Speech Commands Recognition

Davide Dell’Orto, Matteo Lanzillotti, Simone Tranquillo

Audio Classification

Our goal is to be able to **recognize** and **classify** specific spoken “commands”, among a variety of other words, using state-of-the-art deep learning techniques.

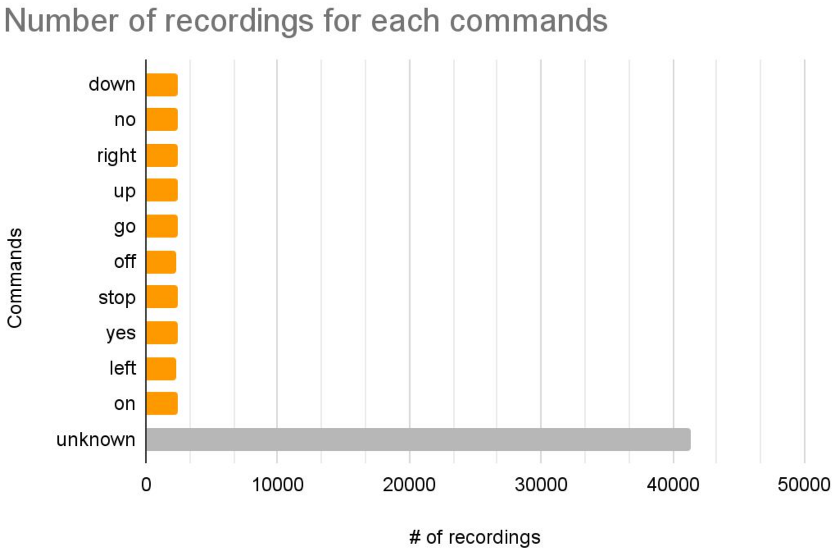
This type of task falls under the **audio classification** category and is employed in industries across different domains like Natural Language classification, Environment sound classification, text-to-speech algorithms ecc...

Most common issues with audio classification mainly relies in the data preparation aspect, as it involves working with **raw audio data** that need to be correctly addressed in order to be useful in deep learning environments.

Data overview

The dataset

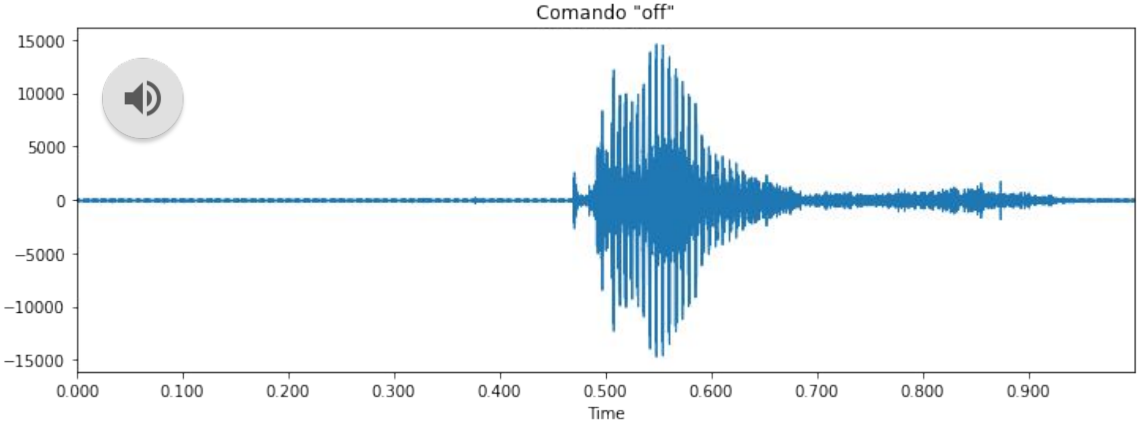
- 65.000 labelled audio files (.WAV) spoken by different person
- 10 speech commands classes
- 20 different words grouped under the “Unknown” class



