21st February 2023

SoundFusion: Music Recognition by Information Fusion of Different Representations

DLNN: PROJECT PRESENTATION

LORENZO AUSILIO

MELIKA KESHAVARZ

GEORGE P. PRODAN

Summary

- **Introduction**
- **Processing Pipeline**
- **Signals and features**
  - Audio representations
  - Dataset
- **Learning Framework**
  - 2D architectures
  - Transfer Learning
  - 1D architectures
  - Information Fusion
- **Results**
  - Regularization, GS, loss plots
  - Width Scaling
  - Confusion matrices
  - Transfer Learning
  - SoundFusion
- **Conclusions**

INTRODUCTION

Music genre classification

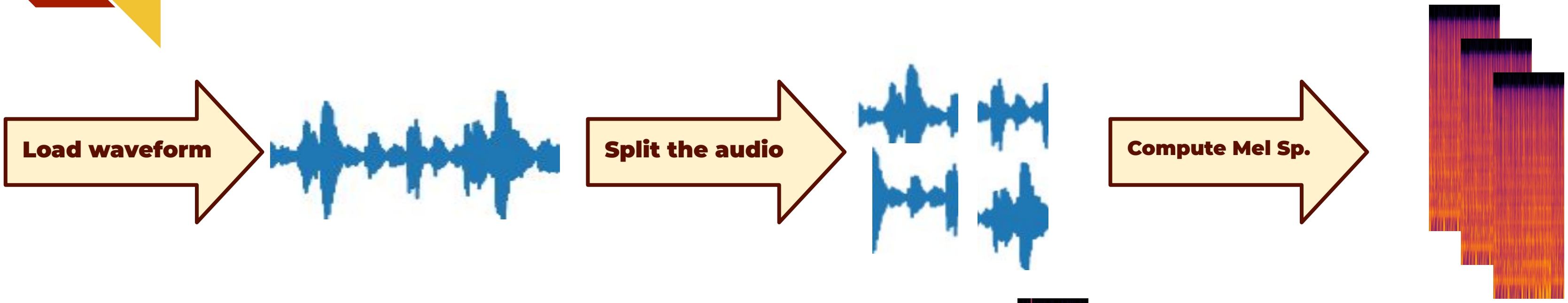
Motivation

- the exponential growth of musical data has stimulated the development of new tools
- one of the most popular research topics in MIR
- applied in music streaming or music shopping platforms

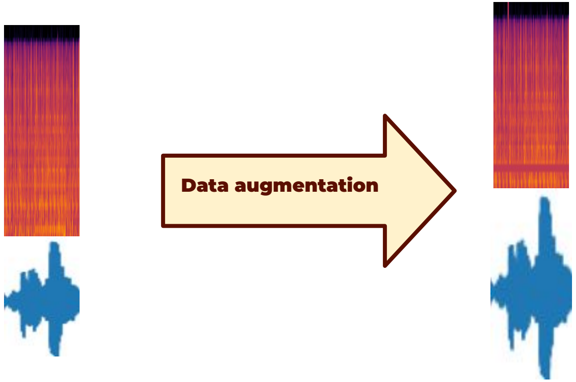
Contributions

- Performance study of different architectures and model scales
- Study of regularization techniques
- We proposed an architecture based on information fusion
- Testing transfer learning

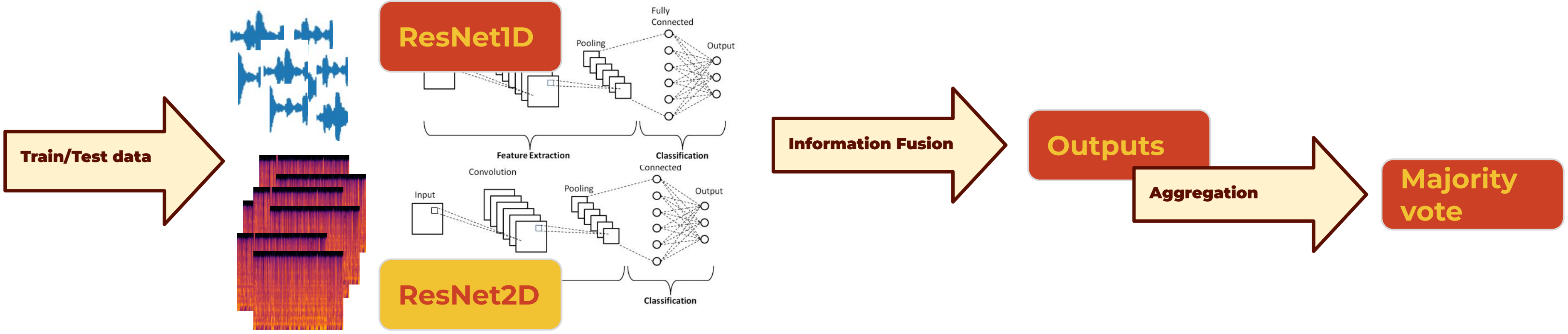
PROCESSING PIPELINE



Optionally:



PROCESSING PIPELINE



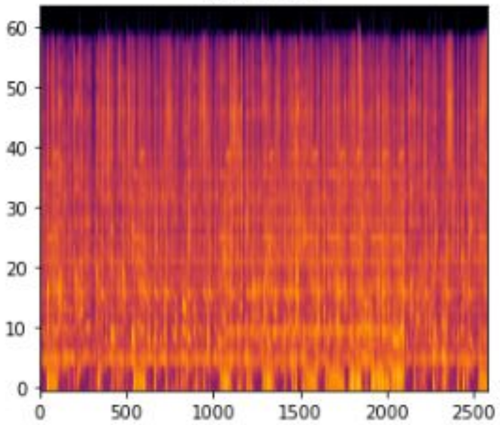
AUDIO REPRESENTATIONS : Raw waveform

Raw audio waveforms

- Loaded using *torchaudio* library
- Mono audio & 22050Hz
- Each clip is cut into 5s subsamples
- Overlap between clips : 25%

AUDIO REPRESENTATIONS : Mel spectrograms

Mono/stereo	Sampling rate (Hz)	Mel Filter Banks	Window size	Hop length	Window function
Stereo	44100	64	1024	512	Hann



DATASET: FMA

Free Music Archive (FMA)

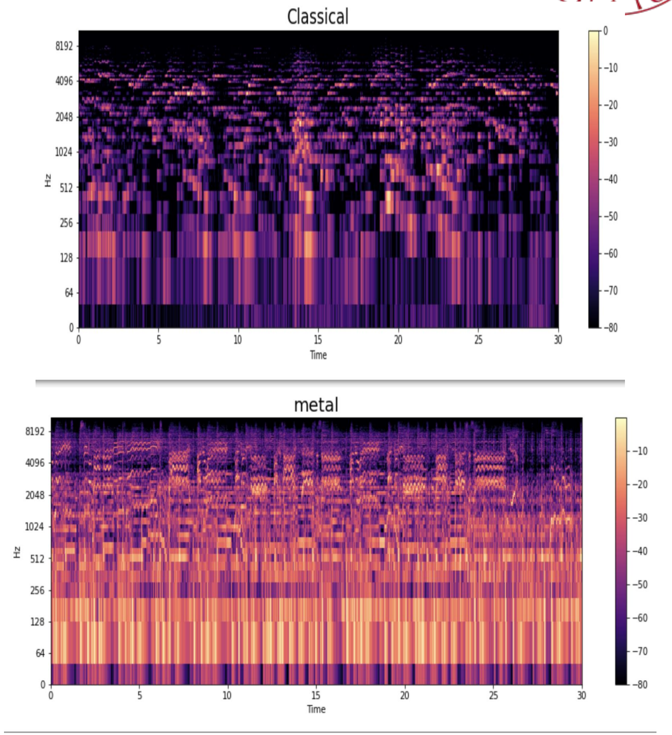
8 genres of music

FMA-small: 8000 clips of 30 seconds

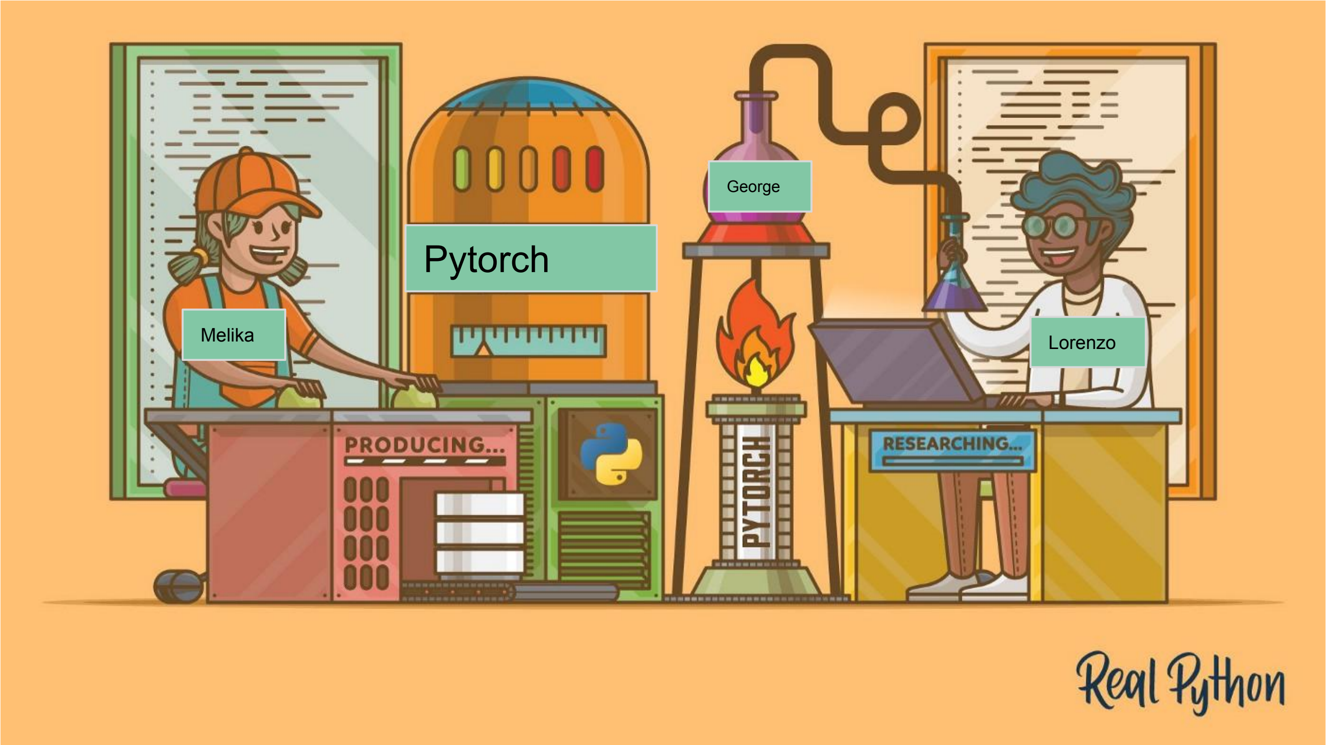
Split the dataset into 80/10/10

Stratified split

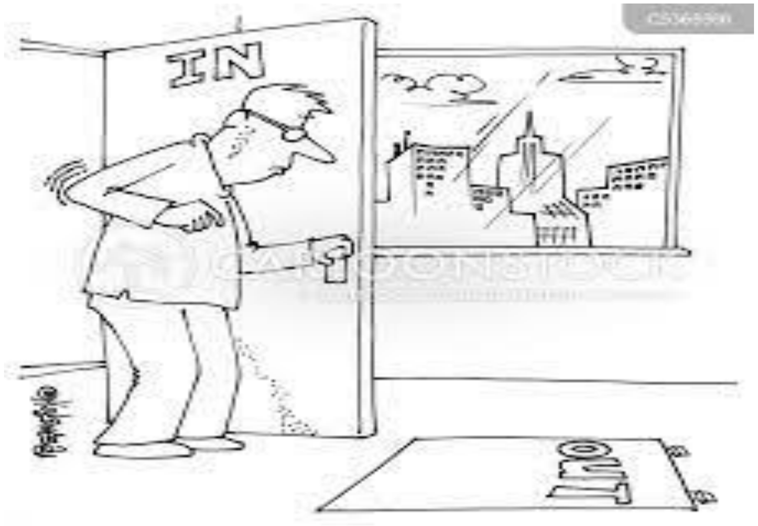
Class distribution in the three sets is representative of the class distribution in the whole dataset

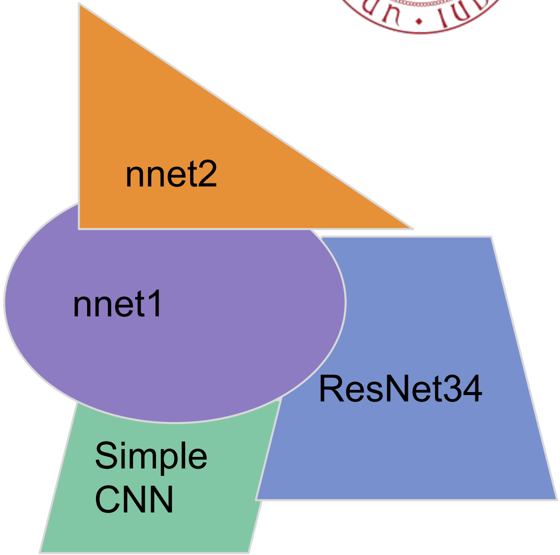


Learning Framework



2D architectures





1. SimpleCNN

- **Simplest model we implemented**
- **25472 parameters**
- **4 convolutional blocks**
    - First with 5x5 Kernel, 2x2 Stride & Padding
    - Remaining : 3x3 Kernel, 2x2 Stride & 1x1 Padding
- **Batch normalisation & ReLU**
- **Final linear layer**

```
SimpleCNN
├─Sequential: 1-1
│        └─Conv2d: 2-1
│        └─ReLU: 2-2
│        └─BatchNorm2d: 2-3
│        └─Conv2d: 2-4
│        └─ReLU: 2-5
│        └─BatchNorm2d: 2-6
│        └─Conv2d: 2-7
│        └─ReLU: 2-8
│        └─BatchNorm2d: 2-9
│        └─Conv2d: 2-10
│        └─ReLU: 2-11
│        └─BatchNorm2d: 2-12
├─AdaptiveAvgPool2d: 1-2
├─Linear: 1-3
```

2. ResNet 2D:

The architecture begins with a convolutional layer that takes in the input image, followed by a series of blocks, each containing multiple convolutional layers and ending with an identity mapping or a projection shortcut. The shortcut connections allow the output of the block to be added to the output of the previous layer, helping to prevent the gradients from becoming too small and allowing for a deeper network.

