Final Report

André Teodosio, Arianna Pierini, Hélène Dos Santos

System Overview

The system for mask detection and classification is divided into two key parts: **face identification and cropping** and **classification of mask usage**. These components work together to detect faces in images, preprocess them, and classify their mask usage status into one of three categories: "mask", "no_mask", and "wrong_mask". The code and pre-trained models are more than 10mb so the project is available on the github repository, on this zip is available the Classifiers and DeepFace notebooks and a set of images to test. It is needed to download the models on the repository on Project-deliver folder.

Part 1: Face Identification and Cropping

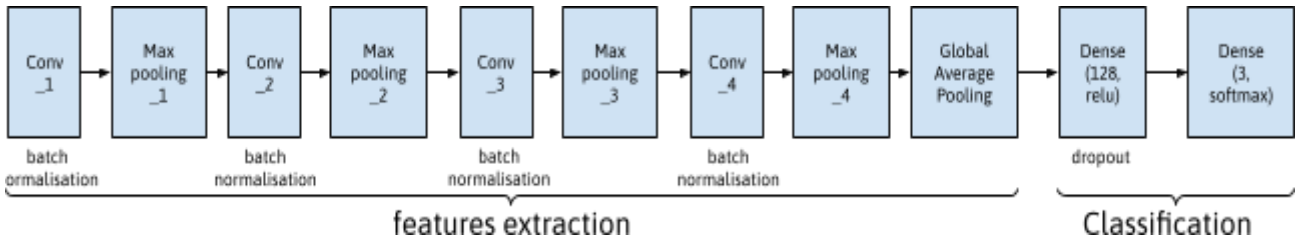
The first step involves identifying and cropping faces from input images. Using the **DeepFace** library with the **RetinaFace** backend, the system detects faces in each image and retrieves their bounding box coordinates. These bounding boxes are slightly expanded using offsets to ensure the cropped region fully captures the face. For every detected face, a cropped version is saved to a designated folder for further processing.

Part 2: Classification of Mask Usage

The cropped face images are divided into training, validation, and test sets to prepare for classification. Two different approaches are implemented for this classification task: a **custom Convolutional Neural Network (CNN)** and a **pretrained MobileNetV2**.

1. CNN Approach

The custom CNN consists of multiple convolutional layers, batch normalization, max-pooling, and a final softmax layer for three-class classification. Data augmentation techniques, including rotation, zoom, horizontal flipping, and shearing, are applied to the training data to improve robustness and generalization. The model is trained using categorical cross-entropy as the loss function and incorporates early stopping and learning rate reduction callbacks to prevent overfitting and optimize training efficiency.

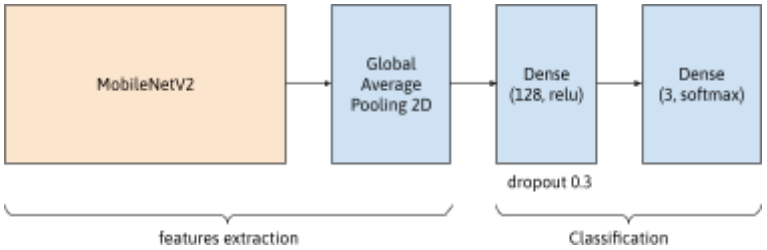


Architecture of the custom CNN

2. MobileNetV2 Approach

The second approach uses a transfer learning method by leveraging the pretrained **MobileNetV2** model. The base model is frozen, and custom dense layers are added for the mask classification task. This approach benefits from MobileNetV2's highly efficient feature extraction capabilities, especially

when dealing with small datasets. Like the CNN, the model is trained with early stopping and learning rate reduction to maximize performance.



Architecture of the mobilenetv2 based CNN

Both approaches are evaluated on the test set, with metrics such as accuracy, a classification report, and a confusion matrix being generated to compare their effectiveness. While the CNN provides flexibility and customization, the MobileNetV2 leverages pretrained knowledge for faster convergence and potentially higher accuracy.

Dataset

The dataset initially exhibited an imbalance among the classes: **mask** (people wearing masks correctly), **no_mask** (people not wearing masks), and **wrong_mask** (people wearing masks incorrectly). To address this, a reduction in the number of samples from the majority classes, combined with efforts to increase the number of **wrong_mask** samples, was crucial. This process ensured the creation of a more balanced and robust dataset, which was then effectively used for training, validation, and testing.

