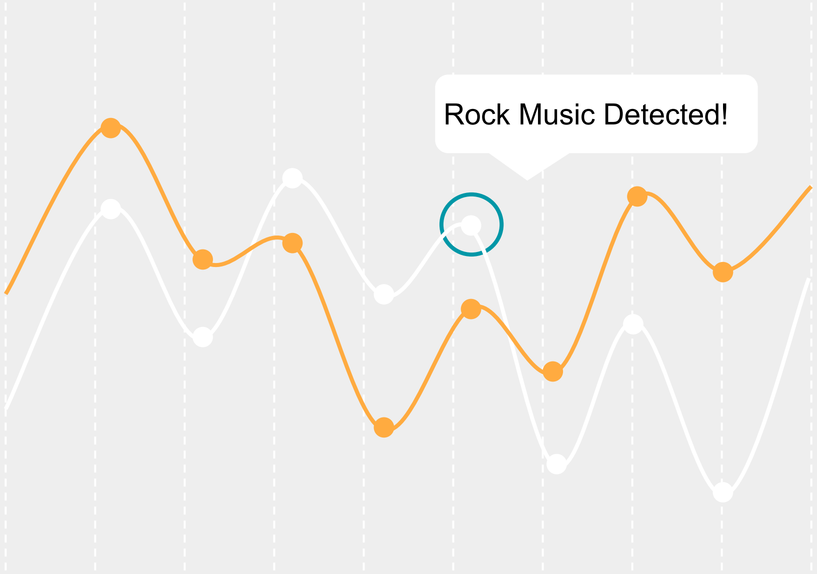
Music Genre Recognition from Audio Tracks and Metadata

CS464 Group 9 Project Final Presentation

Can Kocagil, Berkay Erkan, Efe Eren Ceyani, Mehmet Calıskan, Hammad Khan Musakhel



Plan of Attack

1) What is Music Genre Recognition?

2) The Dataset

3) Metadata-level Computing
   a) Metadata Level Analysis
   b) Metadata Manifold Learning
   c) Machine Learning for Metadata
   d) Meta Learning for Metadata
   e) Model Explainability

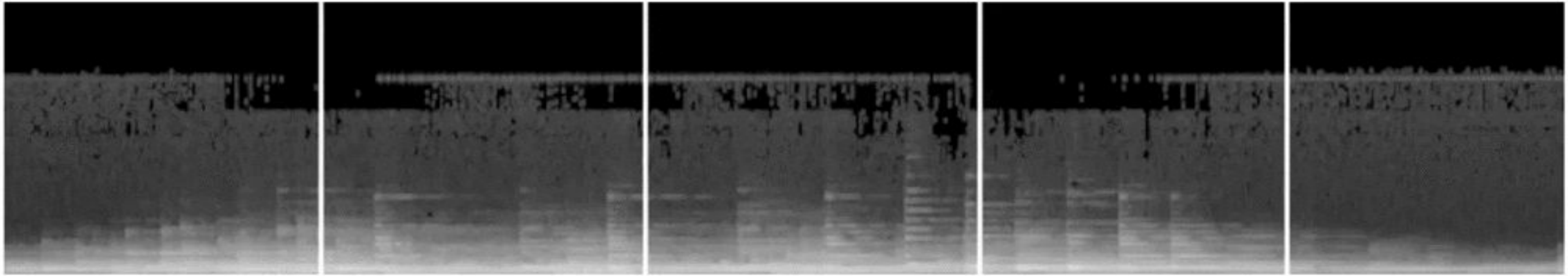
4) Audio-level Computing
   a) Audio Level Signal Analysis
   b) Mel Spectrum Computing
   c) 2-D CNNs for Spectrums
   d) Vision Transformer for Spectrums

What is Music Genre Recognition?

Automatic recognition of audio tracks from music waves has been a challenging task in Music Information Retrieval (MIR).

The mathematical representation of natural music, composed of rhythm, tones, intervals, patterns, and harmonies, is a complex subject for human specialists as there is a geometric interpretation in the humming of the strings and inherent subjective nature.

That's classical !

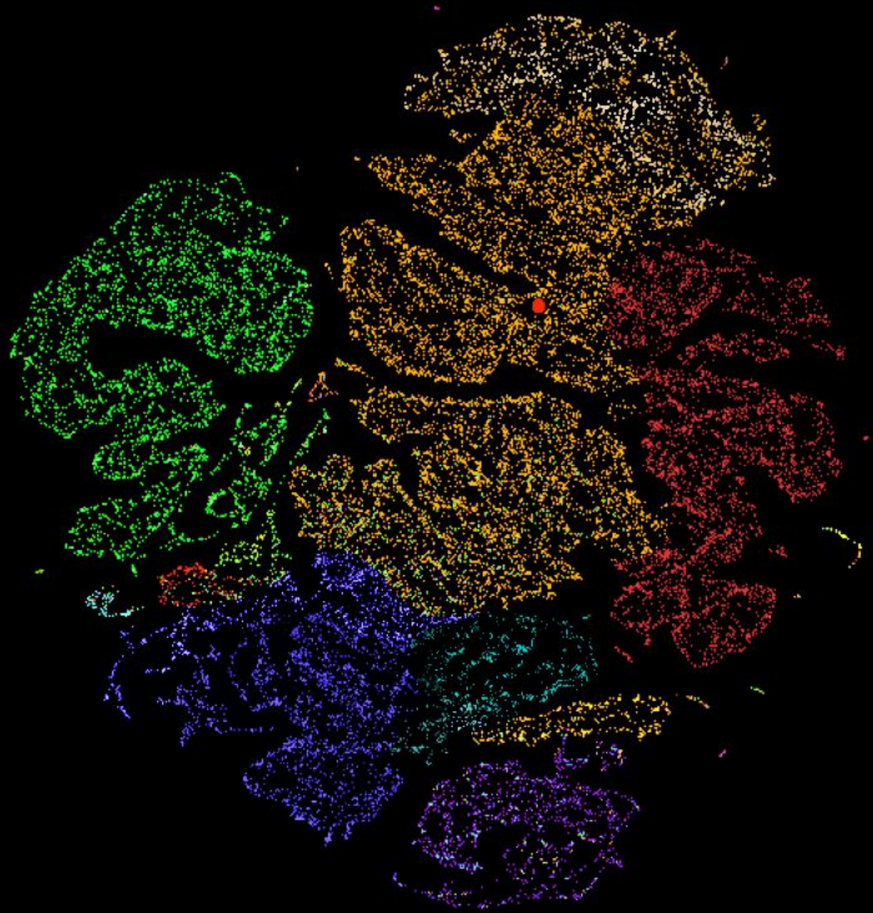


The Dataset

As the dynamics of common genres are temporally varied, representable and scalable datasets are required to perform cognitive tasks in Music Information Retrieval (MIR).
The Free Music Archive (FMA) [1] is a dataset for everyday MIR analysis tasks and is supervised for browsing, searching, and organizing extensive music collections. The FMA dataset is a MIR benchmark dataset that consists of 106,574 tracks from 16,341 artists and 14,854 albums, arranged in a hierarchical taxonomy of 161 genres