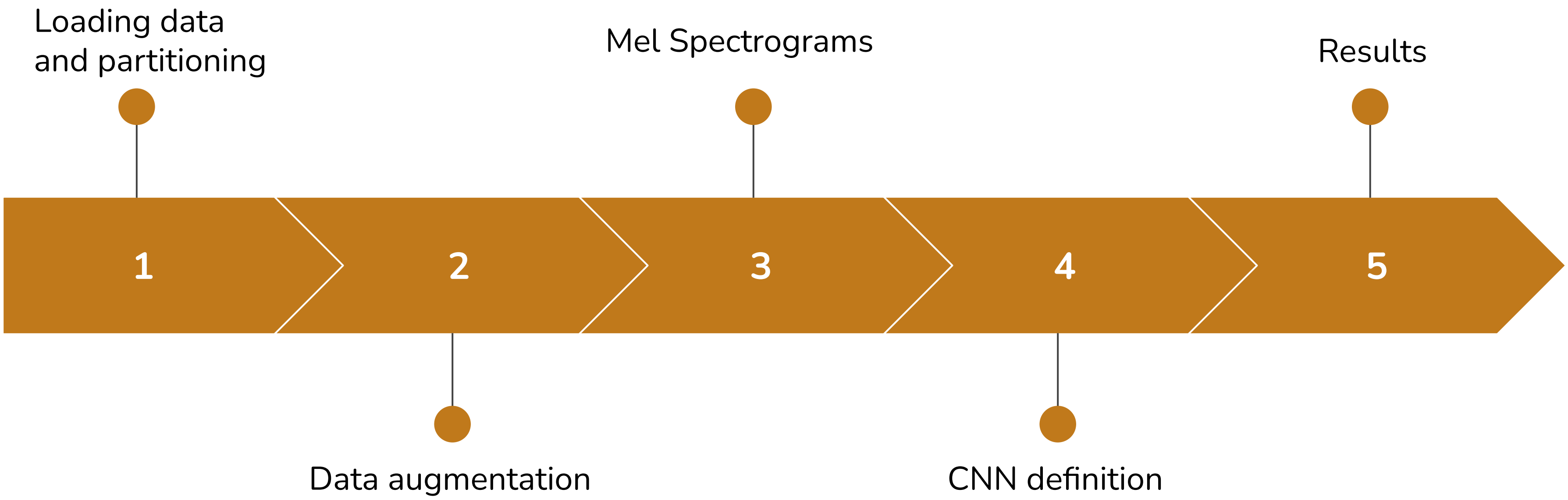
Data overview

The audio file

- one second long
- 16.000 hz sample rate
- mono
- noise-free
- clean recording



Course of action



1 Loading data and partitioning

2 Data augmentation

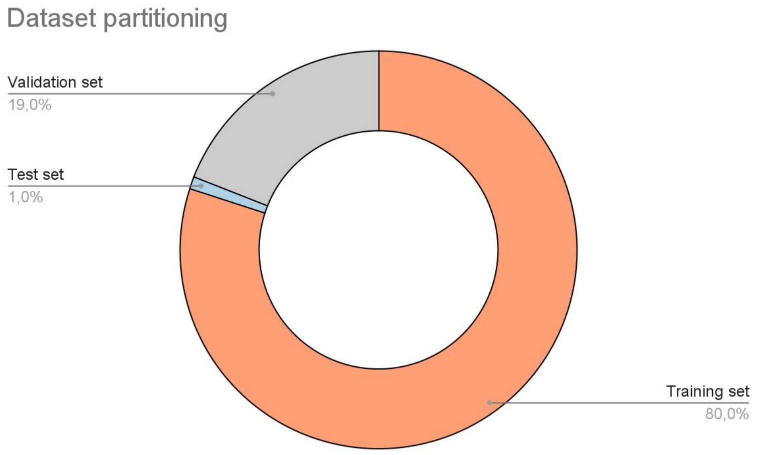
3 Mel Spectrograms

4 CNN definition

5 Results

1- Loading data and partitioning

After downloading and extracting the dataset we have proceeded with **partitioning** it, allocating **80%** of the data to the **training set**, 19% to the validation set and the remaining 1% to the test set.



