A Data-Driven Approach to Structure-Based Large Scale Audio Version Identification

WILL GILLETTE

RESEARCH ADVISOR: DR. CHRIS TRALIE

Overview

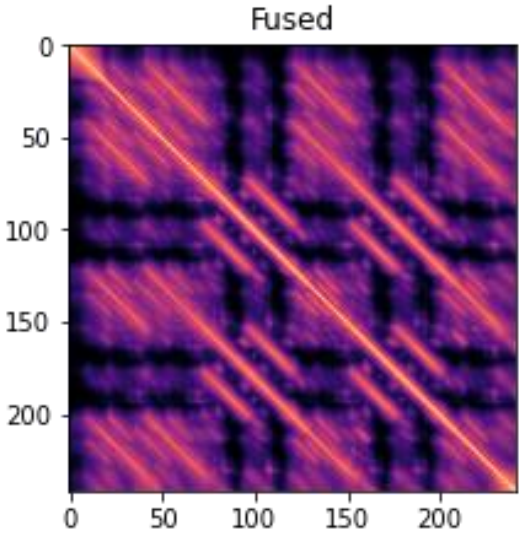
- Problem: Detecting “cover songs”, which are songs that are different versions of each other
- Experiment Strategy: Develop a scalable metric, or pairwise distance, that is high for pairs of songs that are versions of each other and low for unrelated pairs of songs
- Mission: Determine the most accurate and scalable method for cover song identification through performing a series of supervised experiments

Experiment Context

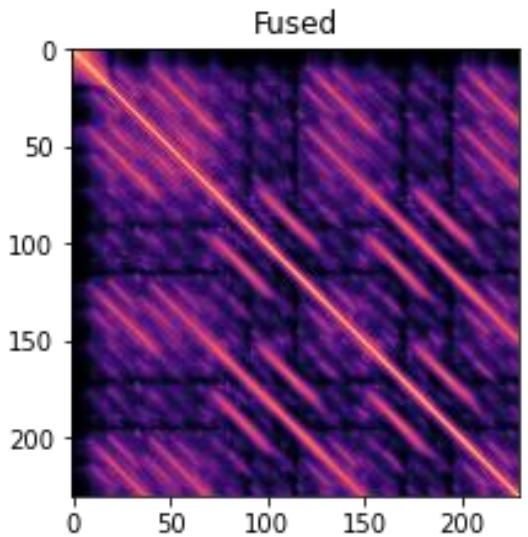
This section includes information regarding the data and metrics relevant to the experiments performed in Python.

Self-Similarity Matrices

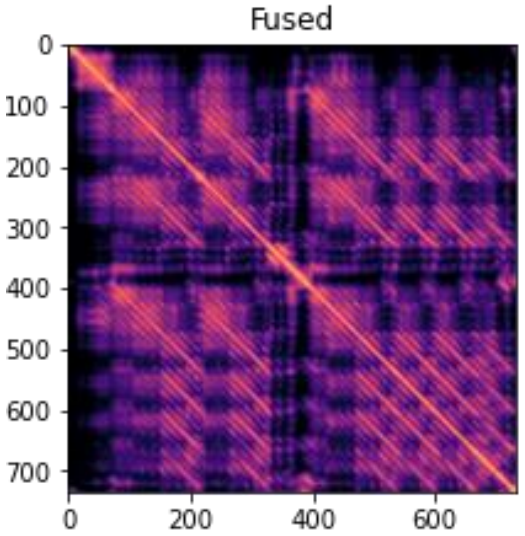
- Convert songs to self-similarity matrix (SSM) images, allowing us to encapsulate pitch, tempo, and timbral features into a single image