Preprocessing and Data Augmentation

Data augmentation plays a critical role in improving the robustness and generalization of the mask classification model. By artificially increasing the diversity of the training dataset, augmentation helps the model become less sensitive to variations in the input data, such as changes in orientation, scale, and position, which are common in real-world scenarios. The following augmentation techniques were applied during the training process:

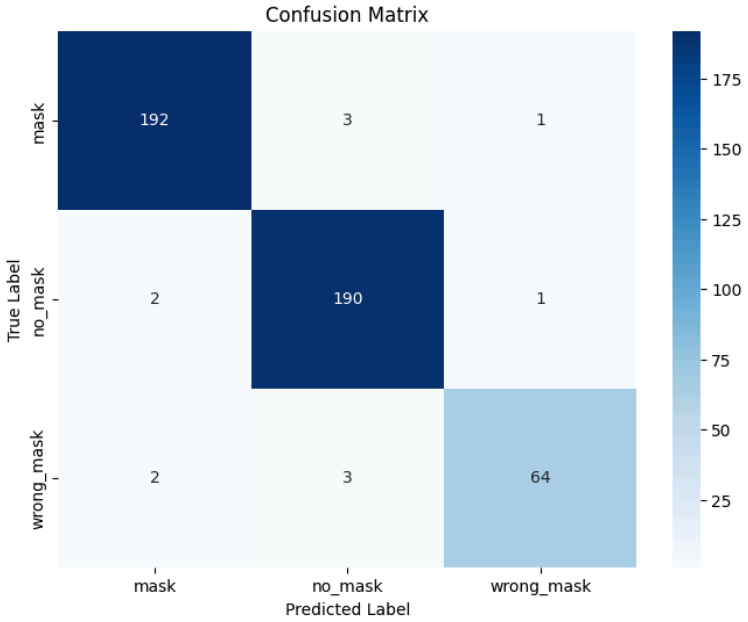
1. **Rotation**: Images are randomly rotated by up to 30 degrees to simulate various head tilts and orientations, ensuring the model can handle angular variations in face orientation.
2. **Width and Height Shifts**: Small random shifts, up to 20% of the image width or height, are applied. This helps the model become robust to slight misalignments or variations in the positioning of faces within the frame.
3. **Shear Transformations**: Shearing introduces slight geometric distortions, making the model resilient to changes in perspective or face angles.
4. **Zooming**: Random zooming in or out by up to 20% is applied to simulate variations in camera distances, ensuring the model can handle both close-up and distant face images.
5. **Horizontal Flipping**: Images are flipped along the vertical axis to introduce mirrored versions of the faces, accounting for symmetry and making the model robust to orientation differences.

6. **Rescaling**: All pixel values are normalized by dividing by 255, ensuring the inputs fall within the range [0, 1]. This normalization stabilizes training and prevents issues caused by varying pixel intensity scales.

These augmentation techniques are applied on-the-fly during training using the **ImageDataGenerator** class from Keras. This dynamic augmentation ensures that the model sees a new, slightly altered version of each image in every epoch, effectively increasing the size and variability of the training dataset without requiring additional data collection. By leveraging data augmentation, the model gains the ability to generalize better to unseen data, improving its real-world performance.

