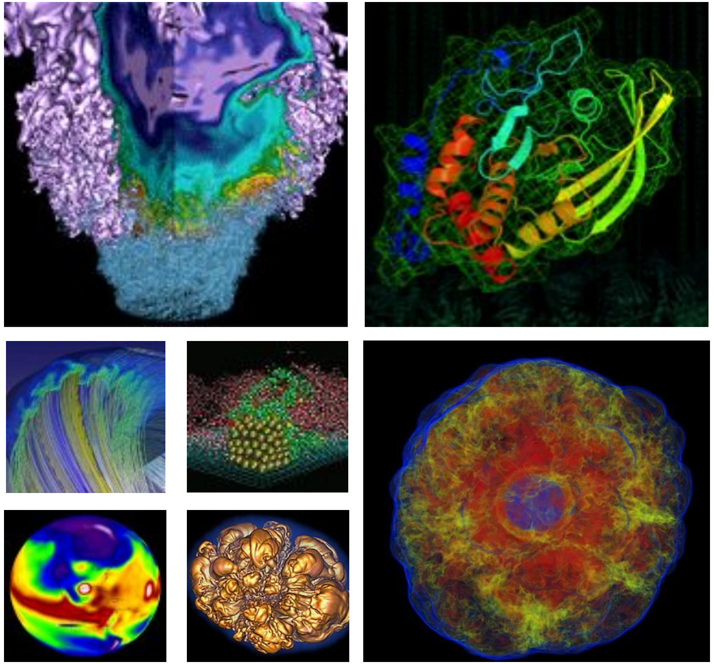
Using unsupervised learning to understand magnet acoustic events



L. Stephey, M. Mustafa, X. Jia, M. Marchevsky

Nov 13, 2019

HEP-ML Seminar

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Introduction and vocabulary

- Superconducting magnets have to be “trained” to get them to reach high magnetic fields (through increasing current)
- Training ramps end in a “quench” when the magnet stops superconducting
- Training is costly⟶ can it be reduced or avoided altogether?

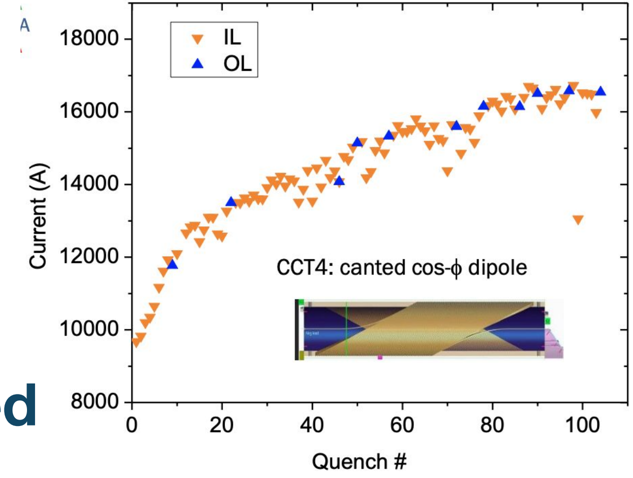


Figure courtesy M. Marchevsky

Why use deep learning?

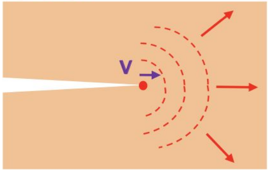
Hypothesis: magnet training is dominated by two kinds of physical events:

- **Cracking (transient event)**
- **Stick-slip (slower, more continuous event)**

1) Can we identify these events? Do we observe clustering in our data?

2) If we can, can they be used to predict the quench event?

Crack

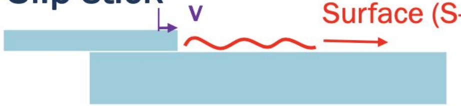


Bulk (P-)wave

$$c_p = \sqrt{\frac{K + \frac{4}{3}G}{\rho}}$$

"Fast" event (~1 μs) cracks propagation velocity can be up to the speed of sound

Slip-stick



$$c_s = \sqrt{\frac{G}{\rho}}$$

"Slow" event (~0.1-1 ms) some mass has to move

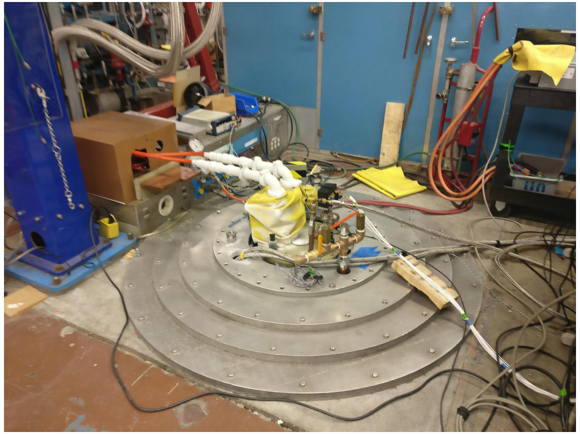
K is the bulk modulus
G is the shear modulus
ρ is the density

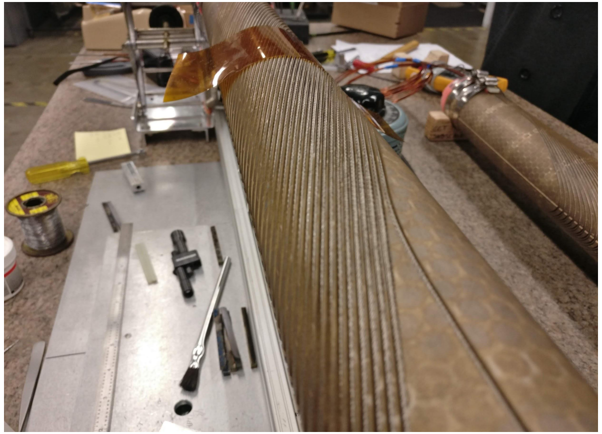
Figures courtesy M. Marchevsky

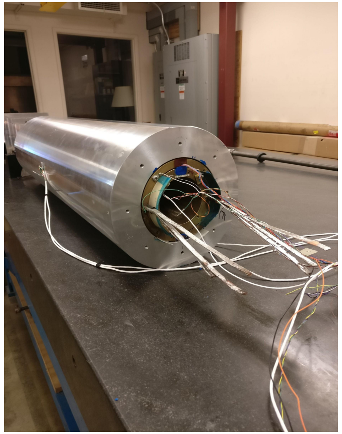
Experimental setup for acoustic data

NERSC

- All data from canted-cosine-theta Nb3Sn dipole magnet CCT4 at LBL
- Use acoustic sensors to measure events
- Data courtesy M. Marchevsky and collaborators