The data is available for download at:
http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

2- Data augmentation

Data augmentation in deep learning has been proven to **increase models performance**, **reduce overfitting** and **increase models accuracy** prediction. The same is, of course, true with audio data.

Augmentation techniques can be applied on **raw audio** or on specific elements derived from them.

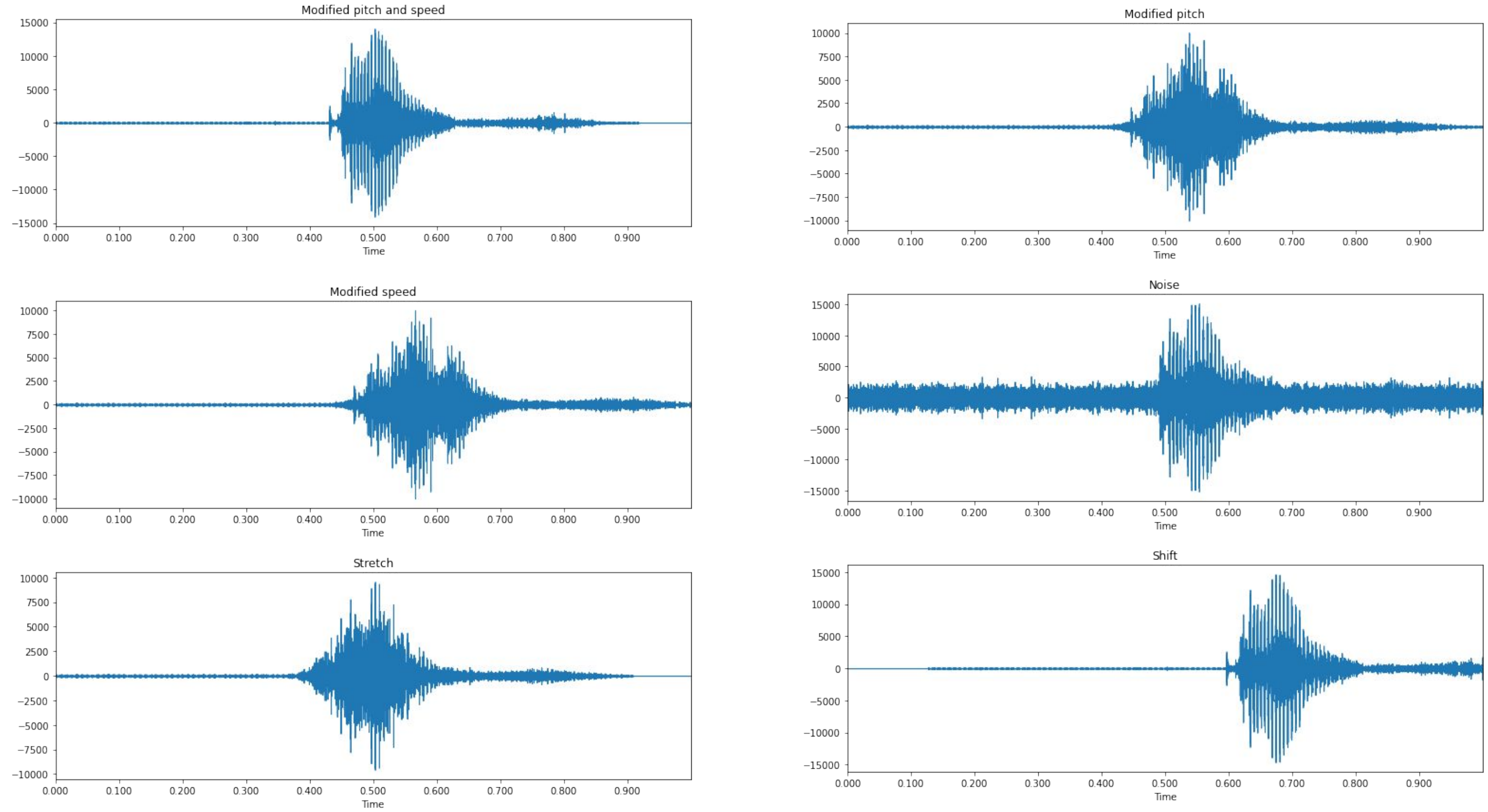
In our case we focused on the former.

