Face and Gesture Analysis: Face Recognition
Final Project

Alejandro Fernández Alburquerque, 242349
Andreu Garcies Ramon, 240618

March 2024

NOTE BEFORE READING: we provide the code for this practice in the GitHub repository of the project: `https://github.com/Alejandro-FA/UPF-Face-and-Gesture-Analysis`

1 Introduction

For the final project of the Face and Gesture Analysis subject, we have developed a face recognition pipeline that performs well in identifying the ID of 80 famous people (see Appendix A for a complete list of their identities). This report aims to explain the project's thinking and development process and highlight the project's evolution between Lab 4 and the final challenge.

2 A Deep Learning based method

Since the first deep neural networks appeared in 2014, they have become state-of-the-art in solving many computer vision problems. This is also true for face recognition (FR), where networks such as DeepFace Taigman et al. (2014) and DeepID Sun et al. (2014) have proved to achieve a similar or better performance than humans. The power of deep learning (DL) in face recognition lies in its ability to autonomously learn intricate patterns and features directly from raw data, enabling remarkably accurate and robust identification capabilities. Unlike traditional methods that rely on handcrafted features and explicit rules, deep learning models can automatically extract hierarchical representations of facial features, discerning complex patterns that may not be apparent to human observers.

Given the excellent performance of DL-based FR models and our prior (limited) experience with deep learning, we chose this methodology to solve the challenge.

3 The pipeline

This section of the report will analyze each step of the face recognition pipeline we have developed. The process of taking an input image and returning a person's ID entails five steps: a face detector preprocessor, a face detector, a feature extractor preprocessor, a feature extractor, and a feature classifier (responsible for determining to which ID the extracted features correspond to). However, our approach combines the feature extractor and feature classification into a single neural network. Figure 1 shows a graphical representation of our implemented pipeline.

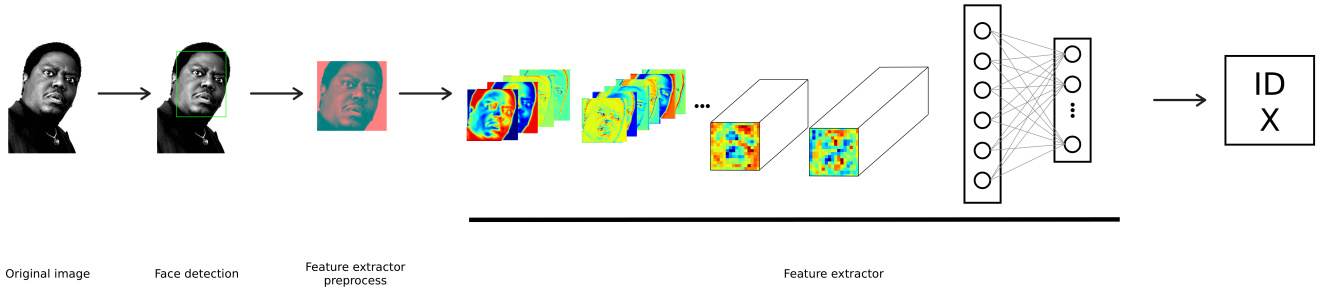


Figure 1: Pipeline overview

3.1 Face detector preprocessor

This step is responsible for applying any image transformation the face detector requires. Currently, we only change gray-scale images to RGB images with three channels (to have a uniform image shape).

3.2 Face detector

To correctly identify a face within an image, we first need to determine the location of the face, if it is present. As its name suggests, the face detector is responsible for detecting the coordinates within the image where a face is situated. In the first Lab on this subject, we implemented a satisfactory face detector based on the Viola and Jones algorithm Viola and Jones (2004). However, a significant drawback of our face detector was its relatively high false negative rate (FNR) of almost 6 %. A high FNR indicates that the model could not detect multiple real faces. Losing real faces early in the pipeline poses a considerable obstacle for an accurate face recognition system, as faces that go undetected cannot be identified at all. Hence, we focused our efforts in this pipeline phase towards minimizing the FNR as much as possible.

