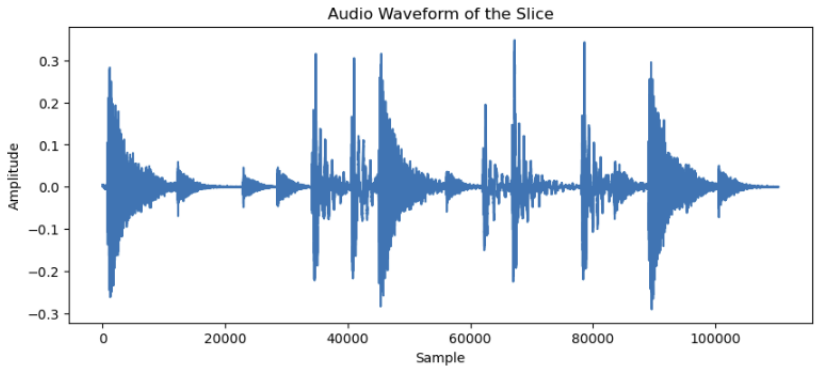
# Machine Learning System: Drum Audio → MIDI

Robert McKean
Global Active, LLC

# Project Goal

Convert recorded drum audio into structured MIDI note events.

**Input**: drum audio recording (wav)




Convolution-based
AI modeling

**Output**: MIDI note events

- note number
- relative timing
- velocity

```
x: 46, y: 1, value: 110
x: 44, y: 85, value: 92
x: 42, y: 87, value: 88
x: 36, y: 130, value: 109
x: 38, y: 173, value: 108
x: 42, y: 176, value: 107
x: 42, y: 260, value: 89
x: 36, y: 302, value: 110
x: 42, y: 346, value: 108
x: 42, y: 432, value: 95
x: 36, y: 474, value: 111
x: 38, y: 517, value: 108
x: 42, y: 518, value: 110
x: 36, y: 602, value: 111
x: 46, y: 603, value: 112
x: 44, y: 687, value: 92
x: 42, y: 689, value: 88
x: 36, y: 732, value: 110
x: 38, y: 775, value: 108
x: 42, y: 776, value: 109
```

**Modeling Approach**:

Train a convolutional neural network to predict time-aligned MIDI notes from STFT features computed on short audio slices.