We employed, to each audio data in the training set, **1 out of 6 possible transformation,** randomly chosen.

Lists of transformations randomly applied

1. Pitch and speed alterations
2. Pitch only alteration
3. Speed only alteration
4. Noise added
5. Audio stretching
6. Audio shifting

2- Data augmentation - Transformations



Modified pitch and speed

Modified pitch

Modified speed

Noise

Stretch

Shift

3- Spectrograms

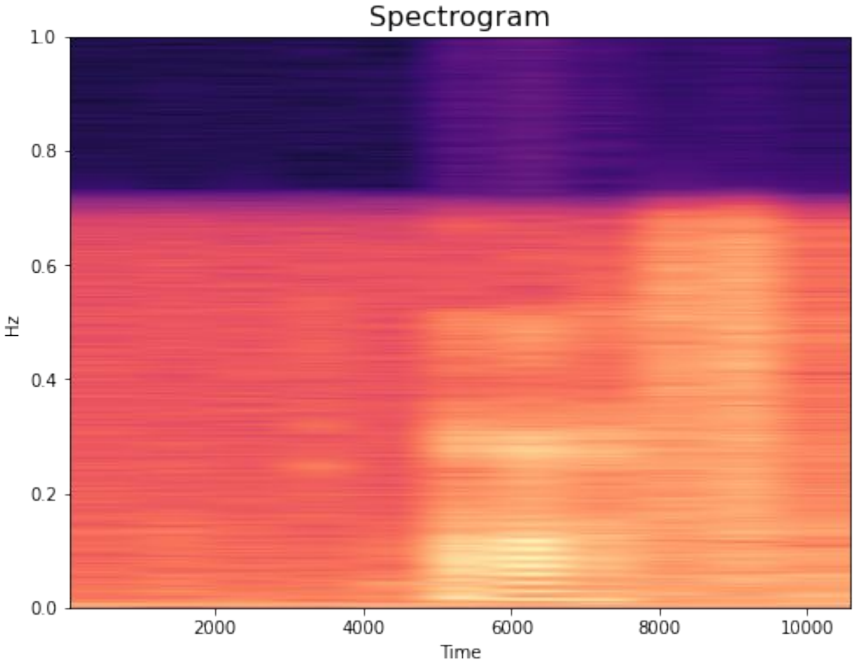
Deep learning models **rarely** take raw audio directly as input.

It is, instead, common practice to convert the audio into a **spectrogram**, a visual representation of the spectrum of frequencies as it varies with time.

Since spectrograms are **images**, they are well suited to being input to CNN-based architectures.

A spectrogram plots **Frequency** (y-axis) **vs Time** (x-axis) and uses different colors to indicate the **Amplitude** of each frequency.

The brighter the color the higher the intensity of the signal.



3- Mel Spectrograms

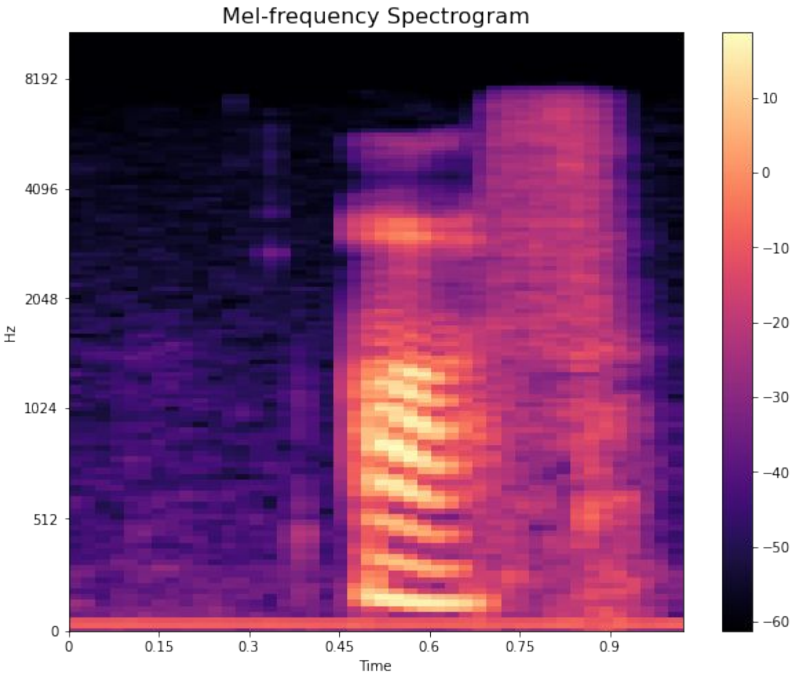
However, humans perceive frequencies and amplitude in a **logarithmic scale** rather than a linear one; these are called Mel scale and Decibel scale respectively.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