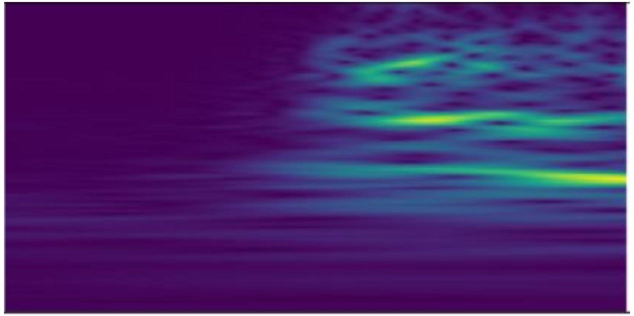




Experimental setup and data

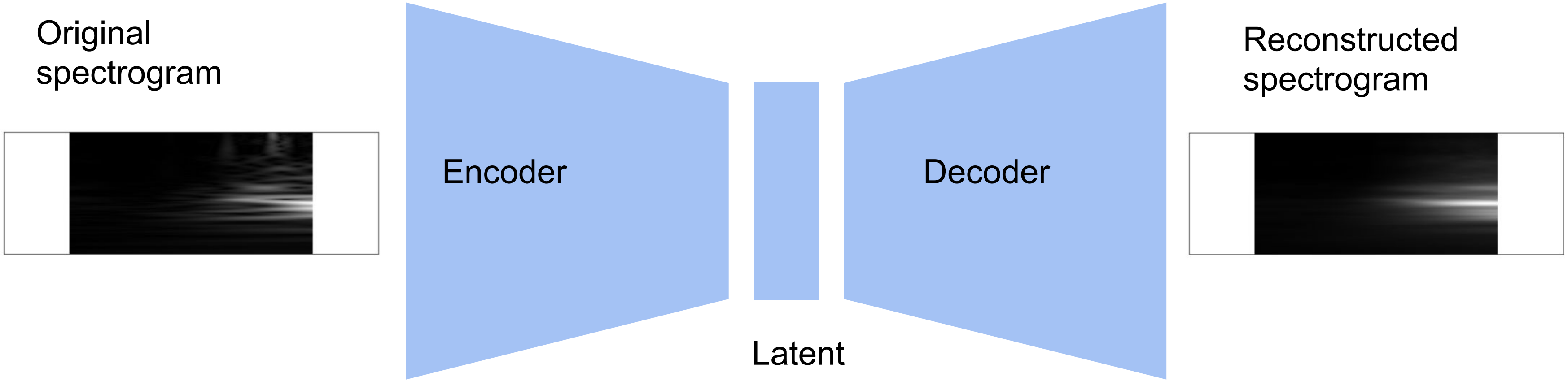
- Using audio data from two sensors (top and bottom of magnet)
- “Events” are automatically identified, windowed, and spectrograms are created
- We apply image processing techniques to these spectrograms (which may or may not be a good idea, since the x and y axis have different meanings)

Frequency



Time

Idea: use autoencoder to find clusters



Latent space

Does the network "learn" anything meaningful while encoding and decoding the images?

If we look at the encoded images in the latent space, do they exhibit clustering?

Autoencoder architecture

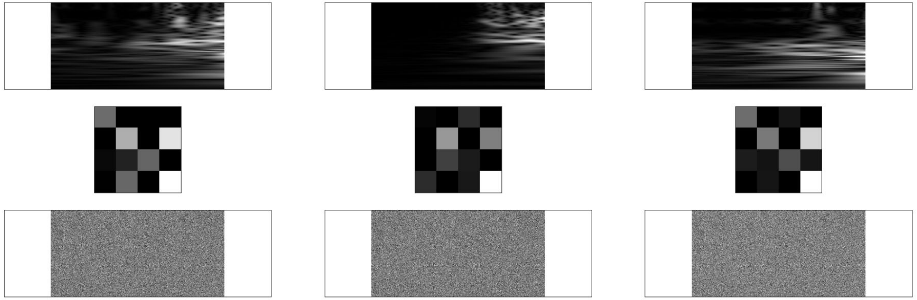
- 3 layer, dense network
- Latent space size 16
- Batch size 8
- Used binary crossentropy loss function
- Trained 20 epochs
- Performed HPO using Talos package to find optimal hyperperameters

```
autoencoder = Sequential()

# Encoder Layers
autoencoder.add(Dense(4 * encoding_dim, input_shape=(input_dim,), activation='relu'))
autoencoder.add(Dense(2 * encoding_dim, activation='relu'))
autoencoder.add(Dense(encoding_dim, activation='relu'))

# Decoder Layers
autoencoder.add(Dense(2 * encoding_dim, activation='relu'))
autoencoder.add(Dense(4 * encoding_dim, activation='relu'))
autoencoder.add(Dense(input_dim, activation='sigmoid'))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                8388672
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 16)                528
_________________________________________________________________
dense_4 (Dense)              (None, 32)                544
_________________________________________________________________
dense_5 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_6 (Dense)              (None, 131072)            8519680
=================================================================
Total params: 16,913,616
Trainable params: 16,913,616
Non-trainable params: 0
```

What features are present in the data?