Results and models' evaluation

MobileNetV2

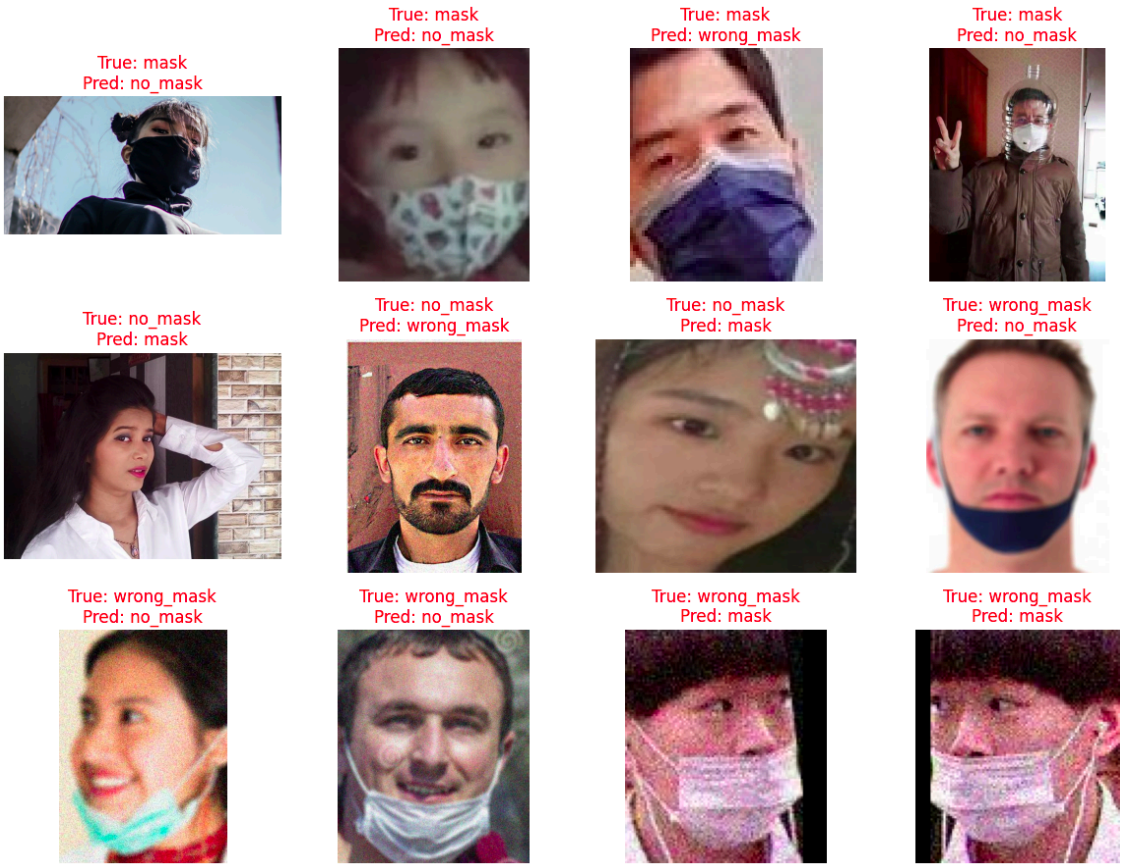


As we can see on the confusion matrix this model seems to be accurate since there are only 12 misclassified images. Which are balanced between all classes.

	precision	recall	f1-score	support
mask	98%	98%	98%	196
no_mask	97%	98%	98%	193
wrong_mask	97%	93%	95%	69
accuracy			97%	458
macro avg	97%	96%	97%	458
weighted avg	97%	97%	97%	458

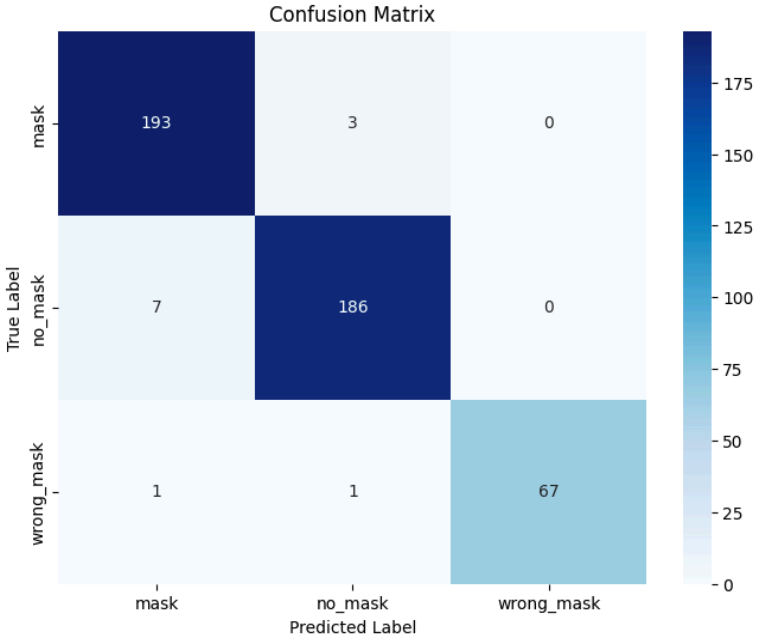
Evaluation coefficient of the mobilenetv2 based CNN

This model is highly effective with really good evaluation on the mask and no_mask labels, nonetheless it seems that it is underperforming on the wrong_mask label.



