Team 7:

Matthew Tang
Ben Nguyen
Mazer Xu
Tianshu Wei

Our Interpretation of the problem statement

- The data provided contain hundreds of features, with some crucial to predicting particle movements while some not as important
- The data is given in 4D, with one time dimension and 3 space dimensions
- Tags for gases and particles are provided in text files
- The goal is to use features in the current timestep to predict Y in same timestep

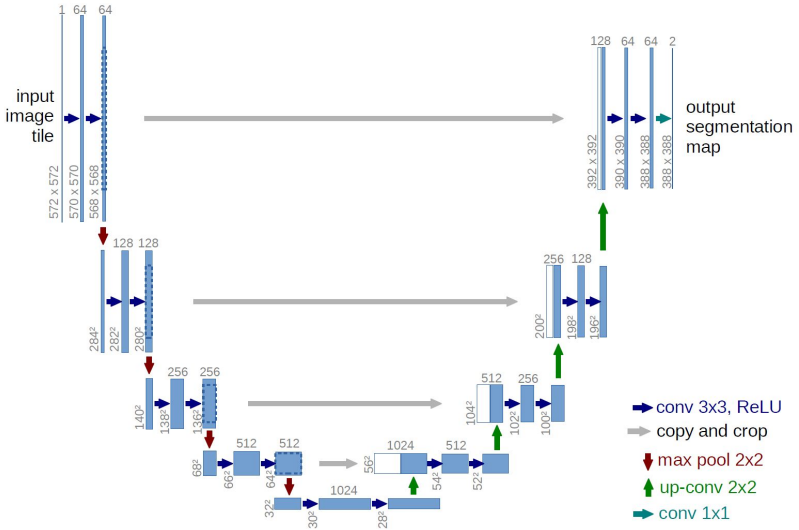
Initial thoughts

Raw data needs to be sanitized so a model can be trained to predict particle movement in future time and space.

How to handle high dimensional image data

Initial approach

- PCA for dimensionality reduction (Cannot use all features $X_1 \ldots X_{101}$)
- UNet architecture adapted from image-image translation/segmentation to improve autoencoder model
- Use LSTM or Transformer for the FC layer to use sequence data
- Adapt UNet to 3D images



Initial approach

- PCA for dimensionality reduction (Cannot use all features $X_1 \ldots X_{101}$)
- UNet architecture adapted from image-image translation to improve autoencoder model
