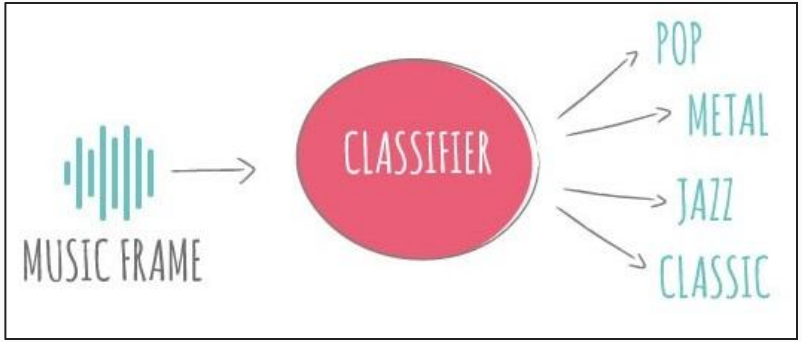
# Music Genre Classification

Davide Pietrasanta - 844824
Giuseppe Magazzù - 829612
Gaetano Magazzù  - 829685

# Task

The task for this project is the **classification** of music genres from **audio** files.

It is assumed that only one musical genre is associated with each audio track.



# Dataset

Free Music Archive (FMA)

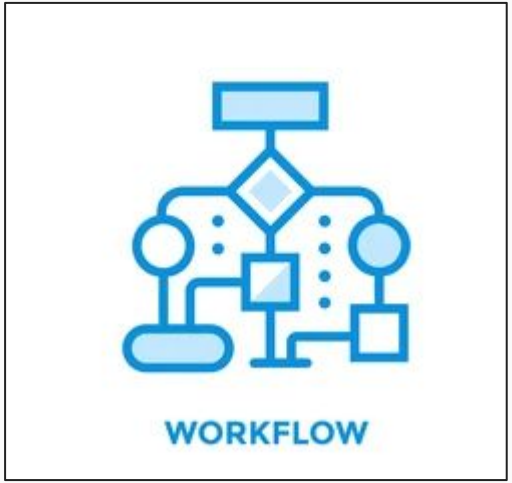
- Raw audio tracks + additional metadata
- Suggested split (Train 80%, Validation 10%, Test 10%)
- Available subsets:
    - **small: 8000 audio tracks and 8 musical genres (balanced)**
    - medium: 25000 audio tracks and 16 musical genres (unbalanced)
    - large: 106,574 audio tracks and 161 musical genres (unbalanced)
- Pre-computed features: statistical moments for different spectral features

Dataset: https://github.com/mdeff/fma

# Workflow

The work followed the following development:

- Handcrafted features
- CNN
    - Simple CNN
    - Hyper-parameters optimization
    - Data augmentation
    - Transfer Learning



# Handcrafted features

FFNN

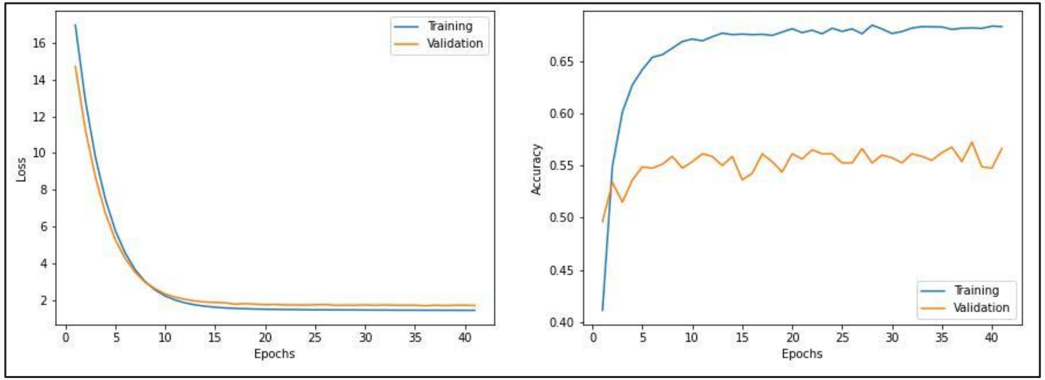
Input: 518 pre-computed features
Output: 8 genres classes