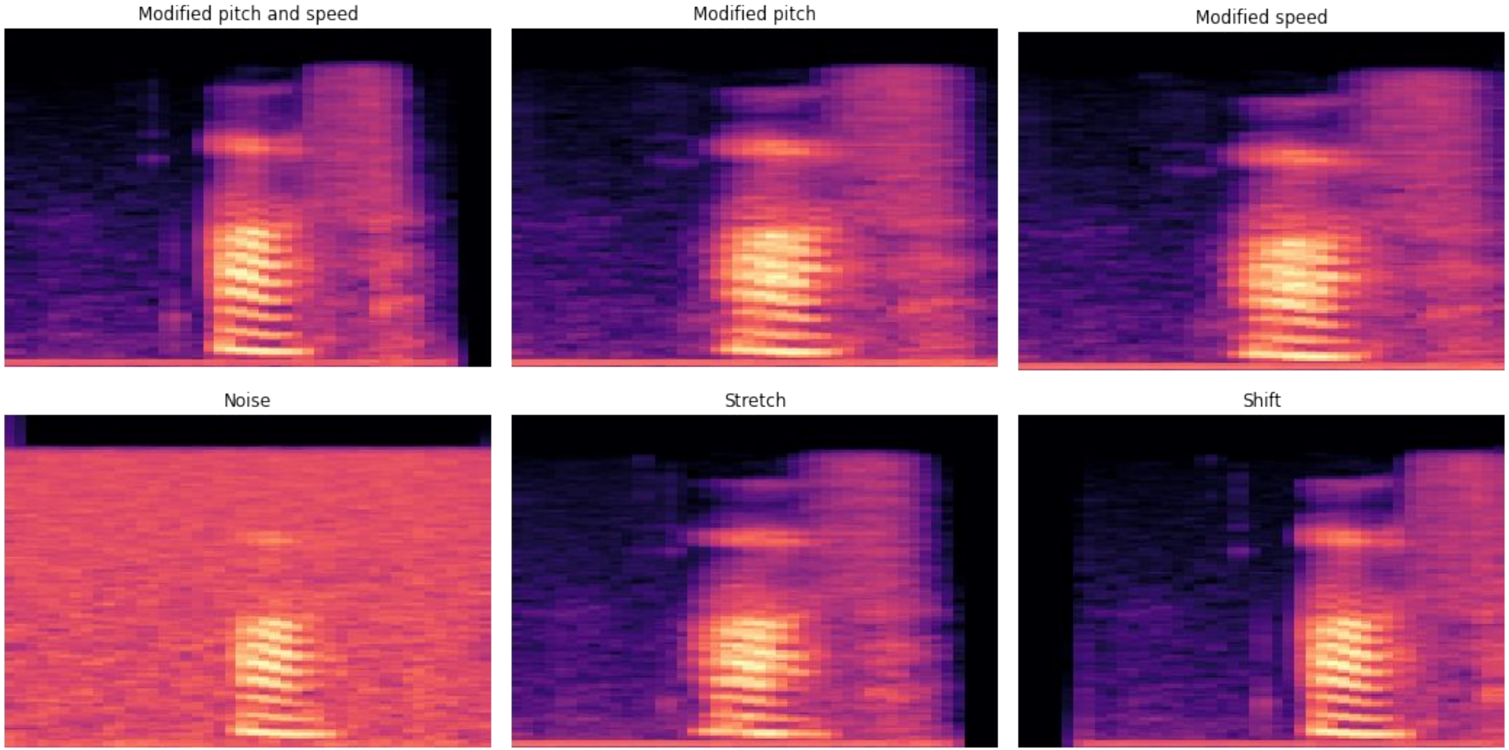
For this reason, a simple spectrogram is rarely used in deep learning while a **Mel Spectrogram** is the preferred choice.

