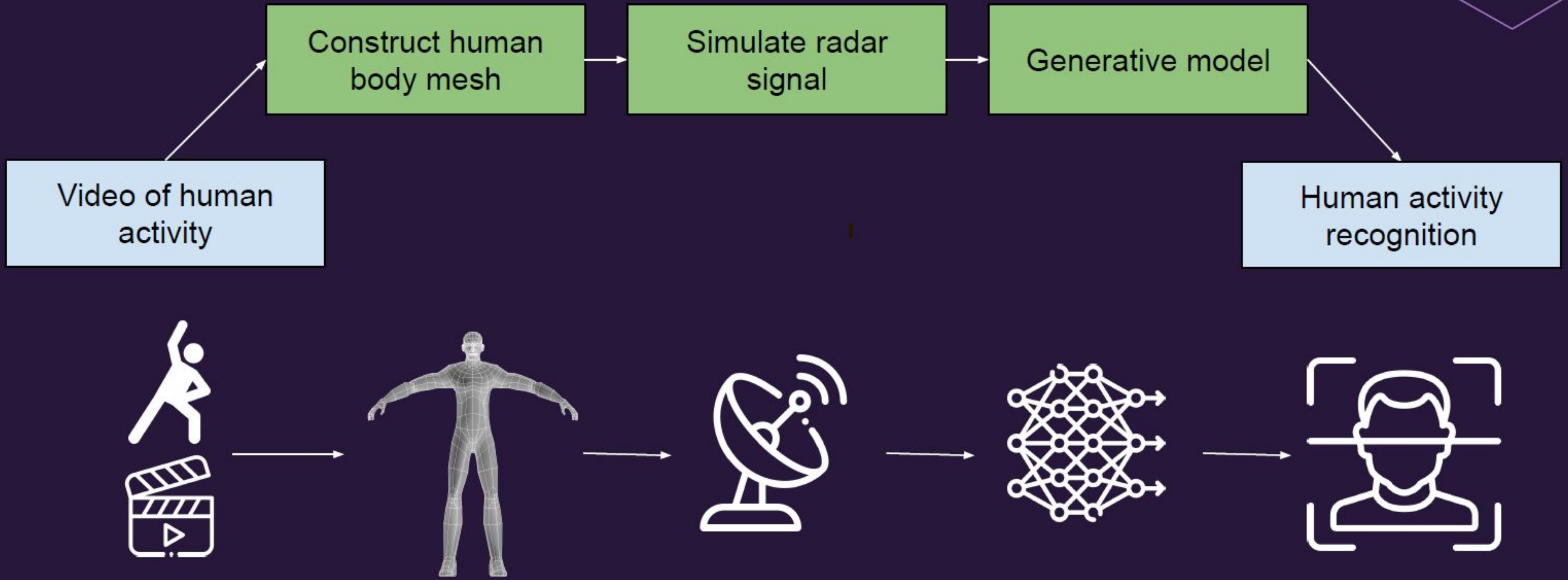
Final Design Review

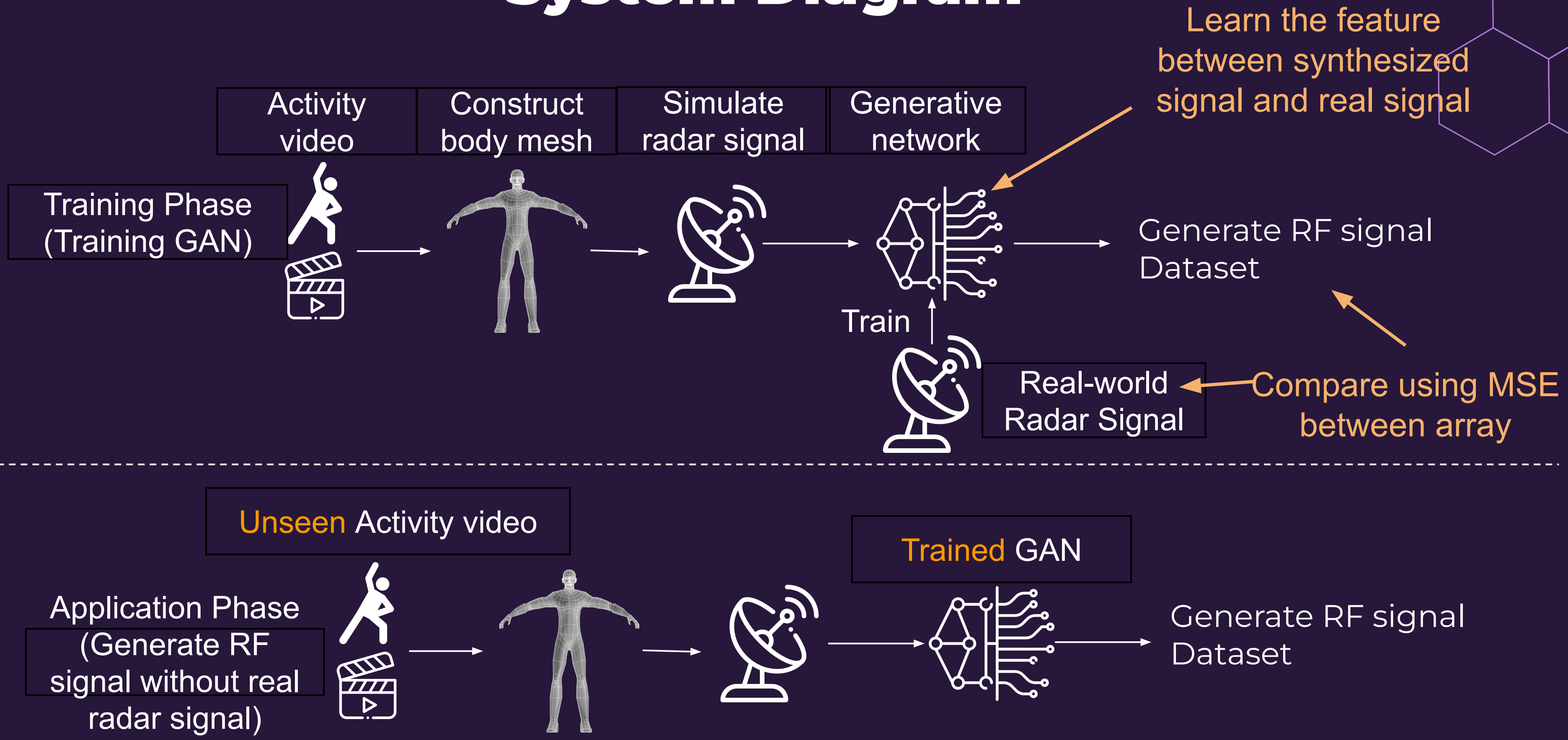
Video to RF Team