Each block is made up of multiple convolutional layers with a kernel size of 3x3, followed by batch normalization and a non-linear activation function. The number of filters in each layer is gradually increased as the network gets deeper.

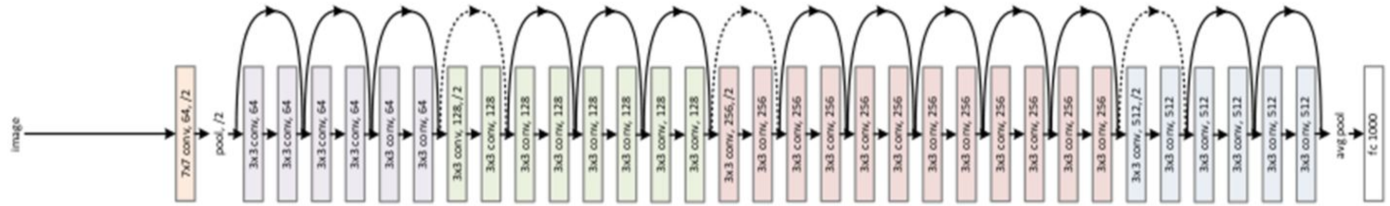
The global average pooling layer aggregates the feature maps of the last convolutional layer into a single vector, which is then fed into a fully connected layer to produce the final output.

Overall, ResNet34 is a highly effective architecture for image recognition tasks and has been widely used in various applications, including object detection, image classification, and segmentation.

Proposed by

Xiangyu Zhang

2015

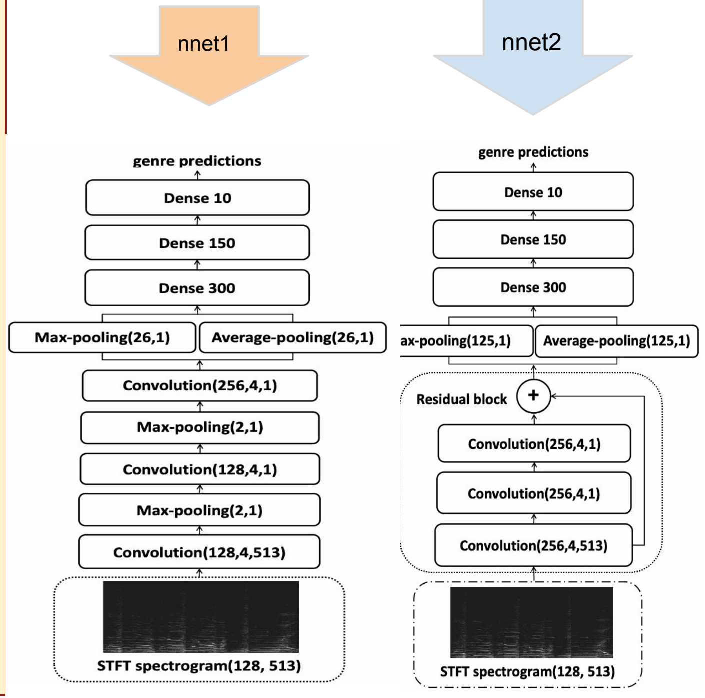


nnet 1 and nnet 2:

Both nnet 1 and nnet 2 are proposed by Weibin Zhang in 2016. These architectures were mainly used for music classifications on the famous GTZAN dataset. Despite their simplicity, they outperformed other architectures.

nnet 1 is a very simple cnn consists of 3 convolutional layers which all were followed by max pooling except after the last layer in which we used a combination of maxpooling and average pooling.

nnet2 is very similar to nnet 1 but it has residual implementation in the architecture, where there is only one residual block with three convolutional layers and a shortcut connection between first and the output of the third layer.



Transfer Learning

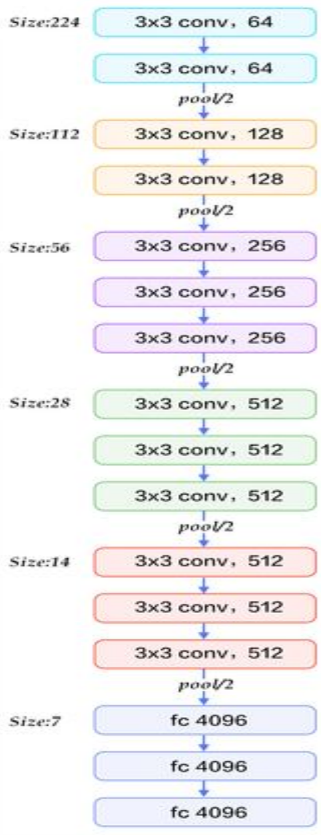
Weight initialization plays an important rule in every CNN, when the weights are randomly chosen, it takes time for the weights to step by step adapt themselves to the learning process features. Adding weights from a similar trained model can both boost the performance and make the model accurate.

