

Master of Science (MS) in Data Science  
Module: ITC6125A1 – Machine Learning & Applications

Instructor: Dimitrios Milioris

Students’ names:

Alkiviadis Kariotis 241735

Konstantinos Margaritis 122451

Term: Spring Term 2023  
Type: Group Term Project

Submission Date: Monday, July 17 at noon

Words: 3104

Table of Contents

[Introduction 3](#_Toc140442258)

[Main Body 4](#_Toc140442259)

[Foundational Theory 4](#_Toc140442260)

[Keras 4](#_Toc140442261)

[CV 4](#_Toc140442262)

[CNN 4](#_Toc140442263)

[Overfitting 6](#_Toc140442264)

[Image Data Augmentation 6](#_Toc140442265)

[CNN for Mask Detection 7](#_Toc140442266)

[Description 7](#_Toc140442267)

[Defining the Model 7](#_Toc140442268)

[Data Preparation 8](#_Toc140442269)

[Defining Callbacks 8](#_Toc140442270)

[Training the Model 8](#_Toc140442271)

[CNN for Face Detection 10](#_Toc140442272)

[Data Preparation 10](#_Toc140442273)

[Model Building and Training 10](#_Toc140442274)

[Real-time mask detection 11](#_Toc140442275)

[Conclusion 12](#_Toc140442276)

[References 13](#_Toc140442277)

# Introduction

The COVID-19 pandemic's impact in the daily lives of people was significant. It started in the winter of 2019 and the place of Origin was Wu han, China (Springer, 2022). The virus spread rapidly around the globe, causing millions of deaths as a result. This gave rise to new measure's like social distancing and the use of face masks. These new measures gave rise to the need of new techniques of monitoring the compliance of people to the new measures.

For the case of the face mask, techniques that were developed for computer vision around object detection and facial recognition can be utilized in order for automated systems to monitor if specific persons are wearing a mask (Research Dive, n.d.). Some possible uses of the specific technology can be in airports, for the monitoring of travelers compliance, governmental organizations and also, hospitals, in order to monitor both staff and visitor compliance to the mask measure (Research Dive, n.d.).

The specific research aims at developing a face mask detector system based on the CNN architectures (Springer, 2022). Two different models were trained on a variety of photographs, one in order for the system to detect if a face exists in front of the camera and a second model which monitors the existence or not of a face mask (Springer, 2022). The outcome of the development was plausible and more analysis of the matter will continue on the other parts of the research (Springer, 2022).

# Main Body

## Foundational Theory

### Keras

According to Ketkar (2017),Keras is a python library which gives emphasis to modularity. By this we mean that it allows the researcher to combine discrete components in various ways in order to create deep learning models. It follows a minimalistic approach which emphasizes simplicity and readability. Its structure provides extensibility which makes it easier for a researcher to integrate now components easily. Keras relies solely on python, something that eliminates the need for custom formats and separate model files. In general, it can be said that it provides more overall simplicity for deep learning tasks compared to other frameworks like Theano.

### CV

According to GeeksforGeeks (n.d.), Opencv is an open source library which is central in computer vision and specific, in tasks that require the identification (face identification, handwriting identification, object identification in general) in images and videos. It is considered crucial in real time systems that execute such tasks.

### CNN

According to IBM (n.d.), CNN’s are nowadays commonly used for computer vision and classification tasks. Convolutional neural networks are a class of neural networks which are widely used for classification and computer vision tasks. CNN'S in general have 3 types of layers which are the following: Convolutional layer, Pooling layer and fully connected layer.

The convolutional layer is the core part of the network, as it is where most of the computation will take place. In general, it requires some specific parts like input data, a filter and a feature map (Kattenborn et al., 2021). CNNS can have hierarchical structures when another convolution layer follows the initial one. With this, the next layers become capable of seeing the pixels from the prior layers.