3.2.1 MTCNN

We tested several face detectors to find the most suitable one for our pipeline: MediaPipe [1], YuNet Wu et al. (2023), RetinaFace with Resnet-50 backbone [2] Deng et al. (2020), and Multitask Cascaded Convolutional Networks (MTCNN) [3] Zhang et al. (2016). MTCNN was the one that produced better results, achieving a FNR of 1.23% while preserving a good F1-score. Check Appendix B for a detailed comparison.

It is important to mention that reducing the FNR comes with a set of trade-offs. An adverse outcome of this reduction is an increased false positive rate, or in other words, identifying human faces in images that lack them. Therefore, we need our feature extractor model to be robust enough to handle input images without faces (by classifying them with a low confidence for example).

3.3 Feature extractor preprocessor

Once the detector has pinpointed the location and bounding box of a detected face, preprocessing is necessary to format it correctly for input into the feature extractor. As detailed in Section 3.4, our feature extractor requires input images of size 128x128 with three color channels. The responsibility of converting the information provided by the face detector into a suitable image for the feature extractor lies with the feature extractor preprocessor. This preprocessor performs three primary functions:

1. Crop the original image around the generated bounding box, ensuring that the resulting crop is square and does not distort the face in any dimension.

2. Resize the square crop to match the 128x128 input size expected by the feature extractor (this may involve either upscaling or downscaling the image).

3. Adjust the color format of the image to the one expected by the feature extractor.

3.4 Feature extractor

The feature extractor stands out as the most critical element of the pipeline. It aims to extract fundamental features of the facial images that it receives. Our DL-based feature extractor not only extracts these features but also performs the classification process to assign the corresponding ID to the input image.

Convolutional Neural Networks (CNNs) have shown remarkable performance at extracting facial features through their convolutional layers. CNNs use convolutional filters to scan through the input image, capturing local patterns and features at different spatial scales. The concatenation of different filter sizes produces a hierarchical feature extraction process that allows the network to capture complex patterns and attributes specific to faces.

State-of-the-art facial recognition networks contain an enormous number of parameters. For example, DeepFace surpasses 120 million parameters Taigman et al. (2014), illustrating the scale of complexity

[1] Open source project available at https://github.com/google/mediapipe

[2] Python library code available at https://github.com/serengil/retinaface

[3] We have used a PyTorch-compatible python implementation, available at https://github.com/timesler/facenet-pytorch

involved in these systems, especially for their training. Our feature extractor had to fulfill two main requirements: to *use a maximum of 1M of parameters* and *include at most ten layers with trainable parameters*. Hence, the main focus at this project stage was to develop a lightweight CNN architecture capable of extracting meaningful and general enough facial features aligned with the imposed requirements.

3.4.1 Light CNN

The Light CNN network, Wu et al. (2018), has proved efficient in terms of computational cost and storage space, achieved by minimizing the parameter count. Additionally, it is a robust network in handling extensive noisy datasets during training, facilitated by a variant of the maxout activation, which their authors refer to as max-feature-map (MFM). Given these advantages, we considered this architecture a good baseline for our feature extractor.

One of the breakthroughs introduced by the authors of the Light CNN architecture is the **MFM activation function**. One of its main advantages is that it effectively handles noisy signals commonly found in large-scale datasets, ensuring that errors from these signals do not bias the learning process of the CNN. Unlike traditional activation functions such as Rectified Linear Unit (ReLU) or some of its variants such as Leaky Rectified Linear Units (LReLU) and Parametric Rectified Linear Units (PReLU), MFM principles are akin to lateral inhibition from neural science Wu et al. (2018), which occurs during visual processes (as it is the case of facial recognition). Moreover, MFM achieves that the inhibition from one neuron is parameter-free, ensuring robustness across diverse datasets without extensive dependency on training data.

Wu et al. (2018) proposed three different variants of the Light CNN architecture. Its second lightest version can be seen in Table 5. Note that despite being considered a light model, the number of parameters is way higher than the maximum of 1M parameters we were allowed to use.

3.4.2 Superlight CNN

To reduce the number of parameters of the Light CNN network, we have created what we have named as **Superlight CNN**. The main change of the new architecture is a reduction in the number of output channels of each convolutional layer. Furthermore, we have swapped the first two convolutional layers with two inception modules since we were allowed to use up to two branches in our model. Finally, we have also decided to train the model with images in LAB color format, which can help with discrepancies in illumination.

