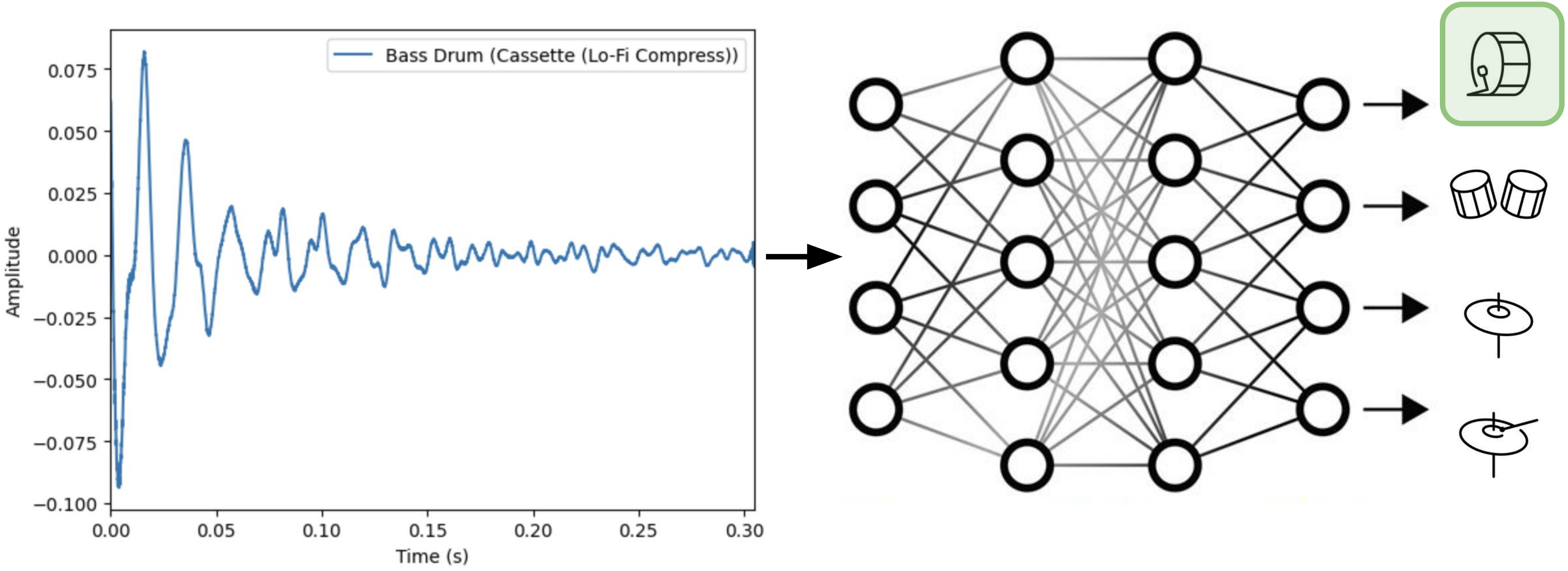
Drum Instrument Classification

Karl Hiner

Problem statement

Given a raw audio clip of an individual drum instruments being struck, classify the instrument as one of a predefined set.



Classifier formulation

$\mathbf{x} \in [-1, 1]^n$: Raw audio clip with *n* real-valued frames

$y \in \{1, \ldots, k\}$: Drum instrument label (e.g. 1 = snare drum)

$\mathcal{D}$: Joint distribution of random variable $Z = (X, Y)$

$h : \mathbf{x} \mapsto \hat{y}$: Learned classifier to estimate drum instrument $\hat{y}$

$l(h(\mathbf{x}), y) = -\sum_{c=1}^{k} y_c \log(h(\mathbf{x})_c)$: Cross-entropy loss for a single sample

$R(h) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[l(h(\mathbf{x}), y)]$: Generalization risk

$\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(\mathbf{x}_i), y_i)$: Empirical risk over *m* samples

$\hat{R}(h) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c=1}^{k} y_{i,c} \log(h(\mathbf{x}_i)_c)$: Cross-entropy loss over *m* samples

