# Surgical mask detection – Report

## -Discriminate between utterances with and without surgical mask-

## Task

Training a model for surgical mask detection in the context of the COVID-19. It is a binary classification task in which an audio file must be labeled as without mask (label 0) or with mask (label 1). The training set consists of 8000 samples (audio files with wav extension), the validation set (1000 audio files) and test set (3000 samples).

## Data preprocessing

In order to process the audio files efficiently and extract only the necessary elements from them, I read the data from the file with librosa which is a python package for music and audio analysis. This library returns a tuple with the signal and sample rate (samples that we have in a second). The audio file will be automatically resampled to the given rate (default for sr parameter is 22050) [1].

```
signal, sample_rate = librosa.load(filepath, res_type = 'kaiser_fast')
```

To extract the features from audio I used *librosa.feature, spectograms* and *MFCCs (*Mel Frequency Cepstral Coefficients*). A* spectogram is a visual representation of the frequencies of a signal in time. The MFCCs of a signal are small set of features which concisely describe the overall shape of a spectral envelope.[2] The MFCCs coefficients contain information about the rate changes in the different spectrum bands.

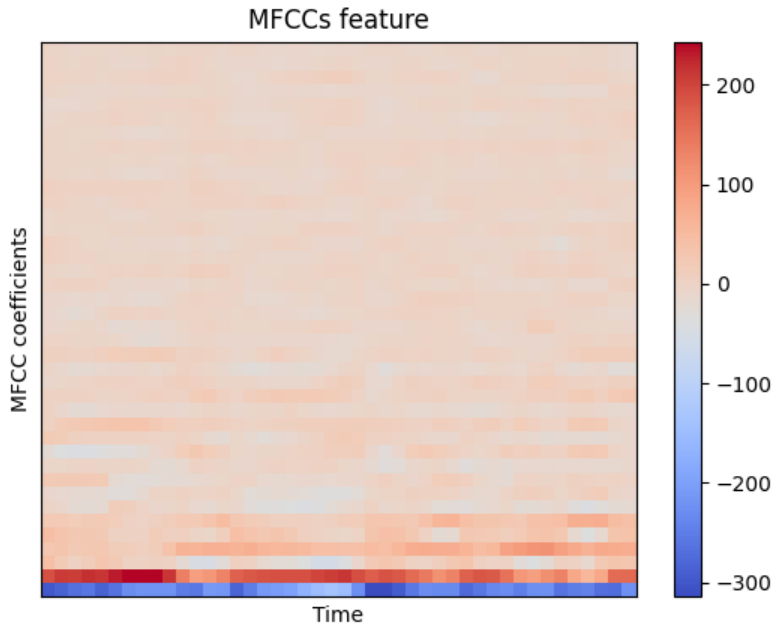
"The most prevalent and dominant method used to extract spectral features is calculating Mel-Frequency-Cepstral-Coefficients. MFCCs are one of the most popular feature extraction techniques used in speech recognition based on the human ear scale. This is a representation of the

real cepstral of a windowed short-time signal derived from the Fast Fourier Transform of that signal. The speech signal is first divided into time frames consisting of an arbitrary number of banks. For each speech frame, a set of MFCC is computed." [3] That's why I chose to use this way to extract the features from audio files given the nature of those records.

The signature of this method: librosa.feature.mfcc(y = None, sr = 22050, S = None, n_mfcc = 20, dct_type = 2, norm = 'ortho', lifter = 0). Of all of these parameters I used only three of them and they are y, sr and n_mfcc(the number of coefficients we extract from the file).

```
mfcc_feature = librosa.feature.mfcc(y = signal, sr = sample_rate, n_mfcc = 40)
```

The representation of mfcc feature for the first audio file from our train set:




## Description of the Machine Learning approach

I have tried various models such as Support Vector Classification, Logistic Regression, Multilayer Perceptron, Keras. But just SVM and Neural Network from keras brought me the highest scores.

**Support Vector Classification** – The objective of SVM is to determine a hyperplane of dimension N(number of features) that can classify the data points. We need to find a plan that has

the maximum margin(C parameter). When we maximizing the margin distance we can classify these points with more confidence.