- Use LSTM or Transformer for the FC layer to use sequence data
- Adapt UNet to 3D Convolutions since images

Difficulties

- Normal PCA does not work on Hyperspectral Images (Each X_i is one channel)
- Not enough GPU memory to hold sequence data
- Likely not enough samples to train Transformer

Preparing Data

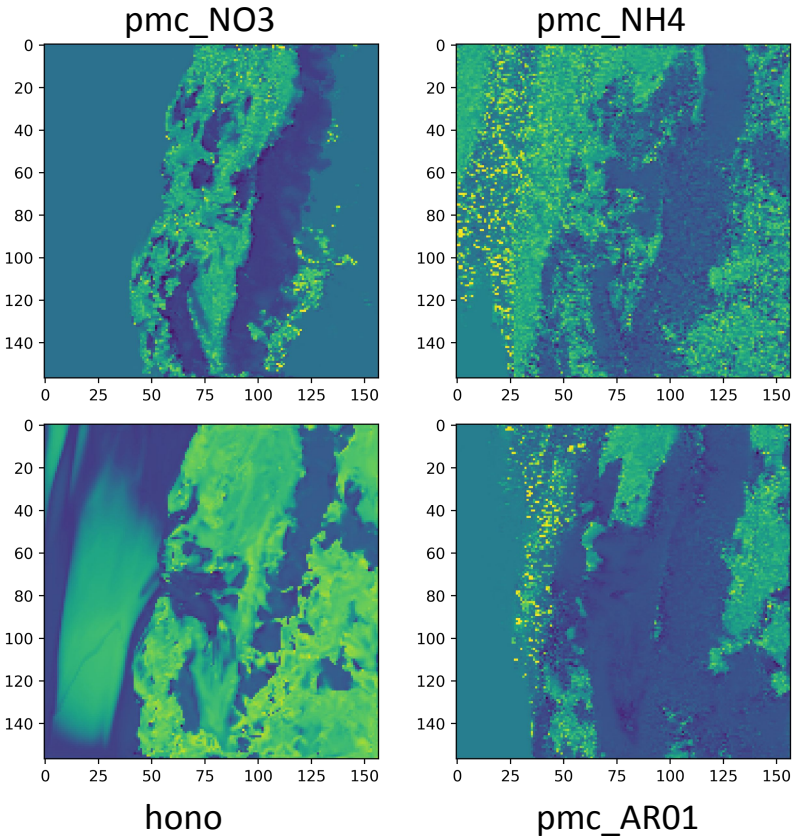
- Read tags from file, removing features that have little correlation to predict particle movements
- Resize the image to size of 157*157 by transforming to fit our architecture
- Normalization: by subtracting mean and then dividing by standard deviation on each channel
- Rearrange data so that data can be accessed by the time variable

```
ch3so2h (133, 39, 159, 169)
ch3sch2oo (133, 39, 159, 169)
ch3so2 (133, 39, 159, 169)
ch3so3 (133, 39, 159, 169)
ch3so2oo (133, 39, 159, 169)
ch3so2ch2oo (133, 39, 159, 169)
SULFHOX (133, 39, 159, 169)
```

```
In [6]: data = torchData
        print(data.size())

        torch.Size([10, 133, 39, 157, 157])

In [26]: X.shape

Out[26]: torch.Size([133, 10, 39, 157, 157])

In [27]: Y.shape
```