Dataset: MIDI-annotated audio recordings

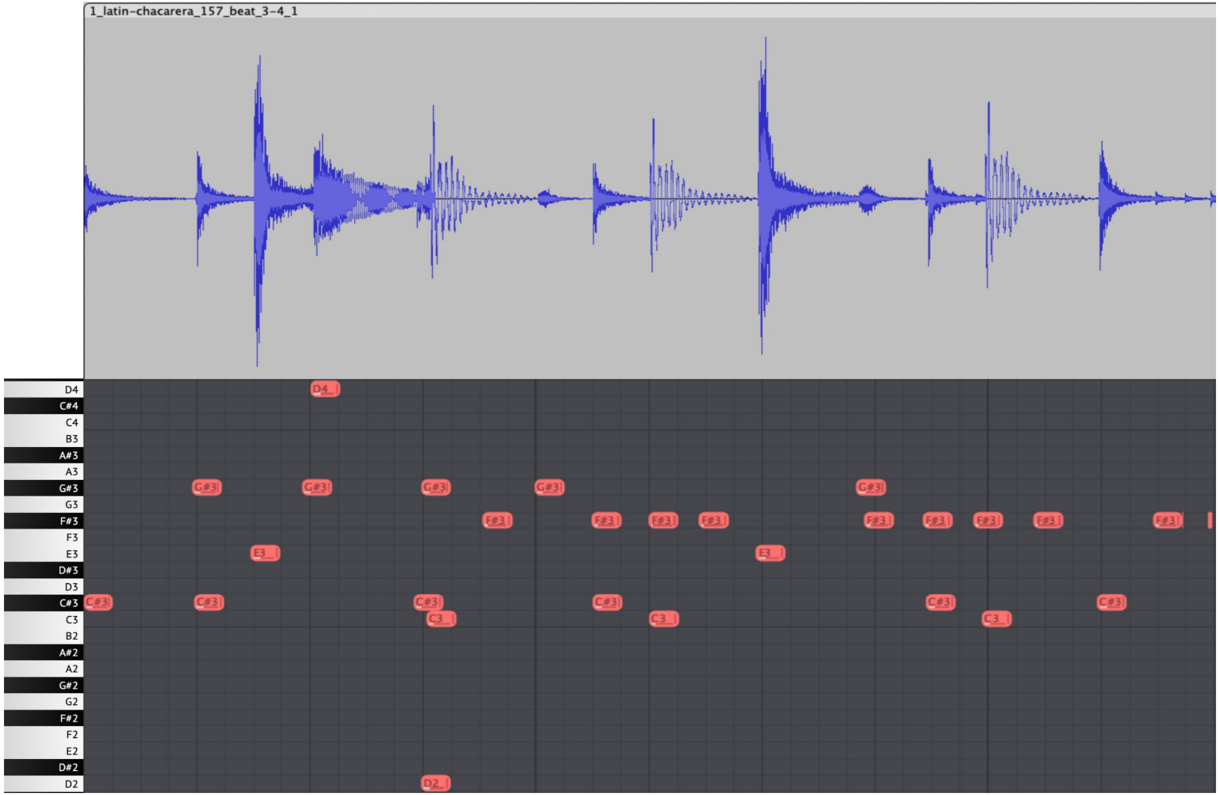
- Expanded Groove MIDI Dataset (EGM-D)*
  - Contains 444 hours of MIDI-annotated audio recordings from 43 electronic drum kits

Split	Unique Sequences	Total Sequences	Duration (hours)
Train	819	35,217	341.4
Test	123	5,289	50.9
Validation	117	5,031	52.2
Total	**1,059**	**45,537**	**444.5**

Field	Description
drummer	An anonymous string ID for the drummer of the performance.
session	A string ID for the recording session (unique per drummer).
id	A unique string ID for the performance.
style	A string style for the performance formatted as "<primary>/<secondary>". The primary style comes from the Genre List below.
bpm	An integer tempo in beats per minute for the performance.
beat_type	Either "beat" or "fill"
time_signature	The time signature for the performance formatted as "<numerator>-<denominator>".
midi_filename	Relative path to the MIDI file.
audio_filename	Relative path to the WAV file (if present).
duration	The float duration in seconds (of the MIDI).
split	The predefined split the performance is a part of. One of "train", "validation", or "test".
kit_name	Name of the drum kit on the Roland TD-17 used for synthesis.