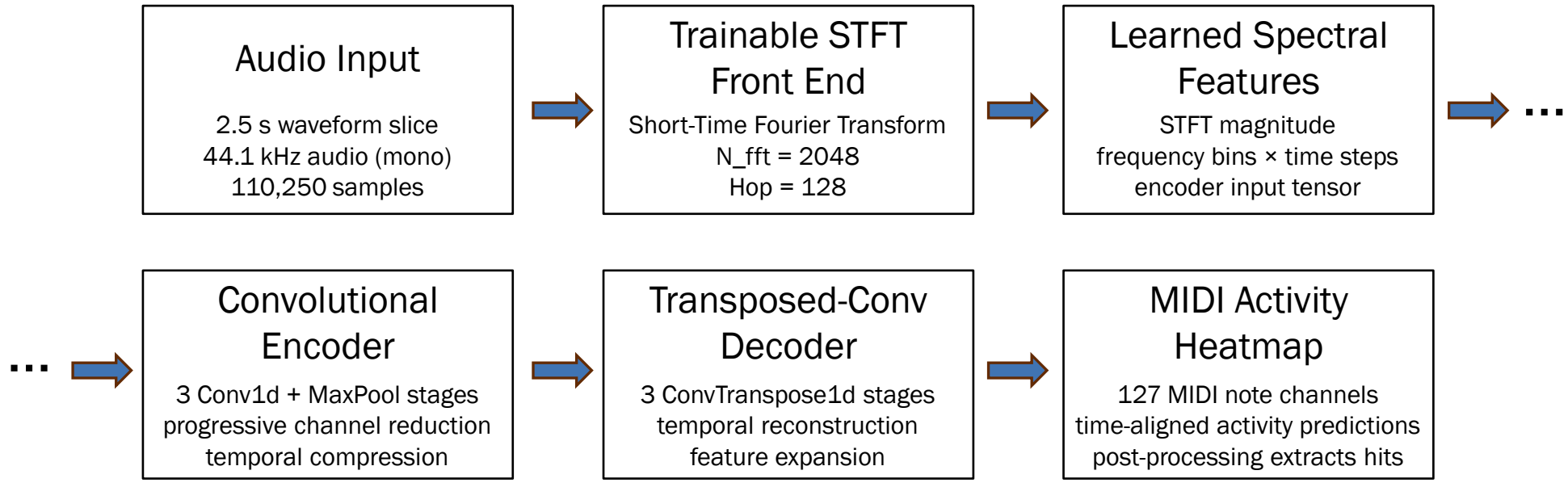
# System Overview

The system converts short audio slices into MIDI activity predictions.

**The Pipeline**

1. Extract short audio slices
2. Convert audio to spectral features (STFT)
3. Predict note activity with a CNN
4. Generate a MIDI activity heatmap
5. Extract note events → MIDI file

# Architecture Diagram




End-to-end architecture converting short audio slices into dense MIDI note activity predictions.

# Dataset Construction

Phase 1 - Generate Training Source Data

- Combine thousands of MIDI drum loops into a continuous sequence
- Render audio from the MIDI file using multiple drum kits and VST plug-ins
- Produce synchronized audio and MIDI recordings (ground truth)

Phase 2 - Create Training Examples

- Segment recordings into aligned 2.5 s audio–MIDI slices
- Preserve exact alignment between waveform and MIDI events
- Use audio slice as model input (X)

Phase 3 - Generate Target Output

- Convert MIDI events into a note–time activity heatmap
- Heatmap represents MIDI note number vs time
- This heatmap becomes the training target (Y)

Phase 4 - Model Training and Verification

- Train the convolutional model on paired X/Y slices
- Compare predicted heatmaps to reference targets
- Extract note events and generate MIDI output