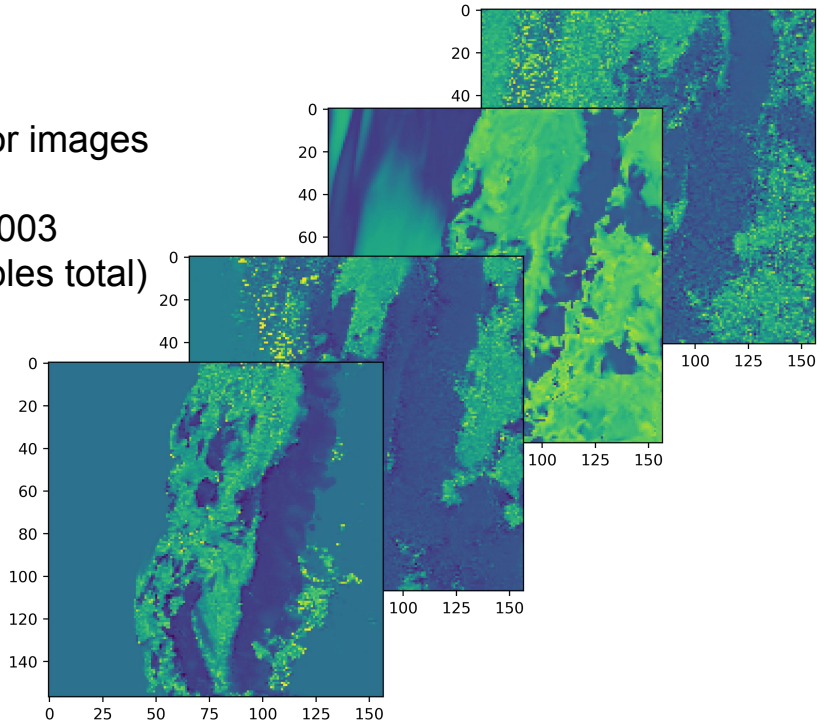
Our Solution

- Treat inputs as Hyperspectral 3D images
- Each feature X_i is an image channel



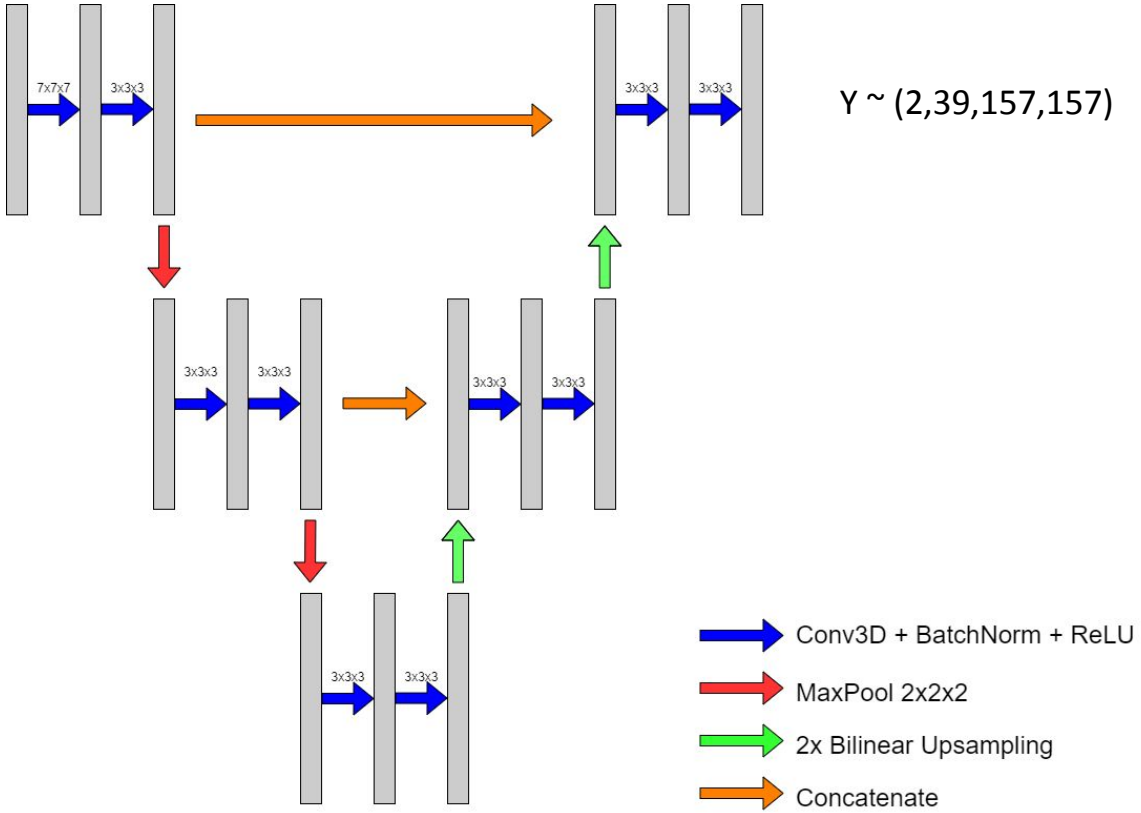
Our Solution

- Treat inputs as Hyperspectral 3D images
- Choose only 10 input features, use them as channels for images
- Input is 3D image with 10 channels ~ (10, 39,157,157)
- Output is 3D image with 2 channels: ccn_001 and ccn_003
- Network maps $X_t \rightarrow Y_t$ for a single timestep t (134 samples total)



