Urban Sounds Audio Classification

By Alessandro Castagna, Daniel Colombaro and Federico Ferretti

3 folders of 1 GB each

10 seconds 24-bit .wav files

44,1kHz sample rate mono-channel

Labels:

- Bus
- Cafe / Restaurant
- Car
- City center
- Forest path
- Grocery store
- Home
- Lakeside beach
- Library
- Metro station
- Office
- Residential area
- Train
- Tram
- Urban park

Balanced classes?

The percentage of the labels in relation to the total number of observation:

Bus	6.48
Cafe / Restaurant	7.48
Car	5.98
City center	6.55
Forest path	6.77
Grocery store	7.19
Home	7.05
Lakeside beach	6.41
Library	7.69
Metro station	7.26
Office	6.62
Residential area	7.34
Train	4.91
Tram	6.34
Urban park	
	5.91

Models developed

SPECTROGRAM CNN

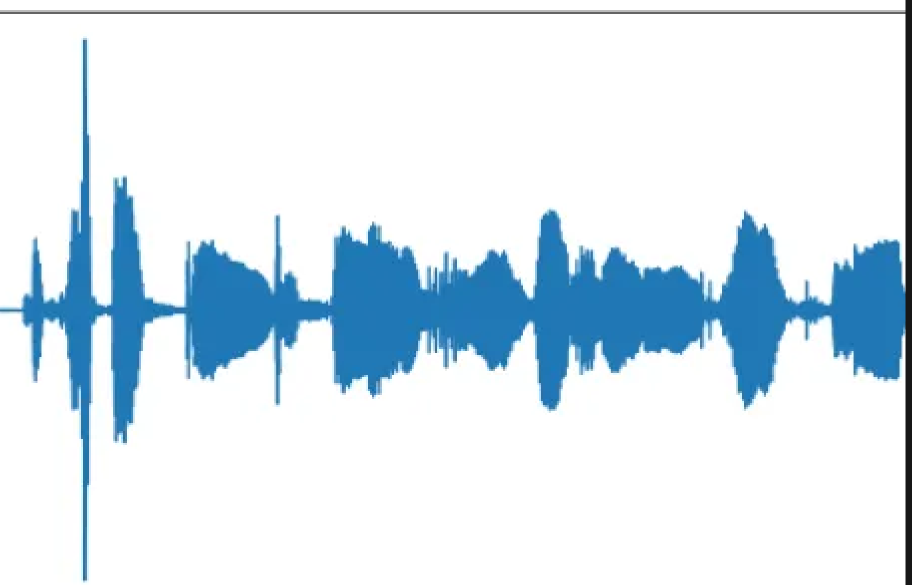
Transform audio files into spectrograms to feed in input to a convolutional neural network.

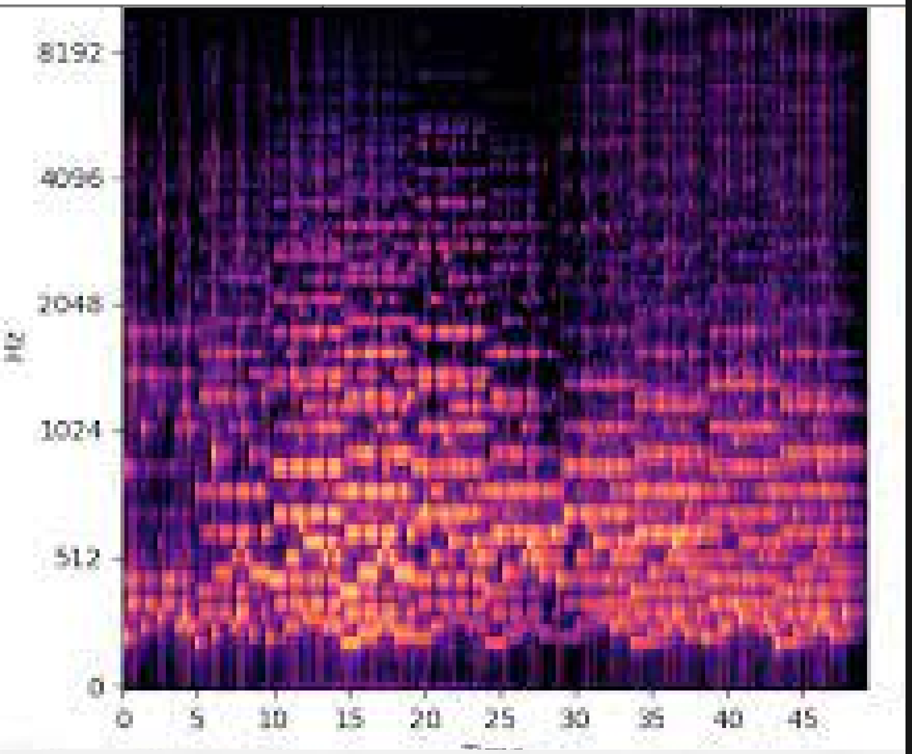
FEATURE EXTRACTION (MFCC)

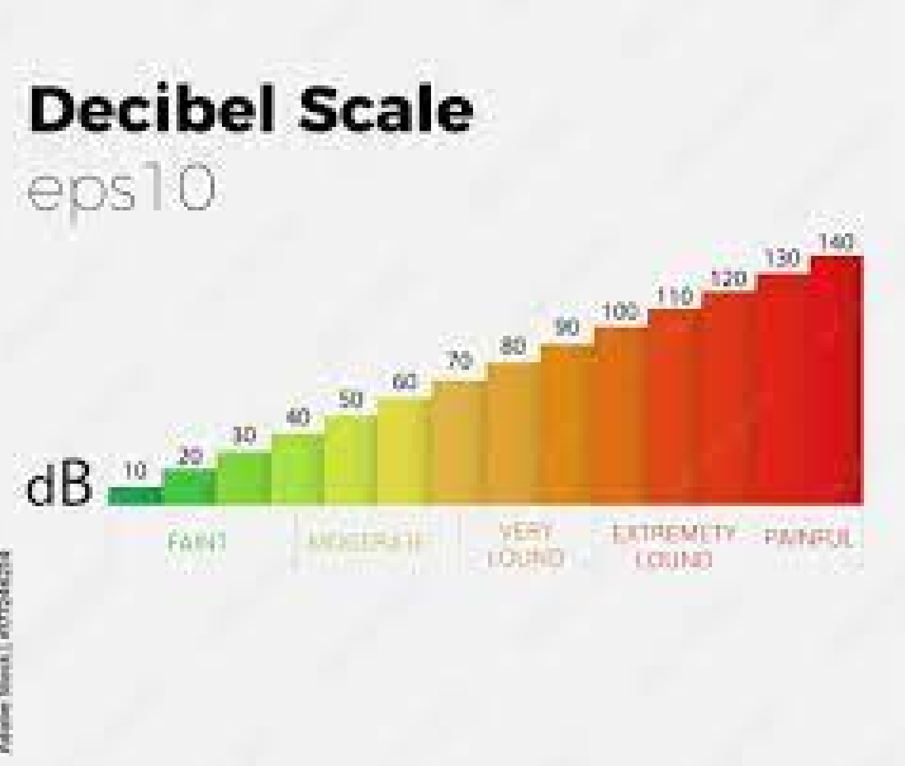
Extracting relevant features using "Librosa" library to train a feed forward network.