**Training dataset:**
9.5 hours of synchronized audio–MIDI data

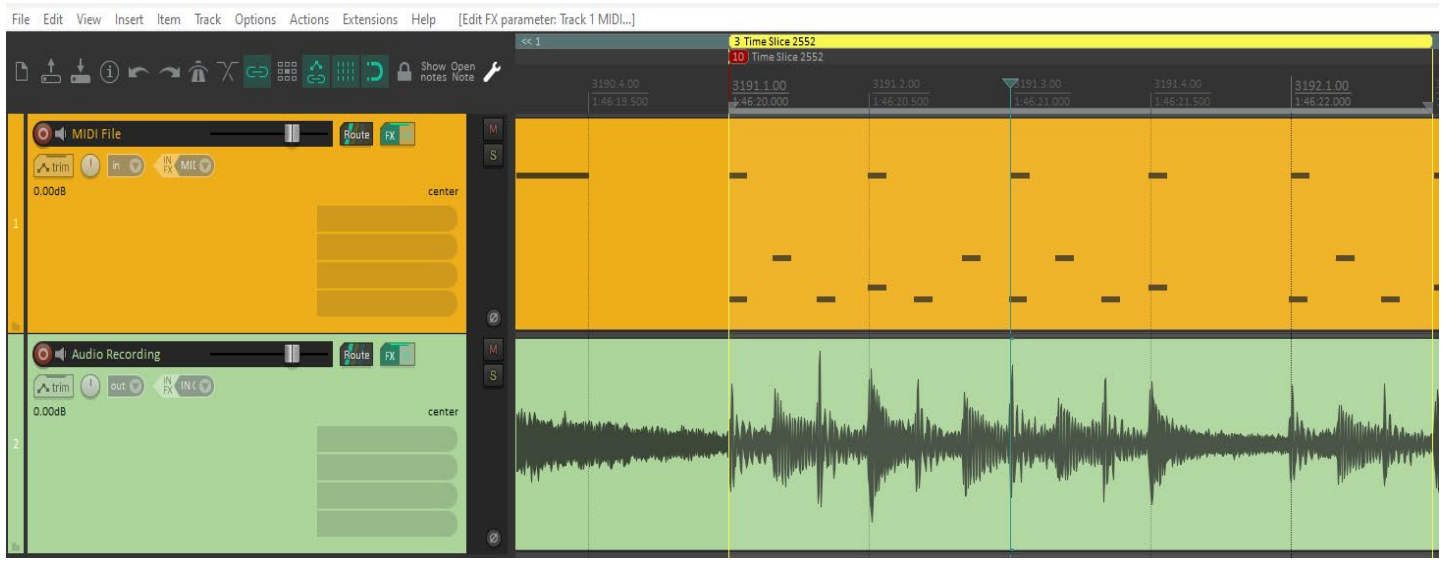
# Create the MIDI and Audio Files

The MIDI file combines thousands of MIDI loops into one long file (9.5 hours).

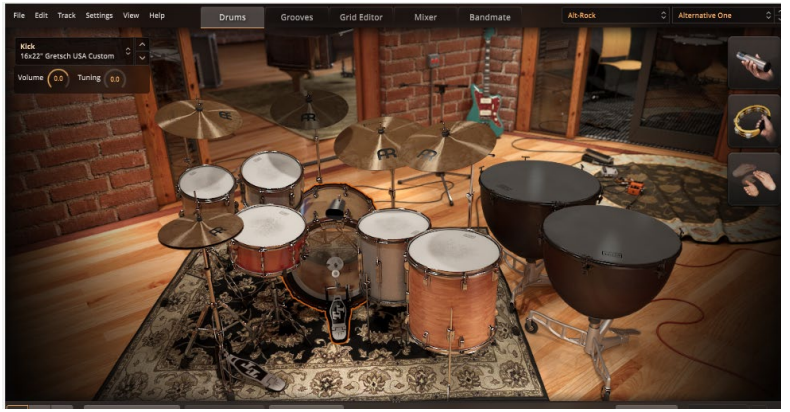
Audio files were then created by using a drum plug-in.

Multiple drum kits were used to provide a variety of sounds for training.

DAW software showing MIDI file and generated drum audio.




Drum plug-in used to generate audio file.

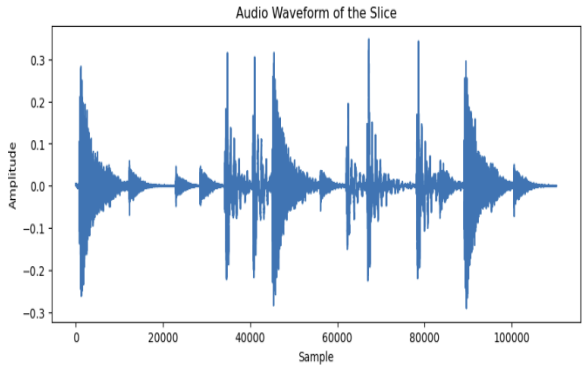


# Training the Model

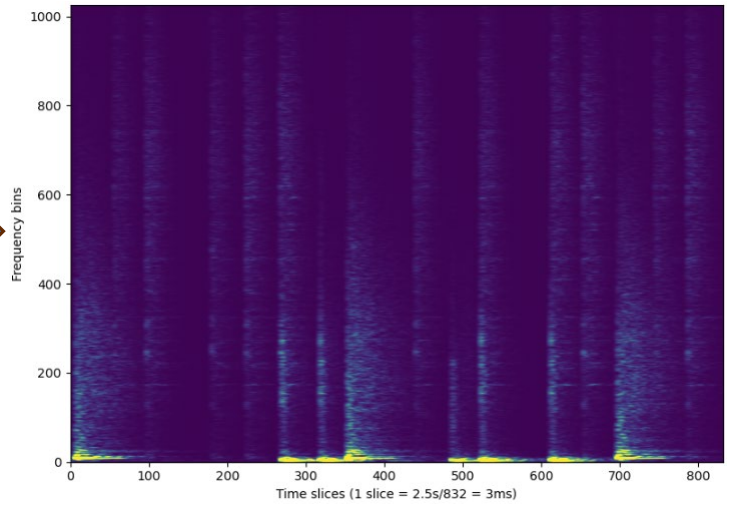
Note that the STFT data aligns with the note data in the target output.