Put the spectrograms into a randomly initialized, untrained network

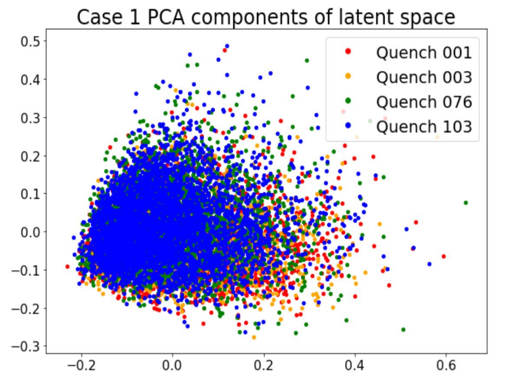


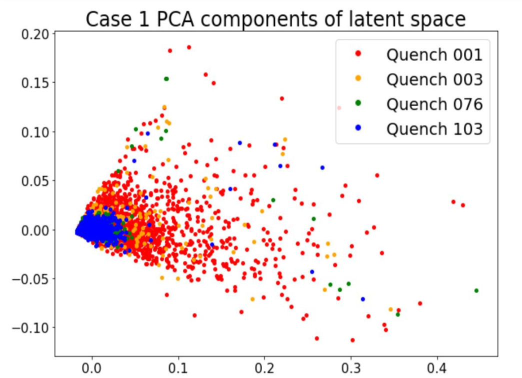
Clearly it hasn't learned anything!

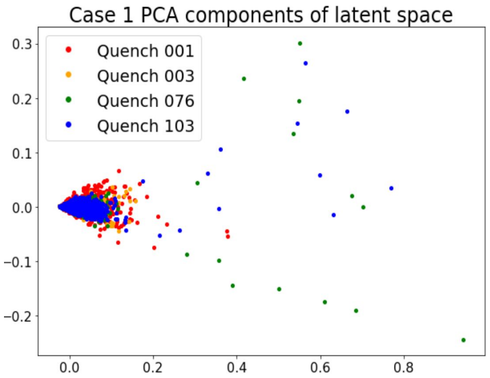
Normalize each spectrogram

Normalize to max value of quench

No normalization

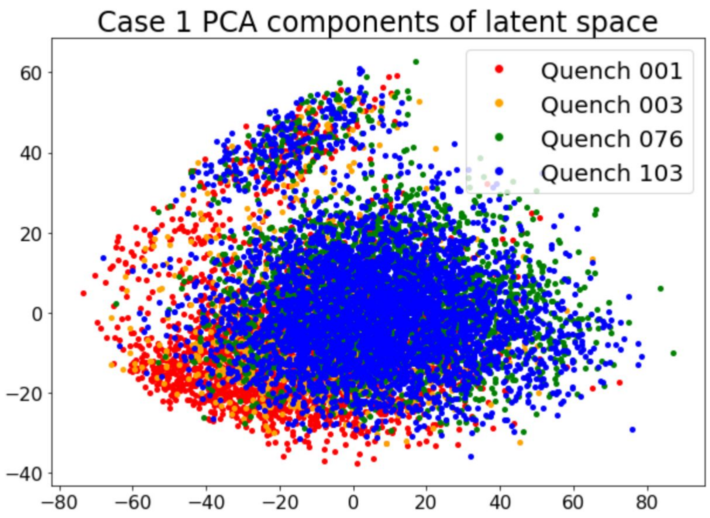


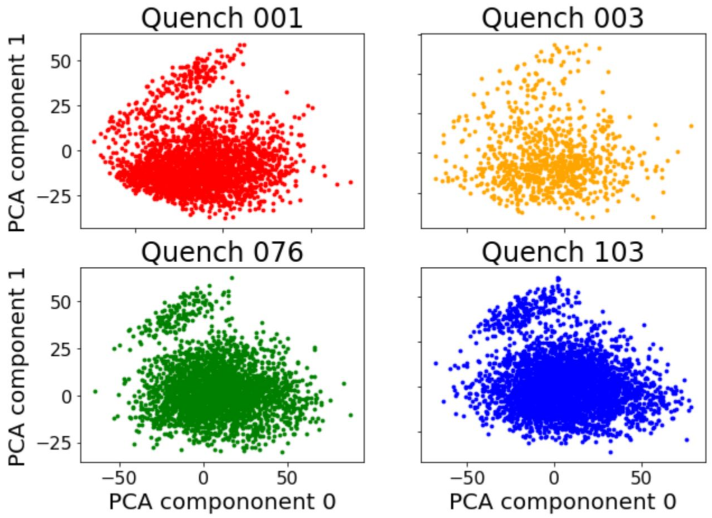




Key was in normalization

It wasn't until we normalized each spectrogram to its max value that we saw clustering our latent space