I kept default parameters values for one of the submissions and then I changed the values of the C and gamma parameters trying to reach an optimal result. C is the cost of misclassifications. A large C gives low bias and high variace. Gamma determines the distance a single data sample exerts influence.

I trained the model on train set and test it on validation data in all of my model.

| Validation Score | C | gamma |
|---|---|---|
| 0.628 | default | default |
| 0.543 | 1000 | 0.01 |
| 0.738(0.5871 kaggle) | 2000 | default |

*Score on kaggle:*

Public score: 0.6311

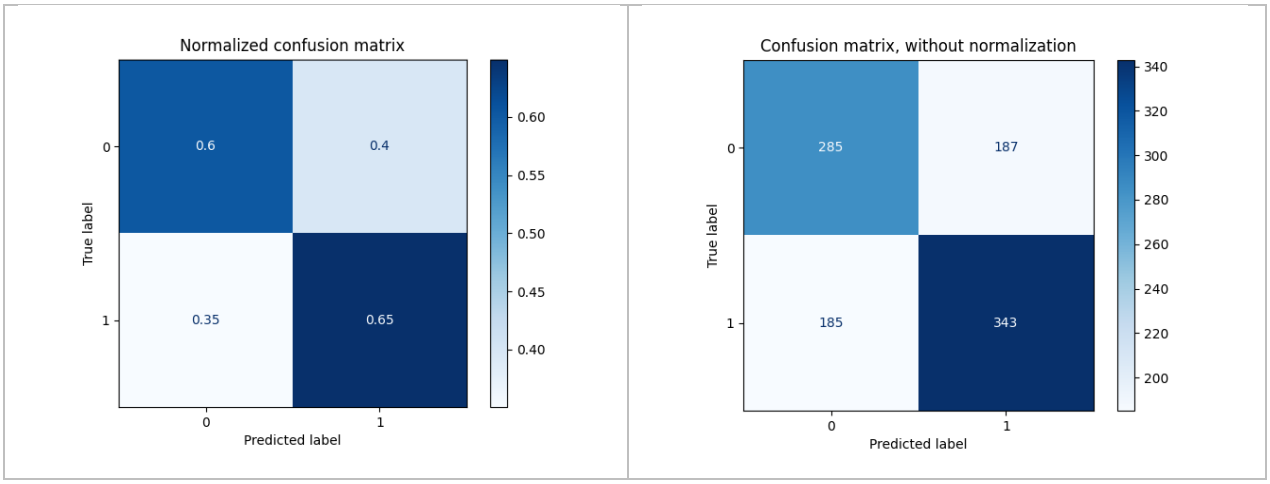
Private score: 0.6419

The precision and recall for default parameters:

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 | 0.6038 | 0.60381 | 0.60510 | 472 |
| 1 | 0.64717 | 0.64962 | 0.64839 | 528 |
| Accuracy | | | 0.6280 | 1000 |

In a classification task, the precision for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive class. Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class[4].

Confusion matrix:




I tried several parameters for C and gamma but the solution that brought me the highest score was the one in which I used the default parameters. While researching I found out that we can use grid search algorithm to find values for C and gamma, but I didn't use it.