Input: drum recording (wav)

torch.Size([1, 110250])

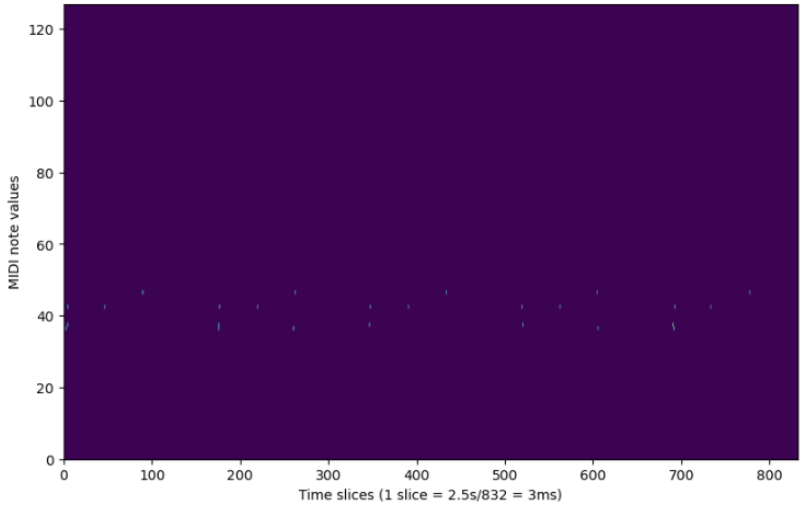



STFT of Input (nnAudio)