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               265728

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 8)                 2056

=================================================================
Total params: 399,112
Trainable params: 399,112
Non-trainable params: 0
```




Early Stopping - Best Epoch: 35

**Inference Time**

CPU: 20.70ms
GPU: 20.07ms

| | |
|---|---|
| Train | loss: 1.470, accuracy: 0.680 |
| Validation | loss: 1.715, accuracy: 0.567 |
| Test | loss: 1.900, accuracy: **0.495** |

# Feature for CNN

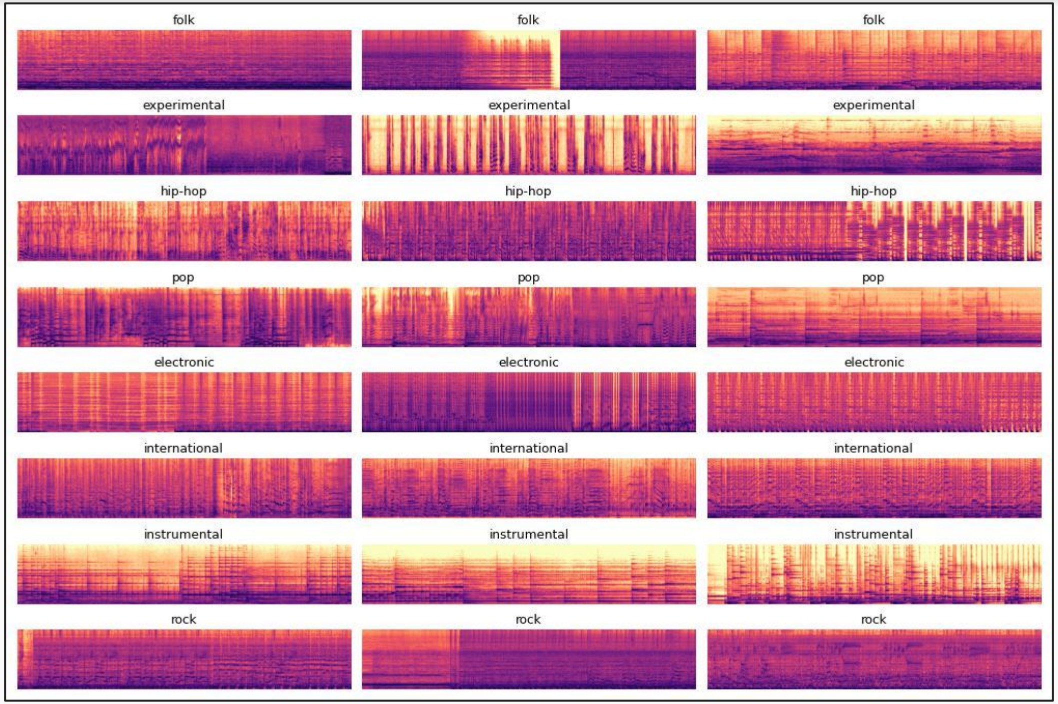
Pre-processing:

- Resample at 22050 Hz
- Trim audio at 29.70 seconds
- Removed corrupted audio files (6)

Log Mel Spectrogram as Images:

- Grayscale [0, 255]
- (128, 1280)

Intraclass Variation




dataset: https://www.kaggle.com/giuseppemagazz/fma-mel-new

# Simple CNN

The architecture used was the one suggested by "Daniel Kostrzewa".

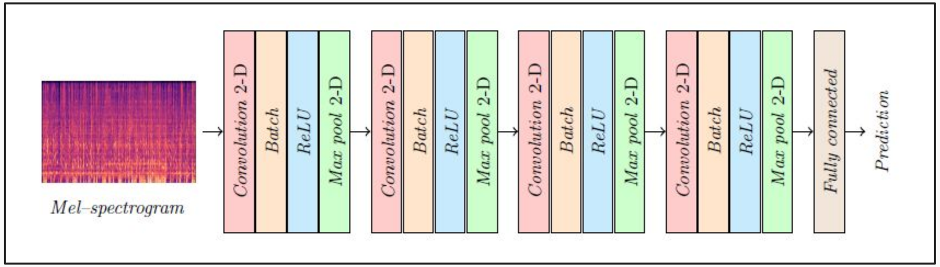
The number of trainable parameters is 735,944.

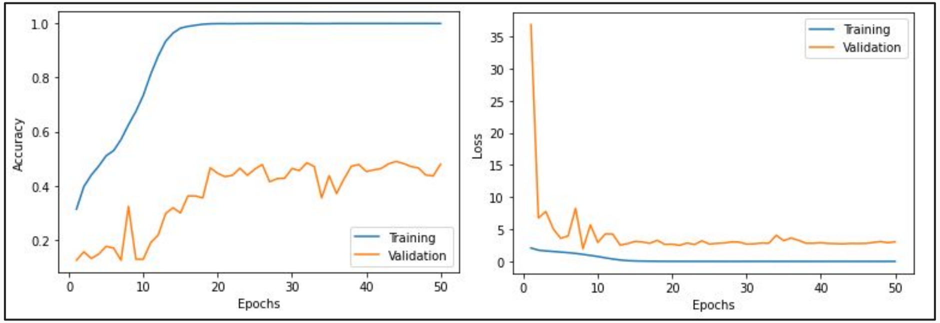
The model reaches an accuracy of **0.3975** on the test set and, as can be seen from the graphs, it overfits.

The accuracy of the paper is, however, equal to **0.5163**.

Time (CPU): 64 ms

Time(GPU): 42 ms




D. Kostrzewa, P. Kaminski, and R. Brzeski, “Music genre classification: Looking for the perfect network,” in International Conference on Computational Science. Springer, 2021, pp. 55–67.

# Hyper-parameters optimization

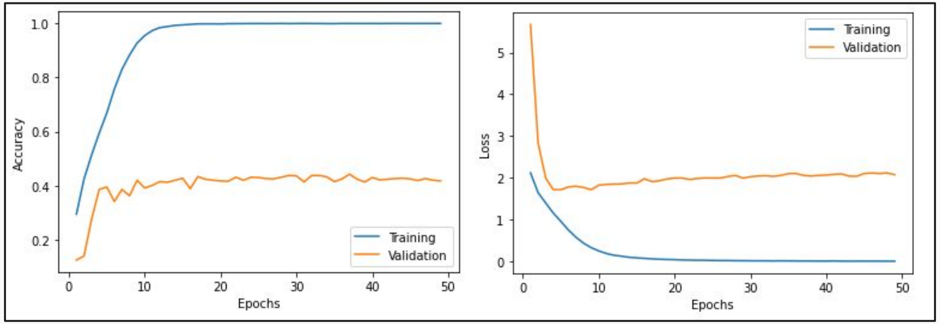
We then perform an hyper-parameter optimization using Hyperband.