TRANSFER LEARNING

Applying parameters of a pre-trained model to other data







STEP 1

Reading files into arrays, whom element are related to the amplitude of the signal

STEP 2

Tranforming into spectrograms using *'STFT':* visual representations of the decomposed frequency ranges

STEP 3

Transform into Mel spectrograms, using '*Mel'* and '*Decibel'* logarithmic scales

Spectrogram

Problem

Spectrograms are fairly memory-intensive and the computations to apply 'stft' caused the RAM to crash

Attempts to fix the issue:

- Using method "*map()*" on the whole tensor:
    - *Applying the function* 'spectrogram()' *to the tensor of arrays would fail to properly take the single arrays*

- Iterating over single elements with a loop:
    - Storing such large amounts of data together would crash the RAM
    - Splitting the dataset in smaller functions did not help

Solution

Batch Data Loader

At runtime, as we train the model one batch at a time, we will load the audio data for that batch and process it by applying a series of transforms to the audio. In this way we keep audio data for only one batch in memory at a time, preventing the RAM from crashing.

- Definition of a custom "PyTorch" dataset object that applies all the transformations to pre-process an audio file.
- Implementation of a built-in DataLoader object that uses the Dataset object to fetch individual data items and packages them into a batch of data.

Structure of the CNN

4 Convolutional blocks composed of:

- conv2d layer
    - stride (2,2) and padding (1,1
- "*relu*" activation function
- batch normalization
- resetting bias

CNN BLOCK 1	CNN BLOCK 2	CNN BLOCK 3	CNN BLOCK 4
2 to 8 features	8 to 16 features	16 to 32 features	32 to 64 features

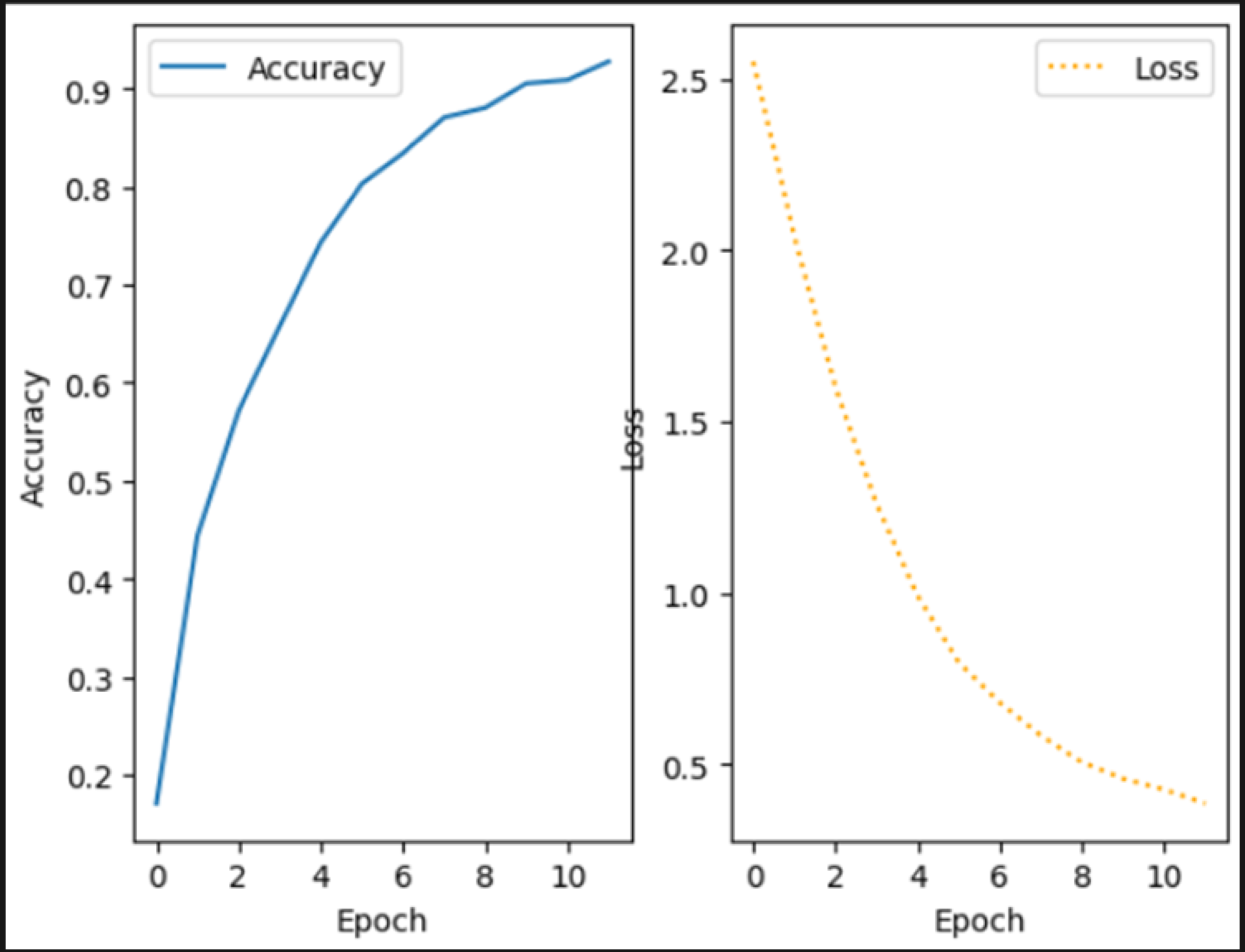
- Adaptive average pooling 2d
- Linear layer with 64 features in input and 15 in output (number of labels)