* https://magenta.tensorflow.org/datasets/e-gmd

Dataset: MIDI-annotated audio recordings



Dataset: MIDI-annotated audio recordings

MIDI Implementation

Model: TD-17 (TD-17-L)
Date: May. 1. 2018
Version: 1.00

```
NAME_FOR_NOTE = {
    35: 'Acoustic Bass Drum',
    36: 'Bass Drum',
    37: 'Side Stick',
    38: 'Acoustic Snare',
    39: 'Hand Clap',
    40: 'Electric Snare',
    41: 'Low Floor Tom',
    42: 'Closed Hi Hat',
    43: 'High Floor Tom',
    44: 'Pedal Hi-Hat',
    45: 'Low Tom',
    46: 'Open Hi-Hat',
    47: 'Low-Mid Tom',
    48: 'Hi-Mid Tom',
    49: 'Crash Cymbal 1',
    50: 'High Tom',
    51: 'Ride Cymbal 1',
    52: 'Chinese Cymbal',
    53: 'Ride Bell',
    54: 'Tambourine',
    55: 'Splash Cymbal',
    56: 'Cowbell',
    57: 'Crash Cymbal 2',
    58: 'Vibraslap',
    59: 'Ride Cymbal 2',
}
```

General MIDI PERCUSSION Key Map

For MIDI Channel 10, each MIDI KEY number ("NOTE#") corresponds to a different drum sound, as shown below. While many current instruments also have additional sounds above or below the range show here, and may even have additional "kits" with variations of these sounds, only these sounds are supported by General MIDI Level 1 devices.

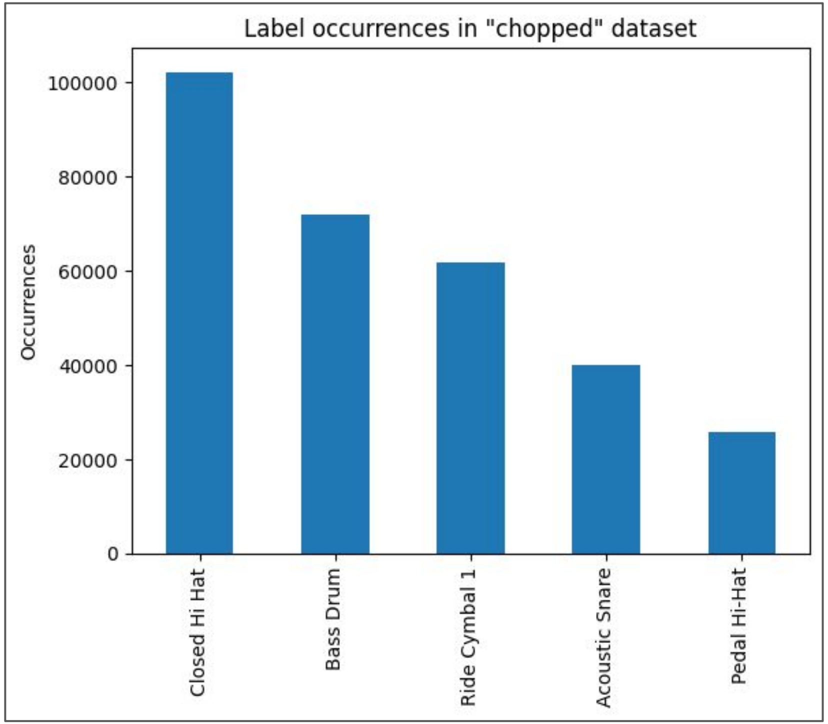
Key#	Note	Drum Sound	Key#	Note	Drum Sound
35	B0	Acoustic Bass Drum	59	B2	Ride Cymbal 2
36	C1	Bass Drum 1	60	C3	Hi Bongo
37	C#1	Side Stick	61	C#3	Low Bongo
38	D1	Acoustic Snare	62	D3	Mute Hi Conga
39	Eb1	Hand Clap	63	Eb3	Open Hi Conga
40	E1	Electric Snare	64	E3	Low Conga
41	F1	Low Floor Tom	65	F3	High Timbale
42	F#1	Closed Hi Hat	66	F#3	Low Timbale
43	G1	High Floor Tom	67	G3	High Agogo
44	Ab1	Pedal Hi-Hat	68	Ab3	Low Agogo
45	A1	Low Tom	69	A3	Cabasa
46	Bb1	Open Hi-Hat	70	Bb3	Maracas
47	B1	Low-Mid Tom	71	B3	Short Whistle
48	C2	Hi Mid Tom	72	C4	Long Whistle
49	C#2	Crash Cymbal 1	73	C#4	Short Guiro
50	D2	High Tom	74	D4	Long Guiro
51	Eb2	Ride Cymbal 1	75	Eb4	Claves
52	E2	Chinese Cymbal	76	E4	Hi Wood Block
53	F2	Ride Bell	77	F4	Low Wood Block
54	F#2	Tambourine	78	F#4	Mute Cuica
55	G2	Splash Cymbal	79	G4	Open Cuica
56	Ab2	Cowbell	80	Ab4	Mute Triangle
57	A2	Crash Cymbal 2	81	A4	Open Triangle
58	Bb2	Vibraslap			