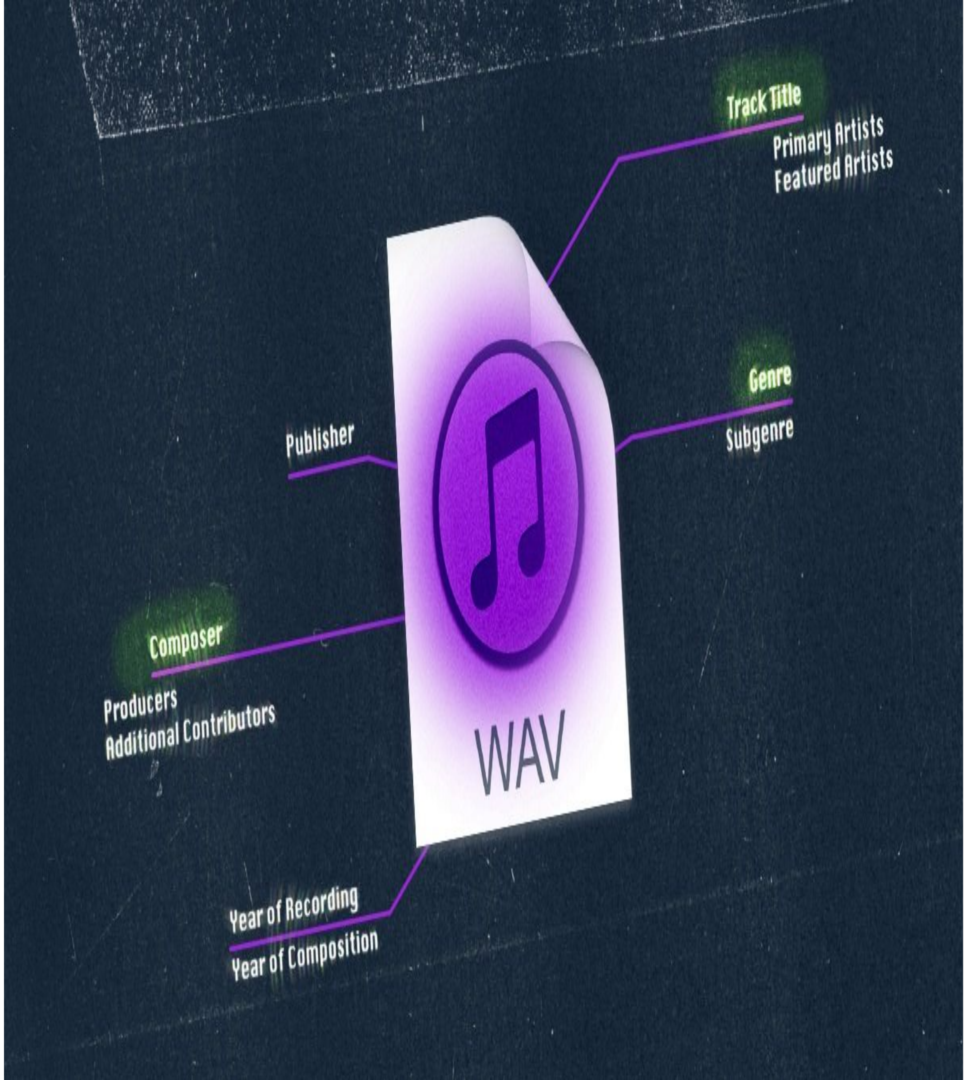
Class Names

- 'Pop'
- 'Folk'
- 'Rock'
- 'Hop'
- 'Experimental'
- 'International'
- 'Electronic'
- 'Instrumental'

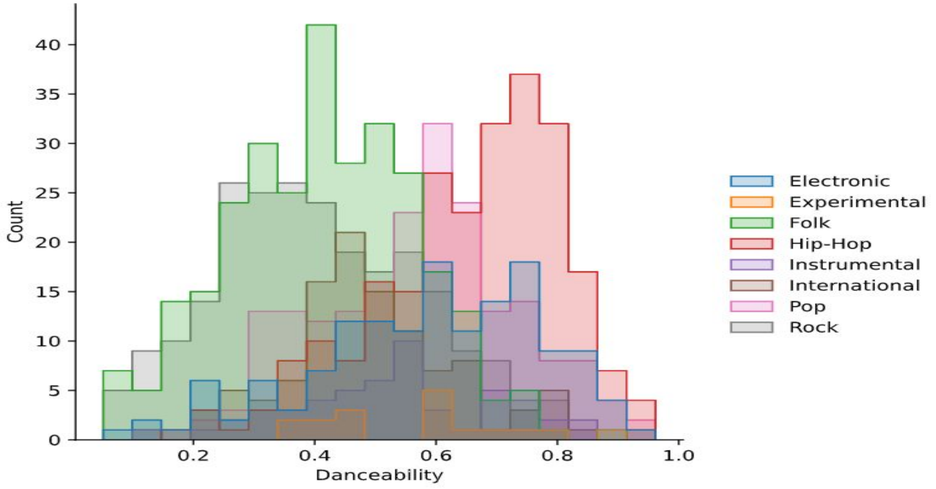


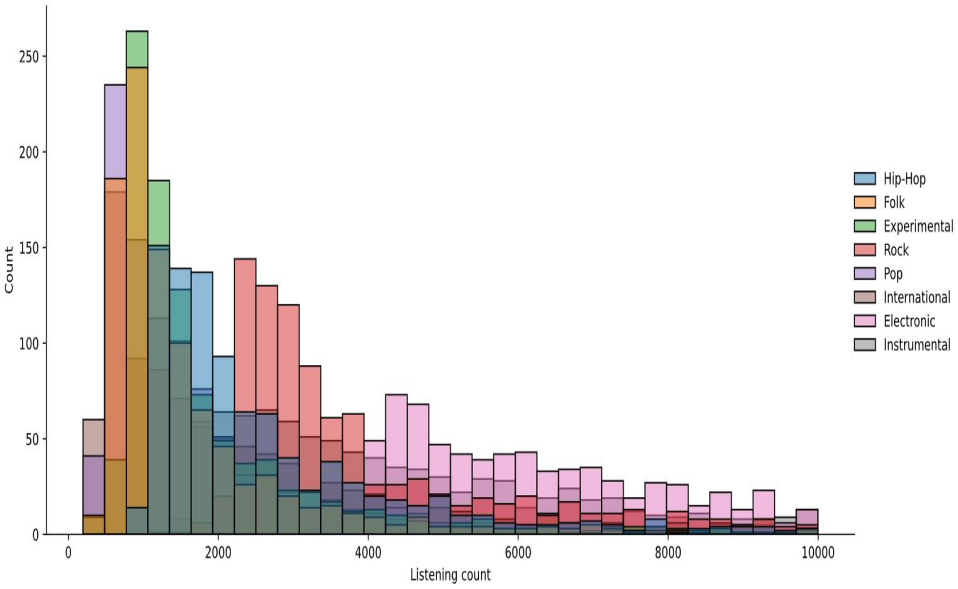
Metadata-level Computing

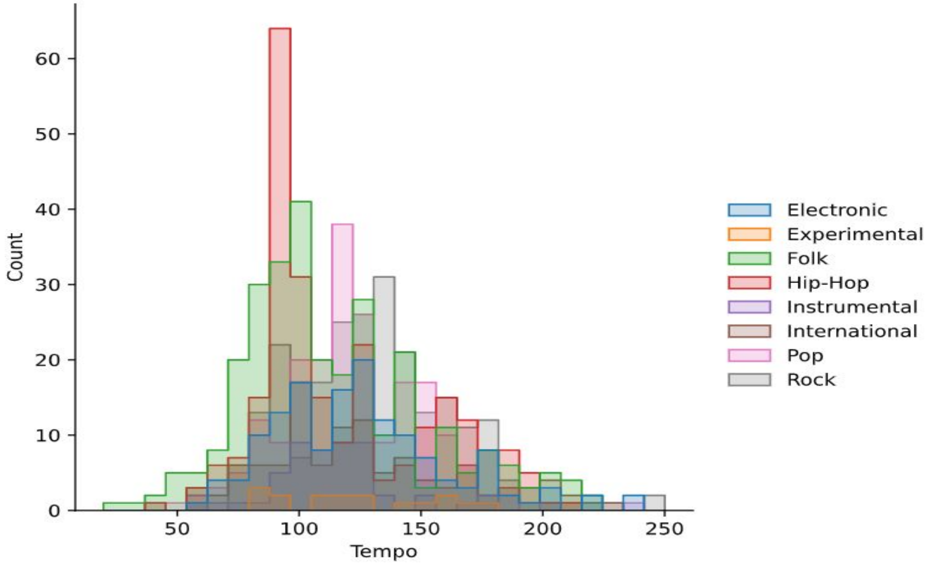
a) Metadata Level Analysis
b) Metadata Manifold Learning
c) Machine Learning for Metadata
d) Meta Learning for Metadata
e) Model Explainability