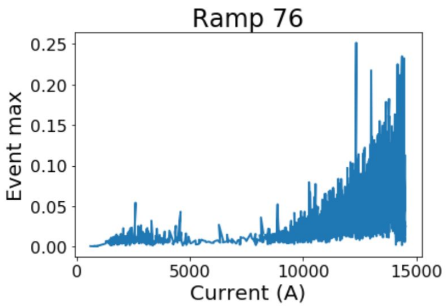


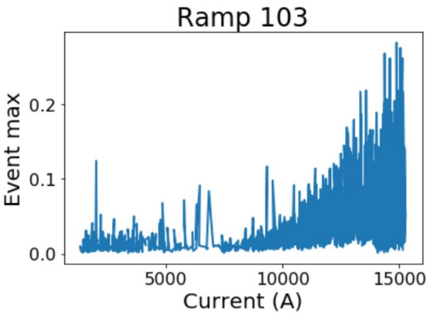


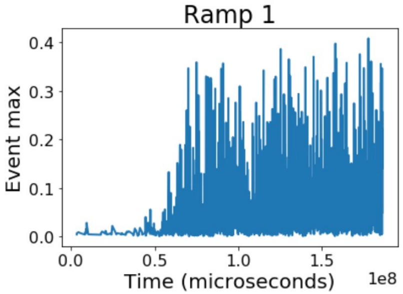
Event max as function of current, time

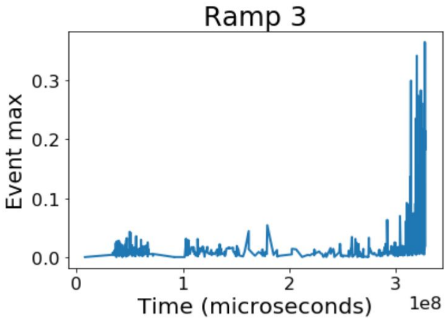
No current data

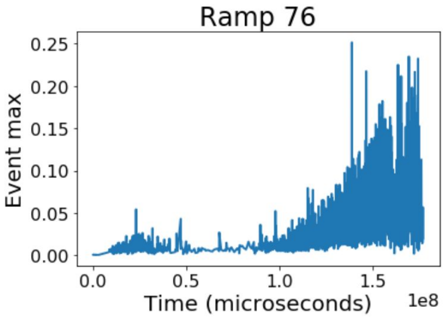
No current data

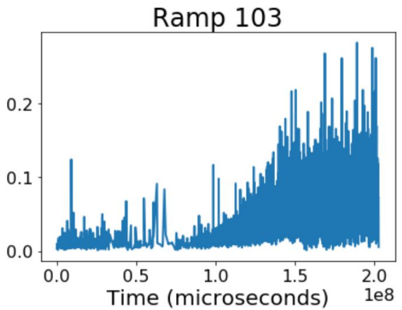








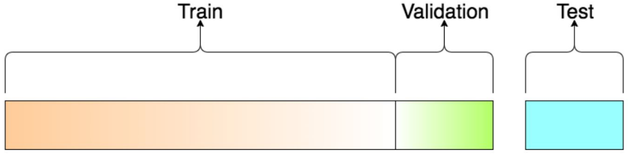




Clustering behavior is robust

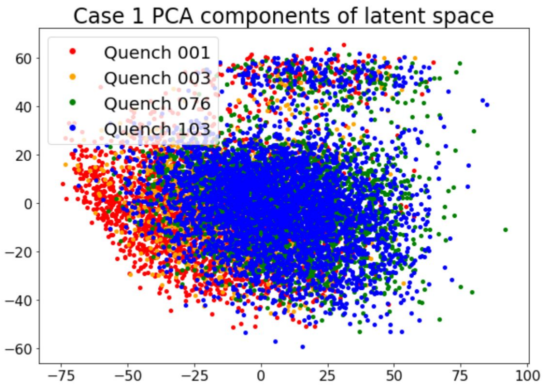
Training the network while leaving out a ramp (103) results in robust clustering in the latent space

