Masked Faces: Overcoming Recognition Challenges with Transfer Learning in CNNs

Italo T. Noé[1], Lucas H. L. Costa[1], Talles H. Medeiros[1]

Department of Computing and Systems, Federal University of Ouro Preto, Brazil
italo.noe, lucas.lirio@aluno.ufop.edu.br, talles@ufop.edu.br

Abstract. Amidst the coronavirus pandemic, the use of masks has become one of the main ways to prevent and control the spread of this virus. However, masks impacted the performance of several face recognition models, by reducing visible features in images. The objective of this work is to present a model of convolutional neural networks with transfer learning capable of classifying thirty individuals regardless of the use of masks. The model was trained in a dataset with real images of masks and another with the insertion of computationally simulated masks and the accuracy results obtained were greater than 90%. Due to the number of classes and limitations of the dataset used, the result is consistent with the low number of related works and highlights the complexity of the problem. It is believed that the use of a dataset with superior quality and quantity of images would make the use of the model more viable for the real world, but the tests presented are promising.

CCS Concepts: • **Computing methodologies** → **Machine learning approaches**.

Keywords: Convolutional Neural Networks, Facial Recognition, Transfer learning

1. INTRODUCTION

The COVID-19 pandemic has profoundly impacted people's daily lives, necessitating face masks as an essential means of protection due to the ease of virus transmission through air and surfaces [Vincent Chi-Chung Cheng 2020]. The Facial recognition technology serves as a preventive method in the biometric field, emphasizing the need for research in this area. Even with the COVID-19 pandemic under control, masks remain crucial for controlling the transmission of respiratory diseases in hospitals, industrial areas, and specific cultural contexts.

The National Institute of Standards and Technology (NIST) discovered that the accuracy of pre-pandemic facial recognition algorithms decreased significantly when applied to masked faces [Ngan et al. 2020]. While leading algorithms had a 0.3 percent error rate for unmasked faces, the rate increased to approximately 5 percent for masked faces. Most facial recognition models use Convolutional Neural Networks (CNNs) to extract facial features and these networks classify faces by identifying individual features. However, masks restrict feature extraction to the eye area and regions near it, accounting for the decline in model accuracy.

This work aims to develop facial recognition models for individuals wearing masks and investigate methodologies for adapting existing models. It also seeks to evaluate the accuracy of these architectures in real-world scenarios, especially regarding COVID-19 prevention applications. The methodology involves using CNNs for face recognition and assessing the potential of transfer learning in creating new models.

2. LITERATURE REVIEW

2.1 Convolutional Neural Networks

The Convolutional Neural Networks (CNN) have as their main characteristic the use of the mathematical operation convolution in at least one of its layers, instead of the general multiplication of the matrix. It makes them very useful for the Pattern Recognition problem in images because they present methodologies and specific architectures to reduce the number of parameters to be adjusted by the network. Thus, CNNs can better deal with variations and distortions in the image, and reduce them to the training stage considering the most relevant pixels without the need for massive sampling.

The Convolutional Layer is characterized by the convolution of a filter as a function of shifting in the main image, producing an output called Feature Map, which can be interpreted as a mapping of more similar features between the filter and the sub-regions of the image. After the Convolution Layer, it is common to use a Pooling Layer to reduce the dimensions of the parameters to be trained without losing the main characteristics and making the network invariant to geometric transformations, making a considerable reduction in the cost of the network. Finally, the Dense Layer consists of a network with all neurons interconnected to all input elements, usually represented by multilayer perceptrons.

2.2 Transfer Learning

The concept of transfer learning allows transferring the learning of a very deep network into a different dataset with similar characteristics for solving a new problem [Pan and Yang 2010]. The facial recognition area greatly benefits from this concept because there are several robust transfer learning architectures such as VGG [Simonyan and Zisserman 2014]. The first layers of a CNN are typically used for detection of universal and lower-level patterns, such as curves and edges. In contrast, the last layers have more specific characteristics and therefore can be retrained with new data. In this way, the weights of the first layers can be maintained and the final layers replaced by a new classifier. With this, it is possible to apply part of robust models to solve problems different from the one proposed by it. However, this methodology cannot be applied to any situation and the ability of the chosen architecture to generalize the problem to be solved must be evaluated.

3. DEVELOPMENT

In this section, the process of executing this work will be discussed, starting with the choice of the dataset, and how the data analysis and pre-processing was done. Subsequently, class balancing and data augmentation were applied in order to reduce the computational cost and improve the accuracy of the network. After that, the coding started with the choice of python frameworks and libraries, in addition to a CNN model to perform the work.