The layer with the highest parameter count is the last fully connected layer, responsible for classifying the extracted features into the different IDs. The original Light CNN network extracts 256 features for each input image. We opted to decrease this number to 133 (the highest value we could get without surpassing 1M total parameters), sacrificing some degree of generalization and classification capability but significantly reducing the parameter count.

In the end, we have a total 998,282 trainable parameters distributed among 10 layers (8 convolutional layers and 2 fully connected layers. See Figure 4 for a detail diagram of the Superlight CNN architecture.

3.4.3 Other attempts

We have tried several techniques intended to improve the performance of our Superlight CNN architecture.

One of the most promising techniques was the use of depth-wise separable convolutions, which reduce the number of parameters of convolutional layers while keeping a similar performance Howard et al. (2017). A depth-wise separable convolution performs a regular convolution operation in two different steps: a **depth-wise** convolution, which captures spatial information independently for each input channel, and a **point-wise** convolution, applied after the depth-wise convolution, which performs a combination of the channels.

With the additional parameters we had to spare, we increased the number of output channels of each convolutional layer as much as we could. However, we saw a regression in performance, possibly because we had to remove some 1x1 convolution layers to comply with the restriction of 10 trainable layers at most.

Another modification was the introduction of normalization techniques: first, we tried batch normalization and then instance normalization. However, the transfer learning step was negatively affected by both techniques, probably because the dataset used for training the feature extractor and the dataset used for training the classifier were different.

4 Training methodology

The two steps of our pipeline that might require training are the face detector and the feature extractor + classifier. We were allowed to use a pre-trained model for the face detector but not for the feature extractor and classifier.

As we will explain in more detail in the following section, we decided it best to train our Superlight CNN model with two different datasets, one with more than the 80 IDs we need to classify for the challenge and another one with only those 80 IDs. To do so, we first trained the model with the first dataset, adjusting the last fully connected layer (responsible for classifying the input images) to have an output size equal to the number of classes of this dataset. Once training with the first dataset finished, we used transfer learning to keep the learned facial features, and we only retrained the last fully connected layer, this time with the second dataset and an output size of 80 IDs.

Here, we provide a summary of the hyperparameters used in PyTorch to train both models:

Table 1: Training hyperparameters. The learning rate was progressively reduced with a learning rate scheduler.

Hyperparameter name	**value**
batch size	512
number of epochs	15 for dataset 1 and 200 for dataset 2
loss function	cross-entropy
initial learning rate	1e-3
final learning rate	1e-5 for dataset 1 and 1e-6 for dataset 2
weight decay	0
optimizer	Adam with default values

5 Datasets used

Deep learning-based methods rely on enormous amounts of data for effective learning and generalization. Large volumes of labeled samples enable these models to learn diverse and subtle representations, enhancing their ability to generalize well to unseen instances. Wang and Deng (2021) summarize the data with which state-of-the-art models have been trained. The models that used fewer images have seen around 200K images during training, corresponding to 10K different identities. This contrasts with the amount of data we had been provided with for the challenge, only 1200 images. Moreover, 700 of these images belonged to impostors, so we only had 500 effective images to train the model.

Training a DL model on these few samples would generate an extremely over-fitted model, unable to generalize in front of unseen data. For this reason, the first step to train a model is to **obtain more training data**.

Having a dataset of hundreds of thousands of images of only 80 different people is not a viable solution since finding more than a hundred images for each identity is already difficult. For this reason, we decided to use two different datasets:

1. An existing face recognition dataset with a vast number of images and IDs. We use this dataset to train the feature extractor component of the model.

2. An expansion of the original 1200 images we were given with more data from each of the 80 IDs. We use this dataset to train the classification component of the model.

5.1 The CelebA dataset

The CelebFaces Attributes Dataset (CelebA, Liu et al. (2015)) contains 202,599 face images corresponding to 10,177 identities. The advantage of this dataset is that, despite containing many images, it can be considered small compared to other datasets used for larger FR architectures. One advantage of the reduced size is that the annotations are very precise. Given that the dataset was accessible and relatively easy to use, we decided to train the early versions of our models using the CelebA dataset.