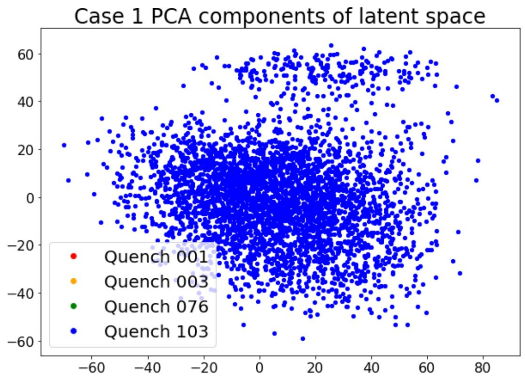
Ramp 001, 003, 076

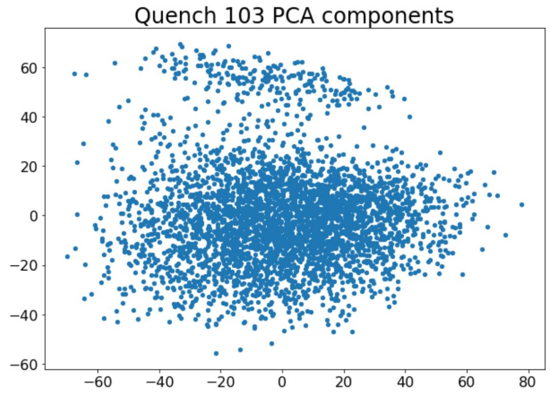


Ramp 103

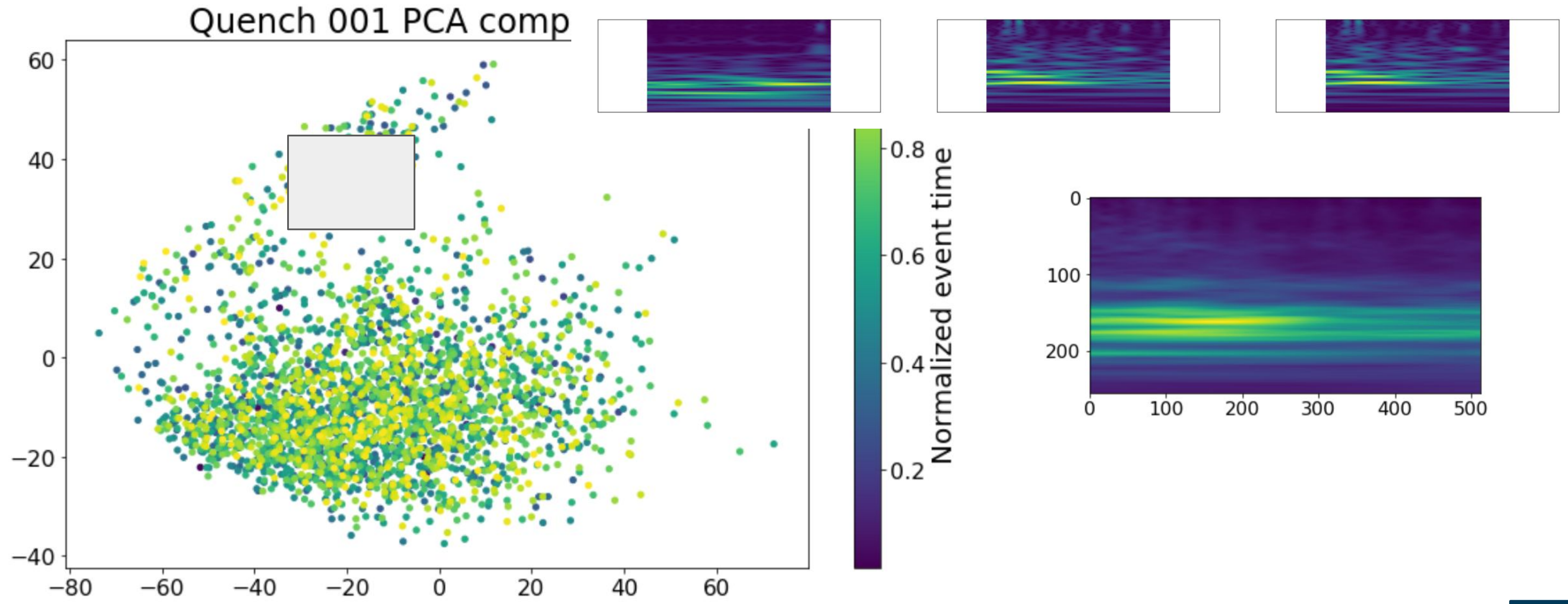
A visualisation of the splits



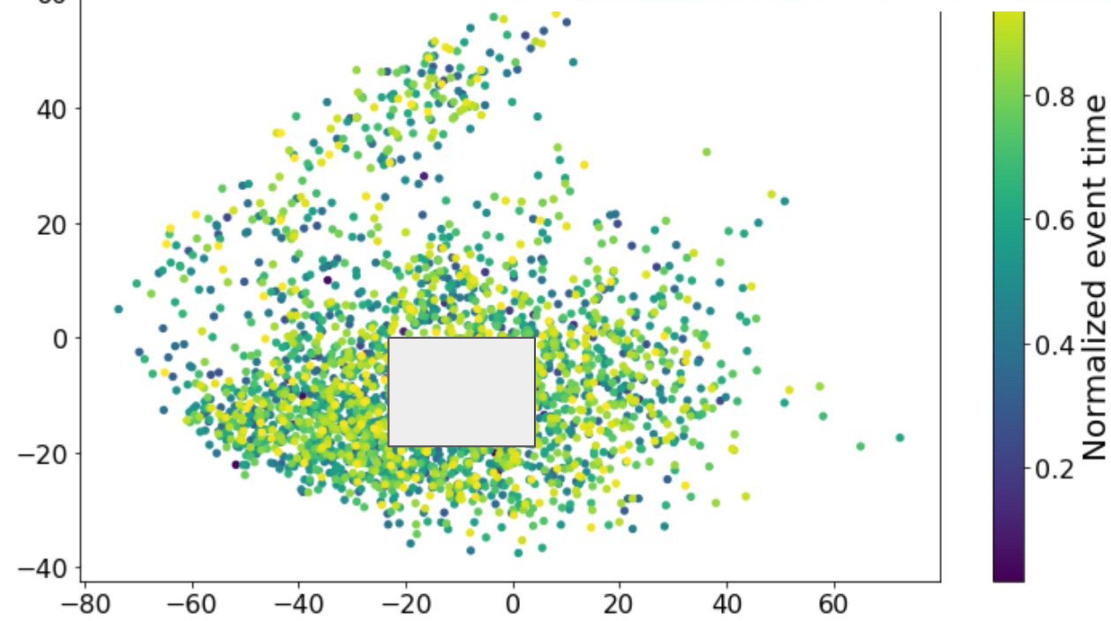


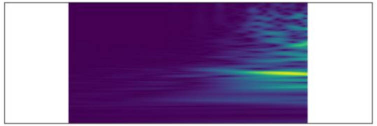


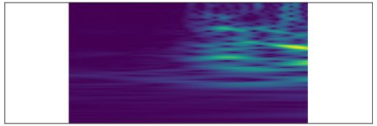
Top cluster Ramp 1

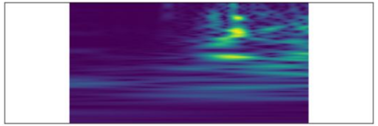


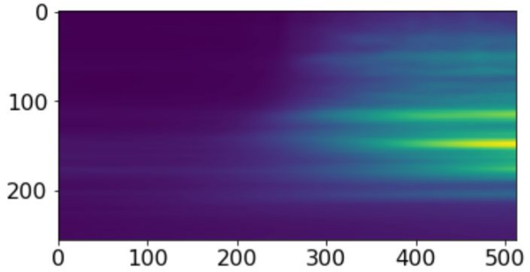
Bottom cluster Ramp 1



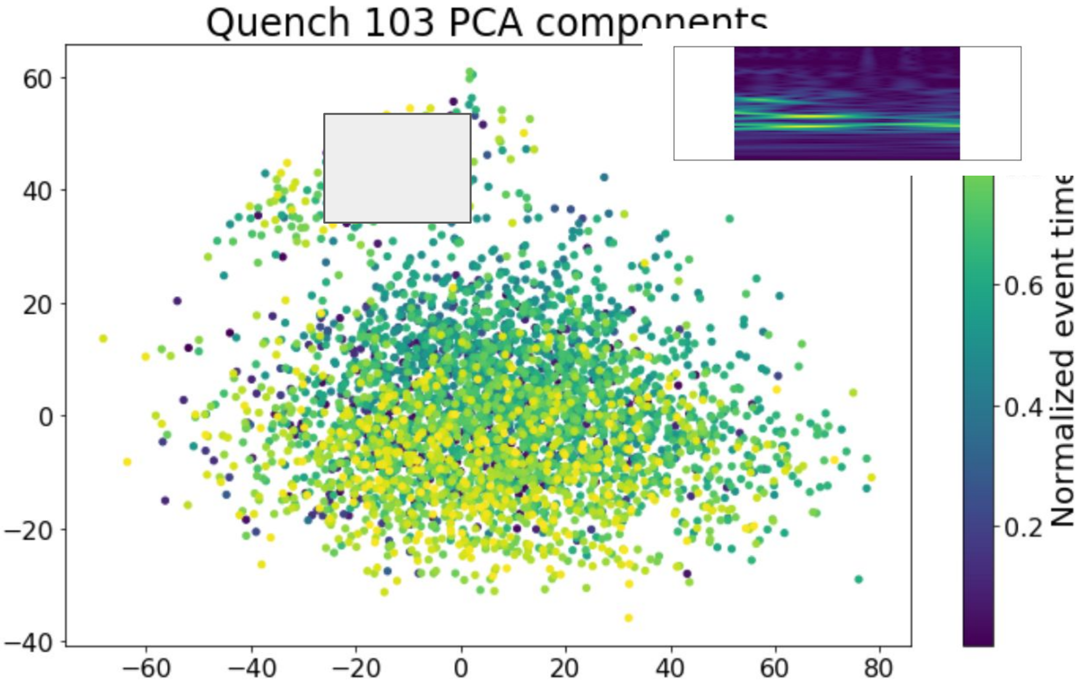


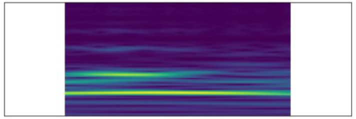


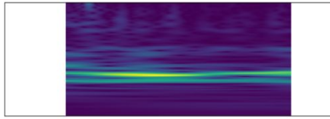