A Mel Spectrogram:

- uses the Mel Scale instead of Frequency on the y-axis.
- uses the Decibel Scale instead of Amplitude to indicate colors.



Augmented data spectrograms



Modified pitch and speed

Modified pitch

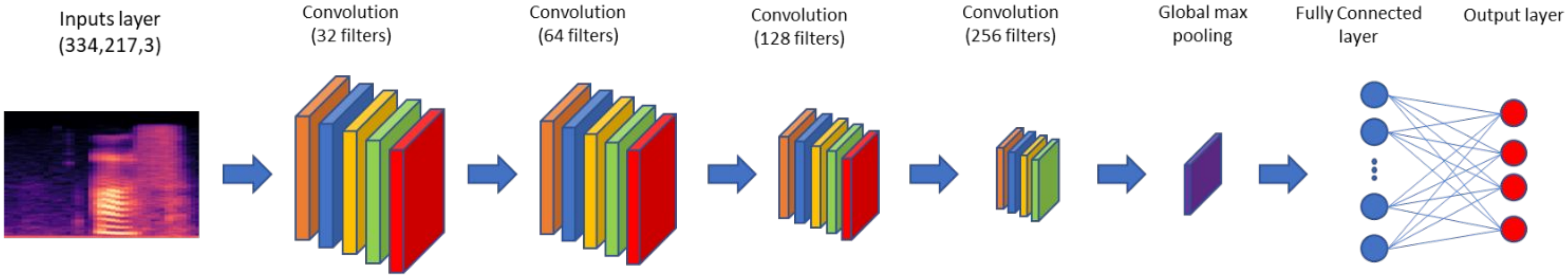
Modified speed

Noise

Stretch

Shift

4- CNN definition



Inputs layer
(334,217,3)

Convolution
(32 filters)

Convolution
(64 filters)

Convolution
(128 filters)

Convolution
(256 filters)

Global max
pooling

Fully Connected
layer

Output layer

🟧 Convolutional 3x3 layer
«Same» padding

🟦 Batch normalization layer

🟩 Max Pooling 2D layer
3x3 kernel, stride=3

● Dense layer
ReLU activation

🟨 Activation layer
ReLU function

🟥 Dropout layer

🟪 Global Max Pooling 2D layer
3x3 kernel, stride=3

● Dense layer
SoftMax activation

Dropout rate = 0.3

L2 regularization factor = 0.001

Categorical cross entropy loss

Adamax optimizer (learning rate = 0.005 reduced on plateau)

