Instrument Activation Detection with CRNN

Ioannis Koutsoukis
MSc Student in Artificial Intelligence
University of Piraeus & NCSR Demokritos
GitHub · LinkedIn

Abstract

This project tackles the task of **frame-level instrument activation detection** in music recordings. Given the multi-label nature of the problem, where multiple instruments may be active at any time, we rely on the **AAM dataset**, which includes metadata `.arff` files and corresponding `.flac` audio tracks. We extract **mel spectrograms** and construct both a simple **MLP** baseline and a more advanced **CRNN** model to evaluate performance. We address challenges such as class imbalance with **oversampling** and **class-weighted training**, and perform detailed evaluation through both **metrics** and **visual inspection**. Our findings show promising results for low-resource baselines and indicate the need for larger datasets and more expressive models for improved generalization.

1. Dataset & Preprocessing

1.1 Dataset: AAM

- Used subset: `tinyAAM`
- Contents: `.arff` files (segment metadata) and `.flac` audio files
- Each `.arff` contains time-annotated instrument activations

1.2 Feature Extraction: Mel Spectrogram

Extracted using `librosa` with the following parameters:

- `n_fft = 2048`: FFT window size
- `hop_length = 512`: hop size
- `n_mels = 128`: number of mel filters
- Converted to decibel scale using `librosa.power_to_db`

2. Model Architectures

2.1 Baseline: MLP

```
class MLPInstrumentClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.bn1 = nn.BatchNorm1d(hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, hidden_dim)
        self.bn2 = nn.BatchNorm1d(hidden_dim)
        self.out = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = F.relu(self.bn1(self.fc1(x)))
        x = F.relu(self.bn2(self.fc2(x)))
        return torch.sigmoid(self.out(x))
```

2.2 Main Model: CRNN

```
class CRNNInstrumentClassifier(nn.Module):
    def __init__(self, n_mels, n_classes, conv_channels=32,
lstm_hidden=64):
        super().__init__()
        self.cnn = nn.Sequential(
            nn.Conv2d(1, conv_channels, kernel_size=(3, 3), padding=1),
            nn.BatchNorm2d(conv_channels),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=(1, 2)),
            nn.Conv2d(conv_channels, conv_channels, kernel_size=(3, 3),
padding=1),
            nn.BatchNorm2d(conv_channels),
            nn.ReLU()
        )
        reduced_mels = n_mels // 2
        self.lstm = nn.LSTM(
            input_size=conv_channels * reduced_mels,
            hidden_size=lstm_hidden,
            batch_first=True,
            bidirectional=True
        )
        self.classifier = nn.Sequential(
            nn.Linear(2 * lstm_hidden, 128),
            nn.ReLU(),
            nn.Linear(128, n_classes)
        )

    def forward(self, x):
        x = self.cnn(x)
        B, C, T, M_half = x.shape
        x = x.permute(0, 2, 1, 3).reshape(B, T, C * M_half)
        x, _ = self.lstm(x)
        return self.classifier(x)
```

3. Training Strategy

3.1 Class Imbalance Handling

- Computed class frequency over training set
- Rare classes (<5 tracks) were oversampled ×4
- Also experimented with **class weighting** in loss

3.2 Training Details

- Batch size: 2
- Loss function: `BCEWithLogitsLoss`
- Optimizer: Adam
- Dataloaders handled frame-level masking and padding

4. Evaluation

4.1 Metrics Used

- **F1 Score** (macro / micro / weighted)
- **Hamming Loss**
- **Jaccard Score** (sample-based)
- **Classification report per instrument**

```
🎯 Baseline MLP Performance:
F1 Macro:      0.50
F1 Micro:      0.50
F1 Weighted:   0.50
Hamming Loss:  0.51
```