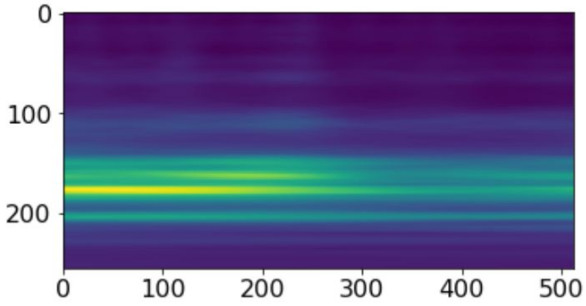


Top cluster Ramp 103

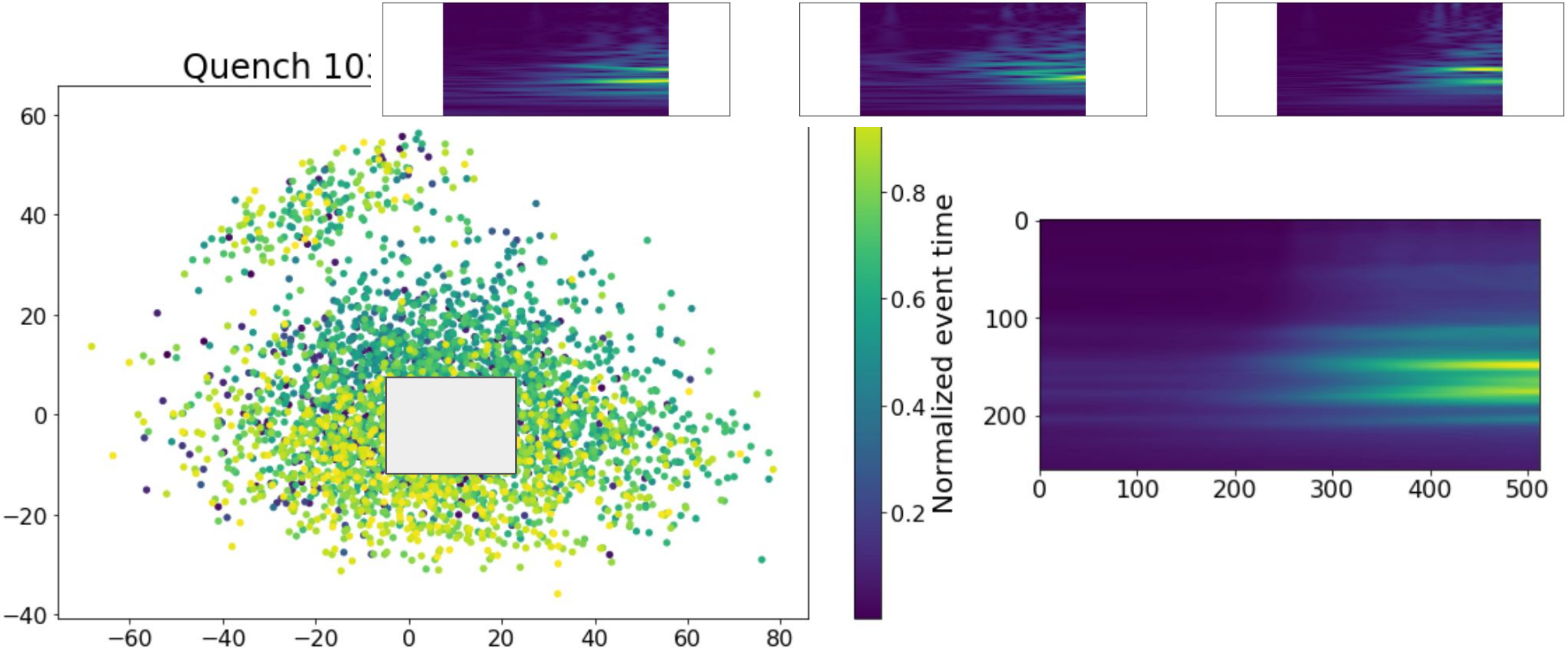




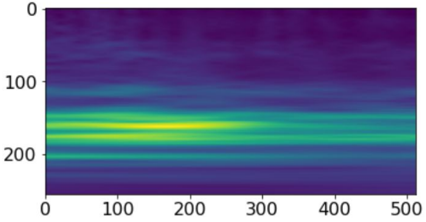
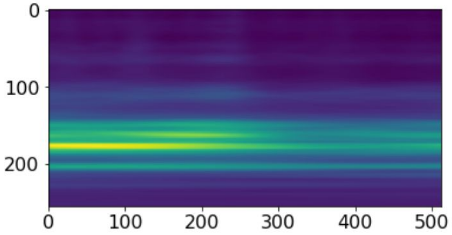
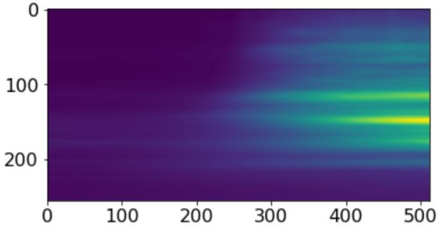
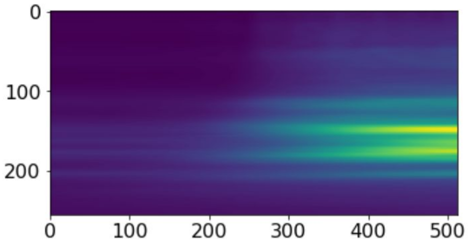


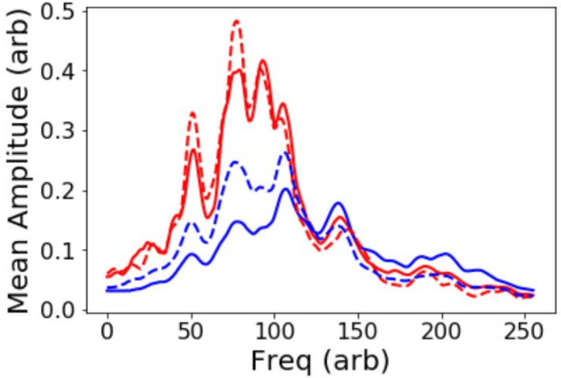


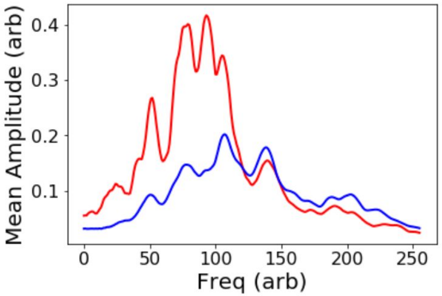
Bottom cluster Ramp 103

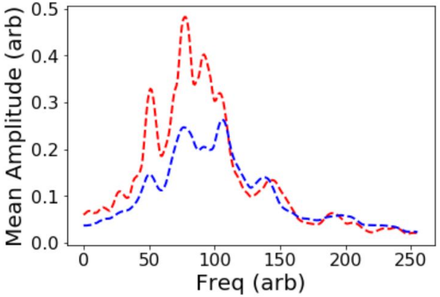


Compare frequency content of mean spectrograms

	Ramp 001	Ramp 103
Top cluster		
Bottom cluster		