For implementing transfer learning we took these 3 steps:

1. **Model without pretrained weights:** We trained the model without pretrained weights on our dataset.
2. **Model with default weights:** We used the default pretrained weights provided by pytorch.(in our case it was ImageNet)
3. **Weights on GTZAN dataset:** We trained a VGG16 model on Gtzan dataset and it's weights for our model.

VGG16

Here we impelemeted VGG16 by Karen Simonyan and Andrew Zisserman.

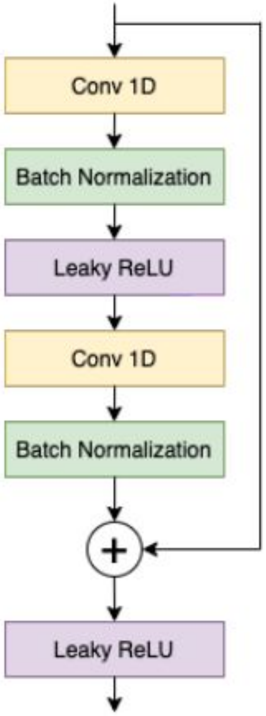


1D ARCHITECTURES

1D ResNet

- 9 residual blocks
- Leaky ReLU activation function

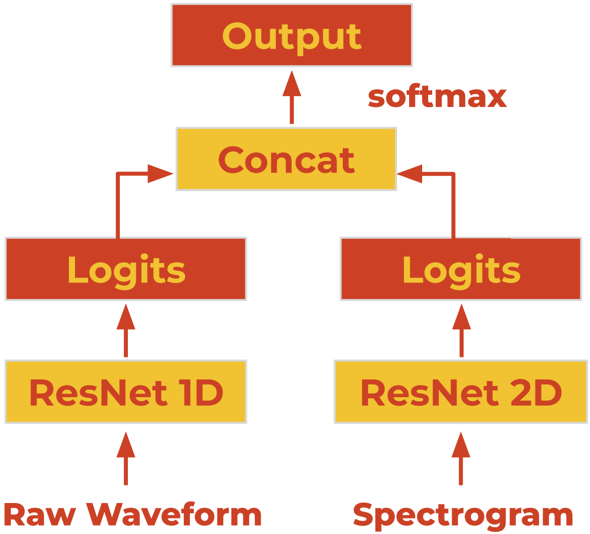
Proposed by Allamy, S. and Lameiras Koerich, A., *1D CNN Architectures for Music Genre Classification*, 2021



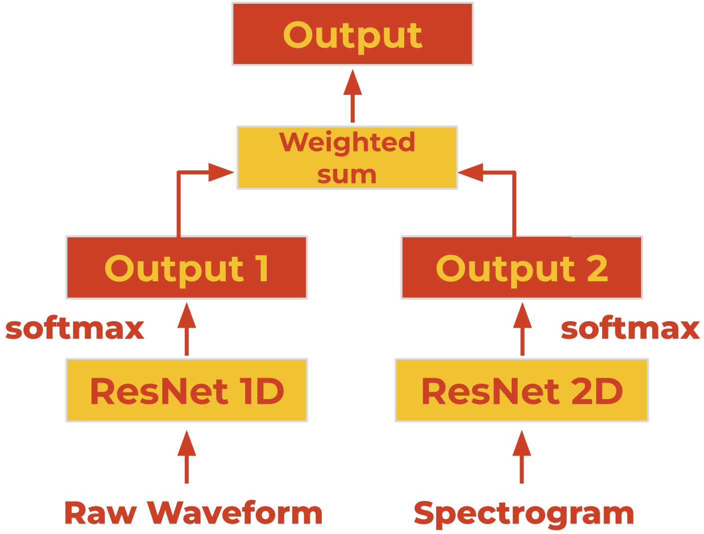
Layer	# Filters	Kernel Size	Pool Size	Stride	Output Shape
Input	-	-	-	-	110,250
Conv1D	128	3	-	3	128×36,750
Res1D	128	3	-	1	128×36,750
MaxPool	-	-	3	3	128×12,250
Res1D	128	3	-	1	128×12,250
MaxPool	-	-	3	3	128×4,083
Res1D	256	3	-	1	256×4,083
MaxPool	-	-	3	3	256×1,361
Res1D	256	3	-	1	256×1,361
MaxPool	-	-	3	3	256×453
Res1D	256	3	-	1	256×453
MaxPool	-	-	3	3	256×151
Res1D	256	3	-	1	256×151
MaxPool	-	-	3	3	256×50
Res1D	256	3	-	1	256×50
MaxPool	-	-	3	3	256×16
Res1D	256	3	-	1	256×16
MaxPool	-	-	3	3	256×5
Res1D	512	3	-	1	512×5
MaxPool	-	-	3	3	512×1
Conv1D	512	1	-	1	512×1
Output	-	-	-	-	10

Trainable parameters: 4,086,794.