The pooling layer is responsible for the dimensionality reduction, the reduction in the parameter number of the input. The pooling operation filters across the input and although it seems similar with the convolutional layer, here the filter does not have any weights (Kattenborn et al., 2021). In the specific case we have the utilization of an aggregation function from the kernel to the values that are within the receptive field. In general, we have two types of pooling which are the Average pooling and the Max pooling.

According to Analytics India Magazine (n.d.), as for the Max pooling layer, the pixel with the maximum value will be selected to be sent to the output array. It is considered a process by which the most significant information in terms of amplitude is extracted by the kernel (Giusti et al., 2013). As an example we can think that if we apply max pooling on a 4x4 channel and we utilize a 2x2 kernel and a stride of 2, the maximum value within each set of 2x2 will be chosen, therefore the important information will not be lost. Max pooling is capable of filtering out unwanted information and at the same time, retaining the important features. On the other hand Average pooling, what it does is that sends to the output array the average value within the receptive field (Giusti et al., 2013).

As for the Dense Layers, it can be described as the fully connected layer is a layer in which every node of the output layer is connected to a node in the previous layer. Here is where the task of classification is performed with the use of the features which were extracted in the previous layers (Kattenborn et al., 2021). In terms of activation function, although the RELU is normally used in convolutional and pooling layers, here we have the usage of the softmax for the classification of inputs, based on a probability which ranges from 0 to 1 (Schmidt-Hieber, 2020).In general and as mentioned also above, the specific class of neural nets are considered better for tasks that are utilizing speech, audio signals and images.

In order for the associations between pairs of features in data points to be captured, dense layers are generally utilized. They get their name because of the connections that they manage to establish between each feature between the previous and a subsequent layer. As a result, the total connections between the layers are n1xn2 (Sperl et al., 2020).

Dropout refers to a technique which aims at the reduction of interconnection among features. This is achieved with the random drop of weights within a certain probability (Phaisangittisagul, 2016). This process results in the prevention of over reliance on specific features of the model, something that provides learning which is more robust (Phaisangittisagul, 2016). It is possible for dropout to be utilized between any layers and it halts the update of weights that are in association with connections that have been dropped. It does possess any associated weights because of the purpose that it serves.

Flatten layers are utilized to convert into a linear format  layer output which is multidimensional in order to be fed in a dense layer. Its numpy equivalent is the function numpy.ravel. The flatten layer produces an output that will be passed to an MLP in order to be utilized in classification-like tasks (Jin et al., 2014). The similarity that a flatten layer and a dense layer have is that they do not possess weights because their purpose is only the transformation of multidimensional data.

### Overfitting

When the model developed tends to become tailored to the training data, then it is most probable that we face overfitting (Ying, 2019). This happens in general when the model is not learning from the generalized underlying patterns and they tend to capture noise or irrelevant patterns. Such models tend to not perform well when tested on new and unseen data. In order to detect overfitting the researcher may use a model evaluation technique likt K-fold cross validation (Ying, 2019).

To prevent the presence of overfitting strategies like early stopping, which stops the training before the model becomes over-specialized, can be utilized (Ying, 2019). The elimination of irrelevant features  with the use of feature selection or pruning is also another valid strategy. Another way is through various regularization techniques that can be utilized in order to reduce the influence of irrelevant features. Finally, we can proceed with data augmentation which means the introduction of variation in the training data in order to prevent over reliance on specific patterns (Amazon Web Services, n.d.).

### Image Data Augmentation

Image data augmentation is used in computer vision tasks in order to introduce variations of images of an existing dataset. This can be done with transformations and with  pixel value manipulation in order for "new"augmented images to appear (Shorten & Khoshgoftaar, 2019). These images can enhance the overall machine learning algorithm by introducing diversity. This technique can be used as a work around when the data set is limited by lowering The time needed for the extraction and detection of new data while improving the models performance. Data augmentation also mitigates overfitting, which tends to occur in small datasets and the possibility for the model to become over-specialized is high (Shorten & Khoshgoftaar, 2019). The specific process is commonly used in tasks that revolve around object detection, segmentation and classification in a CVOps pipeline. The general goal is to provide the model with diverse scenarios in order for its ability to recognize patterns in new data to be amplified. Some techniques for image augmentation include position manipulation, which alters the position of an image with the use of scaling, rotation or flipping and thus, different angles and perspectives are generated. Color manipulations techniques revolve around the adjustment of contrast,brightness, hue and saturation in order to provide variation in lighting. Cropping, blurring and sharpening are image manipulation techniques and revolve around focus, clarity and non - perfect data representation (Shorten & Khoshgoftaar, 2019). These techniques as was mentioned above are increasing the dataset something that helps the training process and amplify the overall performance. In general, although the techniques are many, it is considered complex for a researcher to select and apply the appropriate methods depending on the problem.