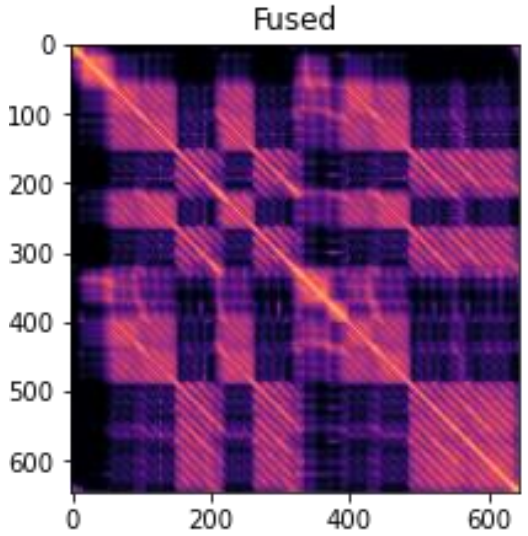
We Can Work It Out
Version #1



We Can Work It Out
Version #2



Take On Me
Version #1

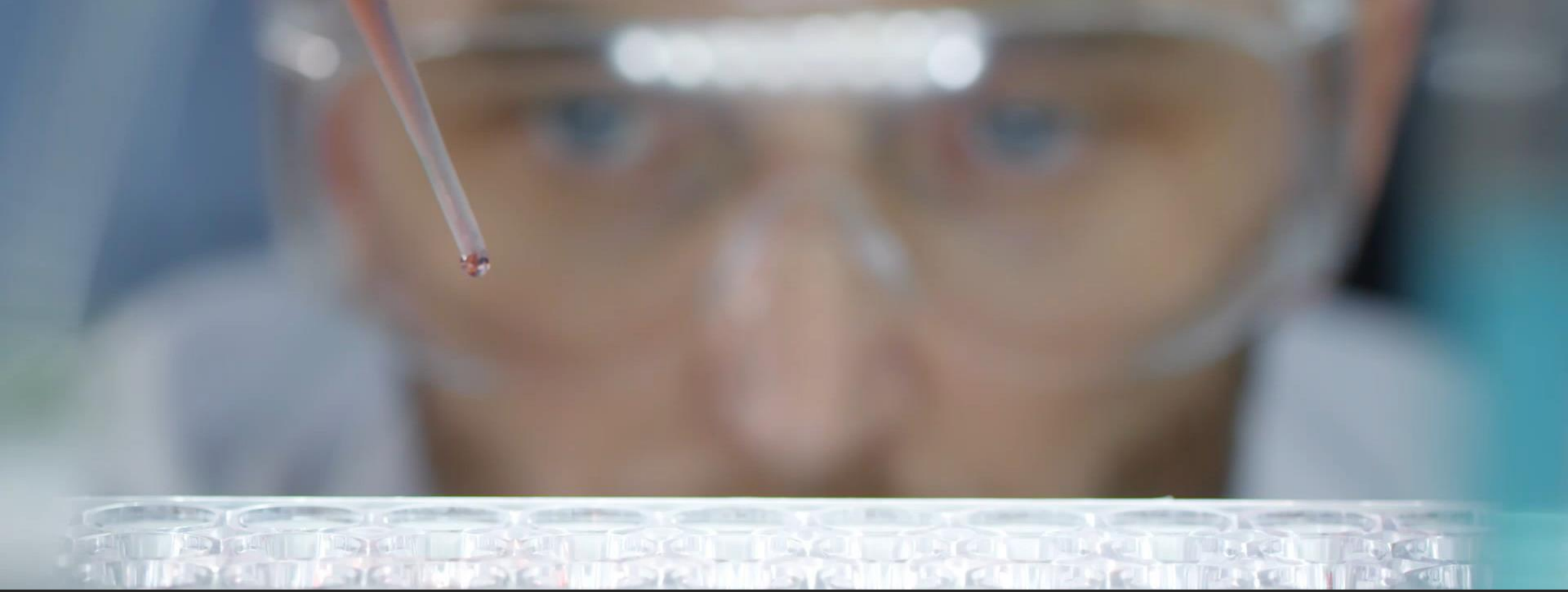


Take On Me
Version #2

Datasets

- Each of these datasets consists of a certain number of cliques each containing songs that are versions of each other:

1. Covers1000: Consists of 1,008 songs and 396 cliques
2. Covers80: Consists of 160 songs and 80 cliques
3. Datacos Testing: Consists of 15,000 songs and 3,000 cliques
4. Datacos Training: Consists of 97,666 songs and 12,122 cliques



Unsupervised Experiments

This section consists of prior work from Ursinus alumnus Ben Klybor. The experiments involve human intervention with experimenting with the structure of the self-similarity matrices.