## Neural Network from Keras

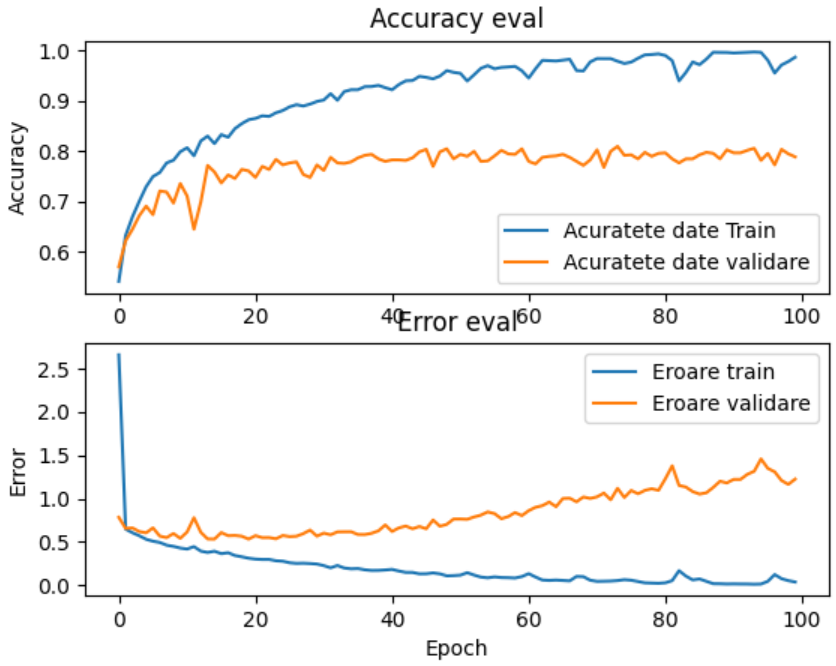
A neural network is a circuit of neurons(nodes). The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. Inputs are modified by a weight and summed and activity functions is a linear combination [5]. These networks consist of an input layer, one or more hidden layers and an output layer. Each layer contains a number of neurons(nodes). The input layer has the number of nodes equal to the number of features extracted from each audio file (n_mfcc parameter) and the output layer contains 2 neurons in our case because this is a binary classification (0/1).

For each layer we need an activation function. For the intermediate layers I used the ReLu activation function because it reduces the vanishing gradient after propagation and has a better convergence. On the last layer I used the softmax activation function. The type of layer used is Dense because the neurons are completely interconnected.

```
model.add(layers.Dense(128, input_shape = (train_data[1],)))
model.add(layers.Dense((128, activation = ‘relu’))
model.add(layers.Dense((128, activation = ‘relu’))
model.add(layers.Dense((2, activation = ‘softmax’))
```

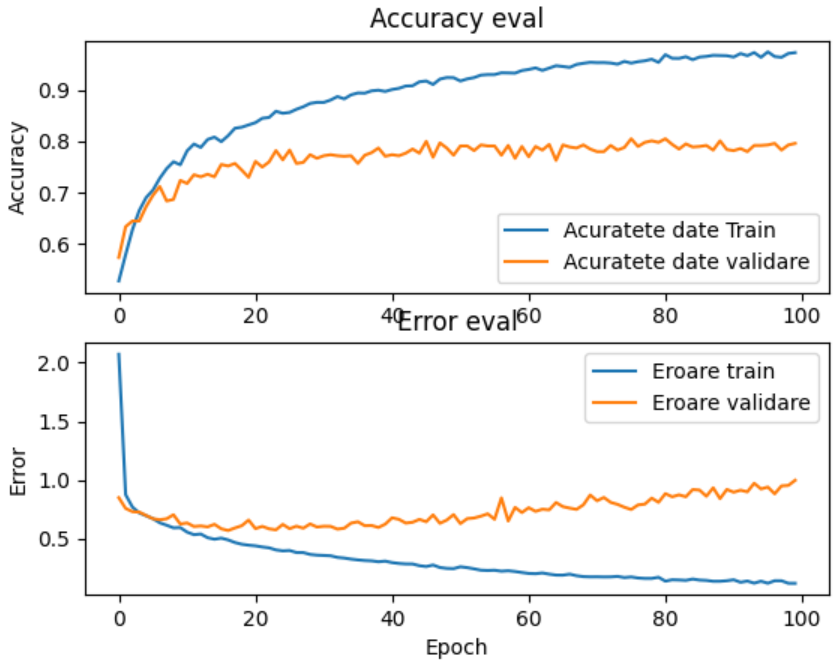
I trained this on 100 epochs and batch_size = 128.

Initially I did not prevent overfitting and the accuracy was 0.78




To prevent overfitting, I chose to use Dropout (randomly drop neurons while training and increase network robustness) and Regularization (minimizes the absolute values of the weights).