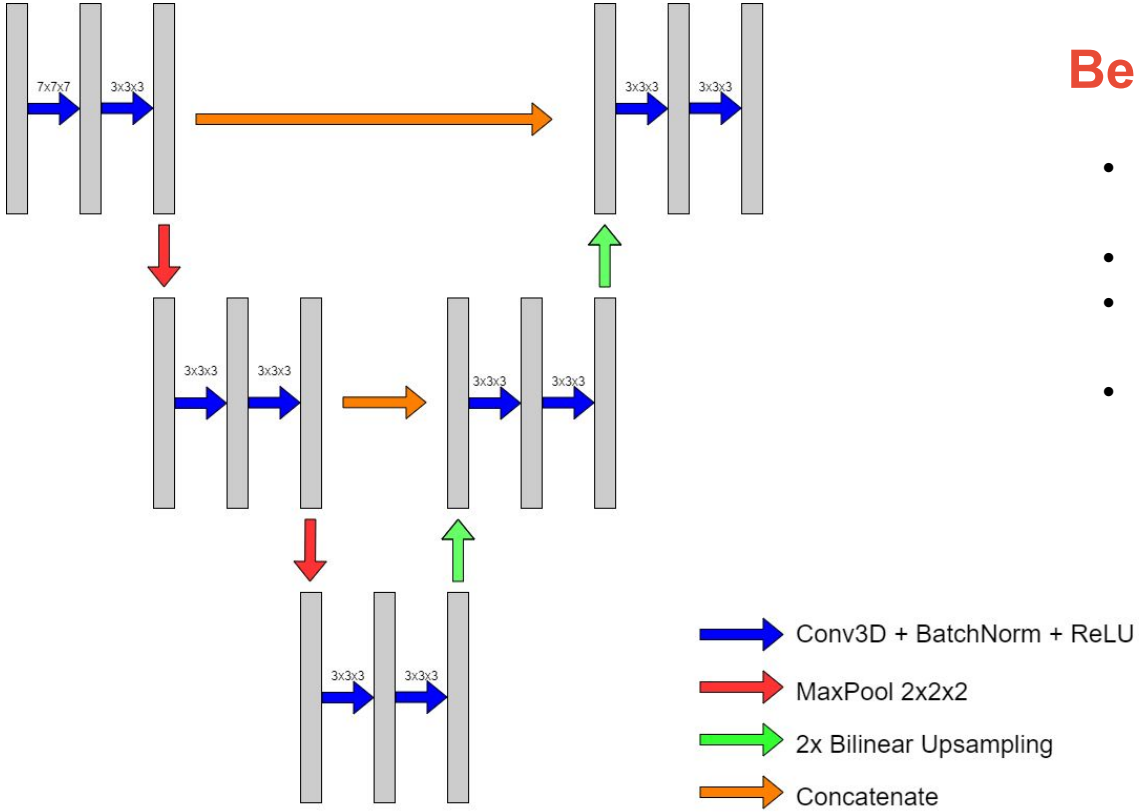
Architecture

X ~ (10,39,157,157)



Y ~ (2,39,157,157)

Architecture

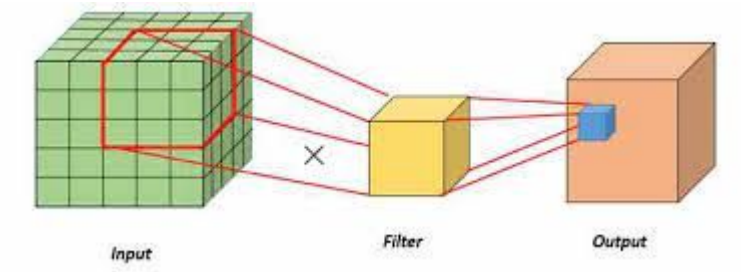


Benefits

- Preserve lower level details from original input
- Features of input seen in output
- Incorporates low and high level features
- Information from different features X_i fit into one image input

3D Conv

- Adapting 2D UNet is non-trivial
- Need to calculate new sizes for each layer
- Adds parameters and gradients → Need more GPU memory



$$D_{out} = \left\lfloor \frac{D_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[2] - \text{dilation}[2] \times (\text{kernel\_size}[2] - 1) - 1}{\text{stride}[2]} + 1 \right\rfloor$$

Challenges & Difficulties

GPU Memory

- Each GPU can only fit 1 sample

Solution

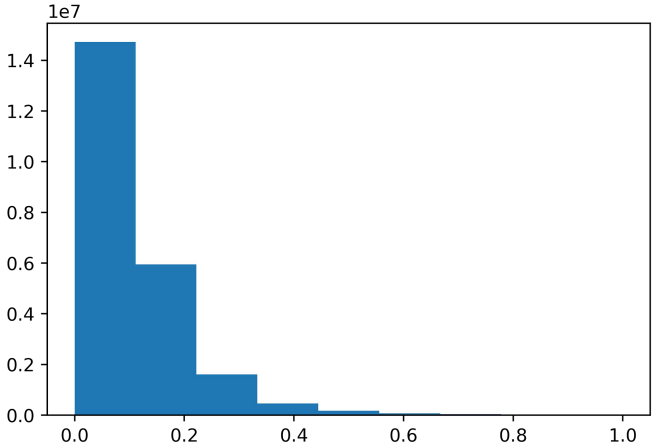
- Parallelize training with Horovod
- Train in batches of 3 with 4 GPUs
  - Max possible batch size on 4 GPU is 5
  - 12GB / 16GB used
- Trade off between memory and convergence