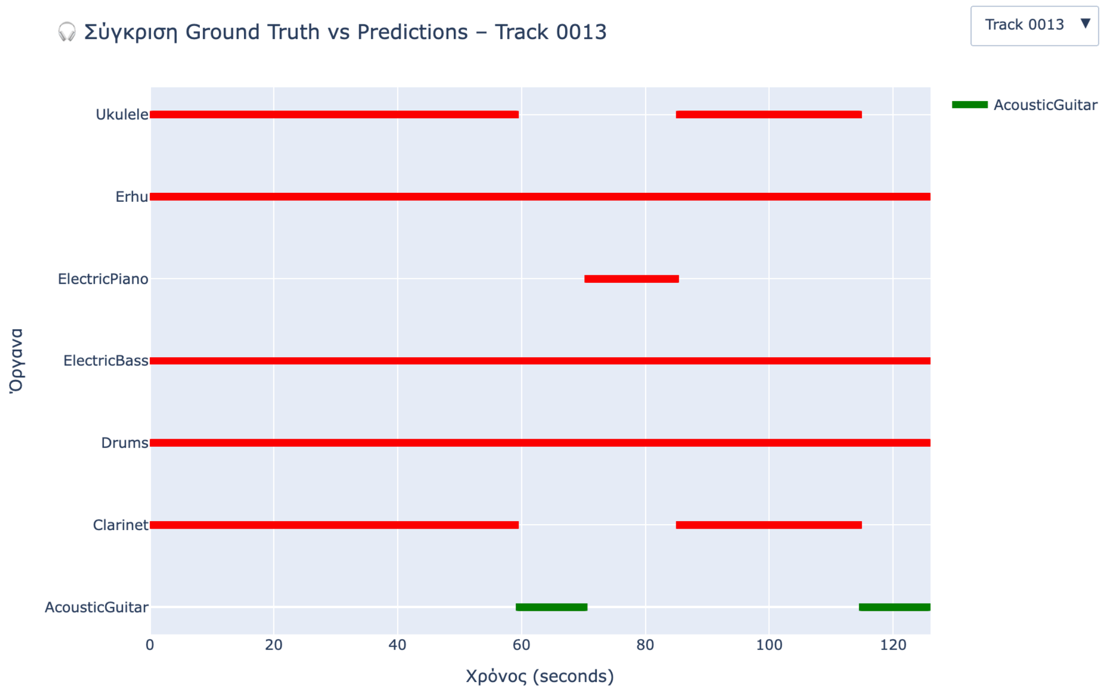
MLP

```
🎯 Evaluation Results:
F1 Macro:     0.253
F1 Micro:     0.535
F1 Weighted:  0.537
Hamming Loss: 0.122
Jaccard Avg:  0.387
```