Accuracy on validation dataset after dropout and regularization:  0.79




Public score: 0.6255

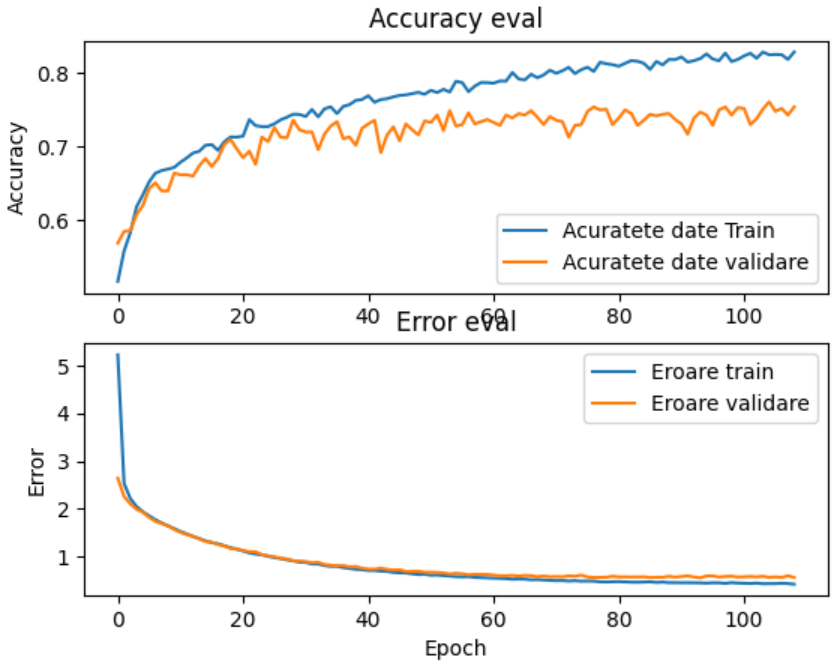
Private score: 0.6235

The second overfitting prevention technique was early stopping. This stops training before starting the overfitting by setting a paticence (delay to the trigger in terms of the number of epochs on which we would like to see no improvement) parameter that monitors the loss values. In this case, we are interesed in saving the model with the best accuracy on the validation dataset and I used for that ModelCheckpoint callback to save the best model observed during training. I saved the model in a .h5 file and then with the help of the load_model function I evaluated the performance of the chosen model on the validation dataset.

Accuracy on validation dataset: 0.76

Public score: 0.5666

Private score: 0.5804




```
early_stopping = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 15, min_delta=0.001)
best_model = ModelCheckpoint(filepath = 'ch_model.h5', monitor = 'val_accuracy', mode = 'max', verbose = 1, save_best_only = True)
```

## *SVM*

```
model = SVC()
model.fit(train_data, train_labels)
predictions = model.predict(validation_data)
```

Neural Networks with Keras:

```
model = Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(train_data.shape[1],)))
```

```
model.add(layers.Dense(128, activation='relu', kernel_regularizer = l2(0.001)))
model.add(GaussianDropout(0.1))
model.add(layers.Dense(128, activation='relu', kernel_regularizer = l2(0.001)))
model.add(GaussianDropout(0.1))
model.add(layers.Dense(2, activation='softmax'))
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
history = model.fit(train_data, train_labels, validation_data = (validation_data, validation_labels), epochs = 100, batch_size = 128)
```

Bibliography


1. https://librosa.github.io/librosa/generated/librosa.core.load.html, last edited on 24 May 2020
2. https://musicinformationretrieval.com/mfcc.html, last edited on 24 May 2020
3. **Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition**; Namrata Dave, International Journal for Advance Research in Enginerring And Technology, Volume 1, Issuse VI, July 2013
4. https://en.wikipedia.org/wiki/Precision_and_recall, last edited on 24 May 2020, at 19:54(UTC)
5. https://en.wikipedia.org/wiki/Neural_network, last edited on 24 May 2020, at 20:45(UTC)
6. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/ (for early stopping and model checkpoint)
7. https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/
8. Audio Music Classification Using Support Vector Machine; Cyril Laurier