Misclassified images by mobilenetv2 based CNN

Custom CNN



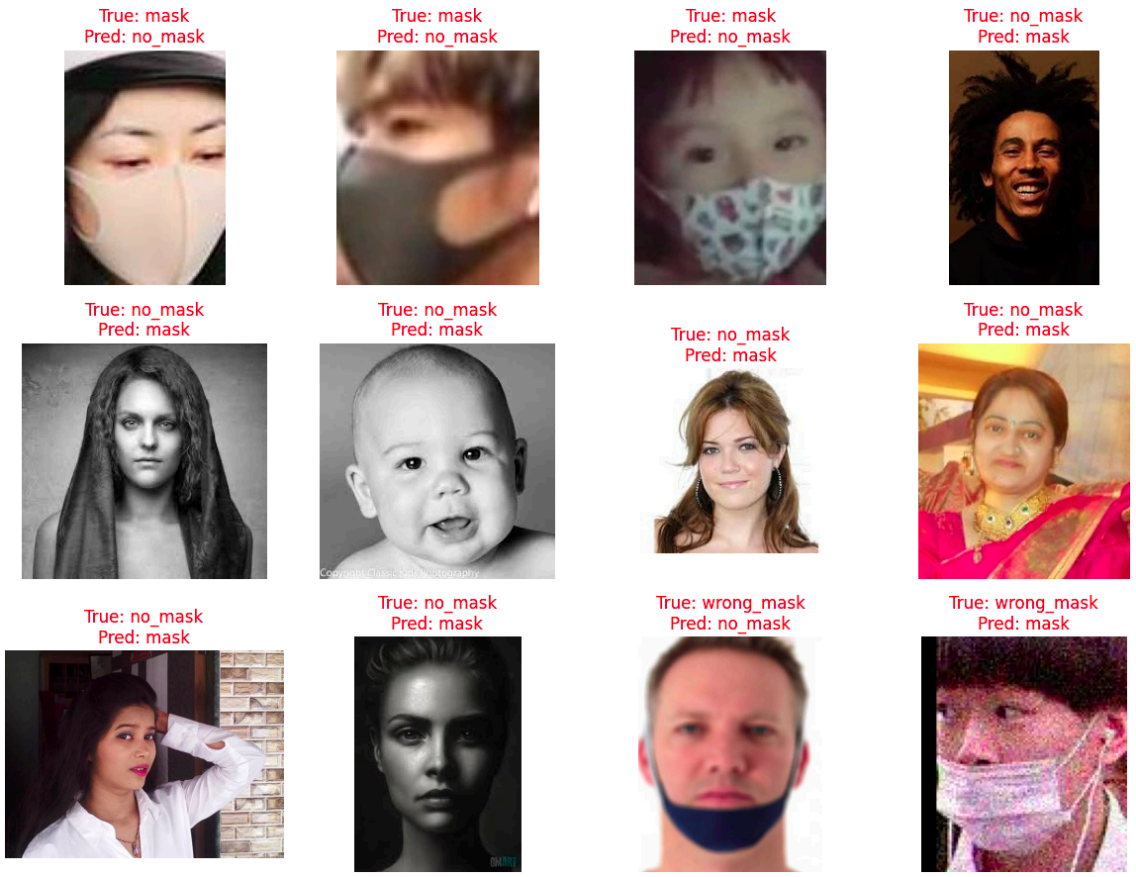
As we can see on the confusion matrix, our model also seems to be pretty effective. There are also only 12 misclassifications.
Misclassifications seem to happen especially between the classes mask and no_mask. However this model seems to be very effective on classifying wrong_mask.

	precision	recall	f1-score	support
mask	96%	98%	97%	196

no_mask	98%	96%	97%	193
wrong_mask	100%	97%	99%	69
accuracy			97%	458
macro avg	98%	97%	98%	458
weighted avg	97%	97%	97%	458

Evaluation coefficient of our CNN

This model is also highly effective, especially on the wrong_mask. It also seems to be slightly weaker on the right mask detection.



Misclassified images by our CNN

Comparison

Comparing the two approaches shows that they are both highly effective, with an accuracy of 97%.

MobileNetV2 demonstrates consistent performance across all classes, with high precision, recall, and F1-scores for "mask" and "no_mask." However, it slightly underperforms in the "wrong_mask" class, where its recall is 93% and F1-score is 95%, compared to our solution which had higher recall (97%) and F1-score (99%). MobileNetV2's strengths lie in its pre-trained, lightweight architecture, making it ideal for deployment on mobile and low-power devices. Despite its slight drop in performance for "wrong_mask," it remains a highly efficient and reliable model for a mask classification task.

As we said before, our solution slightly outperforms MobileNetV2 in classifying the "wrong_mask" category, achieving perfect precision (100%) and better overall class-level metrics. This suggests that this model may capture subtle patterns better in the "wrong_mask" category, making it a stronger option if identifying this specific class is critical for the application. While Conv2d matches MobileNetV2 in overall accuracy and weighted average performance, its class-level edge, especially for the more challenging category, makes it a compelling choice. However, its suitability for deployment may depend on its computational complexity and resource requirements compared to the more efficient MobileNetV2.

Personal contributions to the project

All group members contributed to each phase by providing inputs, and reviewing each other’s work. Helene mostly worked on the pre-processing part : data augmentation and face recognition. Arianna and Andre shared tasks concerning the CNN models.