The hyper-parameters considered are:

- The size of the kernel (fixed on each layer) in [3,6].
- The number of kernels for each layer in [32, 256].
- The probability of dropout in {0.2, 0.25, 0.5}.
- The initial learning rate in {0.01, 0.001, 0.0001}.

The model reaches an accuracy of **0.3900** on the test set and, as can be seen from the graphs, it overfits.

We can see how the validation curve is more stable this time.




Time (CPU): 60 ms

Time (GPU): 43 ms

# Data augmentation

As data augmentation we have used SpecAugment:

- Frequency Masking
- Time Masking

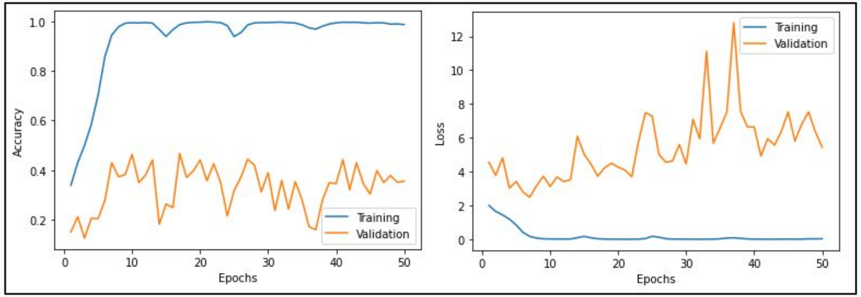
Two new examples for each example in the train set (Train: 19196, Validation: 800, Test: 800).

After the data augmentation the new accuracies for the test set are:

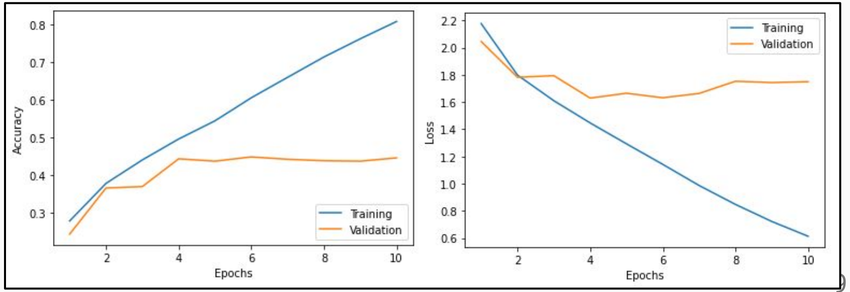
Simple CNN: **0.3225**

Tuned CNN: **0.3787**

We still have overfitting and in the case of the simple CNN we have really unstable results on the validation.




Above, the curves of the simple CNN. Below, the curves of the Tuned CNN

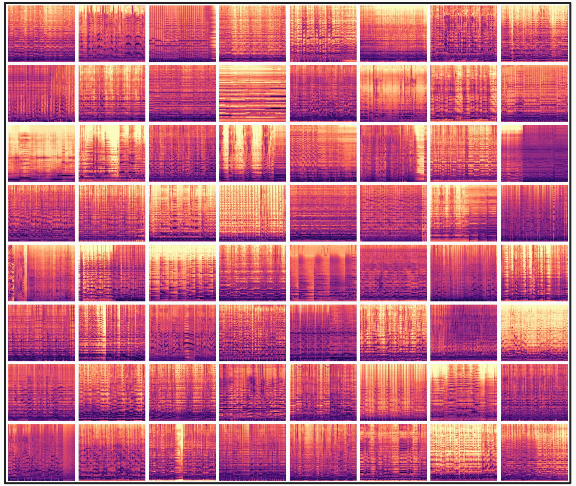


# Transfer Learning

We performed features extraction on two Top-5 Accuracy CNN on ImageNet VGG16 e ResNet50.

Low amount of data for training

Tasks really different



# Experiments

New split training/test with a test of 20%, (6398, 1596).

3 classifiers were trained by a 10-fold cross validation with a grid search for hyperparameters tuning.

Different percentages of cumulative explained variance were tested on FC2 layer of VGG16: 90%, 95%, 99%.