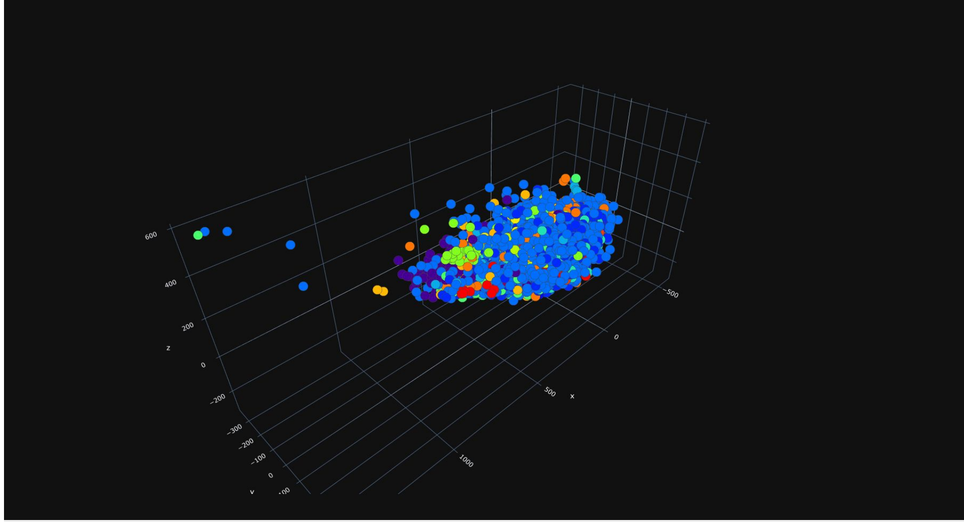
Metadata Level Analysis



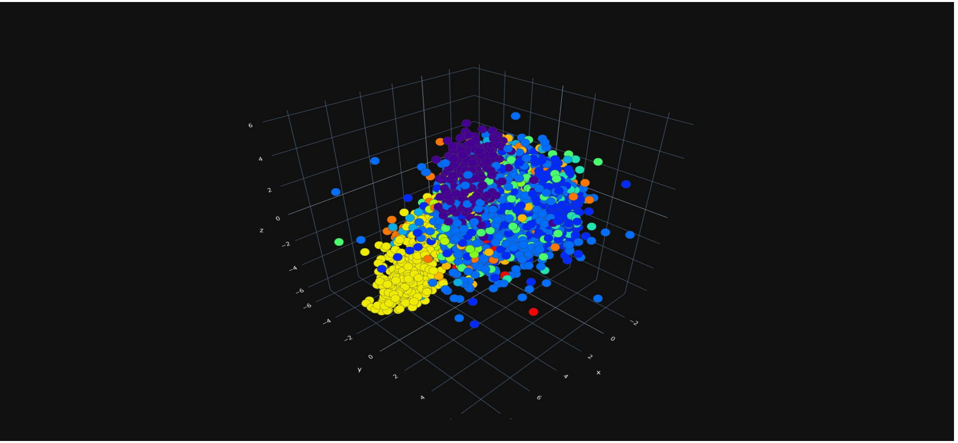




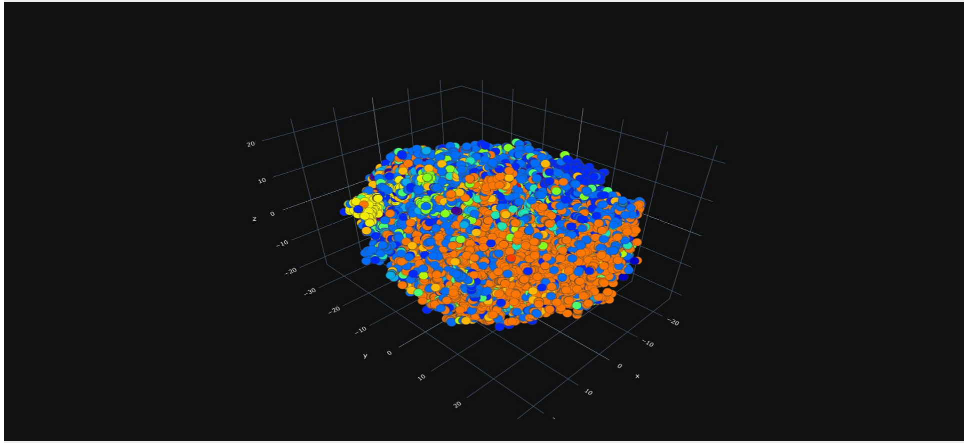
Metadata Manifold Learning



3-D PCA Projection of Metadata



3-D LDA Projection of Metadata

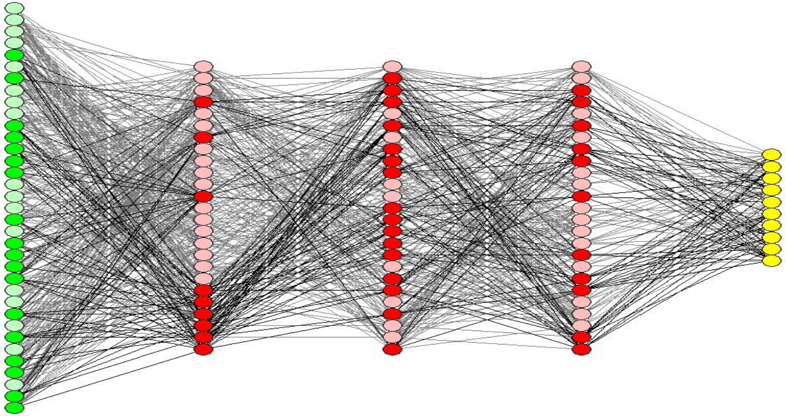


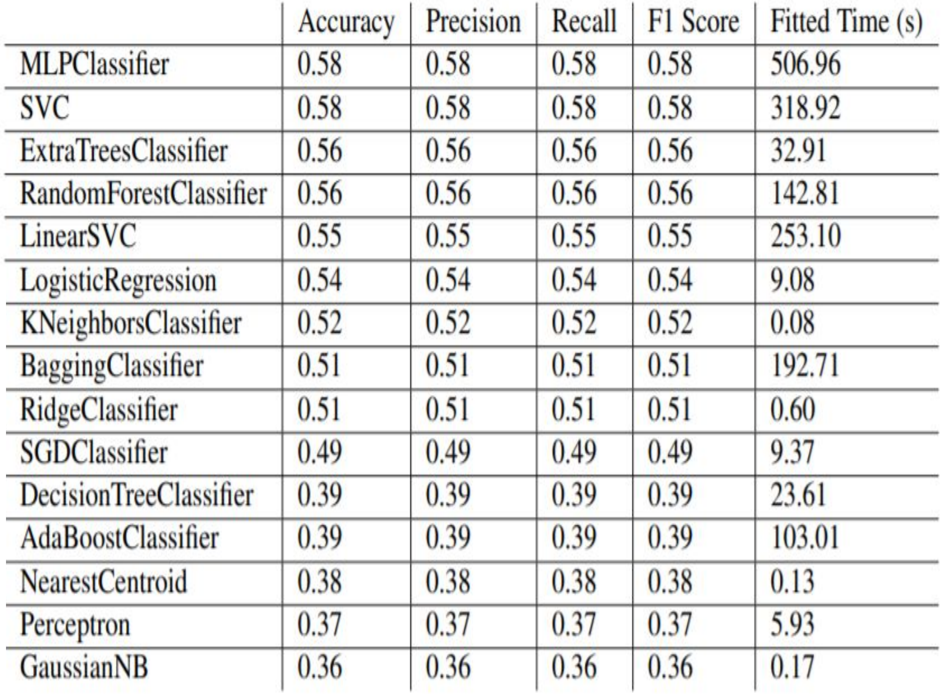
3-D t-SNE Projection of Metadata

Machine & Ensemble Learning for Metadata

Around 55-58% Metadata-level Accuracy

MLP is winner!