## CNN for Mask Detection

### Description

In this python code snippet, we used the Kera’s library in order to create, train and also evaluate a Convolutional Neural Network (CNN) towards image classification tasks. Firstly, we started by importing the necessary libraries to do our processes. Indicative some of them are matplotlib aiming to visualize our training process, OpenCV for operations regarding image processing, numpy for numerical type operations and imutils for the transformation of images.

### Defining the Model

Through a function that we called create\_model, we started defining by using Keras our Sequential model. In more detail, the architecture of our model is:

* In order to extract the features from the images that we have assigned as an input, we created two Conv2D layers, consisting of one hundred filters meaning that our model can learn that many different kinds of features from the input, we used ‘relu’ as an activation function, which helps at combating the vanishing gradient problem, and a kernel size of three times three, each one of them followed by a layer of MaxPooling2D. These layers are used to reduce step by step the spatial dimensions of the input volume, achieving that way minimization of the computational complexity and prevent a bit overfitting.
* Before we feed the outputs in the next layers, we needed to convert the 3D feature maps to 1D feature vectors, so we used a Flatten layer.
* Furthermore, as to help the model prevent overfitting we continued with a Dropout Layer, with a rate of dropout of 0.5, which randomly sets a fraction of inputs units to zero after each update during training phase of the model.
* Afterwards, to learn the nonlinear combinations of the features we proceeded with a Dense layer with fifty units and the ‘relu’ as an activation function.
* Finally, we put a Dense layer, in order to output the class probabilities, consisting of two units and ‘softmax’ as an activation function, which also ensures the probabilities to be summing up to 1.

A diagram of a diagram of a variety of cubes

Description automatically generated

To compile our model, we used the optimizer called Adam, because this a binary classification model we went on with binary cross entropy as loss function, and as an evaluation metric the accuracy.

### Data Preparation

Then we defined a function that called create\_data\_generators that will be responsible for the data generators creation that will be used for the training and the validation dataset. We have respectively assigned the correct paths to find images that we are going to use either for the train and for the test dataset. Then we use a class called ImageDataGenerator in order to not only convert automatically our images into batches of preprocessed tensors but also to perform augmentation of our images of the training data. The augmentation that this included were many, for example rescaling or rotating our images, shifting their width or height, flipping them, zooming in them and more.

### Defining Callbacks

Then we created a function called create\_callbacks in order to define several callbacks in our snippet, in order to ensure the appropriate training:

* In the case that the validation loss was improved during the training, we have established a callback called model checkpoint in order to store our weights of our model at certain intervals.
* Then if the validation loss has not been improving for the last three epochs that our model has performed, in order to prevent overfitting, we set an early stopping callback.
* Finally, to help our model reach its loss function to a global minimum, we established a ReduceLROnPlateau callback as to when our model’s validation loss will plateau then the learning rate will be reduced.

### Training the Model

So, in the end we have created a main function that triggers sequentially all the functions into our python code snippet. The training of our model starts with the Keras function named fit\_generator function and we have set it to train for thirty epochs. As inputs we are using the previous defined functions that we have aforementioned which also set up the callbacks. Then as the training of the model is being executed, we implemented the plot\_history function to the end to plot the model’s training and validation accuracy and loss during each epoch that was being performed. This was found to be very useful at the end because we were able to see the plot of the training and identify issues that we may have regarding overfitting and underfitting.