CONV-LAYERS

Target Output (Y) – MIDI data



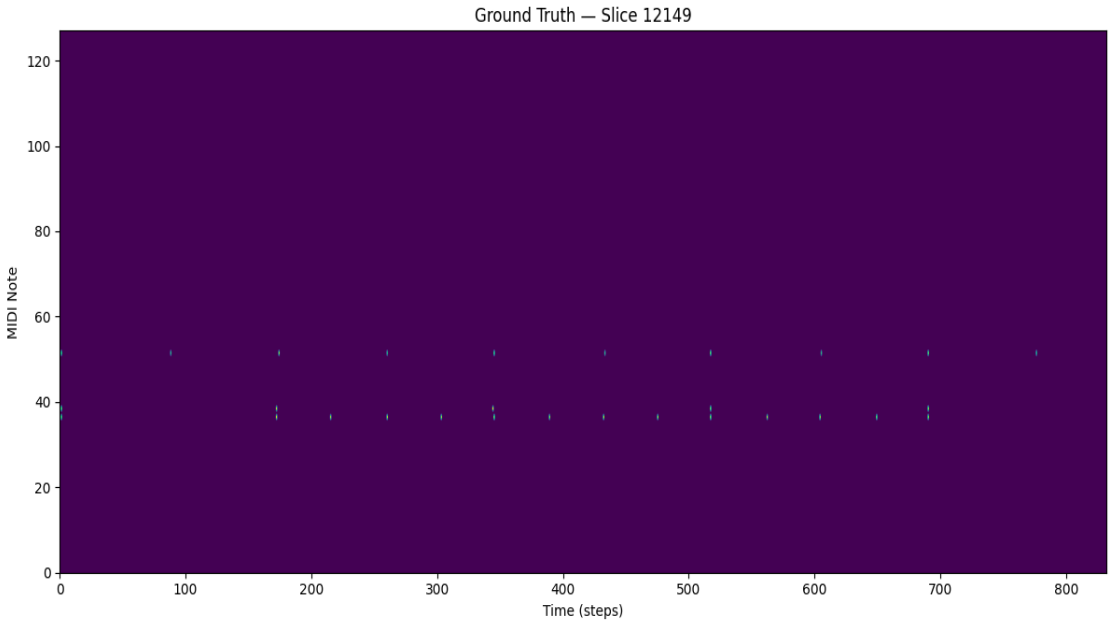

X data for training

Model

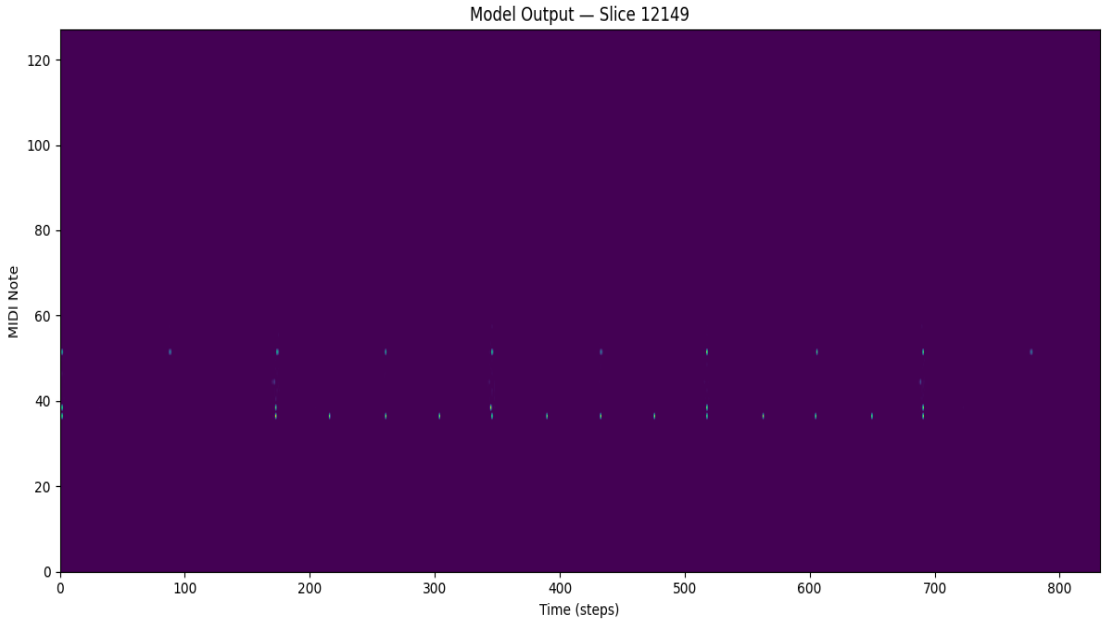
Y data for training
(Ground Truth)

# Results – Target Output vs Actual Output

Target Output – MIDI data




Actual Model Output – MIDI data




The dots represent:

- The relative timing (X axis)
- MIDI note number (Y axis)
- Velocity of the note (dot brightness).

# Results – Actual Values

For this 2.5s slice (2552), the note (x) and relative time (y) values match exactly between the actual and predicted MIDI values.

Conclusion: The model accurately predicted the timing and the specific drums triggered from a random audio sample, even when multiple drums were triggered simultaneously.

The values (note velocity) are higher on the post-modeled files. More detailed scaling analysis is required to improve.

For this example:

1. Slice number: 2552
2. Threshold: 0.5

Original midi file

```
x: 36, y: 0, value: 79
x: 57, y: 0, value: 109
x: 43, y: 54, value: 91
x: 36, y: 108, value: 119
x: 38, y: 170, value: 112
x: 57, y: 171, value: 109
x: 36, y: 227, value: 92
x: 43, y: 286, value: 107
x: 36, y: 343, value: 117
x: 57, y: 346, value: 109
x: 43, y: 400, value: 91
x: 36, y: 456, value: 97
x: 38, y: 514, value: 121
x: 57, y: 514, value: 109
x: 36, y: 686, value: 102
x: 57, y: 689, value: 109
x: 43, y: 744, value: 106
x: 36, y: 799, value: 119
```