	Accuracy	Precision	Recall	F1 Score	Fitted Time (s)
MLPClassifier	0.58	0.58	0.58	0.58	506.96
SVC	0.58	0.58	0.58	0.58	318.92
ExtraTreesClassifier	0.56	0.56	0.56	0.56	32.91
RandomForestClassifier	0.56	0.56	0.56	0.56	142.81
LinearSVC	0.55	0.55	0.55	0.55	253.10
LogisticRegression	0.54	0.54	0.54	0.54	9.08
KNeighborsClassifier	0.52	0.52	0.52	0.52	0.08
BaggingClassifier	0.51	0.51	0.51	0.51	192.71
RidgeClassifier	0.51	0.51	0.51	0.51	0.60
SGDClassifier	0.49	0.49	0.49	0.49	9.37
DecisionTreeClassifier	0.39	0.39	0.39	0.39	23.61
AdaBoostClassifier	0.39	0.39	0.39	0.39	103.01
NearestCentroid	0.38	0.38	0.38	0.38	0.13
Perceptron	0.37	0.37	0.37	0.37	5.93
GaussianNB	0.36	0.36	0.36	0.36	0.17

Table 1: Machine Learning Models and Performance Table

Meta Learning for Metadata

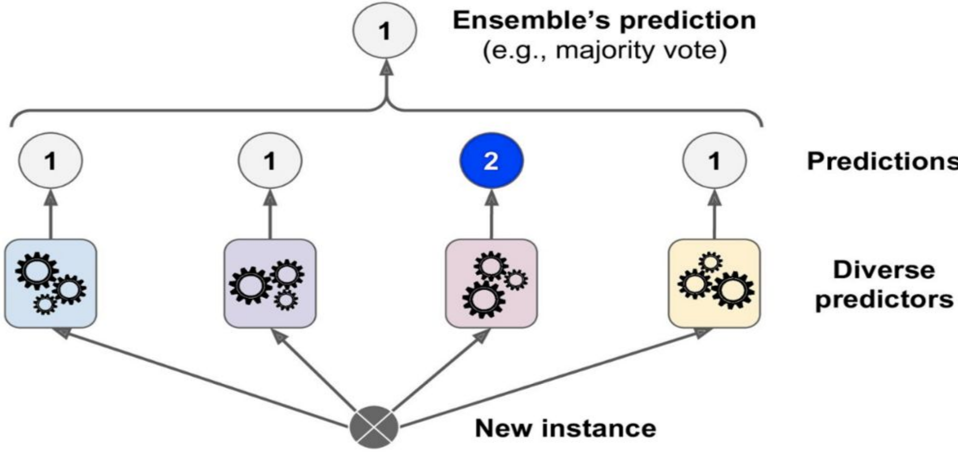
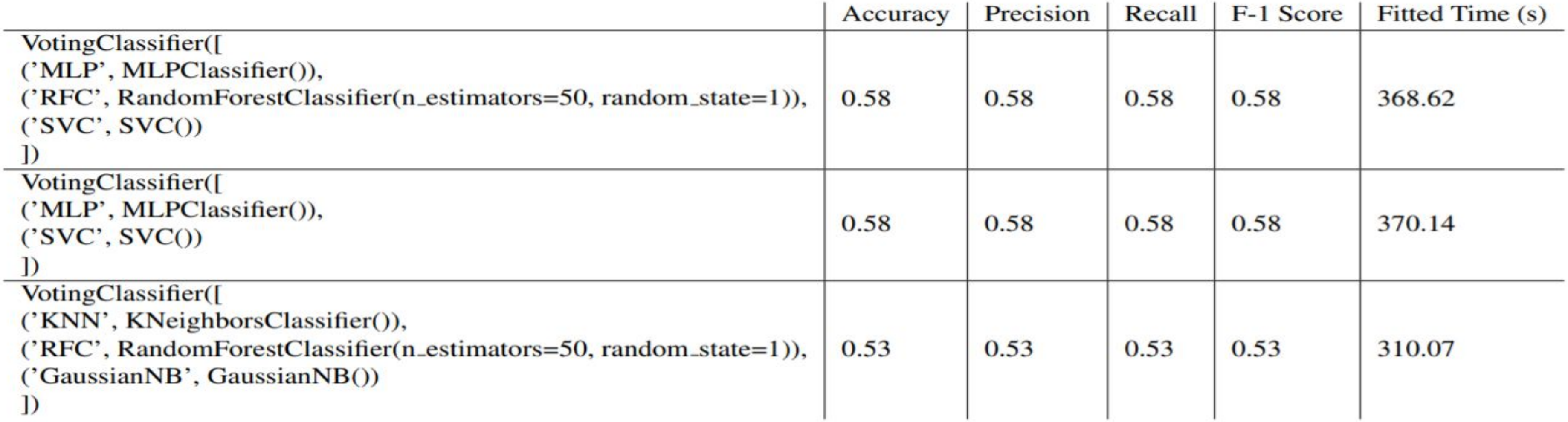


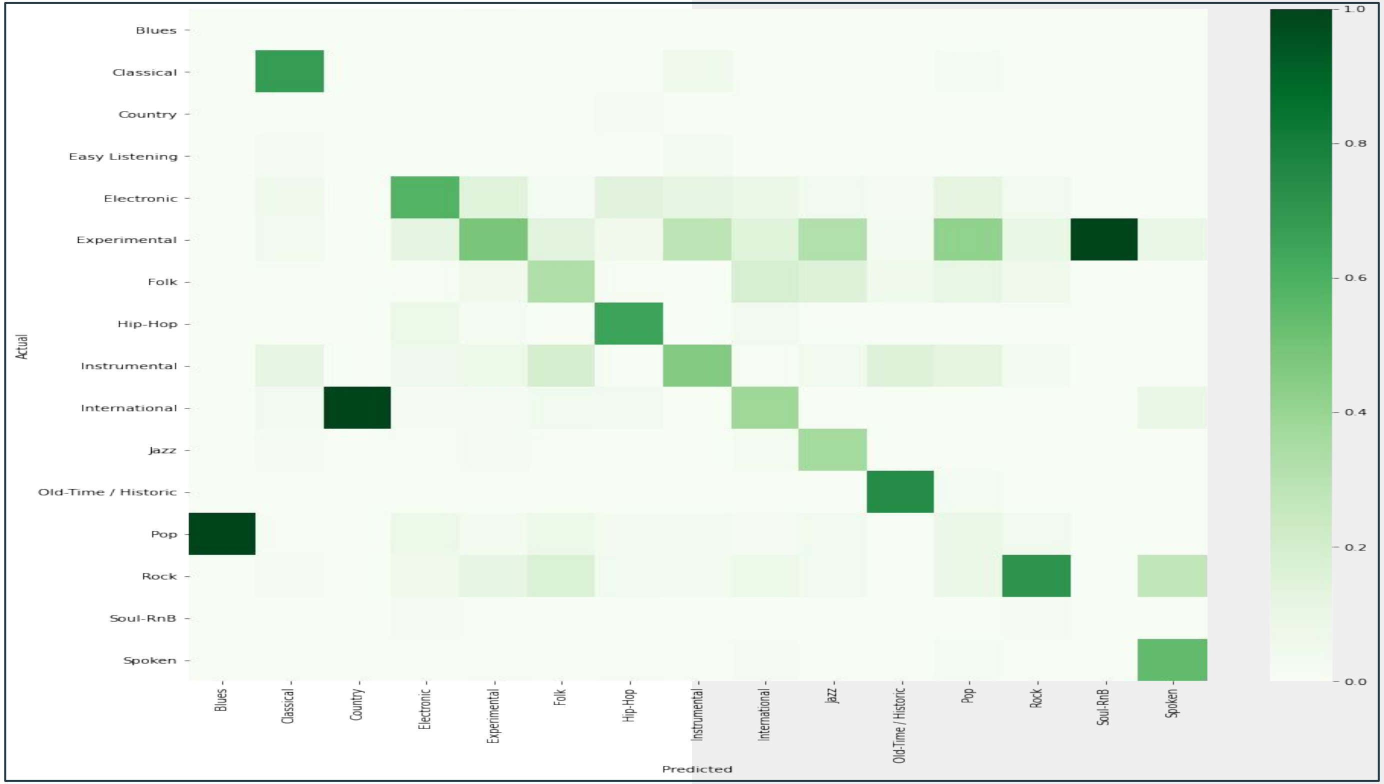
Figure 7-2. Hard voting classifier predictions