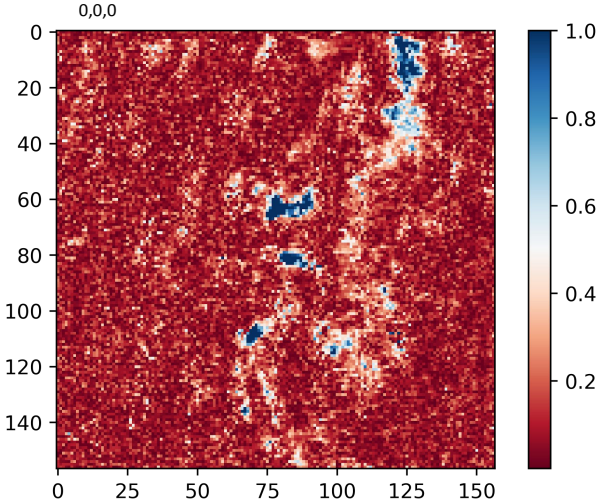
```
[1,3]<stdout>:Loss  8.088553488254547
[1,3]<stdout>:epoch  9
[1,0]<stdout>:Loss  7.749783873558044
[1,0]<stdout>:epoch  9
[1,1]<stdout>:Train Epoch: 9 [0/34 (0%)]        Loss: 0.857045
[1,0]<stdout>:Train Epoch: 9 [0/34 (0%)]        Loss: 0.864655
[1,2]<stdout>:Train Epoch: 9 [0/34 (0%)]        Loss: 0.906699
[1,3]<stdout>:Train Epoch: 9 [0/34 (0%)]        Loss: 0.913215
[1,0]<stdout>:Train Epoch: 9 [4/34 (11%)]       Loss: 0.859567
[1,3]<stdout>:Train Epoch: 9 [4/34 (11%)]       Loss: 0.849823
[1,1]<stdout>:Train Epoch: 9 [4/34 (11%)]       Loss: 0.870123
[1,2]<stdout>:Train Epoch: 9 [4/34 (11%)]       Loss: 0.870615
[1,2]<stdout>:Train Epoch: 9 [8/34 (22%)]       Loss: 0.898890
[1,1]<stdout>:Train Epoch: 9 [8/34 (22%)]       Loss: 0.931529
[1,3]<stdout>:Train Epoch: 9 [8/34 (22%)]       Loss: 0.871692
[1,0]<stdout>:Train Epoch: 9 [8/34 (22%)]       Loss: 0.843230
--------------------------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code. Per user-direction, the job has been aborted.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun noticed that process rank 0 with PID 270553 on node hal16 exited on signal 9 (Killed).
--------------------------------------------------------------------------
```

Evaluation dataset

Calculate relative error between predicted values and ground truth:

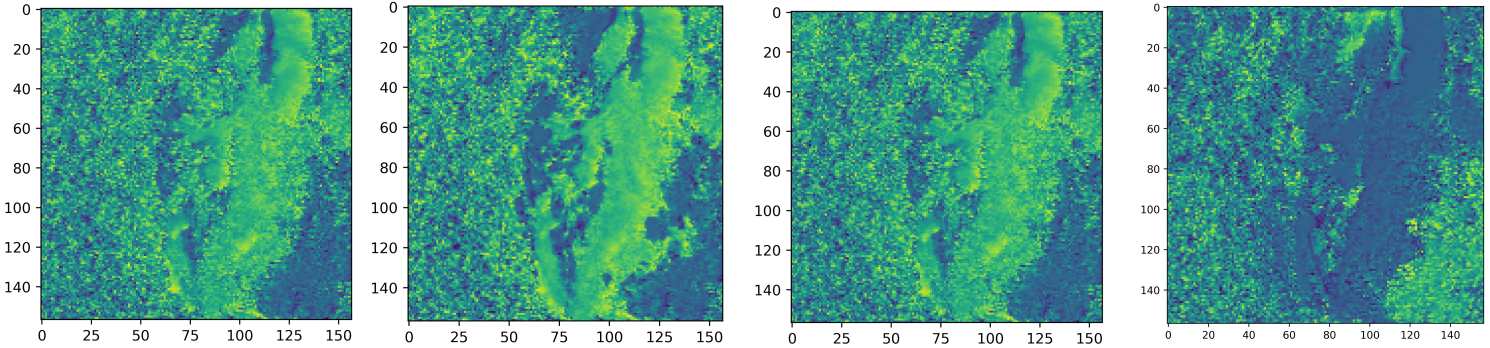
- **86.71%** of voxels under 20% error
  - **80.17%** ccn_001
  - **93.26%** ccn_002
- **10.73%** ±0.32% average error