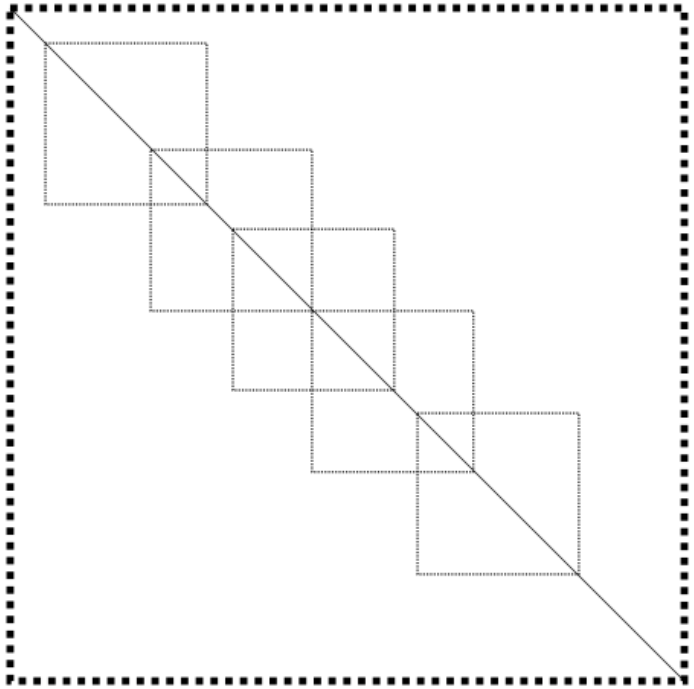
Beat-Synchronous SSM Shingle Features

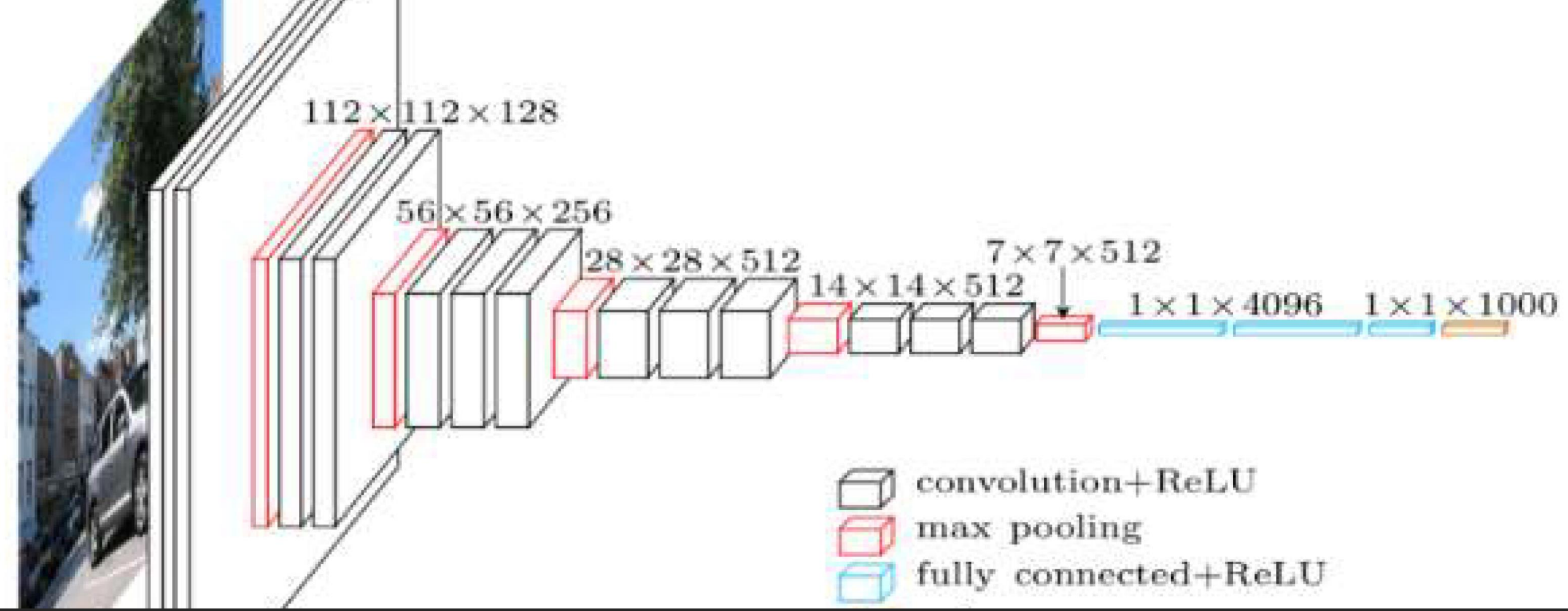
Gather all the 75x75 shingles across the diagonal of each of the 512x512 fused SSMs

For each SSM, stack up these shingles and take the median of each pixel

Convert to vectors, compute pairwise distances and evaluation statistics

75x75 Shingles





Supervised Experiments

These deep learning experiments use the PyTorch library to create convolutional neural networks (CNN) that train a model to detect structural patterns. The model learns from examples.