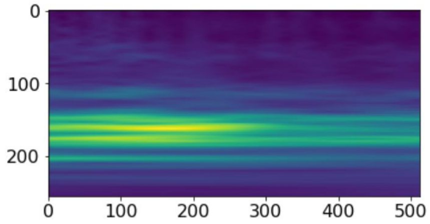
Differences in frequency between Ramp 001 and Ramp 103 occur in the low frequencies

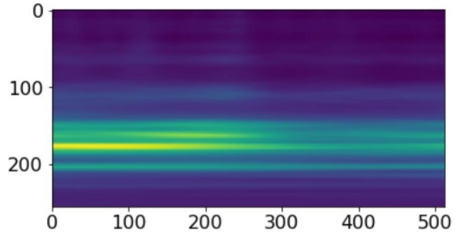
Compare temporal behavior of mean spectrograms

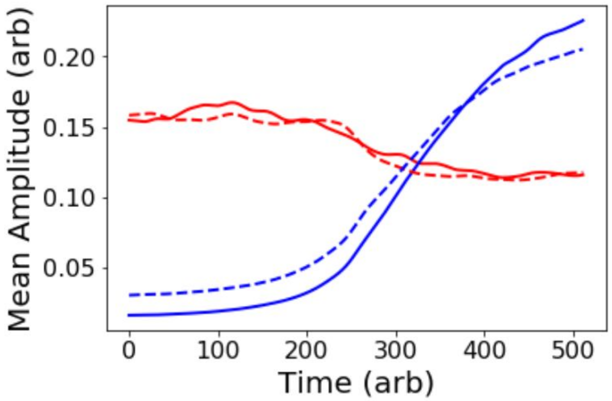
Ramp 001

Ramp 103

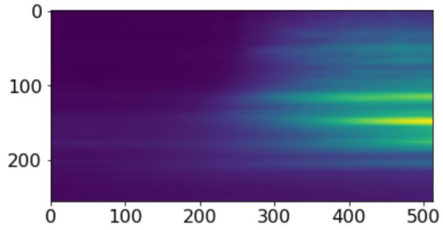
Top cluster

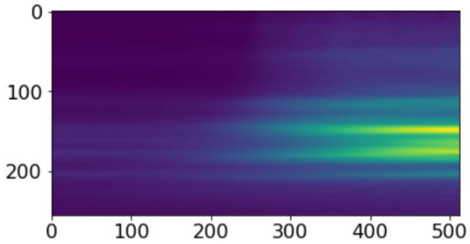


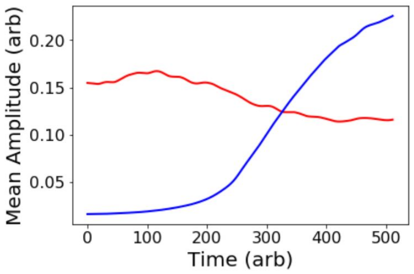


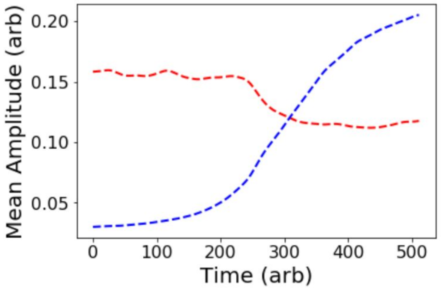


Bottom cluster









Differences in temporal behavior between Ramp 001 and Ramp 103 are primarily in the bottom cluster