3.1 Analysis and Pre-processing

3.1.1 *Dataset.* The Real-World Masked Face Dataset (RMFD) [Wang et al. 2020] was used to generate the models. This was at the time the largest dataset of public people with and without masks. Created during the COVID-19 pandemic, this dataset was developed to help control public safety through facial recognition models. The RMFD has more than 90000 images arranged in two folders (with and without mask) and each one has subfolders grouped by the person's name according to Table I.

Despite containing a large amount of images, it is possible to notice that the binary group is unbalanced in both number of people and total images. Although the masked group has more people, the amount of images it contains is much lower, and it also has noise, empty folders, or folders with a very small amount of images.

Table I. Distribution of the RMFD dataset.

Classes	Total People	Total Images
Without Mask	460	90458
With Mask	525	2156

3.1.2 *Preprocessing.* Since the ultimate goal is to classify people regardless of the use of masks, masked faces have few recognizable regions, the main one being the eyes. Because of this and the unbalance, the masked dataset was filtered for people not wearing masks with a number of images greater than or equal to 10. After filtering, a new dataset was generated in which the classes are the people resulting from the filtering with their respective images with and without masks. As a result, 30 classes of people were separated with a total of 6770 images.

After separating the classes to be trained, noise removal was performed in two ways: automatic and manual. First, automatic removal was performed using the MTCNN (Multi-task Cascaded Convolutional Networks.) library of PyTorch to verify which unmasked images correspond to a human face. In the manual mode, the remaining images that contained most of the covered face were removed.

In addition to the dataset with real images of people with mask, a new dataset was generated with masks artificially inserted into the dataset without mask. Thus, we seek to explore the ability to generate artificial images for face recognition in the real world and circumvent the scarcity of data of people with mask so far. Once the efficiency of this methodology is proven, it can be an alternative for adaptation of existing models.

For the generation of the dataset with simulated masks, the MaskTheFace script [Anwar and Raychowdhury 2020] was tested. As it is desired to compare with the model generated by the RMFD dataset images, only images for the same 30 classes of the first dataset were generated. The settings chosen for this were black masks of the cloth type, which are the ones that presented the closest resemblance to the real ones present in the original dataset. The amount of images of masked faces obtained in a simulated way was much larger, but some real faces were not detected by MaskTheFace due to their low quality.

Finally, in both datasets, the images were resized to 160x160 resolution with the addition of null pixels on the edges to preserve the original proportions of each face and avoid distortions. This way it is possible to obtain better accuracy in the filters generated during training by conserving the size and dimensions of the eyes.

3.1.3 *Class Balancing.* After pre-processing it is possible to see a large discrepancy in the amount of images between the classes. Classification models that are usually evaluated by the accuracy metric and the use of unbalanced datasets tends to a greater accentuation of the error caused by the prioritization of predominant classes.

To work around this issue, the dataset was resampled using oversampling of the minority classes and undersampling of the majority classes. In this work, two resampling algorithms were explored: Random Under Sampler for subsampling and Synthetic Minority Over-sampling Technique (SMOTE) [Chawla et al. 2002] for oversampling.

After running the Random Under Sampler, the dataset with real images was reduced to 4230 images. While the application of SMOTE increased the amount of data in this same dataset to 9270. In addition, the dataset with simulated masks increased its amount of images to 11160. Since the amount of masked images in the first dataset had a small unbalance, the SMOTE technique was applied to the complete dataset. While in the second dataset, the data unbalance between masked and unmasked images was very large in both cases, so the SMOTE technique was applied twice individually and then the data was concatenated.

3.1.4 *Data Augmentation.* Images of human faces have a multitude of variations that can limit the generalizability of CNN models. Among these variations can be cited: changes in angulation, the form of cropping, zoom variations, as well as changes in illumination and saturation. The Convolutional neural networks are sensitive to these changes, so datasets with small amounts of data or few variations between them can generate impractical models for the real world. To reduce these impacts, data augmentation techniques can be used.

Data augmentation techniques consist of generating new images for the data set from transformations of the images present in the dataset. So a single image can generate several different images for the network if transformations such as mirroring, rotation, brightness variations, cropping, and even a combination of these and other possible options are applied. This way, the model can prevent overfitting as it is forced to adjust the filters to accommodate the various possible variations that a given problem may present. In this work the data augmentation technique with the transformations rotation, mirroring, resizing, zooming and shearing was used.

3.2 Frameworks and libraries

The language chosen to develop this work was Python with Jupyter Notebook. To generate the model, the chosen framework was TensorFlow together with the libraries Keras, Sklearn, Imblearn and Pillow. For validation with real data, we chose to use the TensorFlow and PyTorch frameworks and the MTCNN and OpenCV libraries. In this case, multiple frameworks were used because PyTorch demonstrated better performance in face recognition, while TensorFlow was necessary for loading the trained model.