Early stopping if val_loss fails to improve in 4 epochs

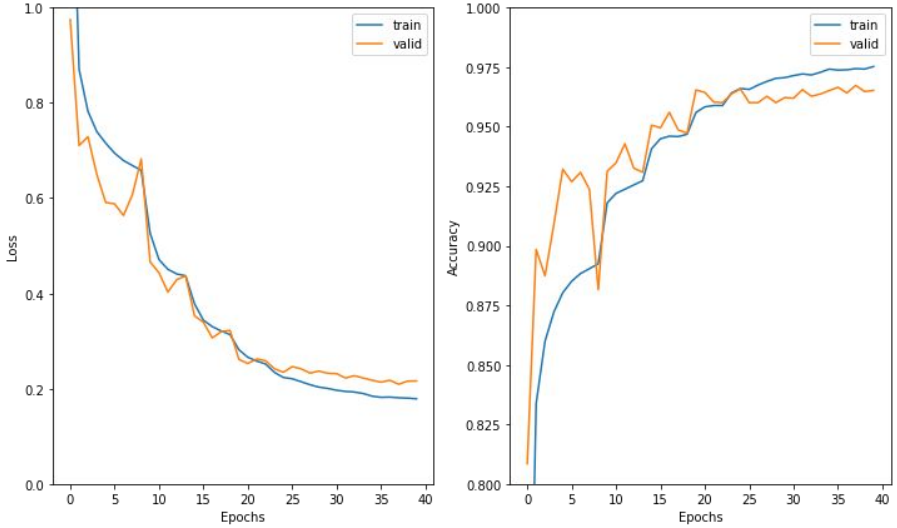
40 epochs

5- Results

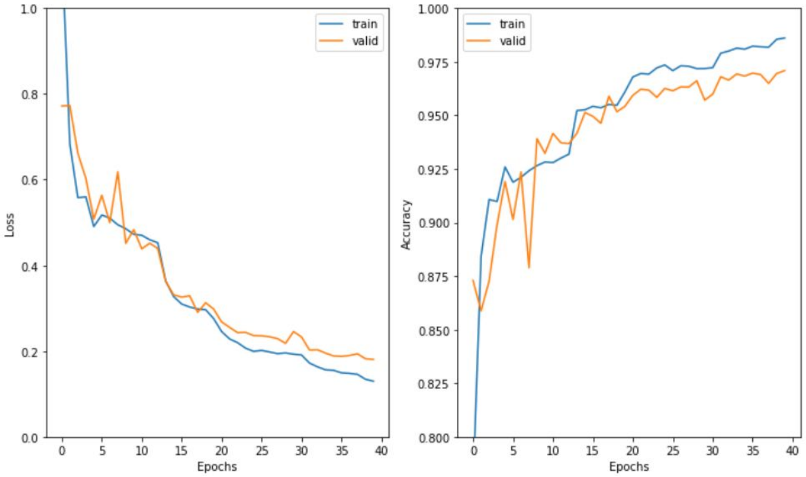
Our model has nearly achieved **97% of validation accuracy** over a period of 40 epochs.

Starting from the 25th epoch it experienced **slightly overfitting**.

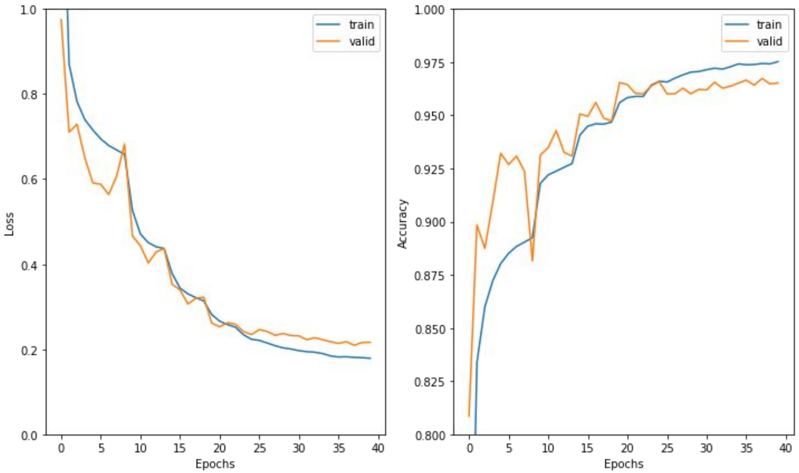
Due to computational limits, **we avoided applying all six transformations to each audio**.
That would have probably prevented the model from overfitting and could have improved performance too.



Comparison with our first solution



Same model, **without** data augmentation.
The model slightly overfitted during the training phase.



Our model.
Though the metrics didn’t improve, the distance between training and validation accuracy **seems to be lower**.

Alternative solutions

Other possible solutions known in literature are:

- Feeding a Conv1D neural network raw audio →
    - although those type of models seems to perform worse than Conv2D or Conv3D with Mel Spectrograms as input

- Extracting more audio features from Mel Spectrograms via MFCCs (Mel Frequency Cepstral Coefficient) →
    - those features are able to recognize specific aspect of someone's voice like texture or timbre

- Time Delay Neural Network →
    - a feedforward neural network used in a variety of tasks ranging from speech recognition to video and text analysis

The End