91%

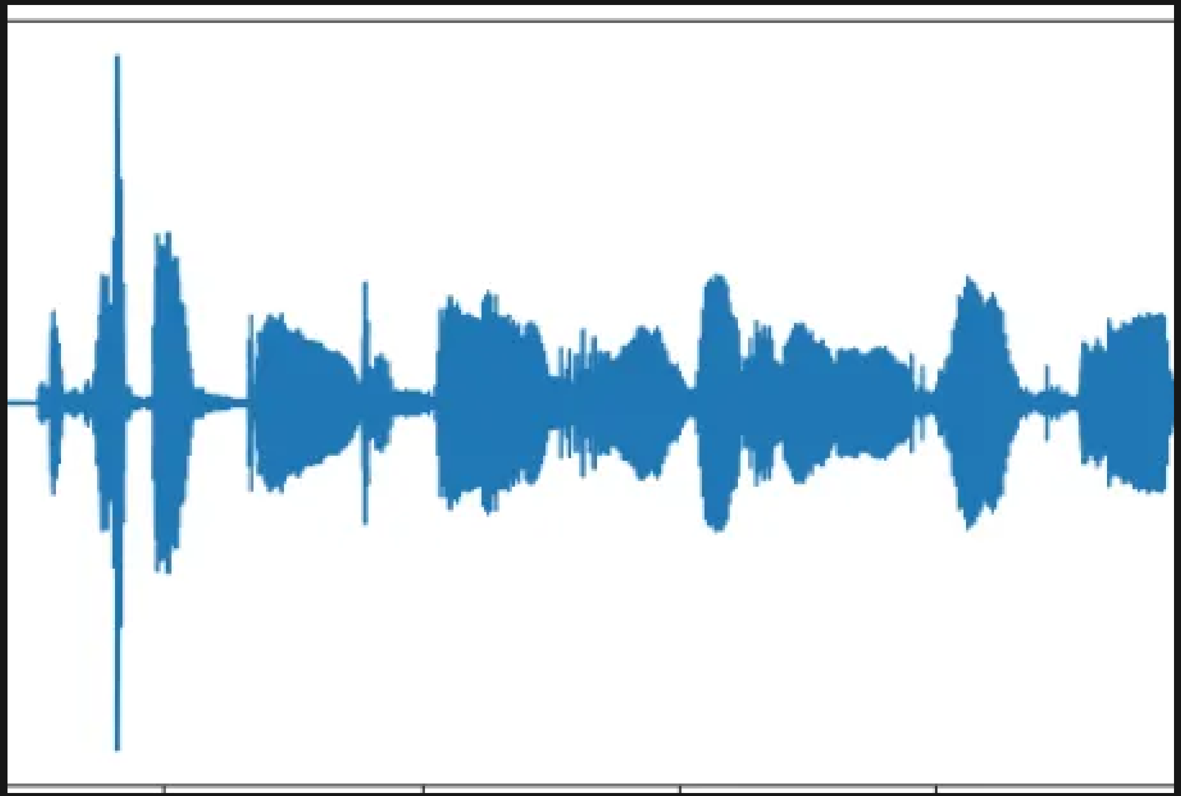
TRAIN ACCURACY

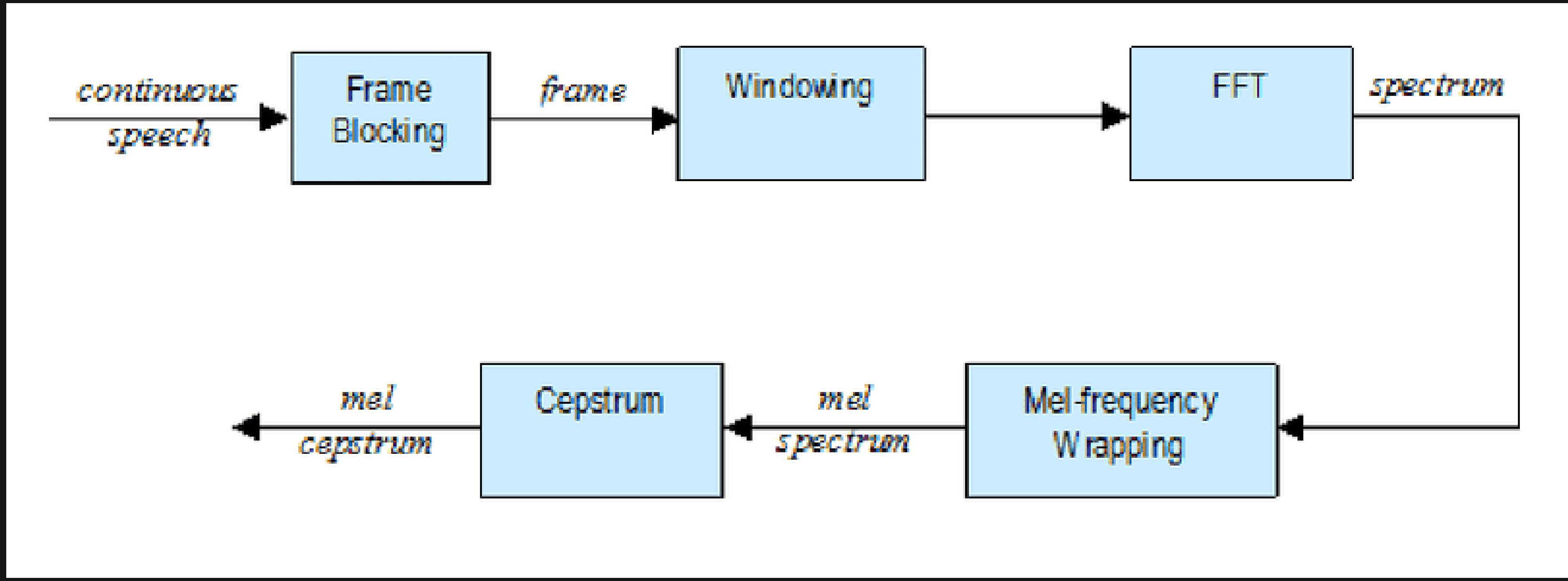
90%

VALIDATION ACCURACY

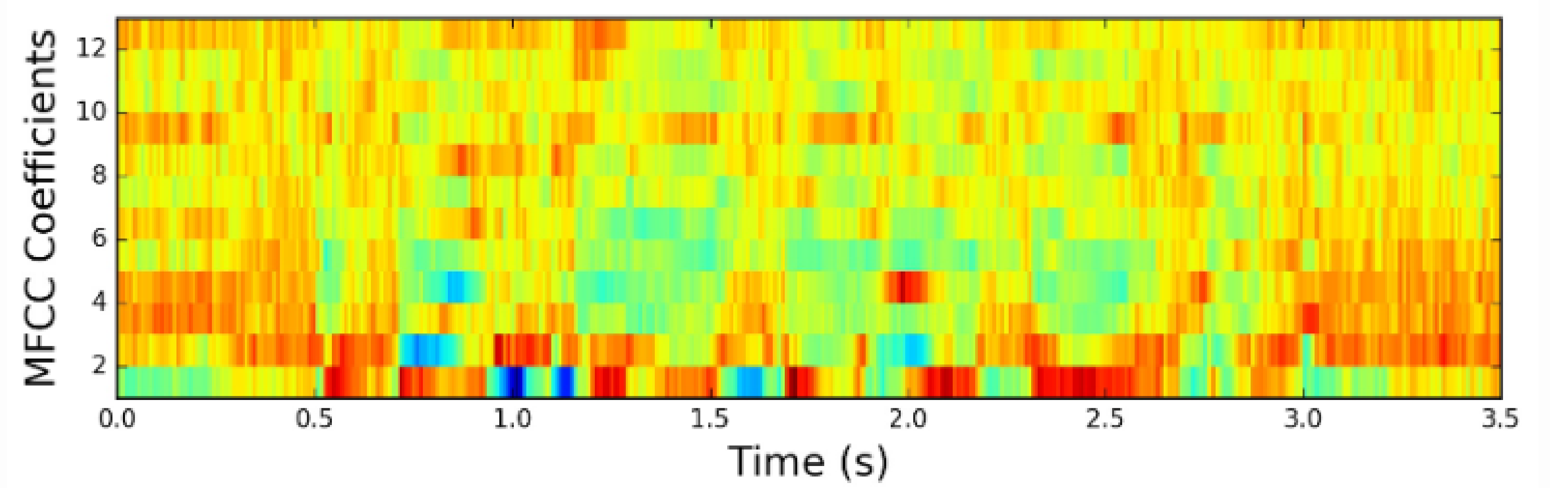


Mel Frequency Cepstral Coefficient (MFCC)





The resulting MFCCs:



MFCCs

Pre processing

Increasing the audio by a few semitones

Rescaling the MFCC values between 0 and 1

MFCC Matrix form

DENSE LAYER

125 nodes

DENSE LAYER

256 nodes

DENSE LAYER

512 nodes

DENSE LAYER

256 nodes

DENSE LAYER

125 nodes

FLATTEN LAYER

DENSE LAYER

15 nodes

Soft max activation

- epoch--> 25
- batch size --> 16

5 Layer dense, each with:

- dropout
- batch normalization

Adjusted MFCC

DENSE LAYER

125 nodes

DENSE LAYER

125 nodes

DENSE LAYER

125 nodes

DENSE LAYER

125 nodes

DENSE LAYER

125 nodes

DENSE LAYER

15 nodes

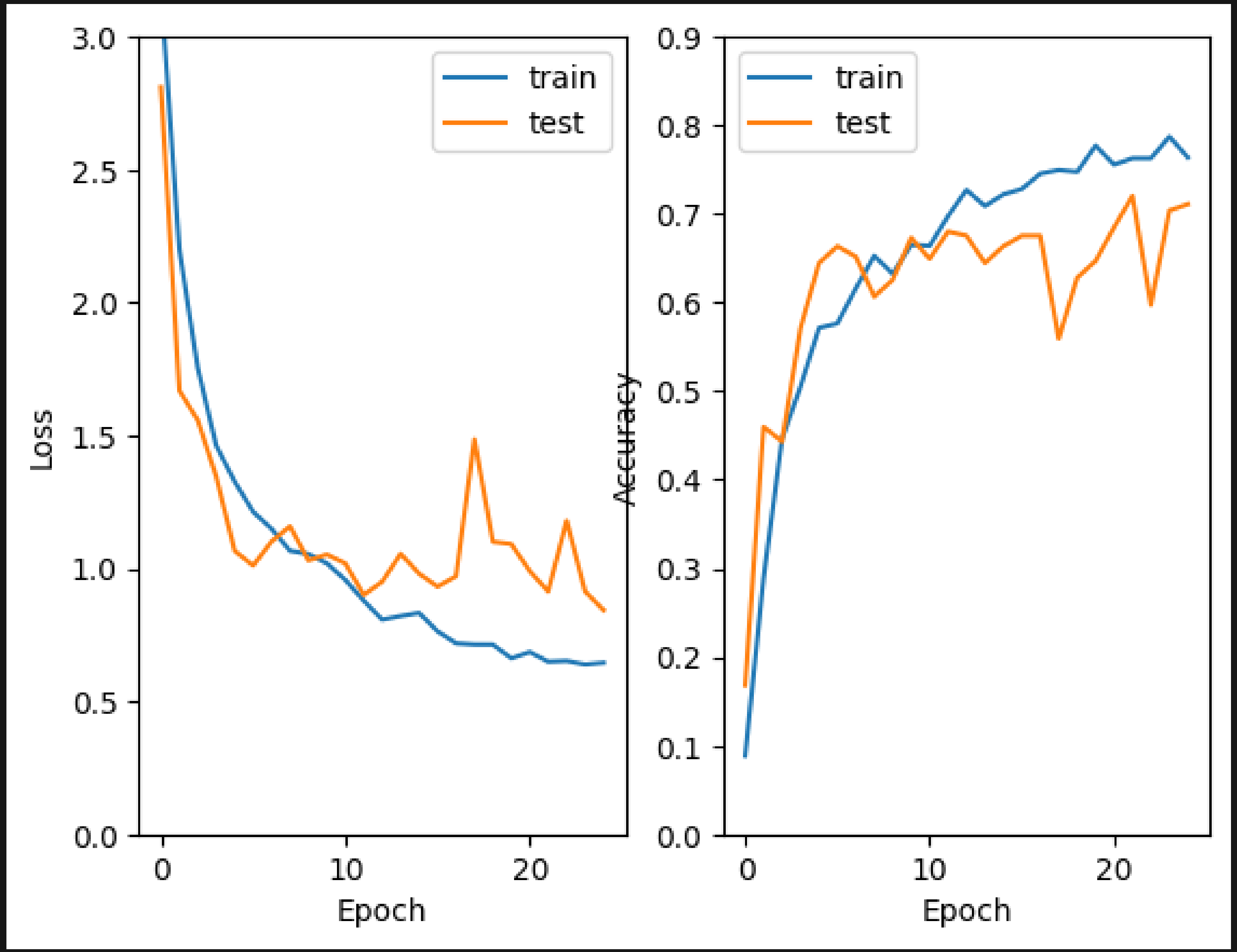
Soft max activation

- epoch--> 120
- batch size --> 32

5 Layer dense, each with:

- dropout
- batch normalization

MFCC Matrix form



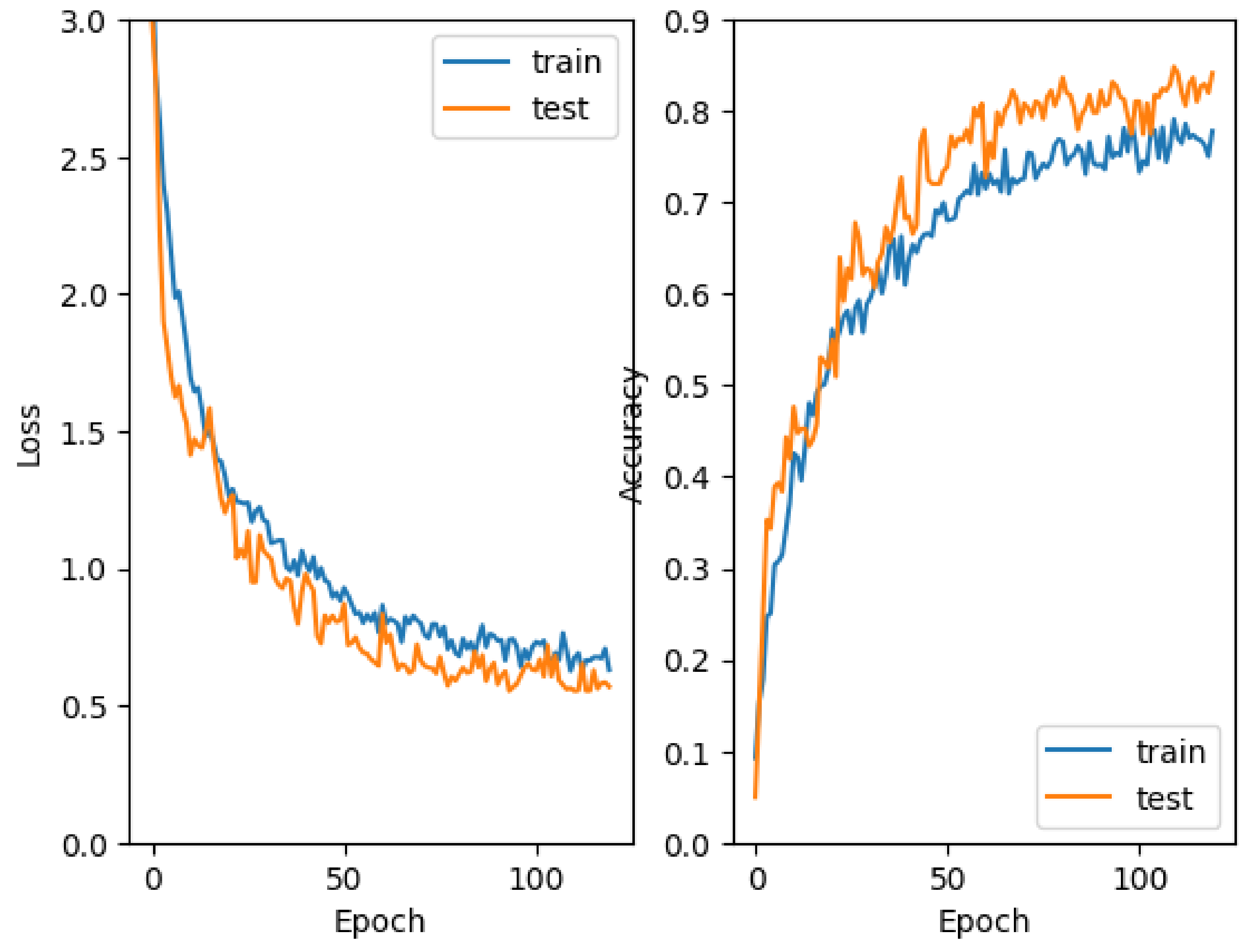
79%

TRAIN ACCURACY

72%

VALIDATION ACCURACY

Adjusted MFCC



79%

TRAIN ACCURACY

84%

VALIDATION ACCURACY

Transfer Learning

Transfer learning is a machine learning technique that allows using the knowledge gained from a pre-trained model on a given task to improve the performance of a model on a related or similar task.

Advantages

Reduces the need for large amounts of training data.
Speed up the process of training new models.
Improve performance on specific tasks thanks to general knowledge learned.

Transfer Learning Process

Pre-training: A model is trained on a large amount of training data.
Fine-tuning: The pre-trained model is tailored or "tuned" to a specific task using a smaller training dataset.

Pre-Trained Model

VGG16

- Deep structure with 16 convolutional layers
- Layered architecture with 3x3 convolutions
- Large number of parameters
- High computational complexity