	Accuracy	Precision	Recall	F-1 Score	Fitted Time (s)
VotingClassifier([(’MLP’, MLPClassifier()), (’RFC’, RandomForestClassifier(n_estimators=50, random_state=1)), (’SVC’, SVC())])	0.58	0.58	0.58	0.58	368.62
VotingClassifier([(’MLP’, MLPClassifier()), (’SVC’, SVC())])	0.58	0.58	0.58	0.58	370.14
VotingClassifier([(’KNN’, KNeighborsClassifier()), (’RFC’, RandomForestClassifier(n_estimators=50, random_state=1)), (’GaussianNB’, GaussianNB())])	0.53	0.53	0.53	0.53	310.07

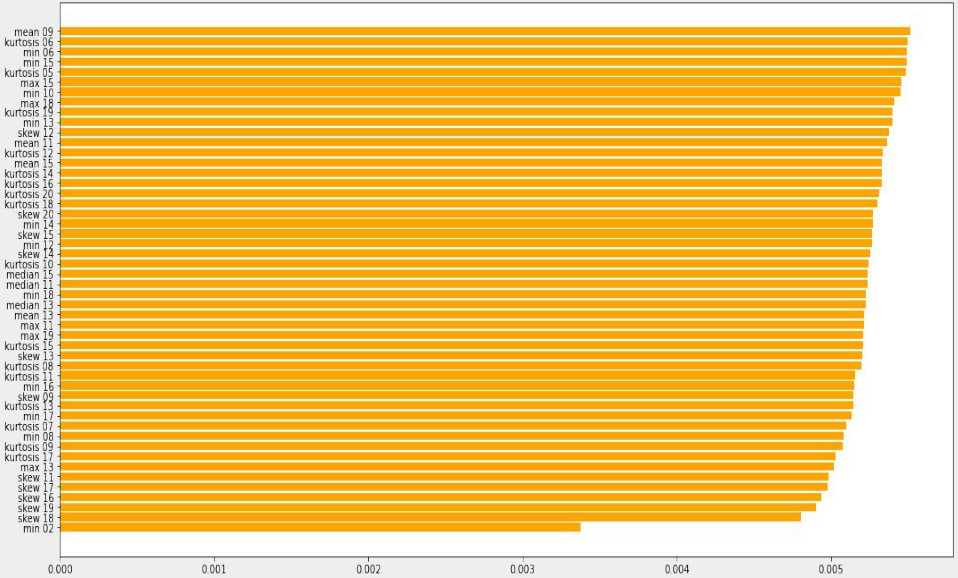
Table 2: Meta Learning Models and Performance Table

Confusion Matrix

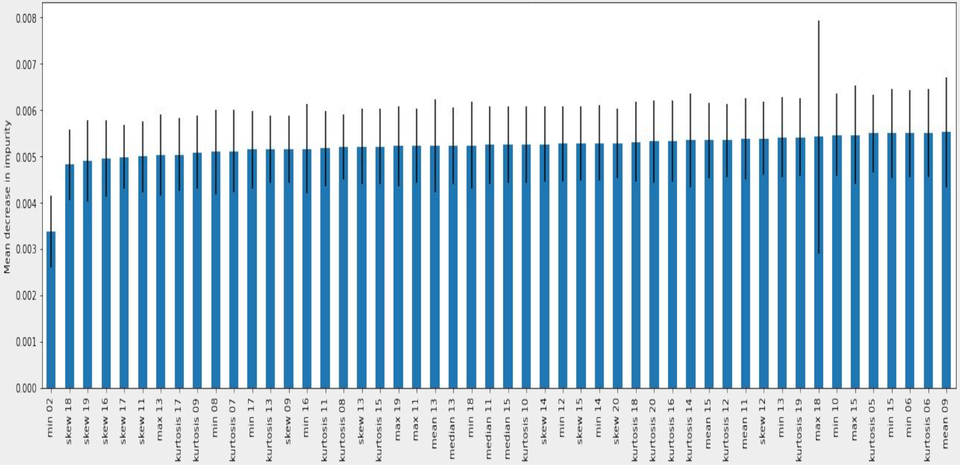


Model Explainability

Random Forest is used for explainability purposes



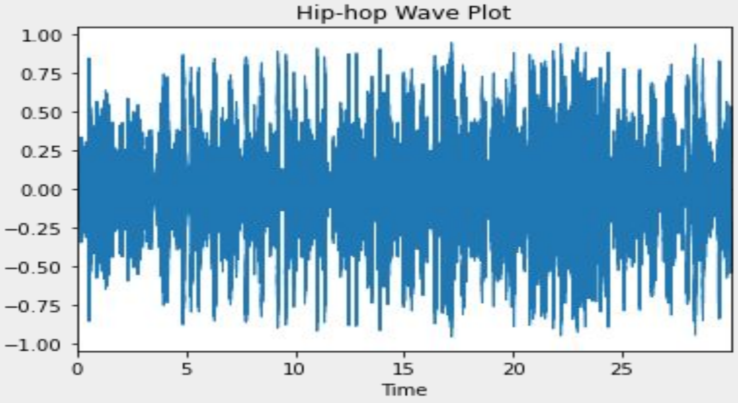
Feature importances using MDI

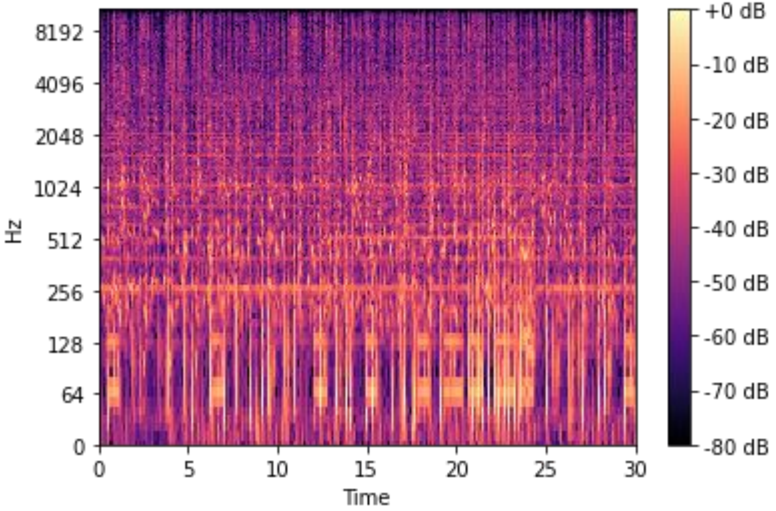


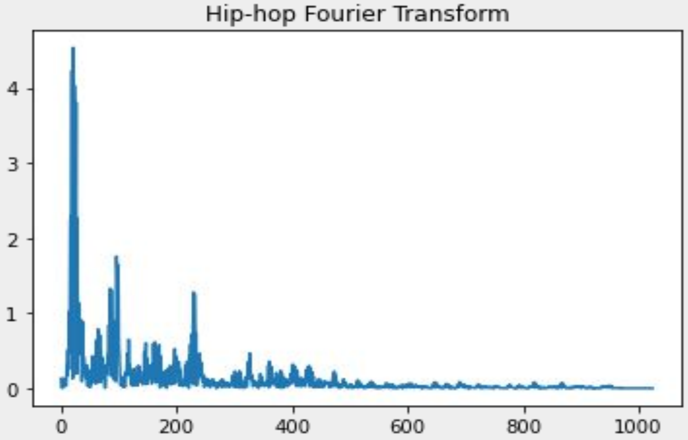
Audio-level Computing

a) Audio Level Signal Analysis
b) Mel Spectrum Computing
c) 2-D CNNs for Spectrums
d) Vision Transformer for Spectrums