5.2 The VGGFace2 dataset

While the CelebA dataset was useful early in development, we had to stop training early to avoid over-fitting. In order to train the model for more epochs and improve its performance, using even more data could be beneficial. Ideally, we were looking for a dataset with more than 1M images. Even though over-parametrized deep learning models can achieve a good performance Geiger et al. (2020), under-parametrized models are often preferred. Otherwise, the training data could be theoretically fitted with zero loss. Therefore, having more data samples than model parameters ensures having an under-parametrized model. For this reason, we decided to use the VGGFace2 dataset, Cao et al. (2018). This dataset contains 3.31M images of 9131 subjects, with an average of 362.6 images for each subject.

5.3 The expanded original dataset

With the CelebA and VGGFace2 datasets, we aimed to develop a robust facial feature extractor capable of capturing intricate facial details. However, to accurately identify the 80 required identities for the challenge, we still needed many images from those 80 IDs for the classifier to work correctly. Consequently, we decided to enrich the original dataset by incorporating additional data. This augmentation strategy involved two distinct approaches:

1. For Lab 4, our idea was to incorporate roughly 100 images for each ID. To do so, we used an automatic Google web scraper. Ultimately, we achieved a dataset with 6k images and 80 unique labels.

2. Based on the findings from Lab 4, we identified a potential deficiency in our model's generalization capacity. Consequently, we decided to augment the original dataset with even more additional data. On average, we added 100 new images per identity, which we manually downloaded from DuckDuckGo. Our selection process aimed to diversify the dataset by including images with varying poses and lighting conditions compared to our existing samples. The dataset grew to 15,265 images across 80 distinct labels with this expansion.

It is crucial to emphasize that these two steps entail acquiring the images and preprocessing and accurately labeling them. This ensures that the new dataset does not contain noisy labels.

6 Results

Having explained the thinking and development process of the project, in this section we will now finally present the results that we have obtained.

The accuracy of the model when trained with the VGGFace2 dataset is 83.2 %. In figure 2 we plot the evolution of the training loss, the training accuracy, the validation loss and the validation accuracy.

The accuracy of the model when trained with the expanded dataset that only contains the required 80 IDs is 75.62%. In figure 3 we plot the evolution of the training loss, the training accuracy, the validation loss and the validation accuracy.

Finally, the challenge results obtained with the original 1200 images are the following ones:

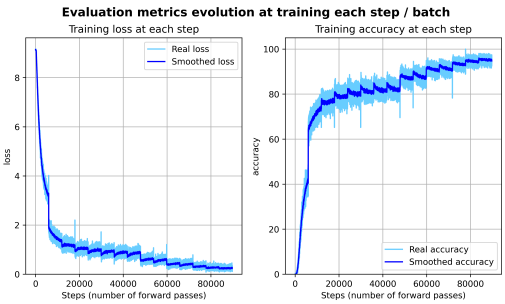
Table 2: Challenge results.

F1-score	**Execution time (mm:ss)**
88.10 %	00:52

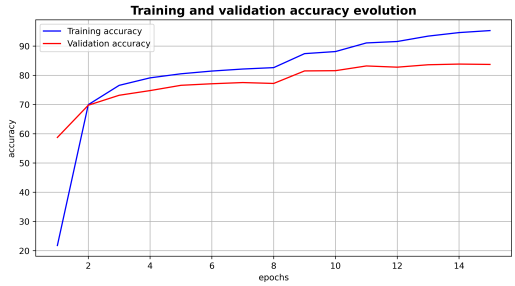
The improvements implemented between Lab 4 and the final challenge have proven beneficial in terms of both model performance and generalization. These improvements have enabled us to continue training the model without excessively overfitting the training data. A comparison of the outcomes presented in Table 2 with those obtained in Lab 4, as depicted in Table 5 of Appendix E, highlights the efficacy of these advancements.