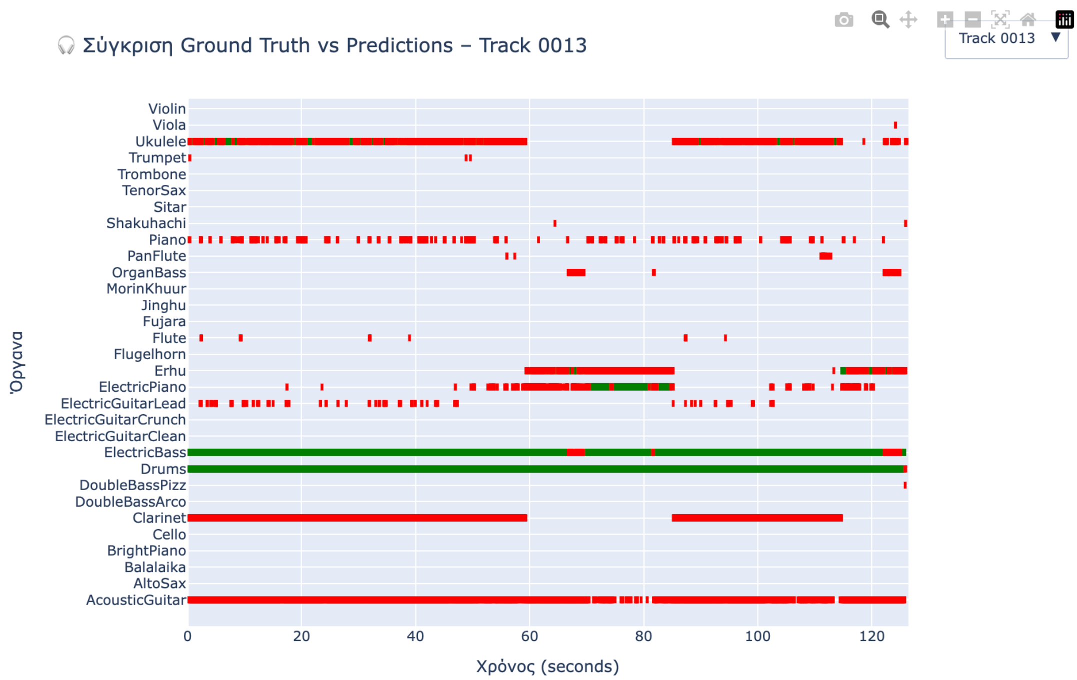
CRNN

4.2 Visualizations

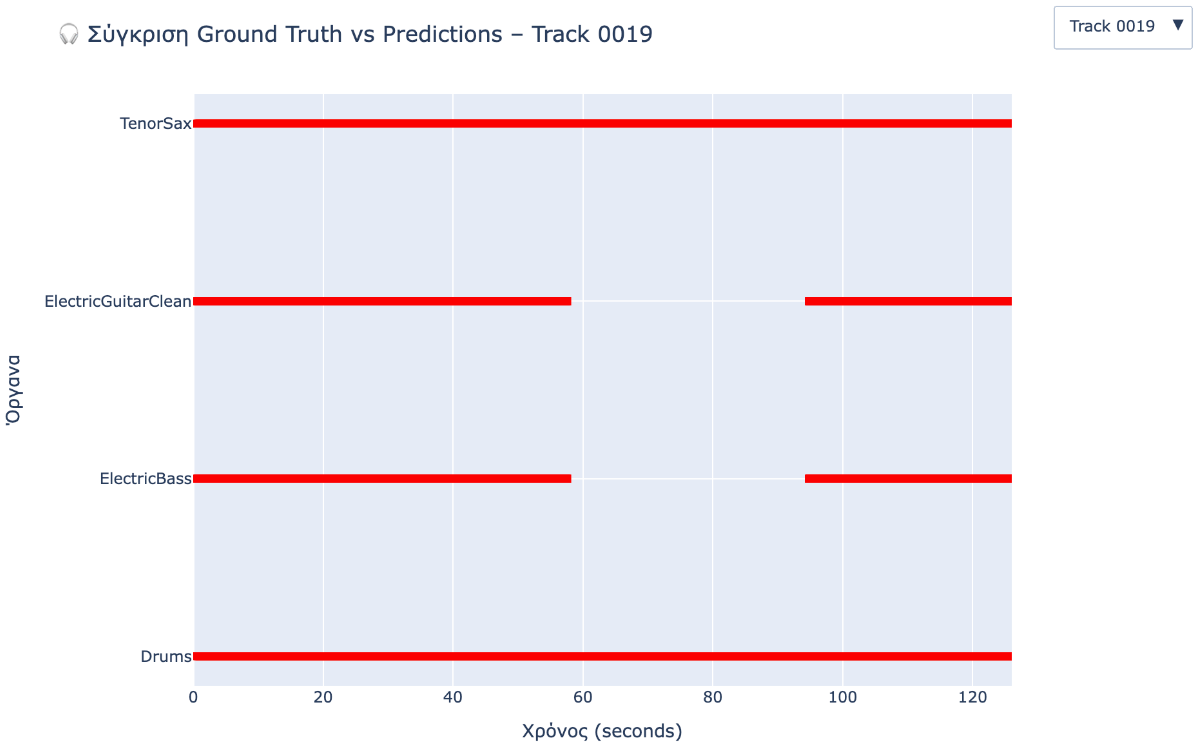
- **Per-track visual inspection** of predicted vs ground truth activations
- **Sorted classification reports** per instrument
- (To be added: screenshots/plots at this section)