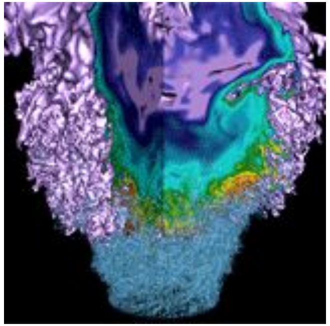
Summary

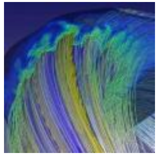
- Used a dense autoencoder to examine latent space of four training ramps for CCT4 magnet
- Robust clustering observed in the latent space **only when each spectrogram is normalized**
- Top, smaller cluster corresponds to slow events
- Bottom, larger cluster corresponds to fast events
- Seems to support original hypothesis of two classes of acoustic events (cracking and slip-stick)

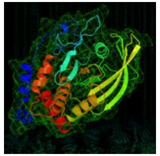
Open questions

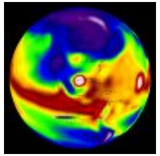
- Does this method generalize to other data? (new magnet CCT5, for example)
- Is this a good approach? Spectrograms are not like images, their x and y axes are fundamentally different quantities
- Is the network sorting by `long` and `short`? By frequency content? → future work to look more deeply into this
- Can we use `supervised` methods to label the precursor events and see if the network can identify similar precursors in a previously unseen dataset?
- Other thoughts, comments, questions, suggestions?

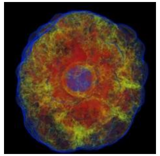
Thank you!

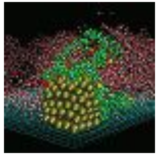










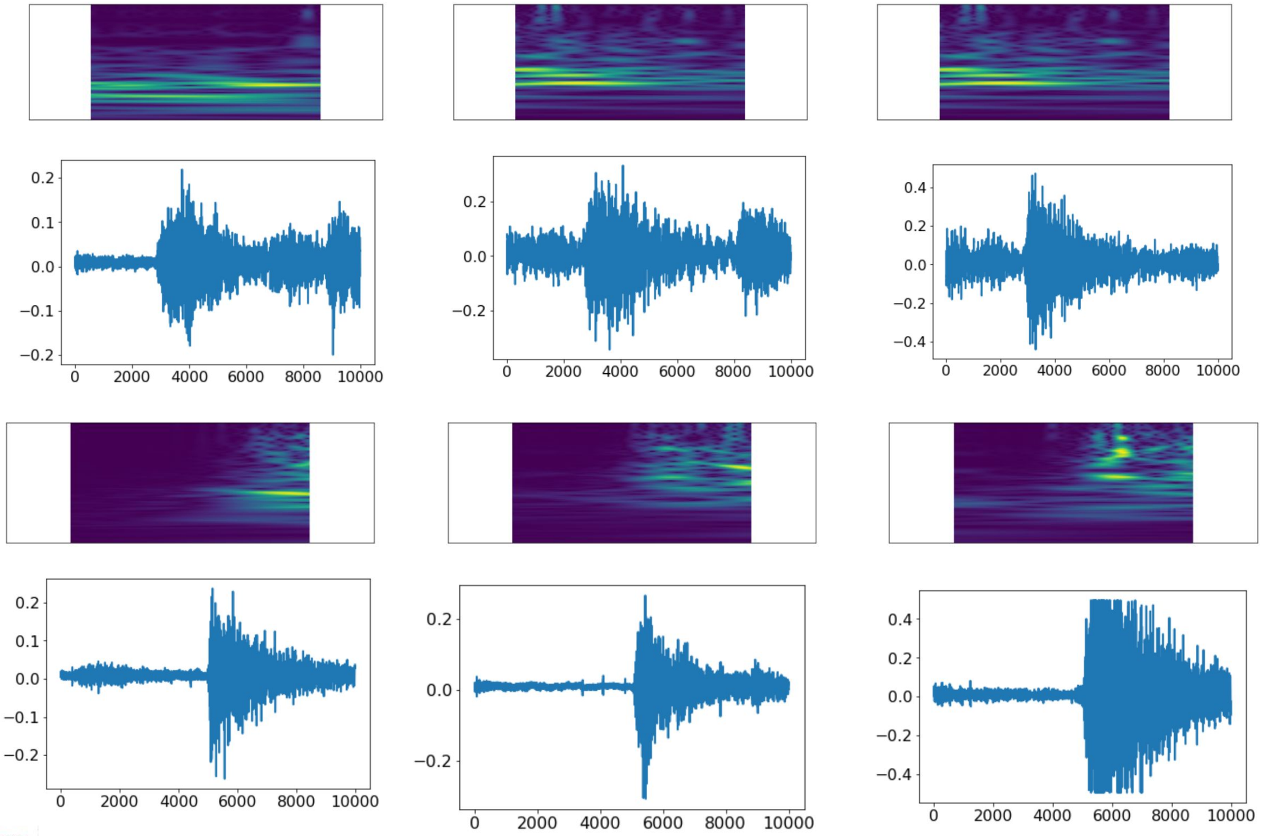


NERSC

Ramp 001

Top cluster

Bottom cluster



Ramp 103

Top cluster

Bottom cluster