CLASSIFIER: 60% FOR US and 66% IN THE PAPER**

**EMOTION CLASSIFIER: 77% FOR US and 95% IN THE PAPER**

## V. REFERENCES

- [1] Octavio Arriaga, Paul G. Plöger, and Matias Valdenegro. Real-time Convolutional Neural Networks for Emotion and Gender Classification. Cornell University, <https://arxiv.org/pdf/1710.07557.pdf>, 2017.
- [2] Challenges in Representation Learning: Facial Expression Recognition Challenge. Kaggle, <https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/overview>, 2013.
- [3] Rasmus Rothe, Radu Timofte, Luc Van Gool. DEX: Deep EXpectation of apparent age from a single image. International Conference on Computer Vision (ICCV), <https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/>, 2015.
- [4] OpenCV, Cascade Classifier. [https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html).
- [5] Octavio Arriaga, Paul G. Plöger, and Matias Valdenegro, GitHub Face Classification Repository. GitHub, [https://github.com/oarriaga/face\\_classification/tree/b861d21b0e76ca5514cdeb5b56a689b7318584f4](https://github.com/oarriaga/face_classification/tree/b861d21b0e76ca5514cdeb5b56a689b7318584f4), 2017.
- [6] Andrew G. Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, <https://arxiv.org/pdf/1704.04861.pdf>, 2017.