In order to evaluate the cost of generating a model for deep networks in everyday equipment, we chose to train this model in a laptop with Intel Core i7-8750H CPU, NVIDIA GeForce GTX 1050 Ti GPU, 16 gb RAM and 480 gb SSD. Although 30 distinct classes were classified with a large dataset, training was only made feasible by performing the processing on the GPU. This requires the use of the NVIDIA CUDA library that already has integration with the libraries used. In addition, the amount of RAM available is not sufficient and therefore 35 GB was set aside for SWAP. The time for the different types of processing can be seen in the Table II. The average time is given per epoch and measured in seconds.

Table II. Comparison of training time between CPU and GPU with batch size of 128 and 800 epochs

Processing	**Average Time**	**Total Time**
CPU	736	$\approx$ 7 days
GPU	60	13h56

Table III. Hyperparameter values for the models.

Hyperparameter	**Value**
batch_size	128
input_shape	(160, 160, 3)
random_state	42
alpha	1e-5
epoch	800

3.3 Model

After preprocessing and data augmentation, for each model, the dataset was separated into 20% for testing and the rest for training. Then, the configuration of the hyperparameters was defined through experimentation. The best hyperparameters can be seen in Table 0**??**. It is possible to highlight the need for 800 epochs to make better use of the dropout layer and to guarantee a sufficient number of epochs for callback actions.

Then two callbacks were defined: ModelCheckpoint and ReduceLROnPlateau. The former will save the last epoch that obtains the highest accuracy value for the validation basis. As a result, an .h5 file is generated, which can be utilized by the model to classify unknown images. On the other hand, the second callback will reduce the learning rate by 0.1 per 100 epochs with no improvement in the hit

rate of the validation data. This technique is useful to extract maximum learning especially at the end of training that the gains are reduced.

The VGG16 model is an architecture of thirteen convolutional layers and three fully connected layers (totaling 16 layers), all of them grouped into five convolutional blocks, in which the first two have two convolutional layers and the last three have three layers. Another characteristic is the use of 3x3 filters with 1 stride, this type of filter is the smallest possible filter to obtain horizontal and vertical orientations. Then the VGG16 model is loaded and at this point it must be defined which convolutional layers will be retrained. As face recognition was one of the classes trained in this architecture, the filters obtained in the first layers are general and reusable. Because of this, only after the fourth block of convolutions were the filters retrained, since they are more specific and important for defining the differences between human faces. Thus, the network will retain 1,735,488 parameters and will retrain 12,979,200 parameters. However, before training, additional layers have been inserted into VGG16.

The first layers inserted were GlobalAveragePooling2D and BatchNormalization. GlobalAveragePooling2D allows the spatial dimensions to be reduced by applying average clustering for each dimension, thus reducing the amount of parameters. Subsequently, BatchNormalization is employed to ensure the output has a mean value close to 0 and a standard deviation close to 1. In this way, it prevents the simultaneous changes of the various applied functions from generating unexpected results that could disrupt the training. Both layers are capable of accelerating learning.

After parameter reduction and normalization, the input data is reduced to a single-dimensional vector by the flatten layer to be trained by fully connected layers. After each fully connected (dense) layer with ReLU activation function a dropout layer was inserted. The first dense layer was set with 1024 neurons and dropout with rate 0.5, followed by 3 dense layers of size 512 and dropout with rate 0.4. Then, to return the probability of a given input matching each trained class, a dense layer with softmax activation function is inserted. The general structure of the architecture can be seen in Figure 1. All implementations performed are available in a GitHub repository [Noe 2021].

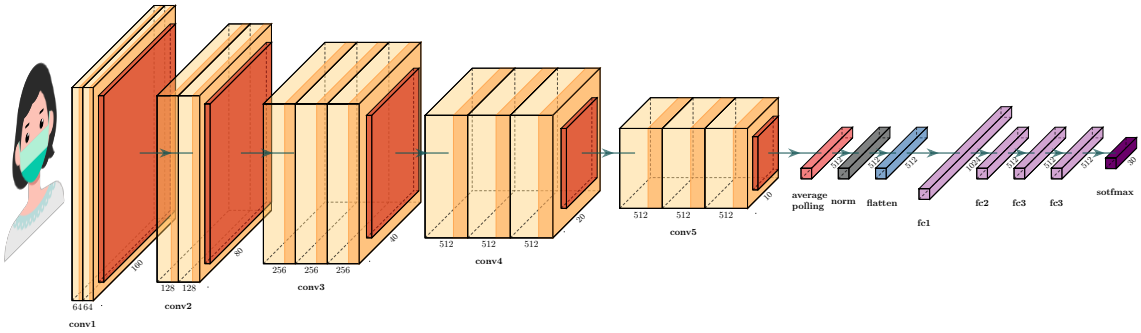

Fig. 1. Architecture created using VGG16.