“Chopped” drum hit dataset generation: Parse MIDI

- Filter down to a reduced dataset by excluding unconventional drum kits.
- Parse MIDI for segments of audio corresponding to solo drum instrument hits.
- A “drum hit” candidate starts with a non-zero velocity note-on event, and continues until the next non-zero velocity note-on event.
- Additional heuristics needed:
  - Only candidates with a length of at least 100ms are considered.
  - At least 100ms long containing exactly one drum instrument.
  - Minimum number of frames to consider a MIDI note as “active” after onset.
  - Trim hit ending to prevent including the beginning of another drum hit.
- A row per drum hit to the provided `dataset`.
- Additional challenge: Lots of bad MIDI event timing.
  - Cleanup pass over audio to drop rows whose peak (max amplitude) start late in the clip.

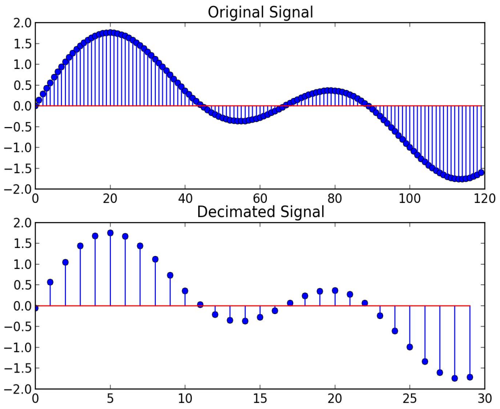
Final “Chopped” dataset

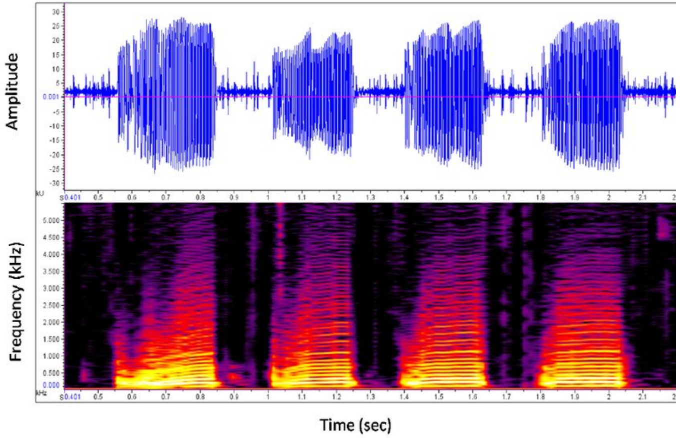
- Total rows (individual drum hits): **301,640**
- Unbalanced!
- Columns:
  - `file_path`: Path to the audio file in the *E-GMD* dataset.
  - `begin_frame`: Frame of the beginning of the hit.
  - `num_frames`: Length, in frames, of the hit.
  - `label`: Drum instrument label - the `id` column in the label-mapping CSV.
  - `slim_id`: Session ID (row in “slimmed” dataset) in which the hit was found, for access to other metadata.