7 Future work

We will conclude this report with a brief mention of possible improvements and future work. First, we would have liked to try other architectures besides Light CNN. However, given the limited time frame, the fact that



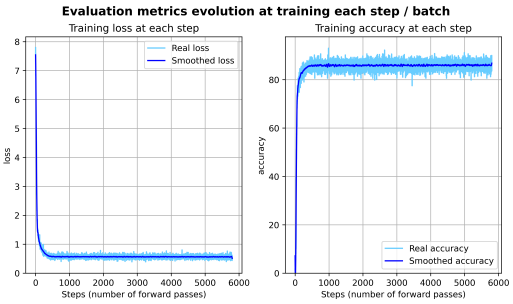
(a) Training loss and accuracy evolution per batch



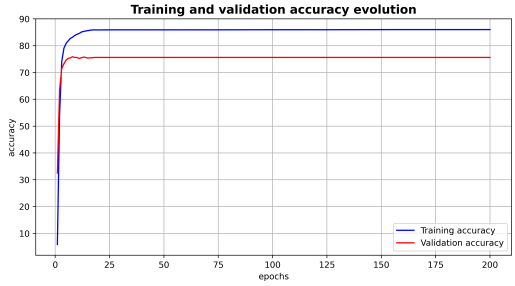
(b) Training and validation accuracy per epoch

(c) Training and validation loss per epoch

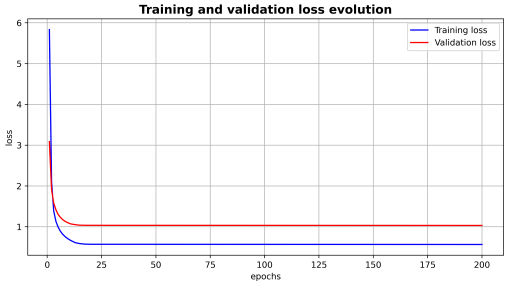
Figure 2: Training results of the Superlight CNN model trained with the VGG-Face2 dataset



(a) Training loss and accuracy evolution per batch



(b) Training and validation accuracy per epoch



(c) Training and validation loss per epoch

Figure 3: Transfer learning results of the Superlight CNN model trained with the Expanded dataset

it already had good performance out of the box, and that it was close to satisfying most of our requirements, we decided to stick with it. Secondly, we could have dedicated more effort to handling noisy labels since we have read that it can help improve the training process and the model's performance. Finally, we could have incorporated alignment techniques into the feature extractor preprocessor, which could improve the results by 1-2 %.

References

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE.

Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5203–5212.

Geiger, M., Jacot, A., Spigler, S., Gabriel, F., Sagun, L., d'Ascoli, S., Biroli, G., Hongler, C., and Wyart, M. (2020). Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401.

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: efficient convolutional neural networks for mobile vision applications. *arXiv (Cornell University)*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014). Deep learning face representation by joint identification-verification. *Advances in neural information processing systems*, 27.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57:137–154.

Wang, M. and Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, 429:215–244.

Wu, W., Peng, H., and Yu, S. (2023). Yunet: A tiny millisecond-level face detector. *Machine Intelligence Research*, pages 1–10.

Wu, X., He, R., Sun, Z., and Tan, T. (2018). A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896.

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.

Appendices

A IDs of the 80 famous people

Table 3: List of Names with IDs.

ID	Name	ID	Name	ID	Name
1	Channing Tatum	2	Christina Applegate	3	Richard E. Grant
4	S. Epatha Merkerson	5	Farah Fath	6	Jim Beaver
7	Cheryl Hines	8	Michael Vartan	9	Hayden Christensen
10	Laurence Fishburne	11	Kathryn Joosten	12	Patrick Warburton
13	Jamie Lee Curtis	14	Jason Sudeikis	15	Billy Burke
16	Robert Pattinson	17	Melissa Claire Egan	18	Morena Baccarin
19	Jolene Blalock	20	Matthew Lillard	21	Alicia Goranson
22	Jennie Garth	23	Wanda De Jesus	24	Tracey E. Bregman
25	Tracey Gold	26	Brendan Fraser	27	Kellan Lutz
28	John Travolta	29	Pierce Brosnan	30	Jasmine Guy
31	Swoosie Kurtz	32	Diego Luna	33	Danny Glover
34	David Cross	35	Farrah Fawcett	36	Paul Walker
37	Matt Long	38	Andy García	39	Casey Affleck
40	Carla Gallo	41	James Brolin	42	Christian Bale
43	Nadia Bjorlin	44	Valerie Bertinelli	45	Alec Baldwin
46	Tamara Braun	47	Andy Serkis	48	Jackson Rathbone
49	Robert Redford	50	Julie Marie Berman	51	Chris Kattan
52	Benicio del Toro	53	Anthony Hopkins	54	Lea Michele
55	Jean-Claude Van Damme	56	Adrienne Frantz	57	Kim Fields
58	Wendie Malick	59	Lacey Chabert	60	Harry Connick Jr.
61	Cam Gigandet	62	Andrea Anders	63	Chris Noth
64	Cary Elwes	65	Aisha Hinds	66	Chris Rock
67	Neve Campbell	68	Susan Dey	69	Robert Duvall
70	Caroline Dhavernas	71	Marilu Henner	72	Christian Slater
73	Kris Kristofferson	74	Shelley Long	75	Alan Arkin
76	Faith Ford	77	Jason Bateman	78	Edi Gathegi
79	Emile Hirsch	80	Joaquin Phoenix		