| Cut Level | Size | Size after PCA | PCA value | Model | Accuracy | | |
|---|---|---|---|---|---|---|---|
| | | | | | svm linear | svm rbf | mlp |
| Fc2 | 4096 | 1480 | 0.99 | VGG16 | 0.4440/0.4003 | 0.5168/0.4454 | 0.4795/0.4191 |
| Fc2 | 4096 | 429 | 0.95 | VGG16 | 0.4965/0.4530 | 0.5206/0.4542 | 0.4890/0.4361 |
| Fc2 | 4096 | 153 | 0.9 | VGG16 | 0.5099/0.4586 | 0.5195/0.4617 | 0.4920/0.4398 |

Train/Test Accuracy

# Experiments

## VGG16

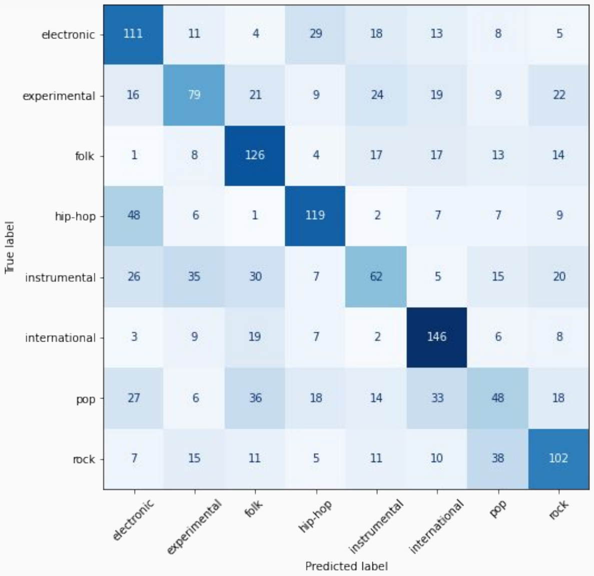
| Cut Level | Size | Size after PCA | Model | Accuracy | | |
|---|---|---|---|---|---|---|
| | | | | svm linear | svm rbf | mlp |
| Fc2 | 4096 | 153 | VGG16 | 0.5099/0.4586 | 0.5195/0.4617 | 0.4920/0.4398 |
| Fc1 | 4096 | 311 | VGG16 | 0.5157/0.4573 | 0.5315/0.4837 | 0.5076/0.4718 |
| block5_pool | 7x7x512 | 1174 | VGG16 | 0.4316/0.4185 | **0.5317/ 0.4968** | 0.4726/0.4429 |

## ResNet50

| Cut Level | Size | Size after PCA | Model | Accuracy | | |
|---|---|---|---|---|---|---|
| | | | | svm linear | svm rbf | mlp |
| avg_pool | 1000 | 87 | ResNet50 | 0.5268/0.4649 | 0.5381/0.4874 | 0.5246/0.4592 |
| conv5_block1_2_relu | 7x7x512 | 1321 | ResNet50 | 0.4193/0.4104 | **0.5484/0.5012** | 0.5221/0.4755 |

Train/Test Accuracy

# Comparison



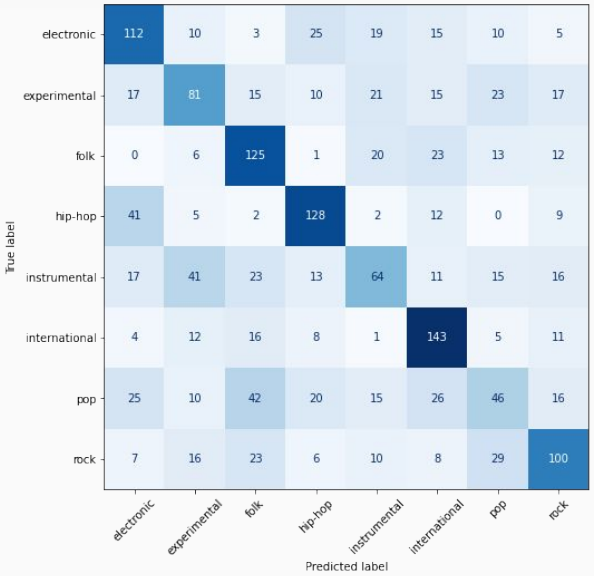

VGG16 best cut + SVM RBF

**Accuracy:** 0.4968

**FE**: 73.21ms(CPU)
**FE**: 23.90ms (GPU)

**SVM**: 02.69ms (CPU)

**Parameters:** 14,714,688




ResNet50 best cut + SVM RBF

FE = feature extraction

**Accuracy:** 0.5012

**FE:** 51.34ms (CPU)
**FE:** 23.75ms (GPU)

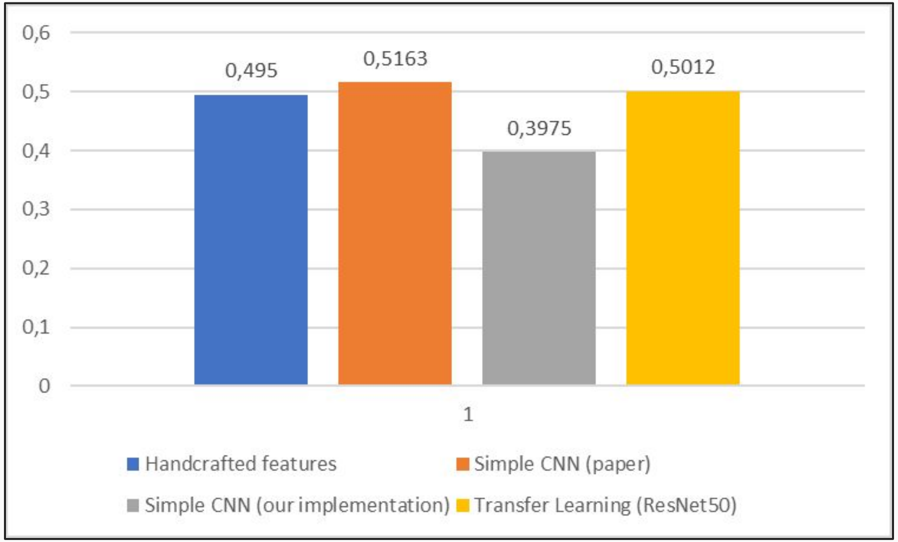