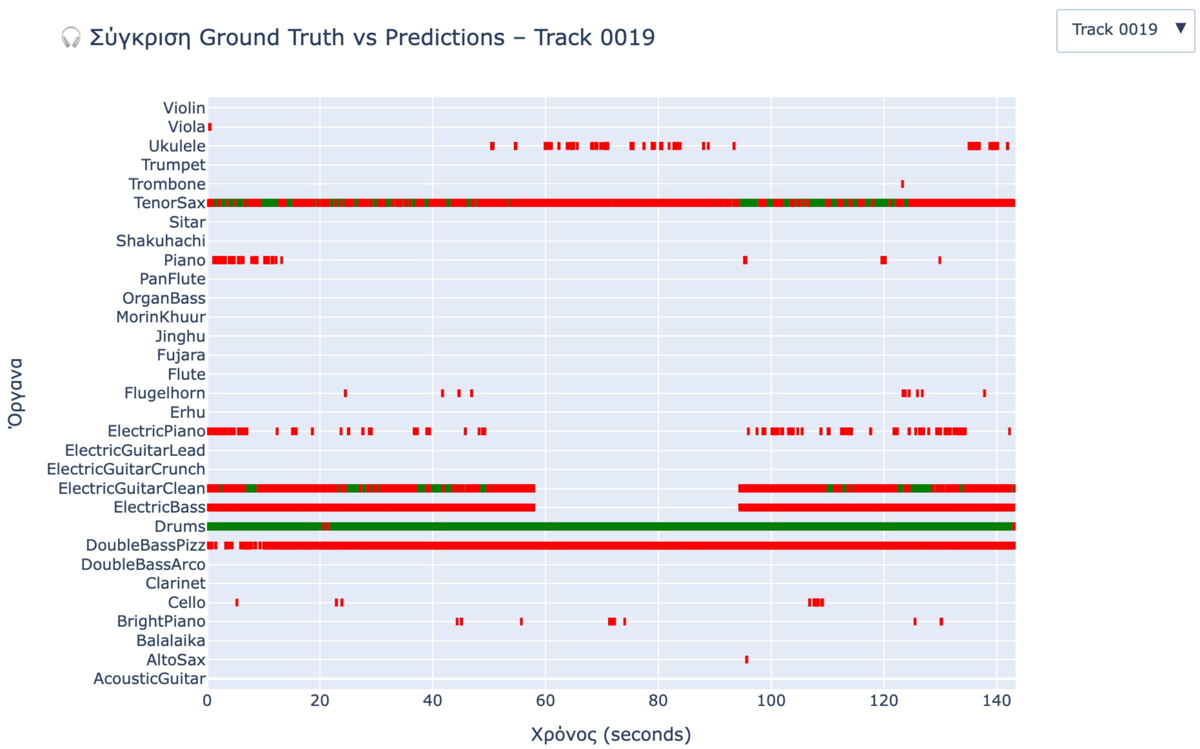
MLP



CRNN


Σύγκριση Ground Truth vs Predictions – Track 0019
Track 0019
TenorSax
ElectricGuitarClean
ElectricBass
Drums
Όργανα
0
20
40
60
80
100
120
Χρόνος (seconds)

MLP


Σύγκριση Ground Truth vs Predictions – Track 0019
Track 0019
Violin
Viola
Ukulele
Trumpet
Trombone
TenorSax
Sitar
Shakuhachi
Piano
PanFlute
OrganBass
MorinKhuur
Jinghu
Fujara
Flute
Flugelhorn
Erhu
ElectricPiano
ElectricGuitarLead
ElectricGuitarCrunch
ElectricGuitarClean
ElectricBass
Drums
DoubleBassPizz
DoubleBassArco
Clarinet
Cello
BrightPiano
Balalaika
AltoSax
AcousticGuitar
Όργανα
0
20
40
60
80
100
120
140
Χρόνος (seconds)

CRNN

5. Confusion Analysis

- Built confusion matrix of most frequent instrument misclassifications
- Found confusion particularly in harmonically similar instruments
- Visual analysis helps reveal overlap patterns due to co-occurrence

True	Predicted	Count
ElectricGuitarClean	ElectricPiano	11791
Shakuhachi	ElectricPiano	10314
Piano	ElectricPiano	9348
ElectricGuitarClean	Ukulele	7641
Erhu	Fujara	7050
Erhu	Sitar	6974
TenorSax	ElectricPiano	6645
Shakuhachi	Ukulele	6614
TenorSax	Piano	6289
Piano	Ukulele	5899
Flugelhorn	ElectricBass	5445
Jinghu	Sitar	5115
AcousticGuitar	ElectricPiano	4975
BrightPiano	ElectricPiano	4930
ElectricGuitarClean	Piano	4896

CRNN

6. Observations

- MLP performs reasonably well in low-data scenarios
- CRNN performs worse at low data but scales better with more data
- Frequent confusions are observed in overlapping spectral instruments (e.g., flute vs clarinet)
- Weighted losses + oversampling help address class imbalance

7. Future Work

- Add more training data from the full AAM set
- Experiment with **pretrained CNNs** for better audio embeddings
- Evaluate **attention-based** and **Transformer-based** models
- Study **temporal consistency** with post-processing smoothing

8. References

- Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). *Convolutional recurrent neural networks for music classification*. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/ICASSP.2017.7952585
- Huang, Y., Wang, J., & Liu, Y. (2022). *Audio Spectrogram Transformer: Learning on the Time-Frequency Representation of Audio*. arXiv preprint arXiv:2104.01778.
- Won, M., Ferraro, D., Han, Y., & Nam, J. (2020). *Evaluation of CNN-based automatic music tagging models*. arXiv preprint arXiv:2006.00751.