A. Overview of Proposed Prototype

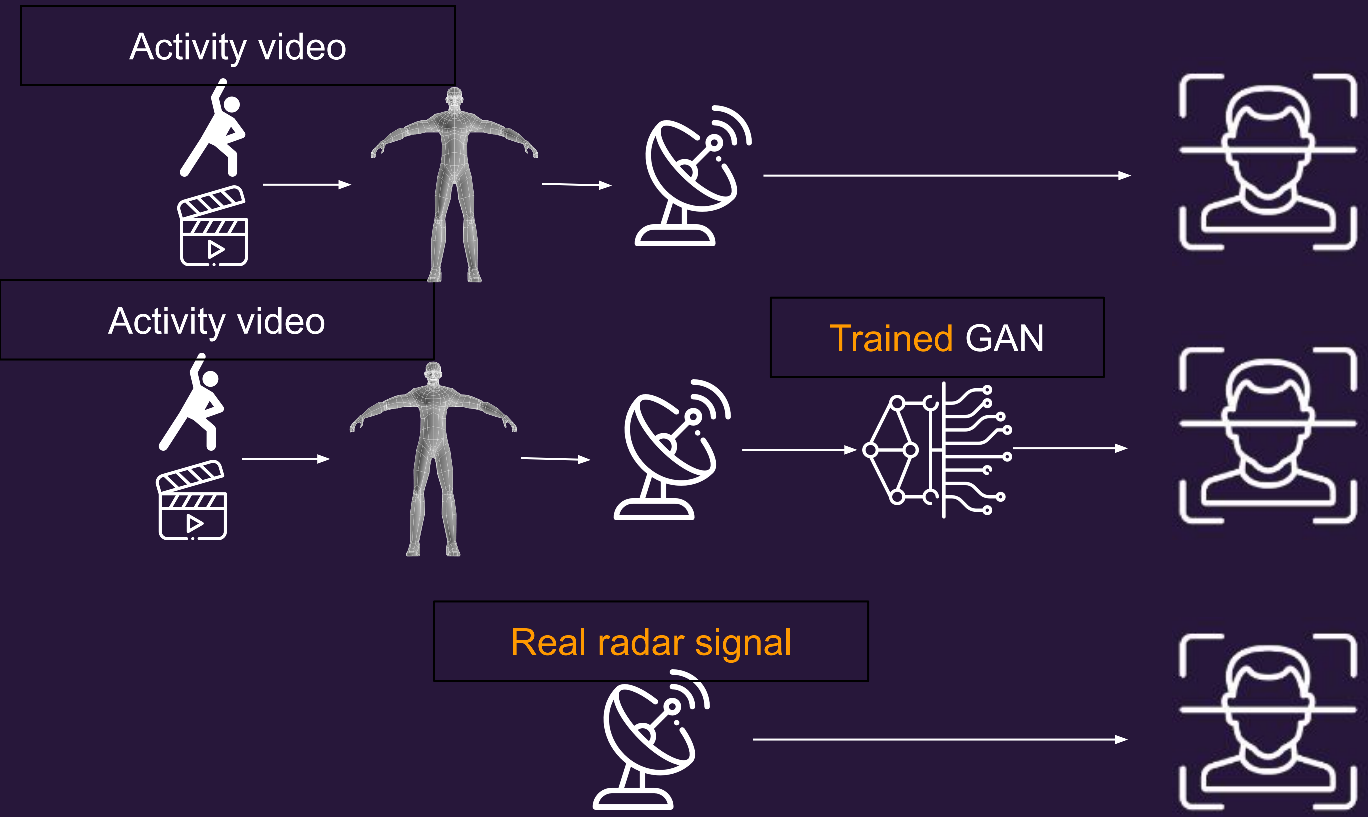
Brief overview of your PROJECT focus