**SVM:** 03.27ms (CPU)

**Parameters:** 11,477,888

# Results

**Accuracies**

- Handcrafted features: **0.495**
- Simple CNN, from paper: **0.5163**
- Simple CNN, our implementation: **0.3975**
- Transfer Learning, with ResNet50: **0.5012**



# Future developments

Possible ways to improve could be:

- Optimization of hyper-parameters from start to finish
- Consider medium and large dataset
- Different pre-processing
- Better data augmentation
- Other features as MFCCs and its derivatives

# Extra

# Handcrafted Features

For each of the following features are calculated:

- min
- max
- mean
- median
- std
- skew
- kurtosi

Total: 518 values

Features:

- Chroma STFT: 12
- Chroma CQT: 12
- Chroma CENS: 12
- Tonnezt: 6
- MFCC: 20
- Zero Crossing Rate: 1
- RMSE: 1
- Spectral Centroid: 1
- Spectral Bandwidth: 1
- Spectral Contrast: 7
- Spectral Rolloff: 1

# FFNN

Adam
Learning Rate: 1e-4

Batch: 32
Epochs: 200
Early Stopping (patience: 3)

Output Layer:
- Categorical Cross Entropy
- Softmax (8 classes)

Input Layer:
- 518 features

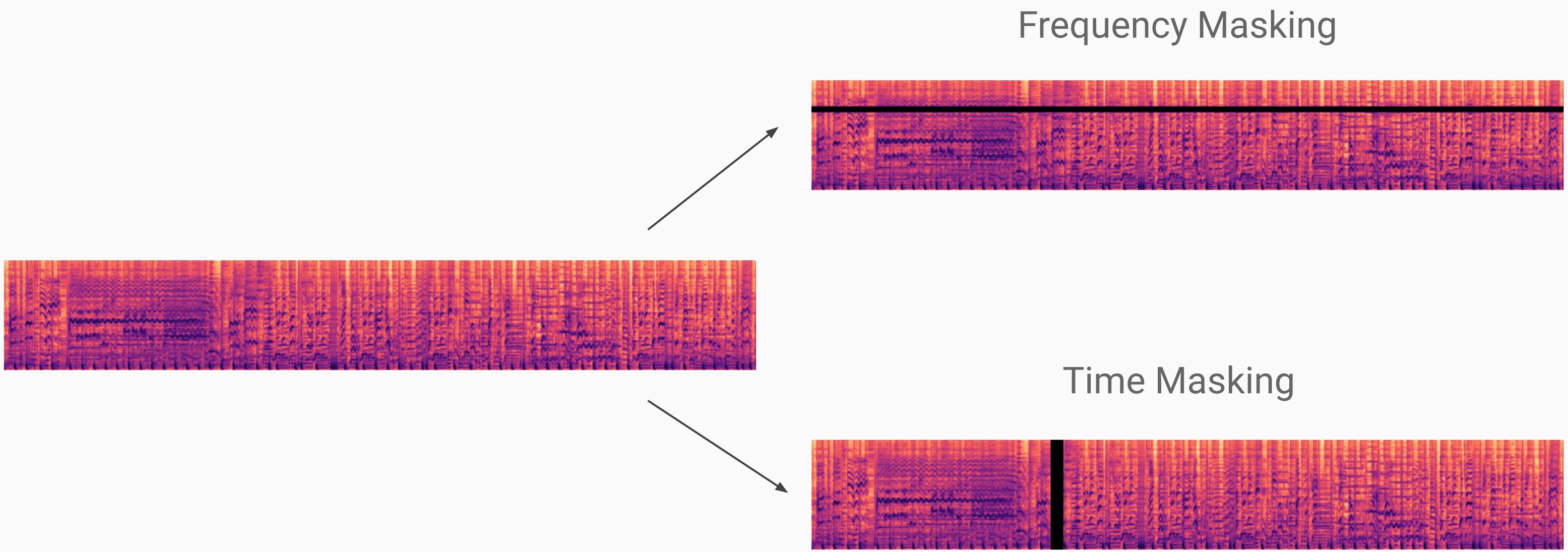
| Units | Parameters | Train Loss | Val Loss | Test Loss |
|---|---|---|---|---|
| (512, 512) | 532,488 | 1.482 | 1.723 | 1.898 |
| (512, 256) | 399,112 | 1.470 | 1.715 | 1.900 |
| (256, 256) | 200,712 | 1.503 | 1.733 | 1.946 |
| (256, 64) | 149,832 | 1.475 | 1.733 | 1.943 |
| (128, 64) | 75,208 | 1.476 | 1.729 | 1.905 |
| (64) | 33,736 | 1.332 | 1.598 | 1.790 |
| (128) | 67,464 | 1.321 | 1.577 | 1.800 |
| (256) | 134,920 | 1.338 | 1.592 | 1.806 |
| (512) | 269,832 | 1.334 | 1.579 | 1.803 |