B Comparison of face detectors

Table 4: F1-score and false negative rate (FNR) of different face detectors.

Method	F1-score	FNR	Execution time (mm:ss)
Viola-Jones + MediaPipe (Lab 1)	88.50 %	5.97 %	03:07
MediaPipe	66.13 %	5.06 %	00:03
MTCNN	73.42 %	1.23 %	01:52
RetinaFace Resnet-50	68.32 %	1.53 %	16:49
YuNet (OpenCV implementation)	72.55 %	4.13 %	00:26

C LightCNN network structure

Table 5: Architecture of the LightCNN network with 9 layers, taken from Wu et al. (2018)

Type	Filter Size /Stride, Pad	Output Size	#Params
Conv1	$5 \times 5/1, 2$	$128 \times 128 \times 96$	2.4 K
MFM1	-	$128 \times 128 \times 48$	-
Pooll	$2 \times 2/2$	$64 \times 64 \times 48$	-
Conv2a	$1 \times 1/1$	$64 \times 64 \times 96$	4.6 K
MFM2a	-	$64 \times 64 \times 48$	-
Conv2	$3 \times 3/1, 1$	$64 \times 64 \times 192$	165 K
MFM2	-	$64 \times 64 \times 96$	-
Pool2	$2 \times 2/2$	$32 \times 32 \times 96$	-
Conv3a	$1 \times 1/1$	$32 \times 32 \times 192$	18 K
MFM3a	-	$32 \times 32 \times 96$	-
Conv3	$3 \times 3/1, 1$	$32 \times 32 \times 384$	331 K
MFM3	-	$32 \times 32 \times 192$	-
Pool3	$2 \times 2/2$	$16 \times 16 \times 192$	-
Conv4a	$1 \times 1/1$	$16 \times 16 \times 384$	73 K
MFM4a	-	$16 \times 16 \times 192$	-
Conv4	$3 \times 3/1, 1$	$16 \times 16 \times 256$	442 K
MFM4	-	$16 \times 16 \times 128$	-
Conv5a	$1 \times 1/1$	$16 \times 16 \times 256$	32 K
MFM5a	-	$16 \times 16 \times 128$	-
Conv5	$3 \times 3/1, 1$	$16 \times 16 \times 256$	294 K
MFM5	-	$16 \times 16 \times 128$	-
Pool4	$2 \times 2/2$	$8 \times 8 \times 128$	-
fcl	-	512	4.1 M
MFM_fc1	-	256	-
Total	-	-	5.5 M