Audio Level Signal Analysis

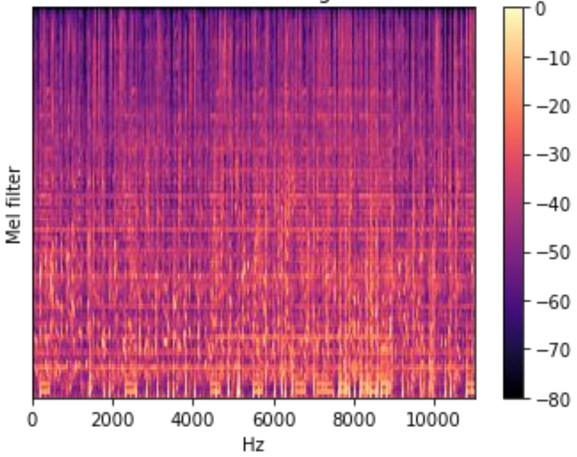




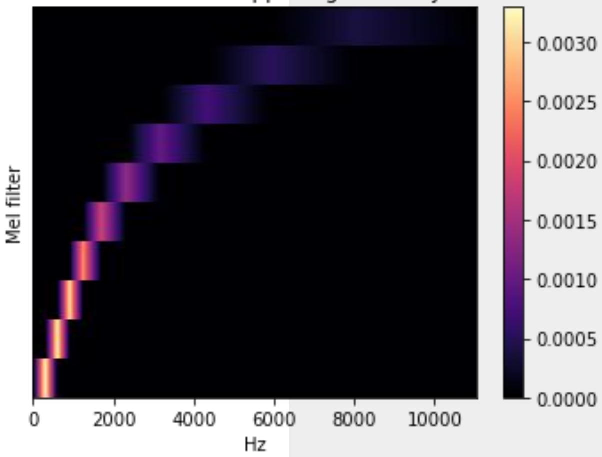


Audio Level Signal Analysis

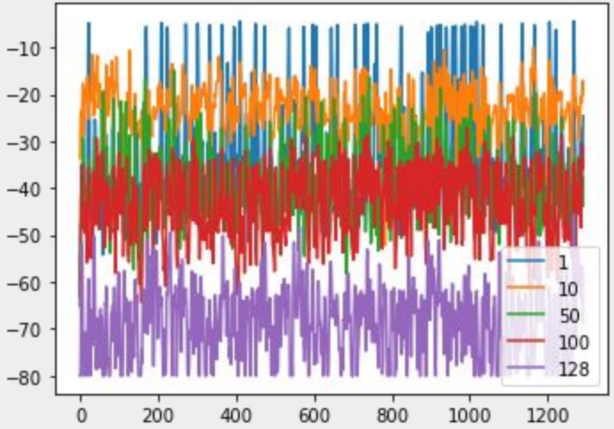
It partitions the Hz scale into bins, and transforms each bin into a corresponding bin in the Mel Scale, using overlapping triangular filters