Table 1: Values of loss for the different number of neurons.

| Units | Parameters | Train Acc | Val Acc | Test Acc |
|---|---|---|---|---|
| (512, 512) | 532,488 | 0.677 | 0.563 | 0.487 |
| (512, 256) | 399,112 | 0.680 | 0.567 | **0.495** |
| (256, 256) | 200,712 | 0.683 | 0.577 | 0.482 |
| (256, 64) | 149,832 | 0.690 | 0.577 | 0.484 |
| (128, 64) | 75,208 | 0.683 | 0.564 | 0.489 |
| (64) | 33,736 | 0.680 | 0.560 | 0.465 |
| (128) | 67,464 | 0.686 | 0.566 | 0.475 |
| (256) | 134,920 | 0.685 | 0.566 | 0.479 |
| (512) | 269,832 | 0.683 | 0.571 | 0.472 |

Table 2: Values of accuracy for the different number of neurons.

# Data Augmentation (SpegAugment)



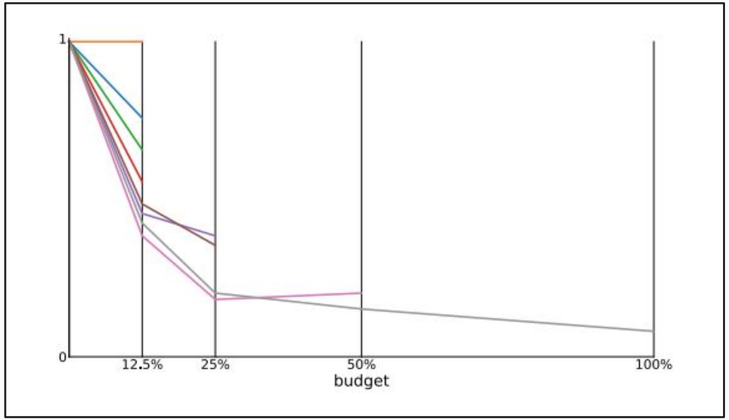

The pixel range is randomly generated from a uniform distribution.
For the frequency (0, 27) was used, while for the time (0, 100).

D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” arXiv preprint arXiv:1904.08779, 2019.

# Extra: Hyperband

**Hyperband** extends the **Successive Halving** algorithm.

The idea behind the original Successive Halving algorithm its about uniformly allocate a budget to a set of hyper-parameter configurations, evaluate the performance of all configurations, throw out the worst half, and repeat until one configuration remains.




It requires the **number of configurations** *n* and some finite **budget** *B* as inputs to the algorithm.
Than *B/n* resources are allocated on average across the configurations.

# Extra: Hyperband

However, in many cases it's not known if it's better to consider many configurations or not.