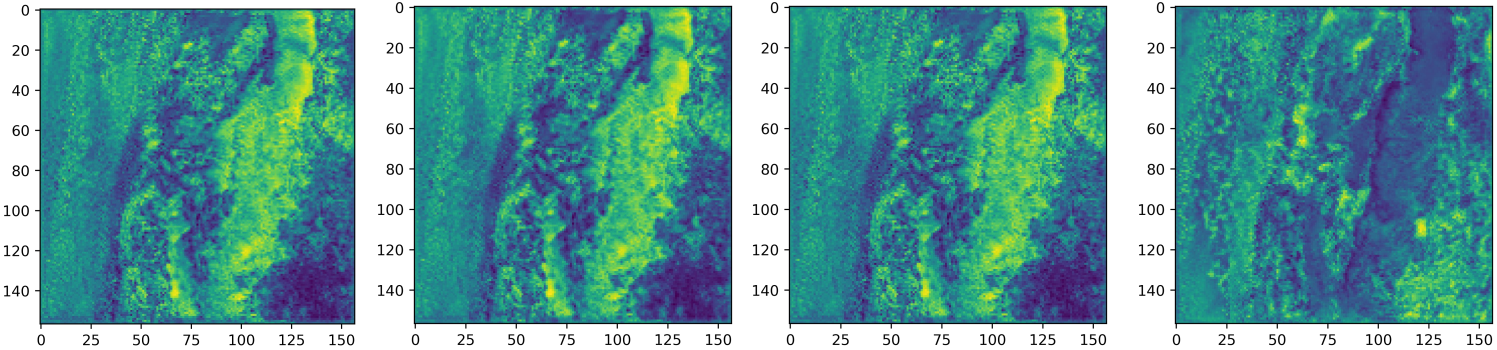


Results

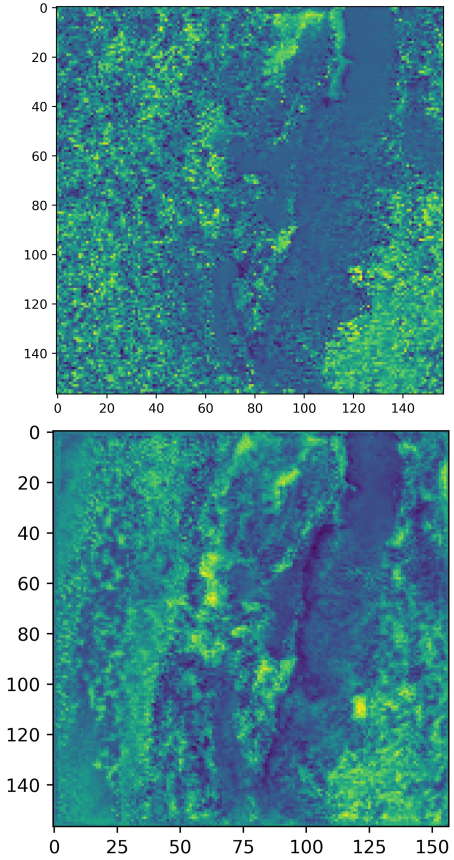
Ground Truth



Predicted



Results



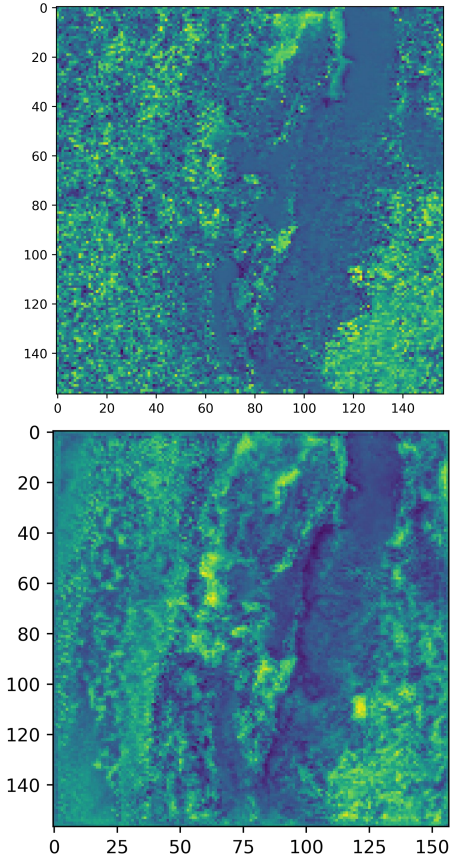
Qualitative comparison

- Clearer images
- Less blurred
- Smoothes out noise

Architecture advantage

- UNet with convolutional layers known to produce clear images compared to autoencoders without skip connections

Why our method works well



Qualitative comparison

- Clearer images
- Less blurred
- Smoothes out noise

Architecture advantage

- UNet with convolutional layers known to produce clear images whereas Autoencoders produce blurry images

Future developments

- Fix current problems
- Incorporate sequence inputs using LSTM or Transformers on image encoding
- Hyperspectral dimensionality reduction using PCA instead of handpicking channels
  https://www.sciencedirect.com/science/article/pii/S221431732030189X
- Increase batch size with more GPUs or better parallelization

Thank You!

Questions?

Contact Information:

- Matthew Tang (mhtang2@illinois.edu)
- Benjamin Nguyen (bnguyen4@illinois.edu)
- Tianshu Wei (tw27@illinois.edu)
- Mazer Xu (mingzex2@illiois.edu)

GitHub: https://github.com/ben44496/hal-ncsa-hackathon-2022