System Diagram



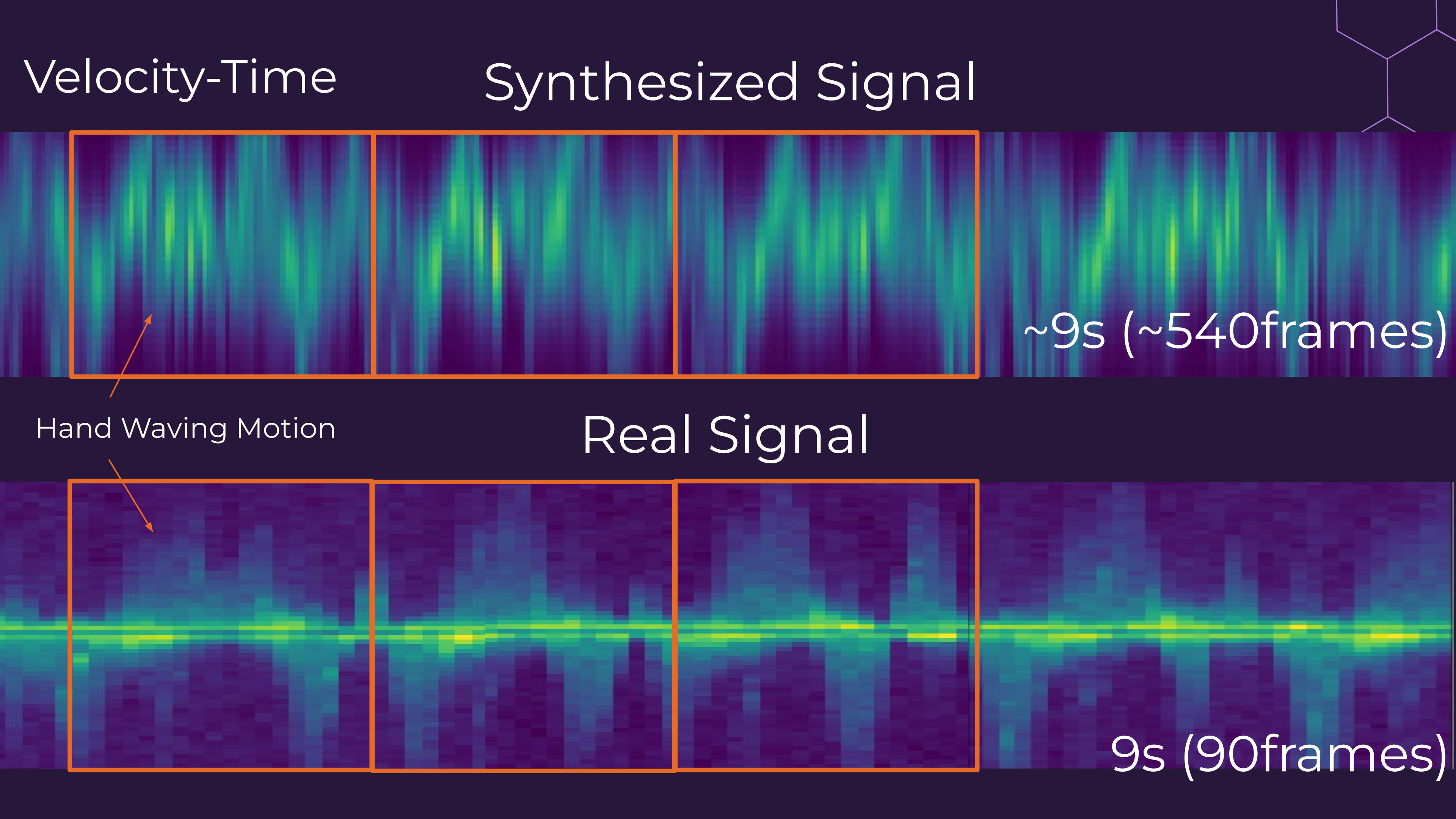
Additional task-HAR



Train the same HAR network using