It runs several Successive Halving runs with different budgets and number of configurations, to find the best set.

It begins with the maximum exploration to ends up with a classical random search, in which every configuration is allocated with R resources.

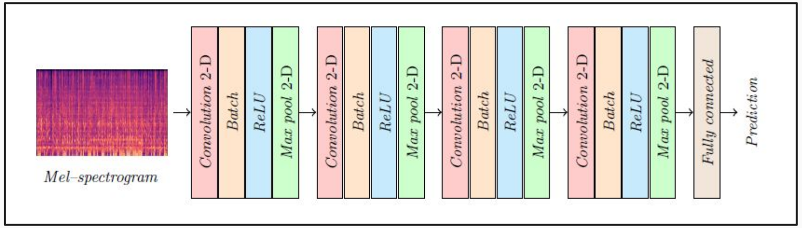
**Algorithm 1:** HYPERBAND algorithm for hyperparameter optimization.

**input** : $R$, $\eta$ (default $\eta = 3$)
**initialization** : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
**for** $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$,  $r = R\eta^{-s}$
  `// begin SuccessiveHalving with` $(n, r)$ `inner loop`
  $T =$ `get_hyperparameter_configuration`$(n)$
  **for** $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{$`run_then_return_val_loss`$(t, r_i) : t \in T\}$
    $T =$ `top_k`$(T, L, \lfloor n_i/\eta \rfloor)$
  **end**
**end**
**return** *Configuration with the smallest intermediate loss seen so far.*

L. Li, K. Jamieson, G. De Salvo, R. A. Talwalkar, and A. Hyperband, “A novel bandit-based approach to hyperparameter optimization,” Computer Vision and Pattern Recognition, arXiv: 1603.0656, 2016

# Extra: Parameters after tuning

| | Simple CNN | Tuned CNN | Tuned CNN (data augmentation) |
|---|---|---|---|
| Layer 1 (num kernels) | 64 | 32 | 64 |
| Layer 2 (num kernels) | 64 | 128 | 64 |
| Layer 3 (num kernels) | 128 | 192 | 64 |
| Layer 4 (num kernels) | 128 | 98 | 96 |
| kernel size | 5 | 6 | 4 |
| Dropout | 0,2 | 0,25 | 0,25 |
| Learning rate | 0,001 | 0,001 | 0,0001 |



# EXTRA: ML classifiers tuning

**linear svm**: C: {0.01, 0.1, 1, 10}
**svm rbf**: C: {0.01, 0.1, 1, 10}, gamma: {0.01, 0.001, 0.000}

**mlp**:

- optimizer=adam