4. EXPERIMENTAL RESULTS

4.1 Testing

The final model was defined through experimentation with various aspects, including input data variations, hyperparameter adjustments, retrained VGG16 layers, and the number of fully connected layers. Initially, the inputs were separated into two datasets: one containing real mask images and the other with computationally inserted masks. An overview of the experiments performed will be presented below, followed by a discussion of the results in the subsequent sections.

Upon analyzing the VGG16 architecture, it becomes clear that as the network goes deeper, more specific parameters are passed to the convolution layers. By opting to retrain all layers or just the first few, the network will need to learn several global features that are limited to the dataset used. Utilizing more weights from VGG16 allows for the reuse of filters that all faces have in common, thus reducing the cost of training the network. However, since the goal is to differentiate between multiple faces, the more specific layers must be retrained.

Lastly, the final variation involved validating dataset balancing options as opposed to unbalanced ones. The RandomUnderSampler and SMOTE methods were explored for balancing. The undersampling technique may be less effective than using an unbalanced dataset because many images would be lost. However, the use of oversampling is expected to yield significantly better results than the previous tests.

4.2 Model accuracy

As can be observed in the Table IV, the dependence of the convolutional neural networks on the increased amount of data is evident. Model 2 performed better even with artificially generated masks because it significantly increased the number of images with this feature. Furthermore, in all cases the SMOTE technique was crucial to obtain satisfactory results due to the limitations of the dataset. It can be seen that not using a technique to oversample the data resulted in unsatisfactory results, so we chose to present only the best result for the dataset with artificial data. As the models with SMOTE were the only ones that presented satisfactory results, only them will be considered in the following discussions.

Table IV. Results of training the datasets with real and simulated masks.

Dataset	**Balancing**	**Accuracy**	**Total images**	**Runtime**
Real	-	82,75%	6898	12h44
Real	RandomUnderSampler	78,01%	4230	6h10
Real	SMOTE	91,94%	9240	13h56
Simulated Mask	SMOTE	93,54%	11160	16h13

Table V shows a low value in the standard deviation of the test set accuracy for five different runs. It is possible to perceive the complexity of the problem of image classification with a high number of classes, but it is evident that the model got most of the people's names right.

Table V. All results in the datasets with SMOTE of real and simulated masks.

Dataset	**Accuracies for test dataset**					**Average**	**Standard Deviation**
Real	90,99%	91,29%	91,39%	91,77%	**91,94%**	91,47%	0,38%
Máscara Simulada	92,60%	92,65%	93,23%	93,28%	**93,54%**	93,06%	0.41%

4.3 Training Analysis

In Figure 2 and Figure 3 it is possible to observe sharp drops in accuracy for the test data. These drops occur due to the action of the dropout in deactivating some neurons in the hidden layer, emphasizing neurons that can be better tuned. In this way, over the course of training the network will be able to generalize better if all neurons have less dependency on each other. Another observation is the effect of the reduced learning rate that occurred at epochs 721 and 468 for Figure 2 and Figure 3 respectively. After a few epochs the network could not converge until the learning rate reduction occurred.

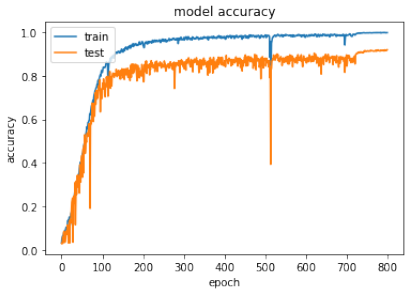

Fig. 2. Accuracy for real images w/ SMOTE.

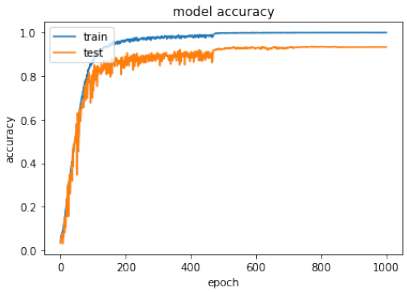

Fig. 3. Accuracy for artificial images w/ SMOTE.