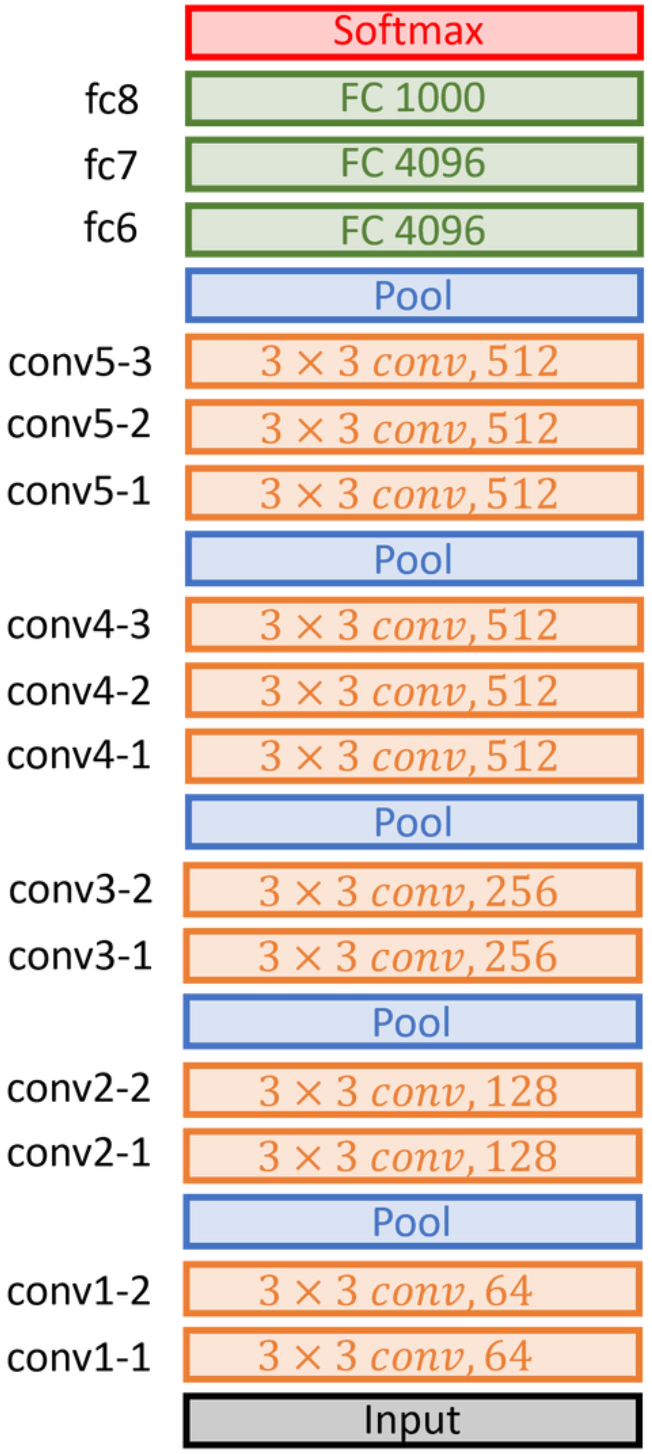
ResNet

- Deep structure with 50 convolutional layers
- Use of residual blocks to facilitate training
- "Skip" connections that improve gradient flow during back propagation

MobileNetV2

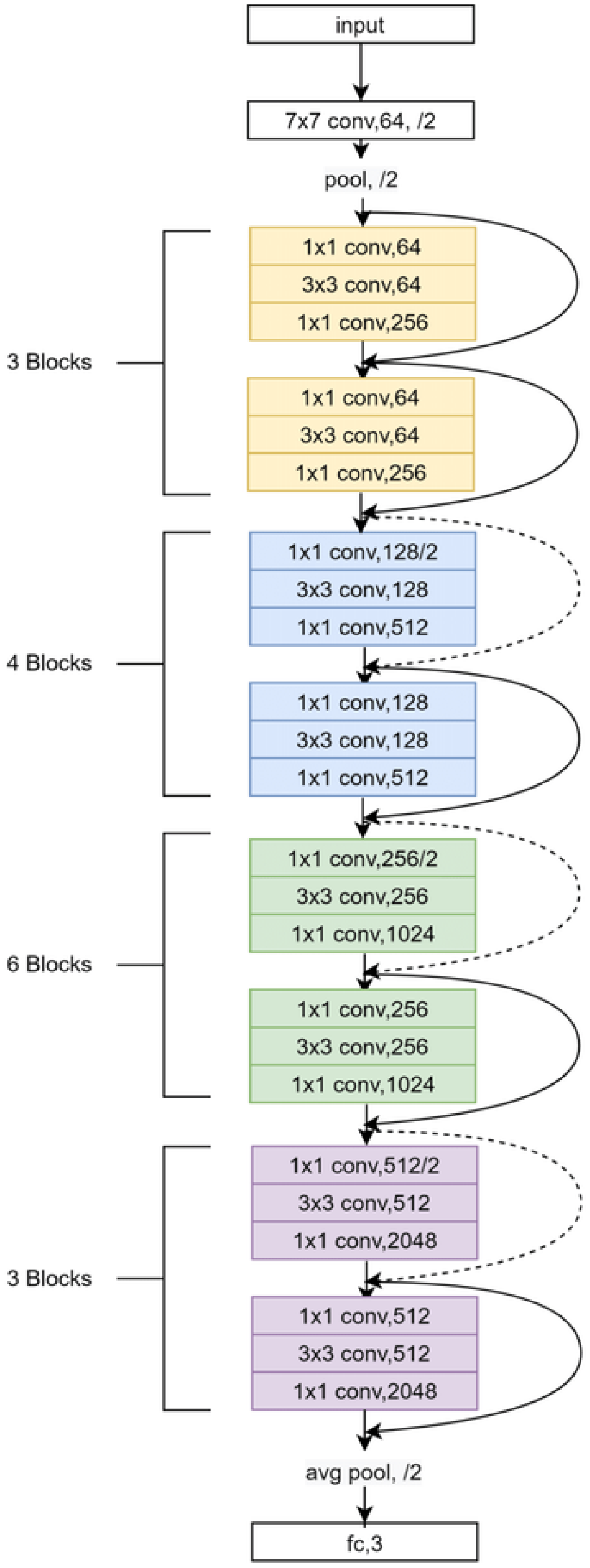
- Lightweight architecture optimized for limited computational resources
- Using depthwise separable convolutions to reduce computations
- Bottleneck layers to increase model complexity without increasing the number of parameters