- learning_rate=1e-4
- batch_size=32
- max_epochs=200 with early stopping
- activations all relu
- last layer softmax(8 classes)
- cross entropy loss

**network architectures**: [(512, 256), (512, 32), (512,)]
**l2 regularization alpha**: [0.01, 0.03, 0.05]

| N | svm linear | svm rbf | mlp |
|---|---|---|---|
| FC2, PCA=0.99 | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.03, 'hidden_layer_sizes': (512, 32)} |
| FC2, PCA=0.95 | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.05, 'hidden_layer_sizes': (512, 32)} |
| FC2, PCA=0.90 | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.05, 'hidden_layer_sizes': (512, 256)} |

Table 8: hyperparameters result for the VGG16 FC2 with different values of pca.

| Cut Level | svm linear | svm rbf | mlp |
|---|---|---|---|
| avg_pool | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.05, 'hidden_layer_sizes': (512, 32)} |
| conv5_block1_2_relu | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.01, 'hidden_layer_sizes': (512,)} |

Table 12: hyperparameters result for different cut of ResNet50 with PCA=0.90.

| Cut Level | svm linear | svm rbf | mlp |
|---|---|---|---|
| Fc2 | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.05, 'hidden_layer_sizes': (512, 256)} |
| Fc1 | {'C': 0.01} | {'C': 10, 'gamma': 0.0001} | {'alpha': 0.05, 'hidden_layer_sizes': (512, 32)} |
| block5_pool | {'C': 0.01} | {'C': 1, 'gamma': 0.0001} | {'alpha': 0.03, 'hidden_layer_sizes': (512, 256)} |

Table 10: hyperparameters result for different cut level of VGG16 with PCA=0.90.