4.4 Validation of results

As the RMFD dataset was initially designed for real-time recognition, a Python script was created to validate images and videos external to the dataset. Its operation consists of extracting faces from images or videos using the MTCNN library and drawing the classified region along with its classification using the OpenCV library. With this test, in videos, it is evident that although the model is able to classify most cases, it also makes errors mainly at changes in angulation or variations in illumination. In Figure 4 and Figure 5 these tests can be observed.

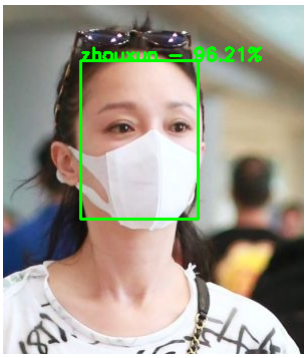

Fig. 4. Zhou Xun's face detection with the simulated mask model.

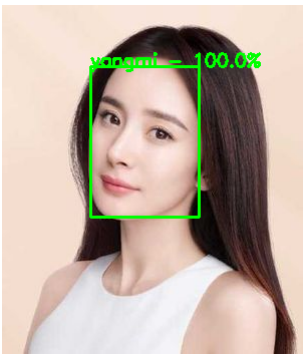

Fig. 5. Yang Mi face detection with the simulated mask model.

4.5 Comparison with other results

As the proposed dataset was created during the pandemic, most of the works aim to identify the presence of masks in images. For this reason, the work by Hariri [Hariri 2022] was the only one found that proposes to classify multiclasses from the RMFD dataset. The main difference between the models is in the number of classes, as the current work classifies 30 classes, while Hariri classifies 50, 60, 70 and 100 classes.

This work chose to classify a smaller number of classes due to the limitation of the dataset in number of images of masked people. While Hariri's work gets around this by aligning the eyes with the horizontal borders and then cropping the image so that the mask is removed before training. The approach proposed by Hariri is interesting but increases the cost of applying the model in the real world, since it requires preprocessing before classification, which may make the identification of people through videos unfeasible.

As shown in Table VI, better results were obtained in both models (with real and simulated masks), with the simulated mask dataset having the highest accuracy. Another point to consider is that Hariri's work shows that increasing from 50 to 60 classes brought better results, and then this accuracy decreased again, suggesting that the choice of classes and the quality of the images had a greater impact than the increase in classes.

Table VI. Comparison of results between models with real and simulated masks.

Methodology	Number of Classes	Accuracy (Real)	Accuracy (Simulated)
Present study	30	**91,94%**	**93,54%**
Hariri [Hariri 2022]	50	91,0%	83,5%
Hariri [Hariri 2022]	60	91,3%	84,7%
Hariri [Hariri 2022]	70	90,1%	88,9%
Hariri [Hariri 2022]	100	89,8%	88,5%

5. CONCLUSIONS AND FUTURE WORKS

This paper presented an work applying convolutional neural networks with transfer learning to recognize masked individuals. It was notable the importance of preprocessing the data for any neural network problem, was emphasized the dependence on a large amount of data for CNNs to differentiate a large number of classes and generalize to real-world scenarios, and how the use of data augmentation and balancing techniques was essential.

In a deep convolutional network problem with multiclasses, the obtained results were able to classify the vast majority of images and be consistent with the current state of the art. However, when applied to a real security problem, the presence of masks can still be considered a problem to be further explored.

Given the big effort to find an optimal neural architecture to solve AI problems, as future work it is intended to apply the AutoML (Automated Machine Learning) in this work in order to search automatically a similar or better neural architecture, reducing the human effort and dependency.

REFERENCES

Anwar, A. and Raychowdhury, A. Masked face recognition for secure authentication. *arXiv preprint arXiv:2008.11104*, 2020.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* vol. 16, pp. 321–357, 2002.

Hariri, W. Efficient masked face recognition method during the covid-19 pandemic. *Signal, image and video processing* 16 (3): 605–612, 2022.

Ngan, M. L., Grother, P. J., and Hanaoka, K. K. Ongoing face recognition vendor test (frvt) part 6a: Face recognition accuracy with masks using pre-covid-19 algorithms, 2020.

Noe, I. T. Redes neurais convolucionais aplicadas ao reconhecimento facial em indivíduos com máscara. https://github.com/ItaloTN10/TCC2, 2021.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22 (10): 1345–1359, 2010.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv*, 2014.

Vincent Chi-Chung Cheng, e. a. The role of community-wide wearing of face mask for control of coronavirus disease 2019 (covid-19) epidemic due to sars-cov-2. *Journal of Infection* vol. 81.1, pp. 107–114, 2020.

Wang, Z., Huang, B., Wang, G., Yi, P., and Jiang, K. Masked face recognition dataset and application. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2020.