Here is how our model performed on the training and where the callbacks stopped at the specific depicted epochs:

A screenshot of a computer program

Description automatically generated

A graph of a model accuracy

Description automatically generatedThis is the plotted history through the epochs that our model performed on accuracy and loss both on the train and on the test dataset:

## CNN for Face Detection

As a brief summary, in this specific python script, we performed a complete machine learning pipeline for a binary classification task which had to do with whether a face was located in an image or not, starting from data loading and preprocessing them and ending to model training and saving. We firstly started by importing all the necessary libraries that we were going to need for our actions. Then through the Kera’s API of TensorFlow, we trained a Convolutional Neural Network to perform a binary classification, which had to do with whether there was a face in the image. Concerning the dataset what we did was that we took some images from our original train folder and cropped them one by one so as only the face would remain. Then though open-source dataset we obtained images containing random objects like cars, bicycles, flowers etc.

### Data Preparation

We started by defining a function that would load and preprocess the images from our given path folder. So, after we read all the image files, we converted them to grayscale and resized them to 64\*64 pixel size, assigning them at the same time the labels based on the folder that the images would come from, meaning 1 if they came from the positive folder (containing human face) or 0 if they came from the negative folder (collection of random images with no humans depicted in them). We then combined the image data and their labels into two separate lists, which we split into training and test sets, and more specifically using 80% for training and 20% for testing. Also, we did augment our images by shifting them, rotating them or even change their RGB color combinations in order to produce more than seven thousand images from nearly 50 original that we have cropped and aligned around the faces. Afterwards, the image data is being converted to array of NumPy type and at the same time normalized by dividing them by 255, which was the maximum pixel value, so as to scale the pixel values between the numbers of 0 and 1. Finally, the image data are reshaped in order match the shape of the necessary input of the network and labels are also transformed into arrays of NumPy type.

### Model Building and Training

Through the Keras Sequential API we then defined our CNN model. The model that we build has two convolutional layers, in which we use the ReLU activation function and then it is followed by max pooling, then a flattening layer and then two dense layers. In the final stage we have a layer which used as an activation function a sigmoid one, which is considered suitable for binary classification. We compiled our model with the Adam optimizer, binary cross entropy loss and as evaluation metric we used accuracy. Finally, we trained our model with a batch size of thirty-two, for ten epochs, and for the validation part we used the test set. After executing and the training is done, we went ahead and saved our model.

Metrics for the loss and the accuracy of the model:



## Real-time mask detection

In this python code script, we started loading our pretrained model on the face detector and on the mast detector. Because we were unable to use libraries like Labelme and labelIMG we couldn’t annotate that’s why in this specific code snippet we show an indication whether out model detects a face or not and we use cascade in order to activate our mask detector model. So, by using a webcam feed and at the same time leveraging the OpenCV library for video capture and for the cascade for face detection we were able to test our models regarding their efficiency. So, we did the following:

* We started a mirror video capture to take video frames from our webcam in order to ensure models functionality.
* We resize the captured frames by a scaling factor of 4, to have faster processing because of the smaller images that were transmitted.
* Our face detector models are being used to detect whether a face is being seen into the live feed camera frame. Then through the Haar Cascade classifier, we detect the face that is present in a good speed and accuracy.
* So, though this we achieve getting back a smaller rectangle containing the face frame through the live detection and then our model of mask detection comes to the foreground.
* The smaller frame passed through our pre trained model to make the prediction whether a mask is worn or not. Using the softmax function the highest probability is being picked and been finally displayed.
* For visual reasons through our script, we draw boxes around faces in the frame transmitted. These frames are changing regarding the state of the face and the label in the corner changes whether a face is not depicted in the screen accordingly.
* This is all in an endless loop until the user presses the ESC button to exit the process and break the loop.