D SuperlightCNN network diagram

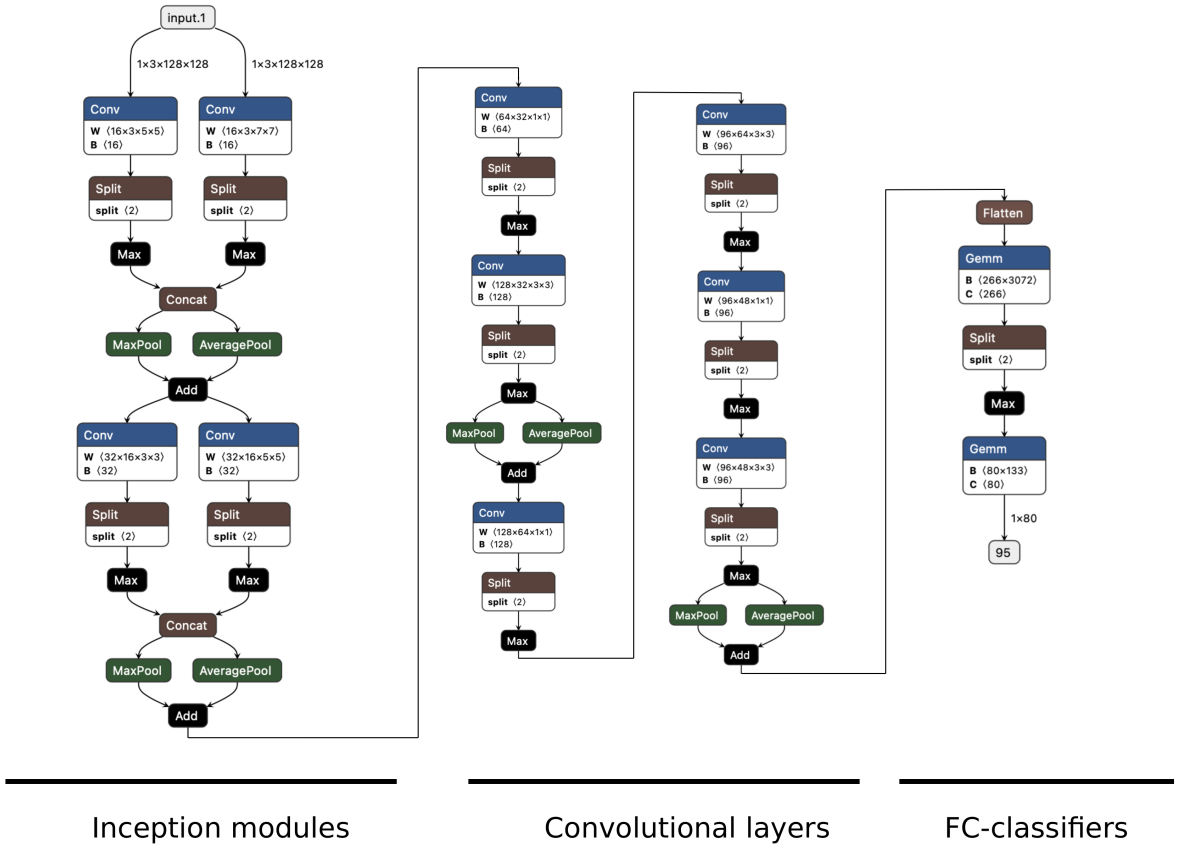
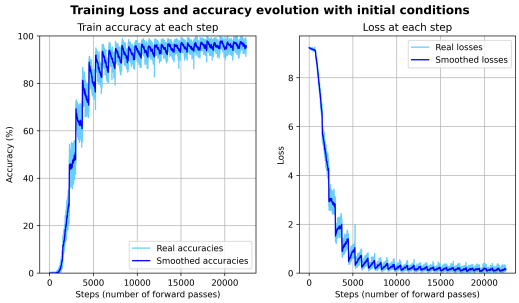
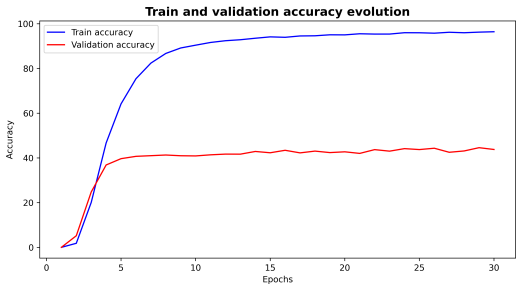


Figure 4: Superlight CNN diagram.

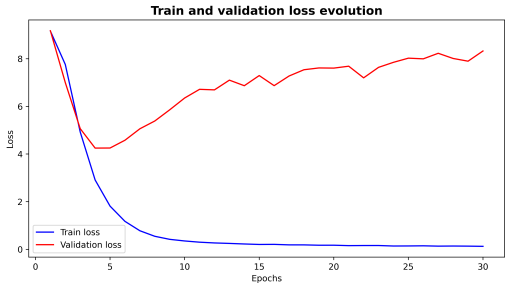
E Training results of Lab 4



(a) Training loss and accuracy evolution per batch



(b) Training and validation accuracy per epoch



(c) Training and validation loss per epoch

Figure 5: Lab 4 training results of the Superlight CNN model trained with the CelebA dataset