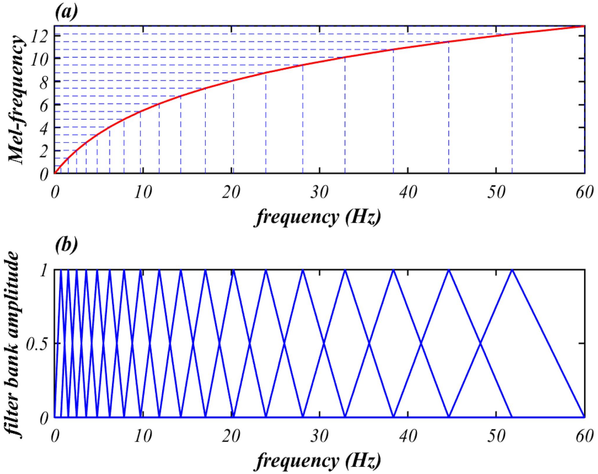


Audio feature pipeline

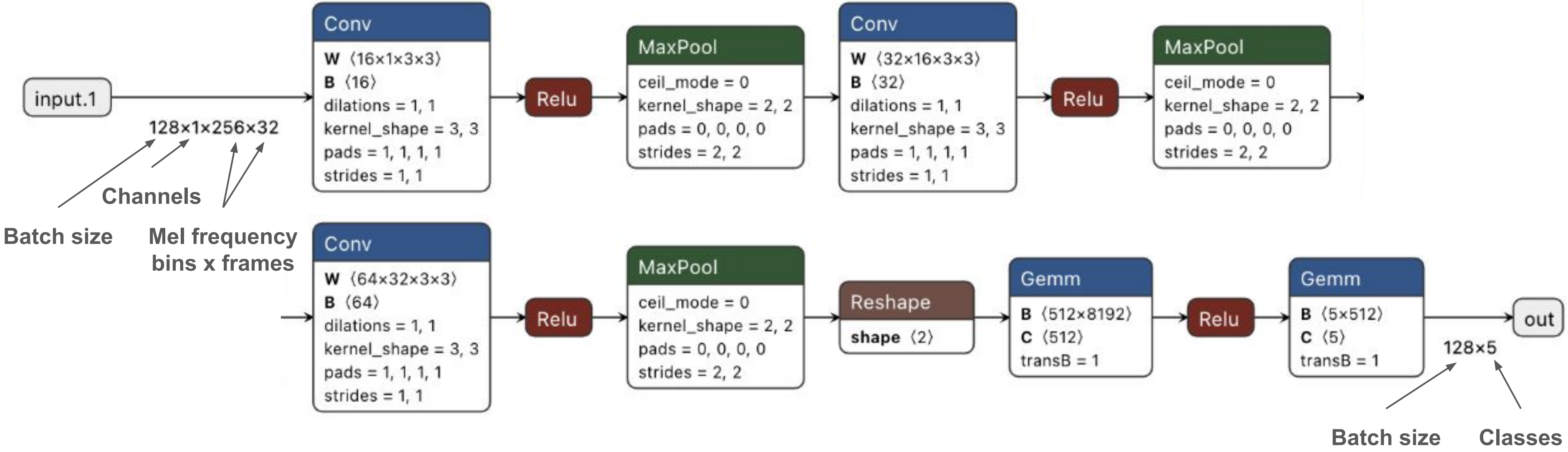
- Trim each drum hit clip to 0.5 seconds.
- (Optionally) downsample audio from 44.1 kHz (final model uses 32 kHz).
- Compute magnitude spectrogram (`n_fft=2048, hop_length=512`).
- Convert spectrogram to Mel scale.