Vanilla Fused SSMs CNN

Network Architecture:

- **Convolution Layers**: Series of repeated sequences of pooling and convolution down layers
- **Fully-Connected Layer**: Transforms the output of the previous layer to a specified-dimensional vector

Training:

1. Input a set of examples to the loss function
2. Perform backpropagation to adjust weights/biases
3. Repeat **steps 1 and 2** for thousands of iterations
4. Repeat **step 3** for a certain number of epochs

Testing:

1. Input the dataset to the model, receiving output vectors of a specified dimension
2. Compute the Euclidean distances between these vectors
3. Calculate the evaluation statistics

CNN Parameter Adjustments

16 Channels, 1000-dimensional Output

```
<class 'torch.nn.parameter.Parameter'> torch.Size([16, 1, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([16])
<class 'torch.nn.parameter.Parameter'> torch.Size([16, 16, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([16])
<class 'torch.nn.parameter.Parameter'> torch.Size([16, 16, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([16])
<class 'torch.nn.parameter.Parameter'> torch.Size([1000, 65536])
<class 'torch.nn.parameter.Parameter'> torch.Size([1000])
```

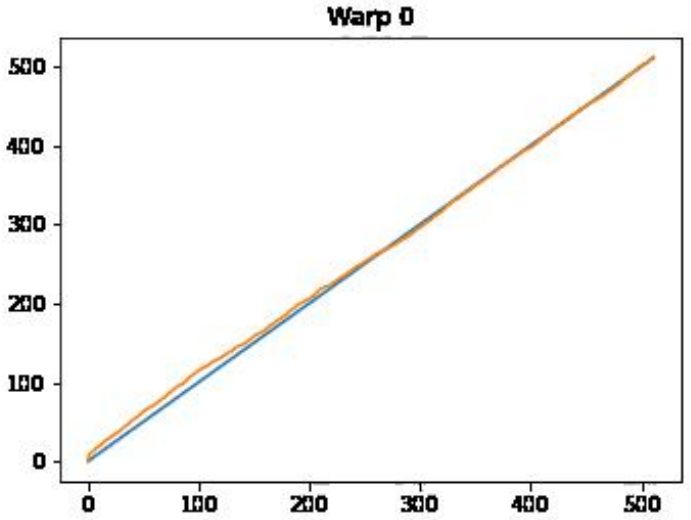
32 Channels, Doubling, 500-dimensional Output

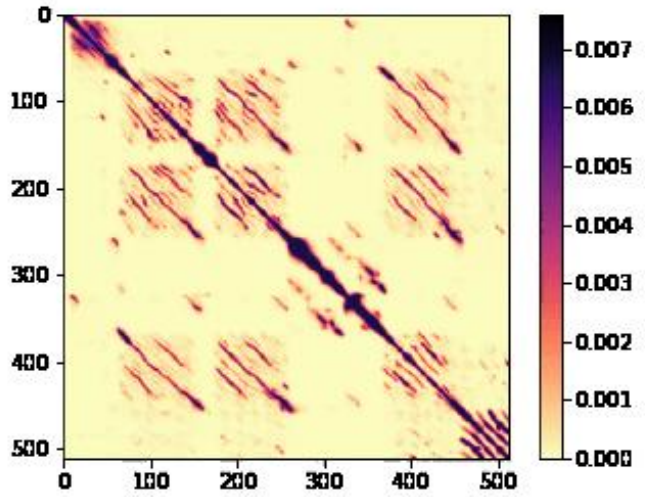
```
<class 'torch.nn.parameter.Parameter'> torch.Size([32, 1, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([32])
<class 'torch.nn.parameter.Parameter'> torch.Size([64, 32, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([64])
<class 'torch.nn.parameter.Parameter'> torch.Size([128, 64, 3, 3])
<class 'torch.nn.parameter.Parameter'> torch.Size([128])
<class 'torch.nn.parameter.Parameter'> torch.Size([500, 524288])
<class 'torch.nn.parameter.Parameter'> torch.Size([500])
```

- Comparing the effect of deeper versus shallow networks on the results
- Doubling the number of channels each time the network halves the dimensionality of the input images
- Changing the output dimensionality of the network
- Changing the number of channels in the first layer of the network

Data Augmentation via Warping SSMs

- Next Step: Substantially increase the size of the training data through time warping a series of SSMs and rerun the supervised experiments, training the model to be insensitive to changes in time warping





Thank You

SPECIAL THANKS TO DR. CHRIS TRALIE FOR THIS OPPORTUNITY AND HIS TIME. THANK YOU TO EVERYONE FOR COMING OUT TODAY!

Questions?