data from 3 different resource

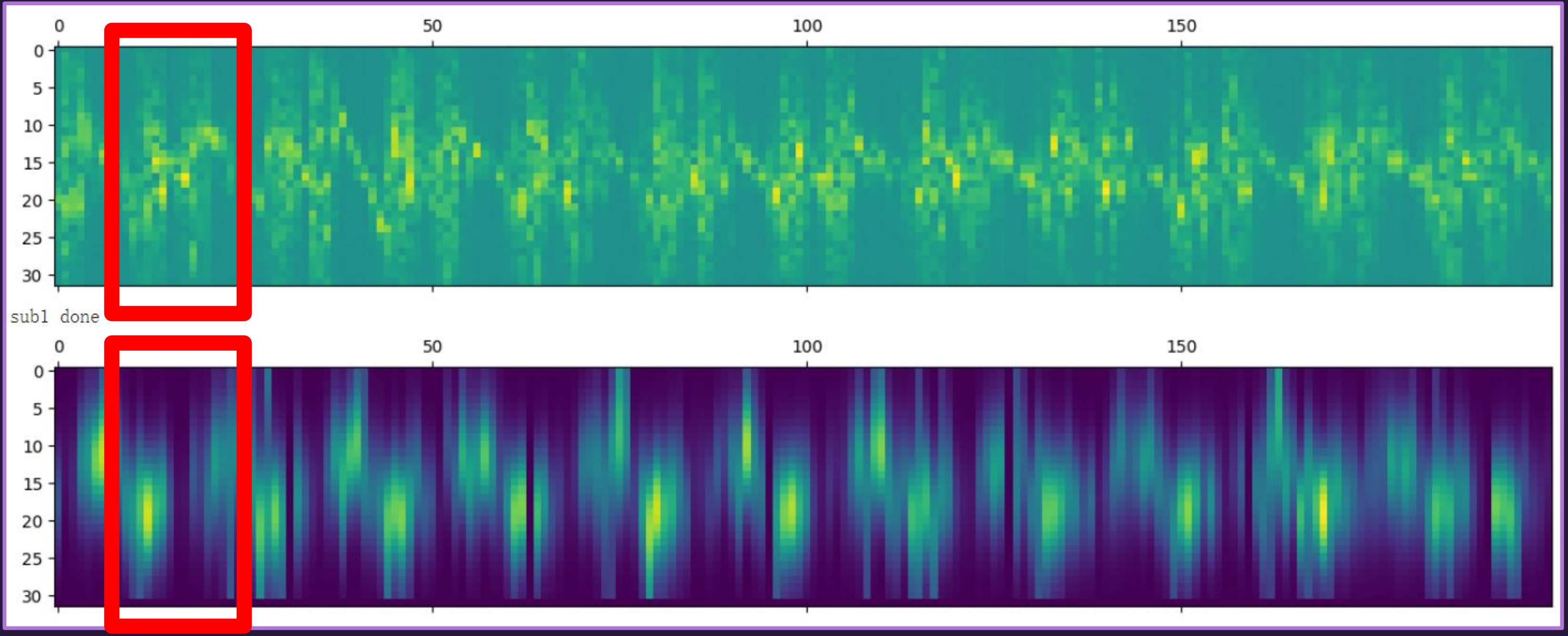
B and C. Technical progress and Schematics