Model: Simple CNN



Model: Simple CNN

```python
# Audio feature extraction pipeline.
# 1. Resample audio
# 2. Convert to power spectrogram
# 3. Convert to mel-scale
class WaveformFeatures(nn.Module):
    def __init__(
        self,
        input_freq=44_100,
        resample_freq=32_000,
        n_fft=2048,
        n_mel=256,
    ):
        super().__init__()
        self.resample = T.Resample(orig_freq=input_freq, new_freq=resample_freq)
        self.spectrogram = T.Spectrogram(n_fft=n_fft, hop_length=n_fft // 4)
        self.mel_scale = T.MelScale(n_mels=n_mel, sample_rate=resample_freq,
                                    n_stft=n_fft // 2 + 1)

    def forward(self, waveform: torch.Tensor) -> torch.Tensor:
        resampled = self.resample(waveform)
        spec = self.spectrogram(resampled)
        mel = self.mel_scale(spec)
        return mel
```

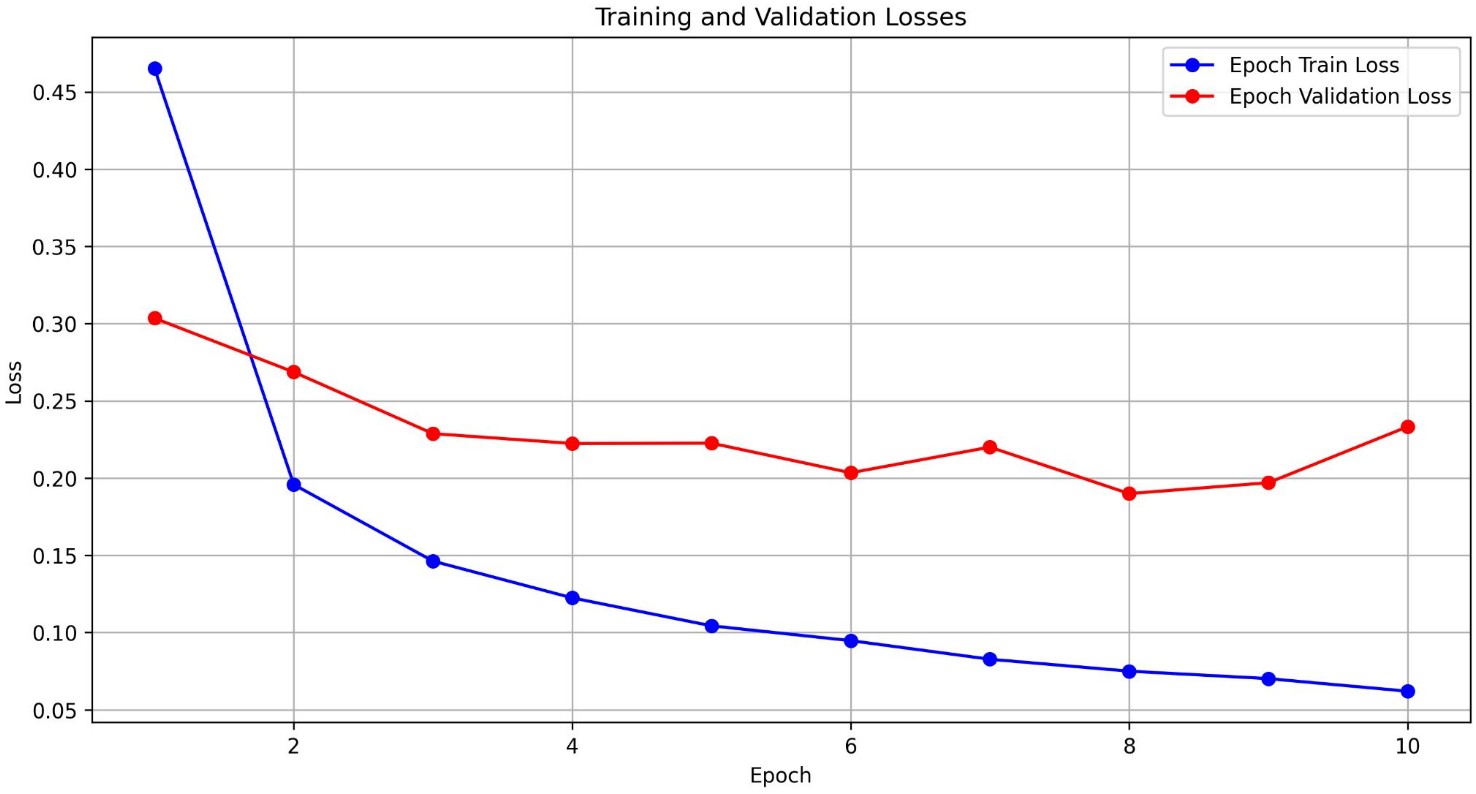
```python
class AudioClassifier(nn.Module):
    def __init__(self, num_classes, n_mel, n_mel_frames):
        super(AudioClassifier, self).__init__()