Information Fusion



By feature concatenation



By decision weighting

RESULTS

REGULARIZATION TECHNIQUES

Data Augmentation

Spectrograms:

- Time Masking → 2
- Frequency masking → 2
- Augmentation probability → 1
- Maximum mask percentage → 0.3

Raw waveform:

- Gaussian noise with std 0.02
- Modifying volume with gain -12dB & 12dB

Dropout Layers

After the output layer

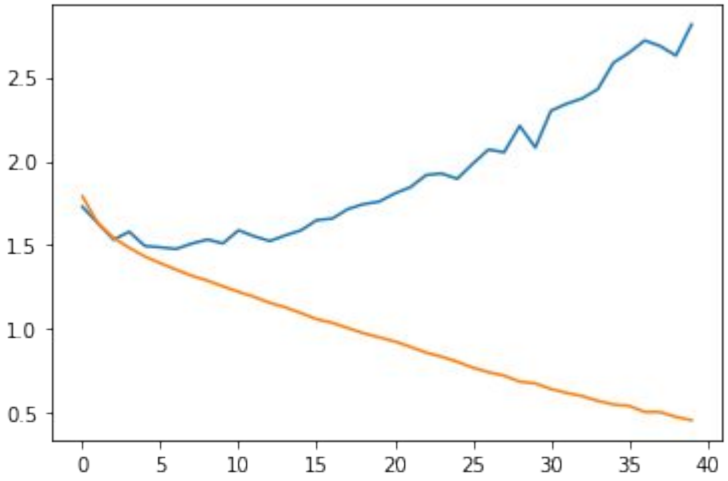
Dropout probability 30-50%

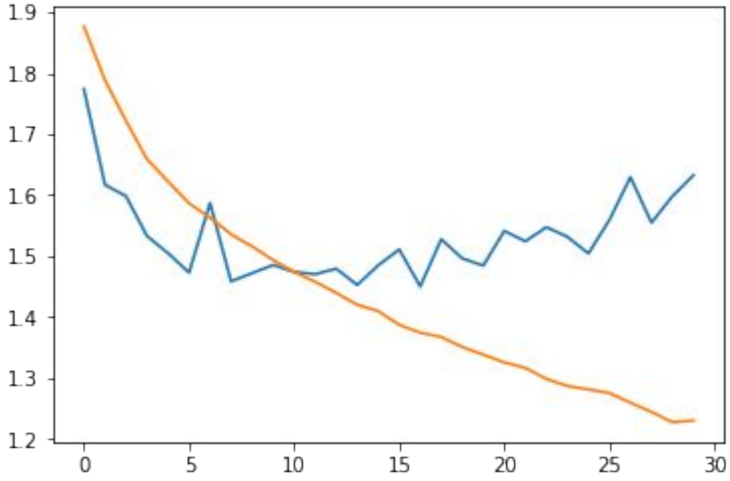
model	regularization	validation	test
ResNet 1D (70k param.)	N/A	0.472	0.380
	dropout	0.515	0.413
	dropout + augmentation	0.507	0.418
ResNet 2D (445k param.)	N/A	0.523	0.485
	dropout	0.520	0.489
	dropout + augmentation	0.531	0.528