1. Our filter bank for converting from Hz to mels.

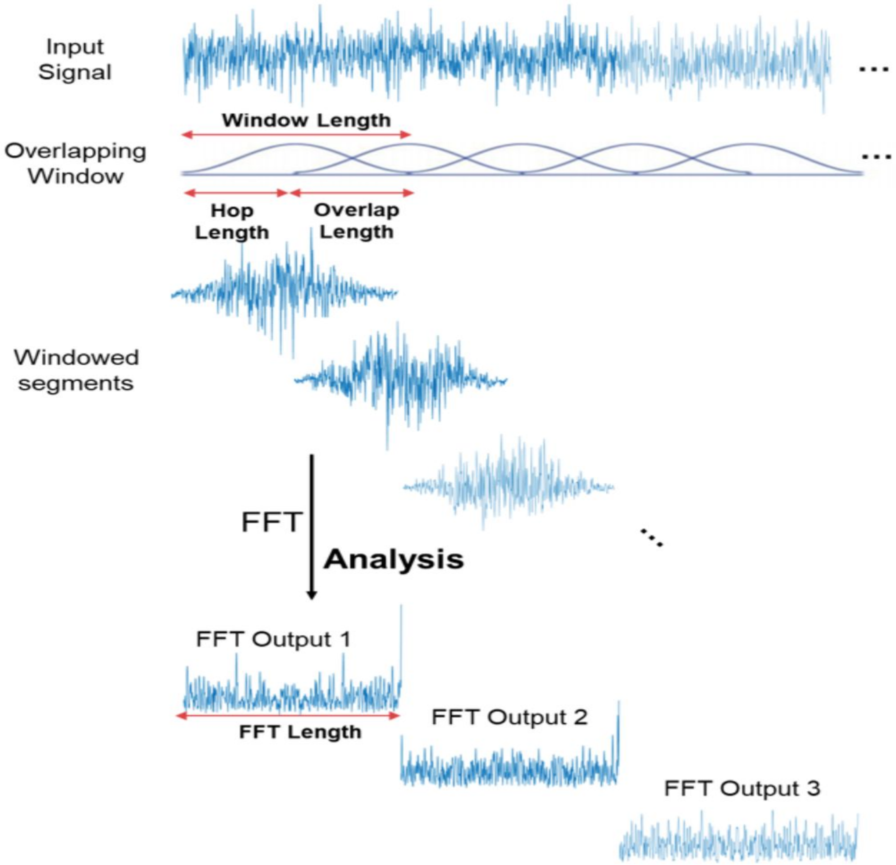


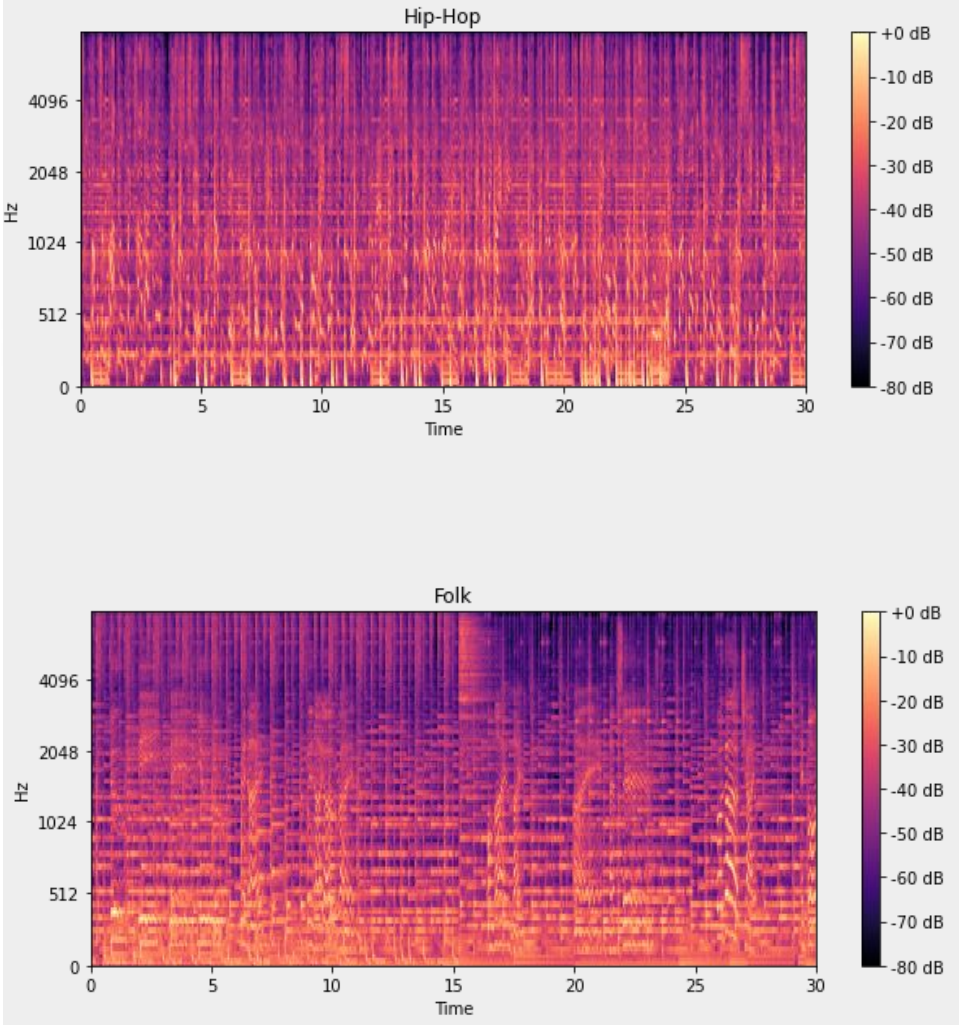
2. Easier to see what is happening with only 10 mels.



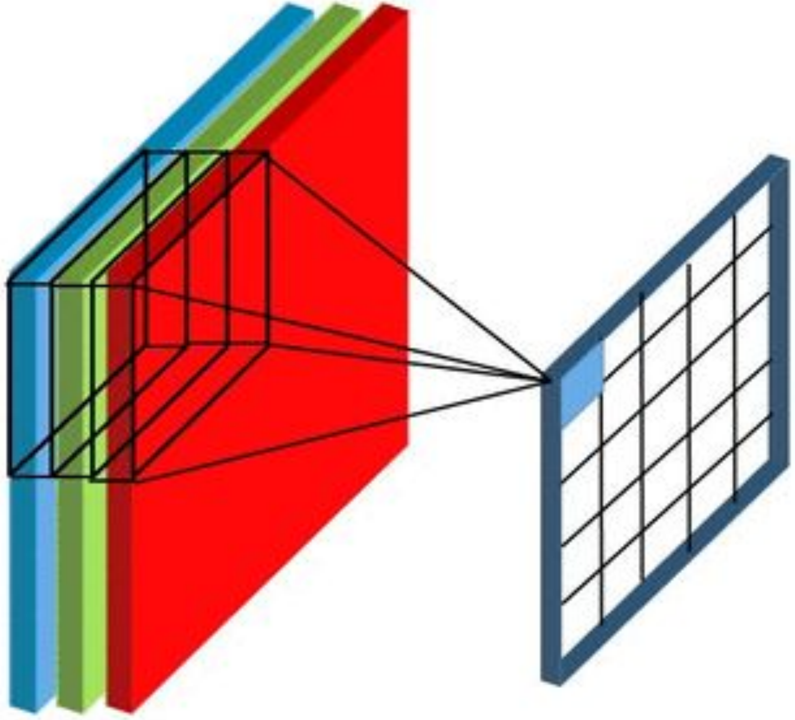
3. Plotting some triangular filters separately.

Mel Spectrum Computing



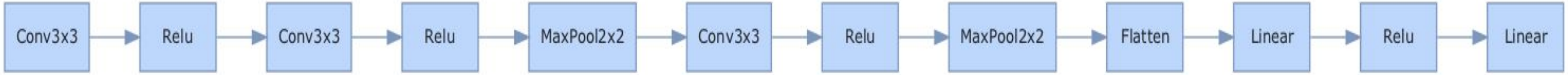


2-D CNNs for Spectrums



Naive CNN

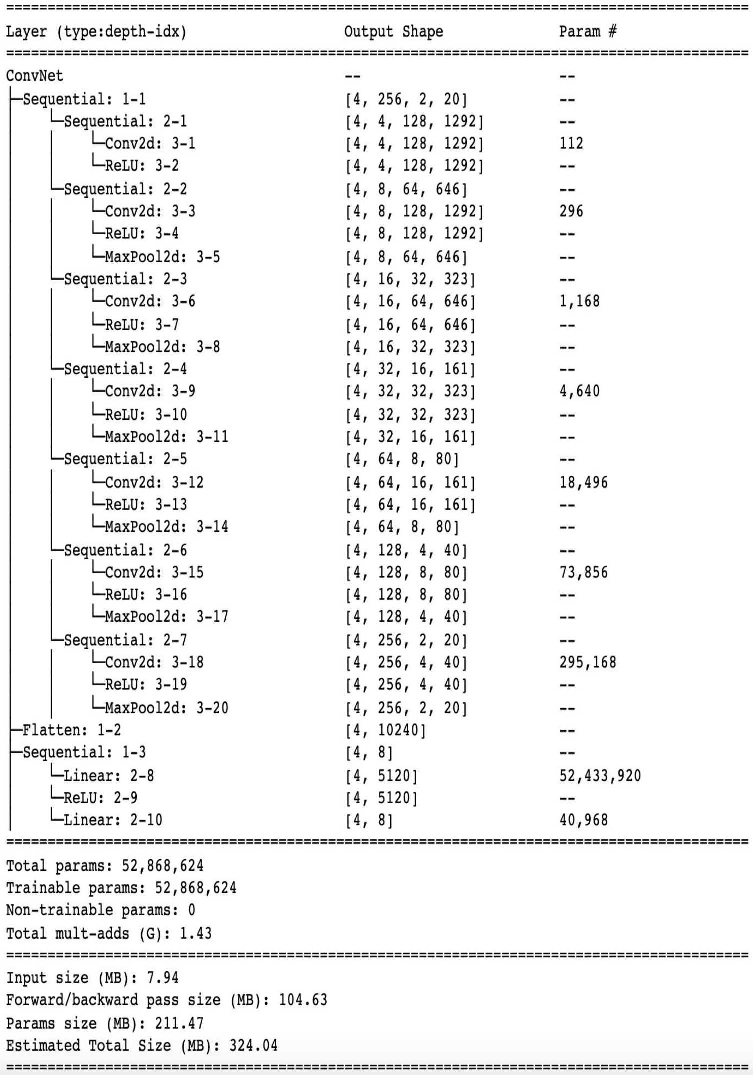
Deeper architecture is utilized but for the visualization purposes we keep it simple.