        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.dropout = nn.Dropout(0.5)

        time_dim_after_pooling = n_mel_frames // (2**3)
        conv_output_size = n_mel // (2**3)
        fc1_input_size = 64 * conv_output_size * time_dim_after_pooling
        self.fc1 = nn.Linear(fc1_input_size, 512)
        self.fc2 = nn.Linear(512, num_classes)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(x.size(0), -1) # Flatten for the fully connected layers.
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

Results

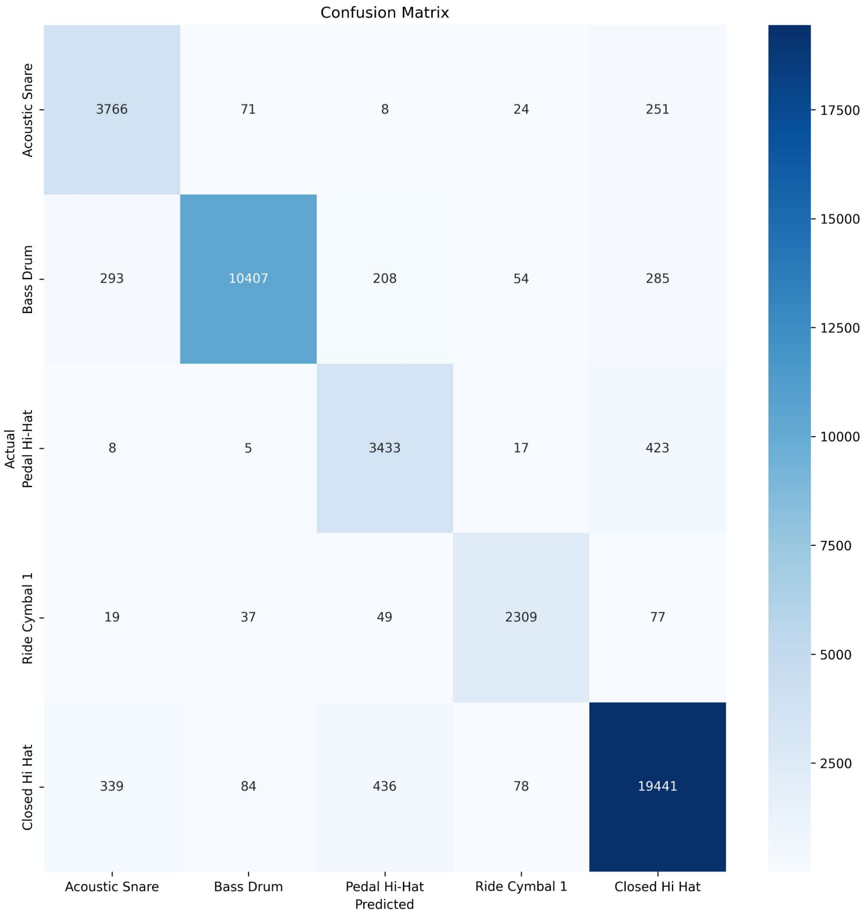


Training and Validation Losses
Epoch Train Loss
Epoch Validation Loss
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Loss
2
4
6
8
10
Epoch

Results

Accuracy over test set:

39,356 / **42,122** correct predictions for final accuracy: **93.4%**



Future work

- Better segmentation
- Account for class imbalance in model/training
- Multi-label classification, for overlapping drum instruments
- Estimate velocity in addition to instrument

Questions