# Conclusion

Concluding**,** we could handle some certain missing error scenarios better, meaning to ensure empty or problematic directories. Then, we could make the hyperparameters more flexible, as to accept arguments through command lines to make the script more flexible. So, by fixing the aforementioned things and by making feasible the task of annotation we could create our own object detector in order to not need the Haar Cascade and do the entire process from scratch. Also, another thing for further research would be to enhance more the dataset with images of people with glasses or hats so it could detect more different and ‘unusual’ cases.

# References

Amazon Web Services. (n.d.). Overfitting. Retrieved from <https://aws.amazon.com/what-is/overfitting/>

Analytics India Magazine. (n.d.). Max Pooling in Convolutional Neural Network and Its Features. Retrieved from <https://analyticsindiamag.com/max-pooling-in-convolutional-neural-network-and-its-features/>

Dense, Conv1D/2D, Dropout, Flatten, and All the Other Layers. (n.d.). Retrieved from <https://datascience.stackexchange.com/questions/44124/when-to-use-dense-conv1-2d-dropout-flatten-and-all-the-other-layers>

Giusti, A., Ciresan, D., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2013). *Fast image scanning with deep max-pooling convolutional neural networks*. <https://doi.org/10.1109/icip.2013.6738831>

Hussain, H. M. (2022). Face mask detection using convolutional neural networks. Journal of Information Science and Engineering, 38(2), 349-360. doi:10.6688/JISE.202203\_38(2).0004

IBM. (n.d.). Convolutional Neural Networks (CNNs). Retrieved from <https://www.ibm.com/topics/convolutional-neural-networks>

Image Data Augmentation. (n.d.). Retrieved from <https://www.tensorflow.org/tutorials/images/data_augmentation>

Jason, S. (2020). Python for Data Analysis. O'Reilly Media. Li, C., Yang, Q., Guo, H., Zhao, B., & Liu, D. (2022). Deep learning-based face mask detection during the COVID-19 pandemic. Soft Computing, 26(9), 6863-6873. doi:10.1007/s00500-022-07700-w

Jin, J., Dundar, A., & Culurciello, E. (2014). Flattened convolutional neural networks for feedforward acceleration. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1412.5474>

Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *Isprs Journal of Photogrammetry and Remote Sensing*, *173*, 24–49. <https://doi.org/10.1016/j.isprsjprs.2020.12.010>

Ketkar, N. (2017). Introduction to Keras. In Deep Learning with Python. Apress, Berkeley, CA. <https://doi.org/10.1007/978-1-4842-2766-4_7>

Max Pooling in Convolutional Neural Network and Its Features. (n.d.). Retrieved from <https://analyticsindiamag.com/max-pooling-in-convolutional-neural-network-and-its-features/>

MDPI. (2023). A survey on computer vision-based face mask detection systems. Computers, 11(2), 107. doi: 10.3390/computers11020107

OpenCV Overview. (n.d.). Retrieved from <https://www.geeksforgeeks.org/opencv-overview/>

Phaisangittisagul, E. (2016). *An analysis of the regularization between L2 and dropout in single hidden layer neural network*. <https://doi.org/10.1109/isms.2016.14>

Research Dive. (n.d.). How does the face mask detection system act as a beneficial device in the epoch of the COVID-19 pandemic? Retrieved from <https://www.researchdive.com/blog/how-does-the-face-mask-detection-system-act-as-a-beneficial-device-in-the-epoch-of-covid-19-pandemic>

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics*, *48*(4). <https://doi.org/10.1214/19-aos1875>

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). <https://doi.org/10.1186/s40537-019-0197-0>

Sperl, P., Kao, C., Chen, P., Lei, X., & Böttinger, K. (2020). *DLA: Dense-Layer-Analysis for Adversarial Example Detection*. <https://doi.org/10.1109/eurosp48549.2020.00021>

Springer. (2022). COVID-19 pandemic and the importance of mask detection systems. Soft Computing, 26(7), 6499–6507. doi: 10.1007/s00500-022-07700-w

Ying, X. (2019). An Overview of Overfitting and its Solutions. *Journal of Physics*, *1168*, 022022. <https://doi.org/10.1088/1742-6596/1168/2/022022>