nnet 2 training

nnet2 training + regularization

LOSS





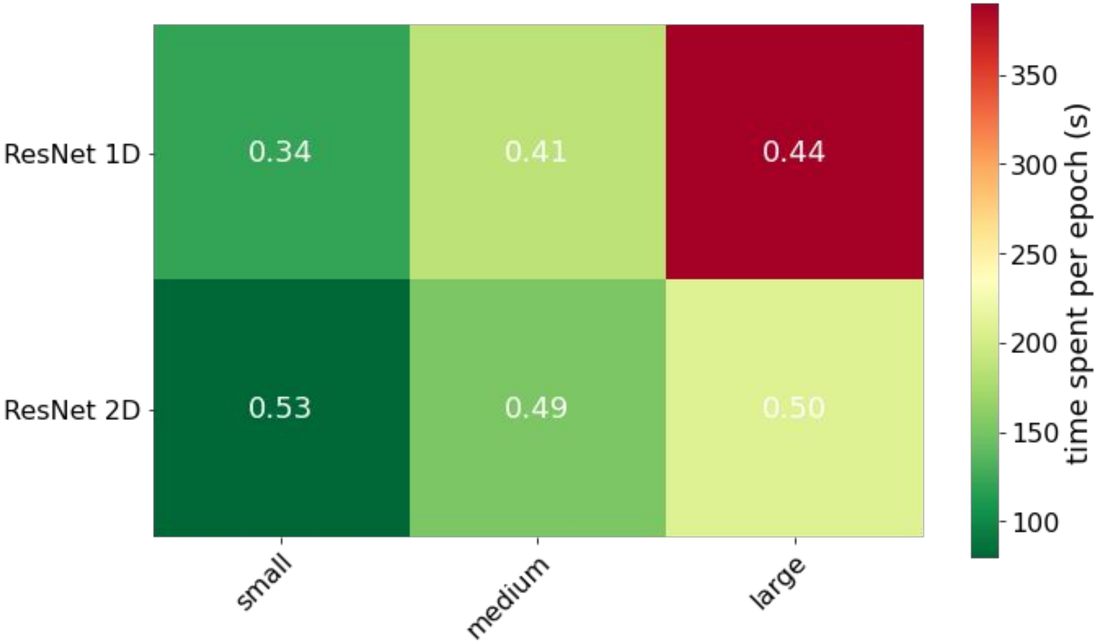
EPOCH

WIDTH SCALING

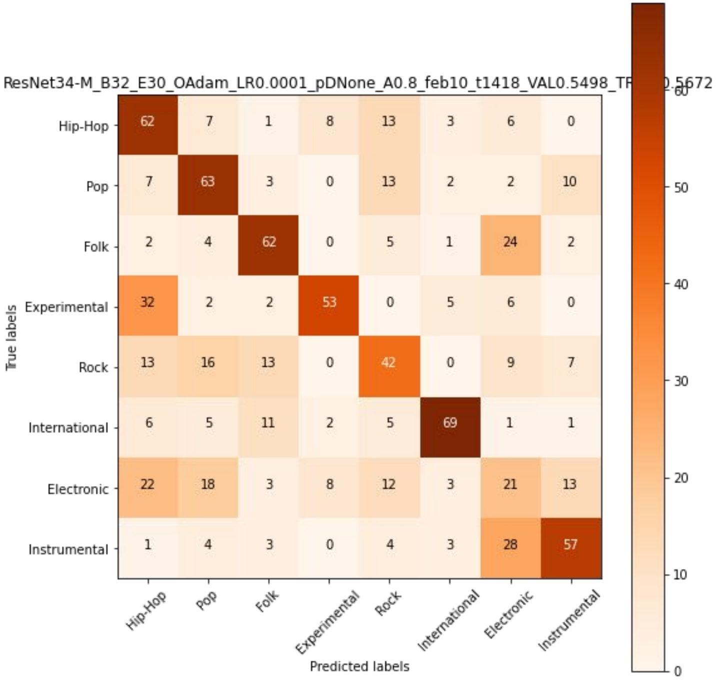
F = 16 - > small

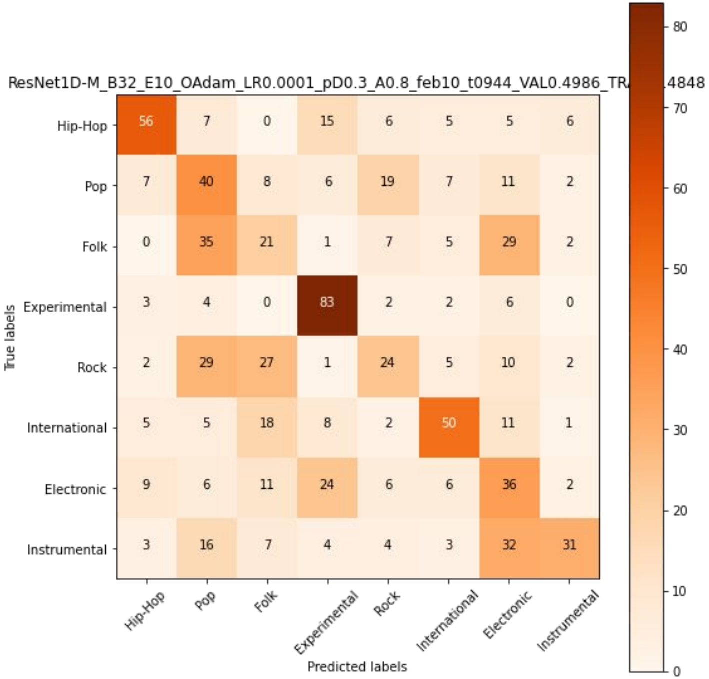
F= 32 -> medium

F = 64 -> large

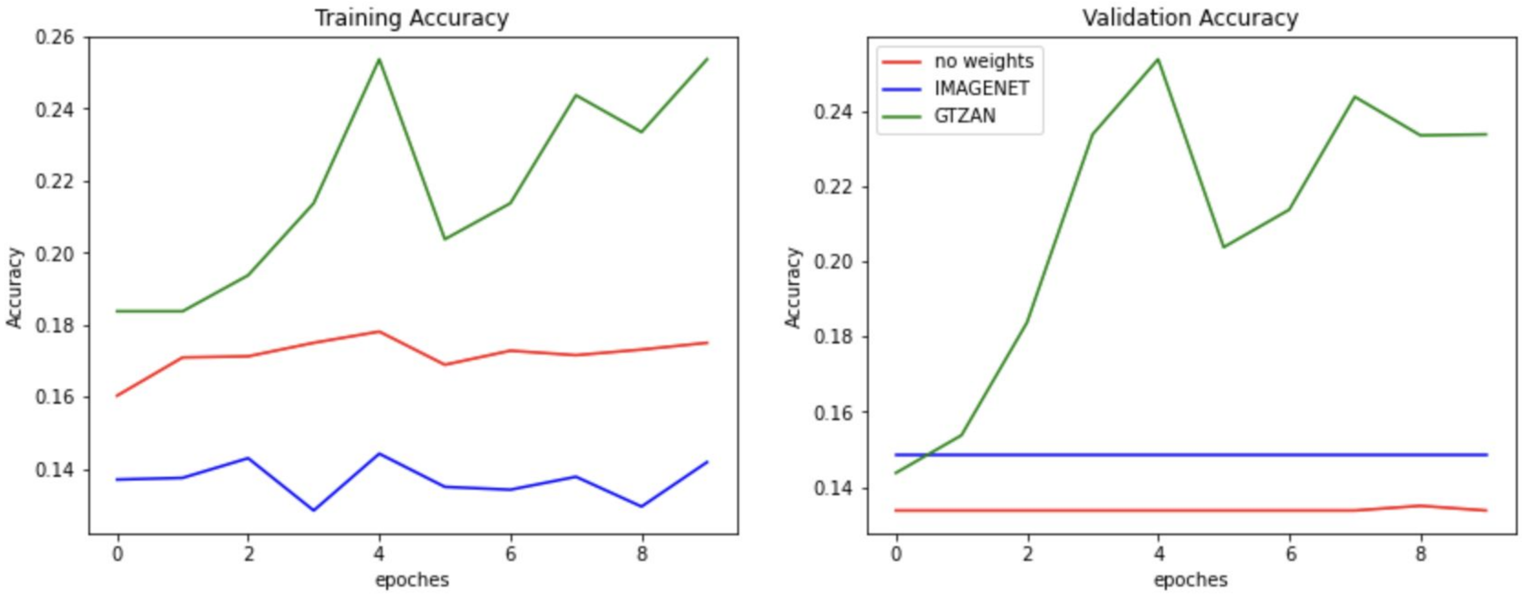


1D - 2D COMPARISON



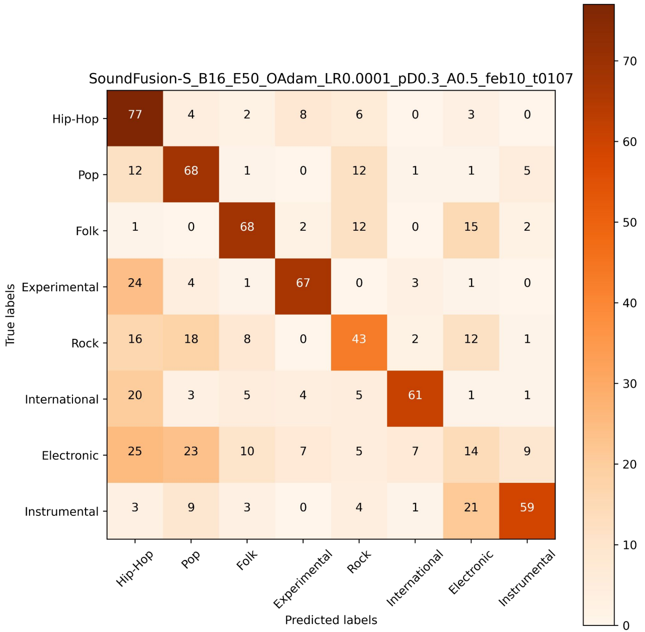


Transfer Learning



SoundFusion performance

Input	Model	Test accuracy	Aggregation
1D	ResNet1D-m [7]	0.406	0.426
2D	pre-trained VGG16	0.492	0.526
	ResNet2D-s [1]	0.528	0.567
	nnet1 [3]	0.344	0.366
	nnet2 [3]	0.475	0.493
	Simple CNN w/o segmentation	0.484	n/a
Mix	**SoundFusion-1**	0.535	0.571
	SoundFusion-2	0.483	0.506



CONCLUSION

- **Smaller** models would be a better option in the FMA-small scenario
- SoundFusion shows an **improvement** compared to the baselines
- Learning from **2D features** is a better option in the FMA-small scenario

Further work

- Fine-tune other hyper-parameters such as the learning rate, optimizer or batch size
- Implement cross-validation
- Try mixing different architectures in SoundFusion (i.e. a larger model of ResNet1D, ResNet18 instead of ResNet34, and so on)

Thank you for your attention!

Do you have any questions for us?