Post-modeled midi file

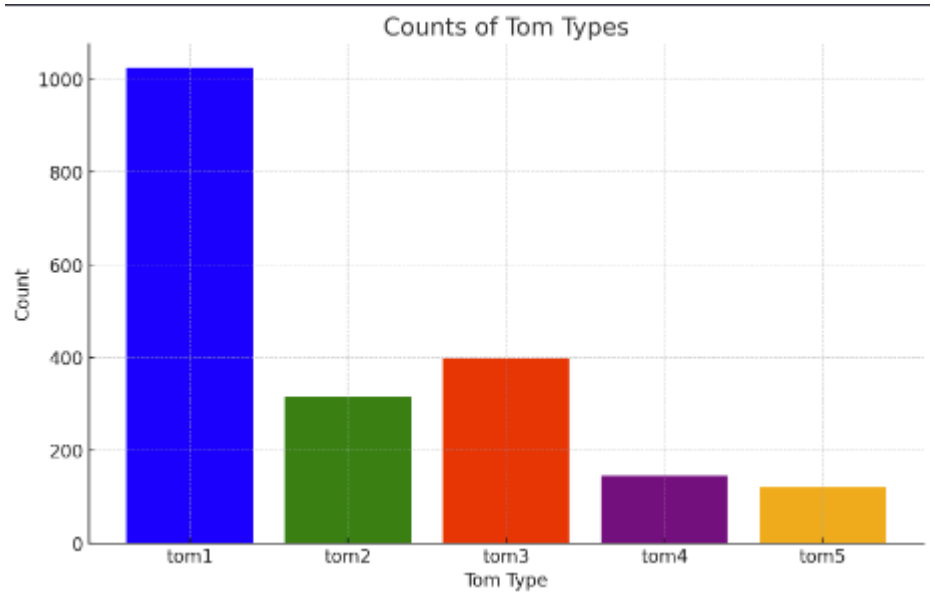
```
x: 36, y: 0, vel: 111
x: 57, y: 0, vel: 196
x: 43, y: 54, vel: 136
x: 36, y: 108, vel: 121
x: 38, y: 170, vel: 145
x: 57, y: 171, vel: 91
x: 36, y: 227, vel: 146
x: 43, y: 286, vel: 136
x: 36, y: 343, vel: 164
x: 57, y: 346, vel: 120
x: 43, y: 400, vel: 124
x: 36, y: 456, vel: 134
x: 38, y: 514, vel: 144
x: 57, y: 514, vel: 138
x: 36, y: 686, vel: 123
x: 57, y: 689, vel: 132
x: 43, y: 744, vel: 134
x: 36, y: 799, vel: 138
```

# Biggest Challenges

1. Quality drum data. Used drum samples from Kontakt, EZ Drummer 3, and individual one-shots.
2. Even with a folder full of individual drums (toms, for example), there will be outliers. Electronic drums sound different than acoustic drums, for example.
3. Required: A larger curated drum set. With 10,000's of drums - curating this data was the largest challenge.
4. Developed unsupervised K-clustering to sort and classify drums. Effective.
5. Memory. Deeper algorithms use more memory and take longer to model. Needed to keep GPU memory below 8GB to run on laptop's NVIDIA GeForce RTX3080.

Below are the results from K-clustering approximately 2,000 individual tom tom drum samples.

The primary separator was the tone, with tom1 being the lowest in frequency.



# Lessons Learned

- Results were strong for drums used during training. Broader generalization requires a larger curated dataset.
- More tuning of the model would be helpful, including steps that would increase GPU memory consumption.
- MIDI note resolution: 3ms. Could be decreased at the expense of more GPU memory and longer training.
- Tip: Input tensor length must be compatible with convolution algorithms. This program needed to truncate input tensor from 862 to 832 samples.
- Tip: Create a test tensor with random data to confirm that the model produces the expected output data format. The output from the model must match the format of the Y data.

In summary, I was able to:

- Identify a practical solution to generate MIDI from audio.
- Develop a strategy for generating supervised X/Y training data.
- Create large synchronized audio and MIDI datasets.
- Generate curated and cleaned-up data for modeling.
- Implement a convolutional neural network from scratch.
- Use k-means clustering to categorize drum samples.
- Deliver a working system using Python and PyTorch.

# Thank you!