Conv3x3 → Relu → Conv3x3 → Relu → MaxPool2x2 → Conv3x3 → Relu → MaxPool2x2 → Flatten → Linear → Relu → Linear

2-D CNNs for Spectrums

The Complete Architecture

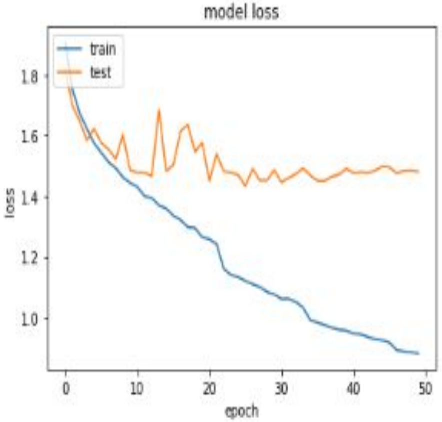


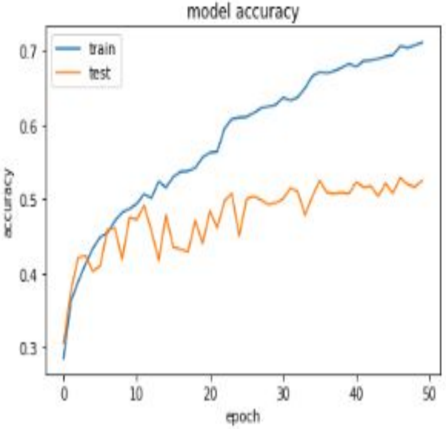
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
ConvNet                                  --                        --
├─Sequential: 1-1                        [4, 256, 2, 20]           --
│    └─Sequential: 2-1                   [4, 4, 128, 1292]         --
│    │    └─Conv2d: 3-1                  [4, 4, 128, 1292]         112
│    │    └─ReLU: 3-2                    [4, 4, 128, 1292]         --
│    └─Sequential: 2-2                   [4, 8, 64, 646]           --
│    │    └─Conv2d: 3-3                  [4, 8, 128, 1292]         296
│    │    └─ReLU: 3-4                    [4, 8, 128, 1292]         --
│    │    └─MaxPool2d: 3-5               [4, 8, 64, 646]           --
│    └─Sequential: 2-3                   [4, 16, 32, 323]          --
│    │    └─Conv2d: 3-6                  [4, 16, 64, 646]          1,168
│    │    └─ReLU: 3-7                    [4, 16, 64, 646]          --
│    │    └─MaxPool2d: 3-8               [4, 16, 32, 323]          --
│    └─Sequential: 2-4                   [4, 32, 16, 161]          --
│    │    └─Conv2d: 3-9                  [4, 32, 32, 323]          4,640
│    │    └─ReLU: 3-10                   [4, 32, 32, 323]          --
│    │    └─MaxPool2d: 3-11              [4, 32, 16, 161]          --
│    └─Sequential: 2-5                   [4, 64, 8, 80]            --
│    │    └─Conv2d: 3-12                 [4, 64, 16, 161]          18,496
│    │    └─ReLU: 3-13                   [4, 64, 16, 161]          --
│    │    └─MaxPool2d: 3-14              [4, 64, 8, 80]            --
│    └─Sequential: 2-6                   [4, 128, 4, 40]           --
│    │    └─Conv2d: 3-15                 [4, 128, 8, 80]           73,856
│    │    └─ReLU: 3-16                   [4, 128, 8, 80]           --
│    │    └─MaxPool2d: 3-17              [4, 128, 4, 40]           --
│    └─Sequential: 2-7                   [4, 256, 2, 20]           --
│    │    └─Conv2d: 3-18                 [4, 256, 4, 40]           295,168
│    │    └─ReLU: 3-19                   [4, 256, 4, 40]           --
│    │    └─MaxPool2d: 3-20              [4, 256, 2, 20]           --
├─Flatten: 1-2                           [4, 10240]                --
├─Sequential: 1-3                        [4, 8]                    --
│    └─Linear: 2-8                       [4, 5120]                 52,433,920
│    └─ReLU: 2-9                         [4, 5120]                 --
│    └─Linear: 2-10                      [4, 8]                    40,968
==========================================================================================
Total params: 52,868,624
Trainable params: 52,868,624
Non-trainable params: 0
Total mult-adds (G): 1.43
==========================================================================================
Input size (MB): 7.94
Forward/backward pass size (MB): 104.63
Params size (MB): 211.47
Estimated Total Size (MB): 324.04
==========================================================================================
```

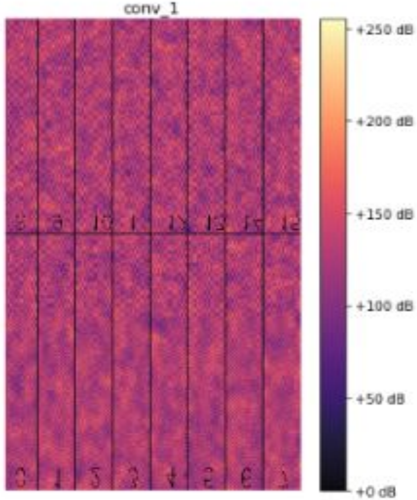
2-D CNNs for Spectrums

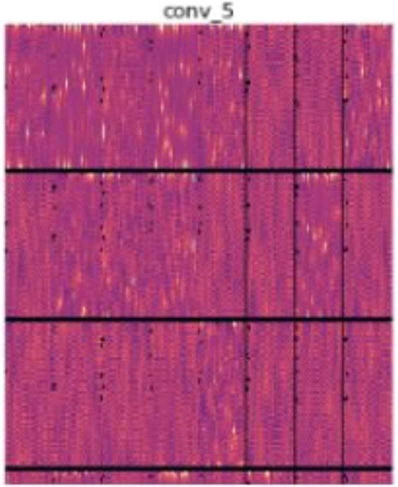
Accuracy 51%

To understand what the filter is focusing on, we look at what kind of input maximizes the activations in that filter. Figure below shows the filter activations of all 16 filters in the first convolution blocks vs the first 24 filters of the fifth convolution block



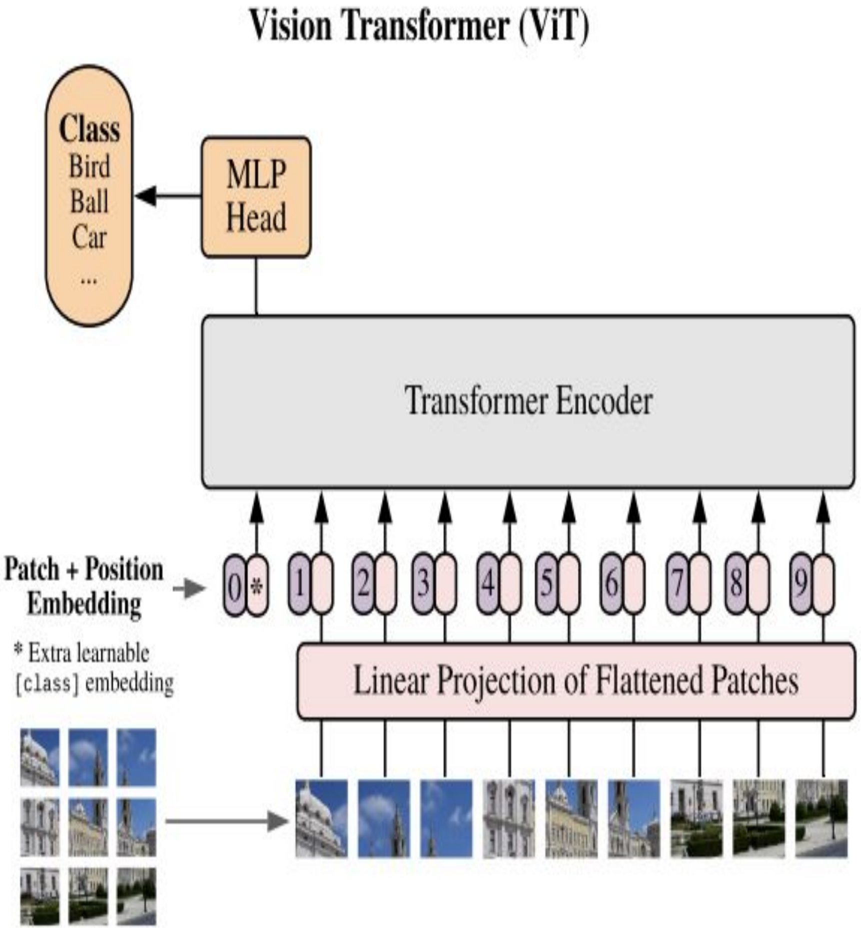






Vision Transformer for Spectrums

The Vision Transformer, or ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable “classification token” to the sequence is used


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
0 *
1
2
3
4
5
6
7
8
9
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

Vision Transformer for Spectrums

Accuracy around 42%

Questions