VGG16

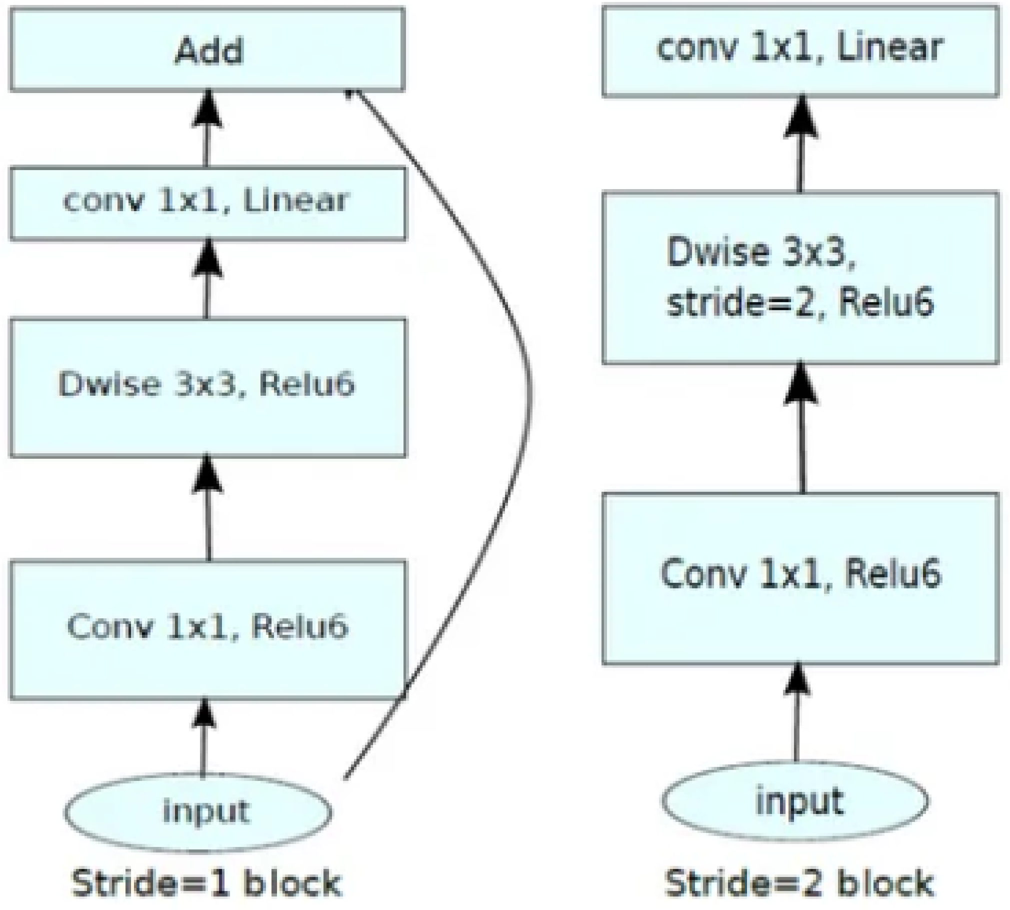


VGG16

RESNET50



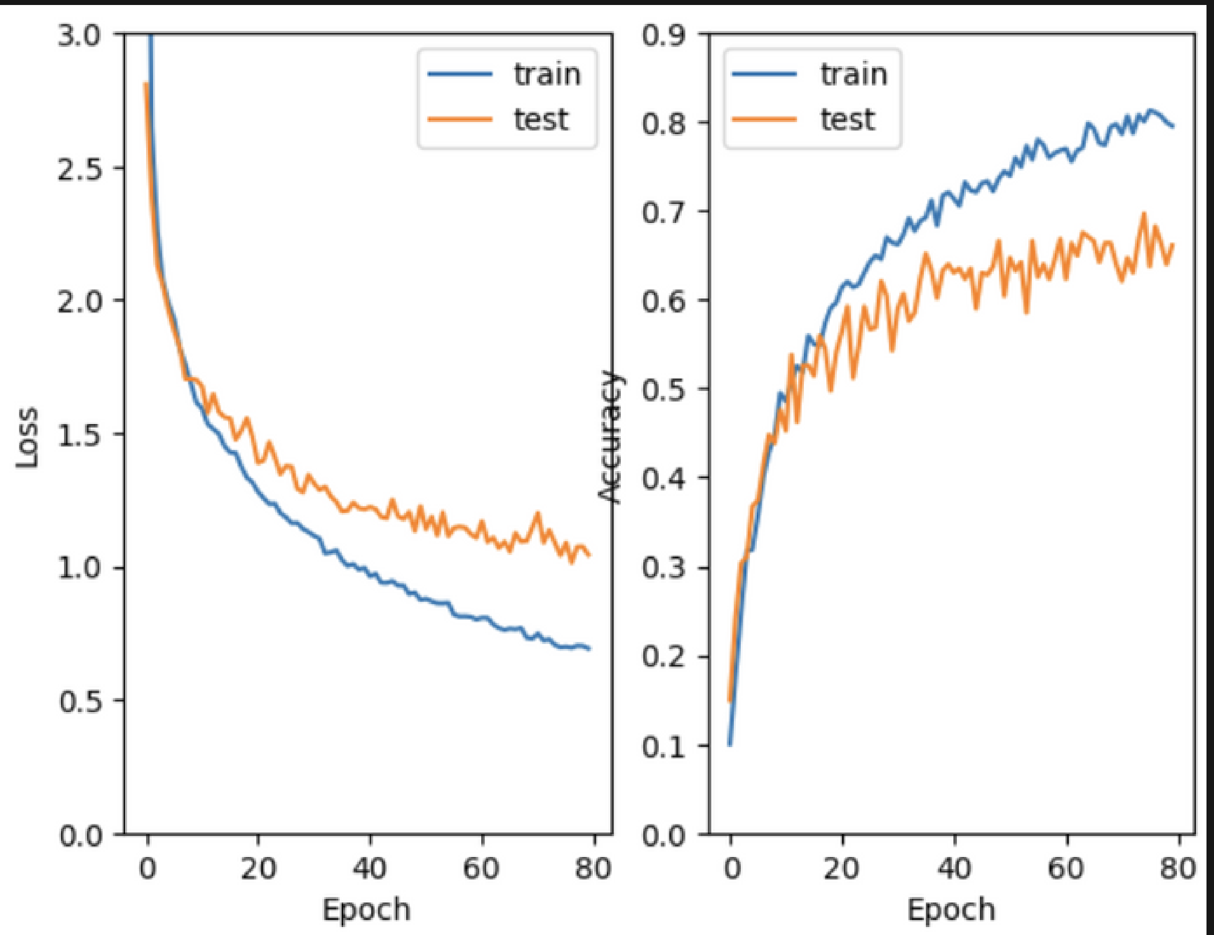
MobileNetV2



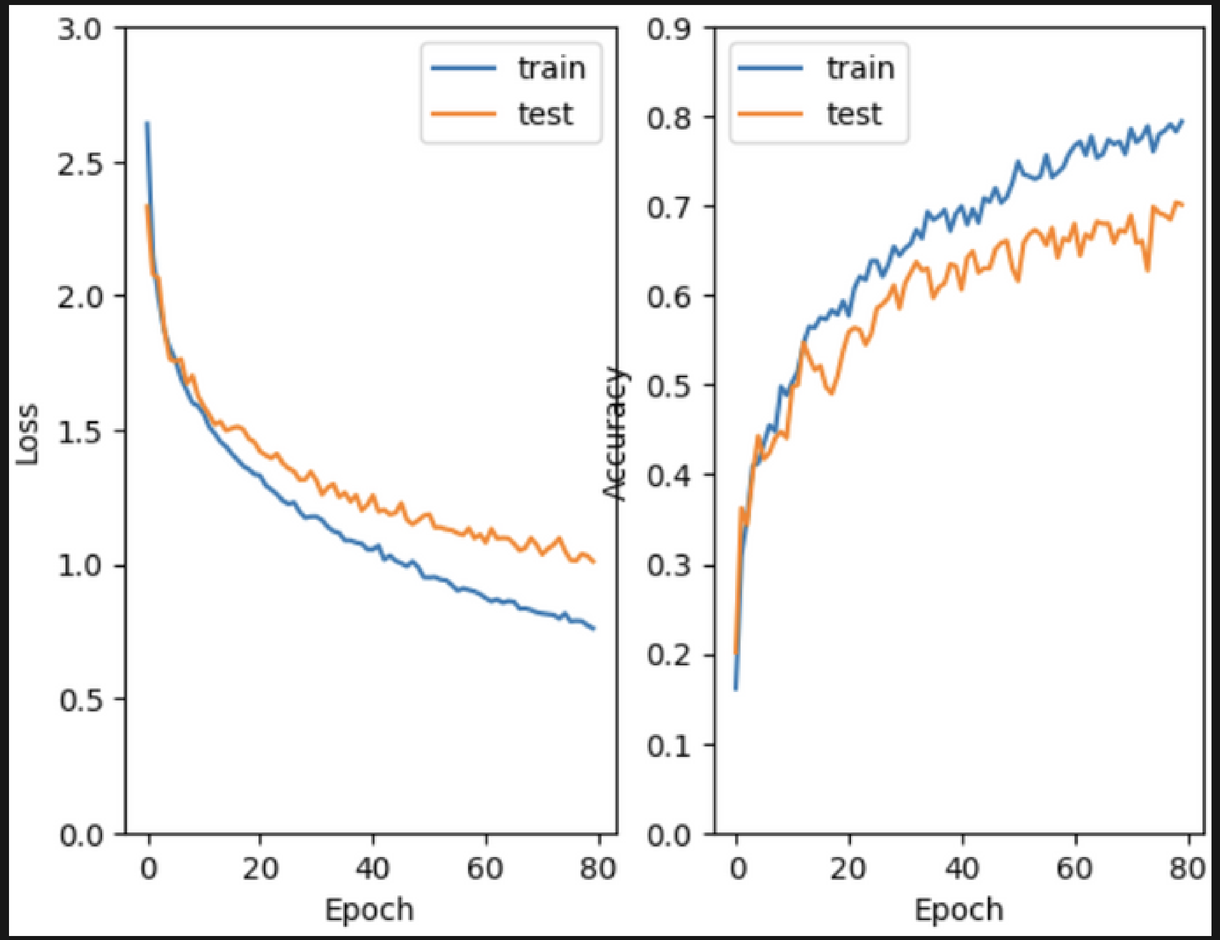
VGG16

RESNET50

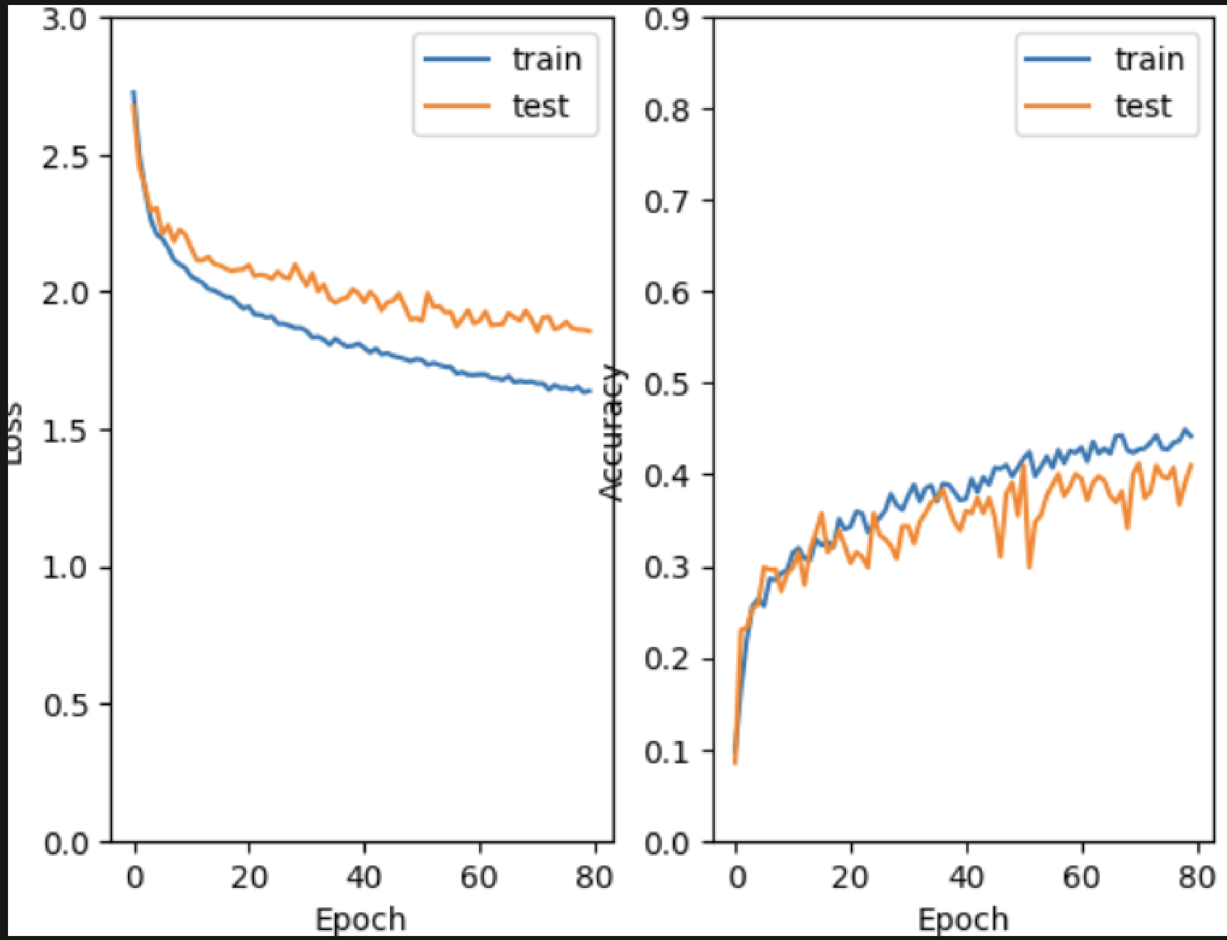
MobileNetV2



loss: 0.6913 - accuracy: 0.7953
val_loss: 1.0427 - val_accuracy: 0.6611
Training completed in time: 504 Sec.



loss: 0.7603 - accuracy: 0.7943
val_loss: 1.0101 - val_accuracy: 0.7014
Training completed in time: 362 Sec.



loss: 1.6379 - accuracy: 0.4420
val_loss: 1.8564 - val_accuracy: 0.4100
Training completed in time: 206 Sec.

Summarizing

MOST COMMON METHOD

Spectrogram model

BEST ACCURACY

0.90

- Spectrogram model

BEST EXECUTION TIME

43.7 seconds

- MFCC adjusted model

HIGHER COMPLEXITY

ResNet