GAN for Signal Improvement


Velocity-Time
Synthesized Signal
~9s (~540frames)
Hand Waving Motion
Real Signal
9s (90frames)

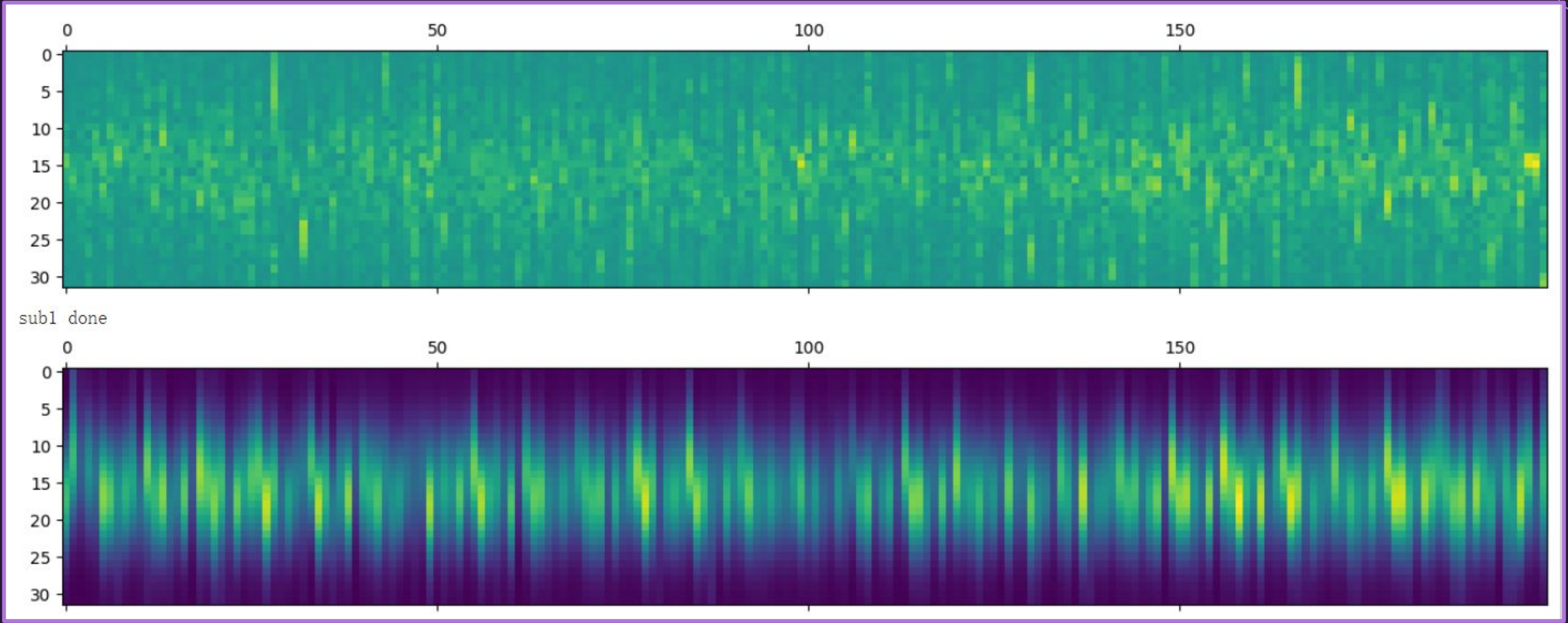
Real Radar Signal

Squat:



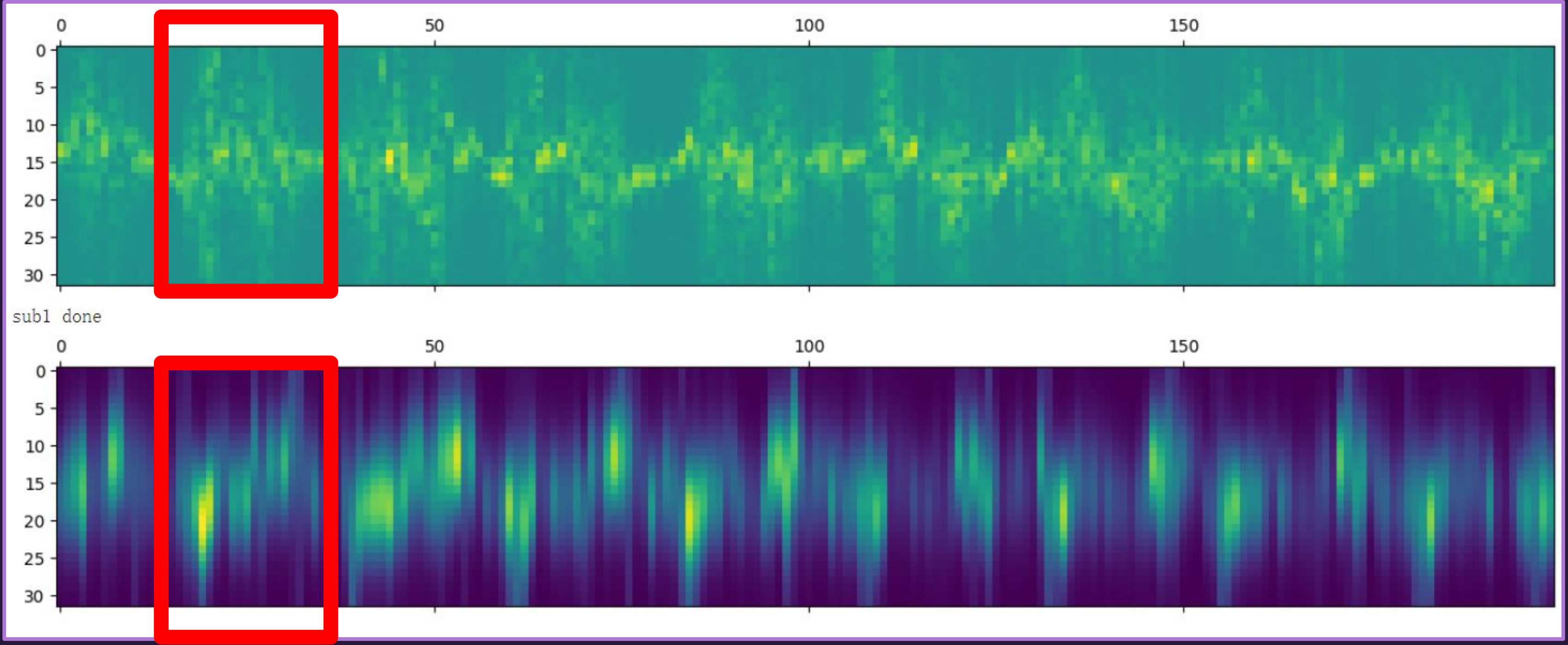
Real Radar Signal

Running:



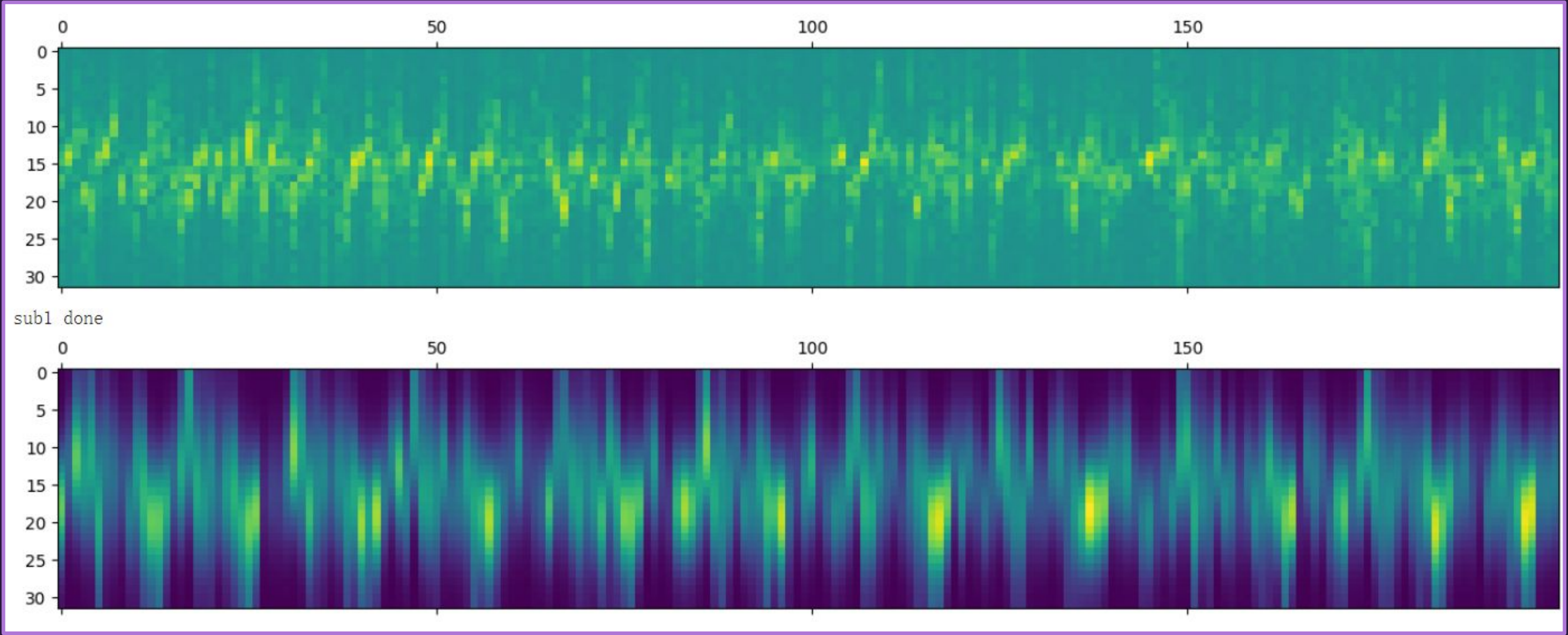
Real Radar Signal

Lunging:



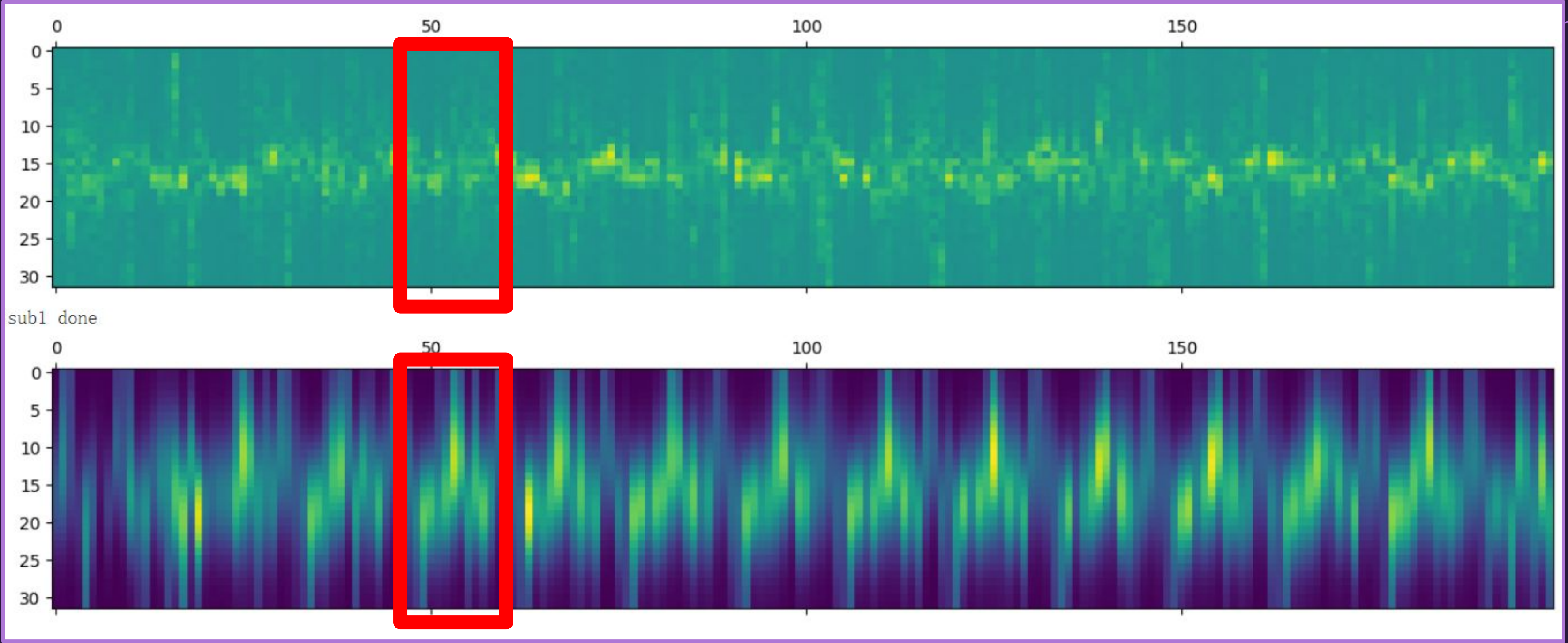
Real Radar Signal

Jumping Jack:



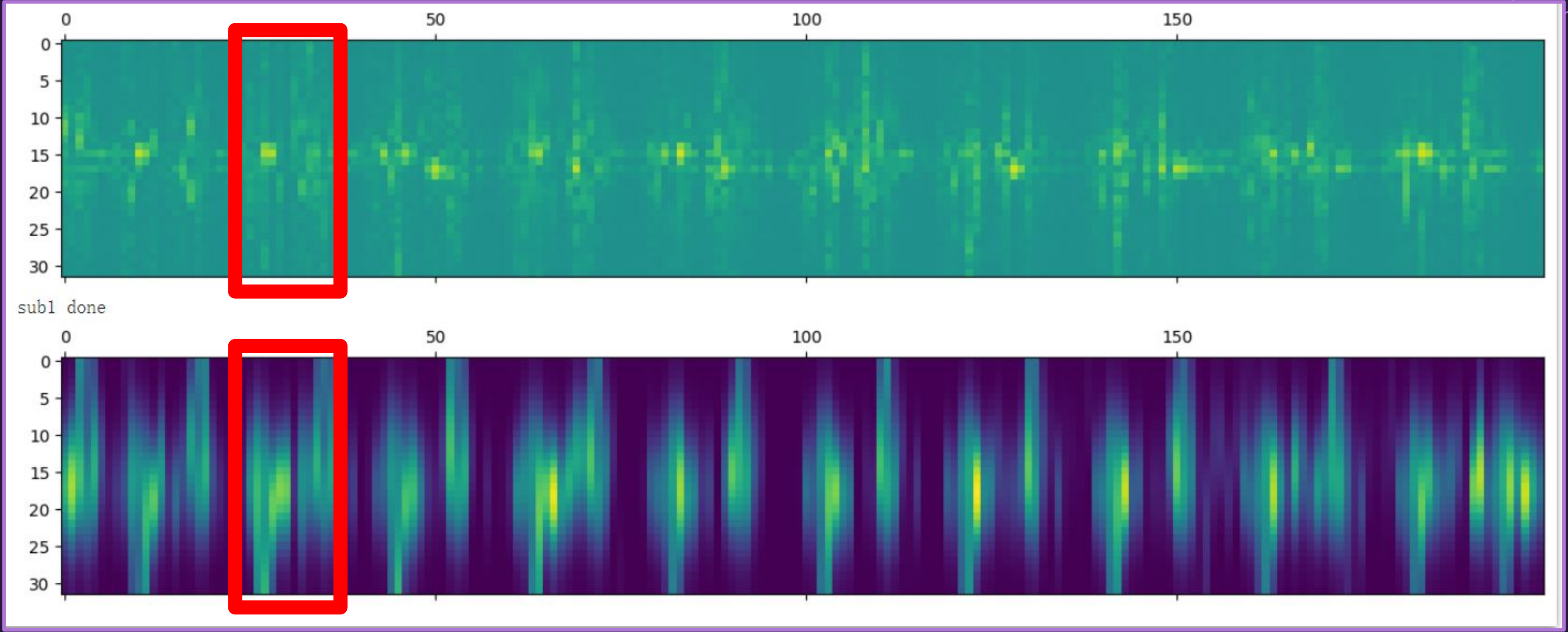
Real Radar Signal

Hand Swing:



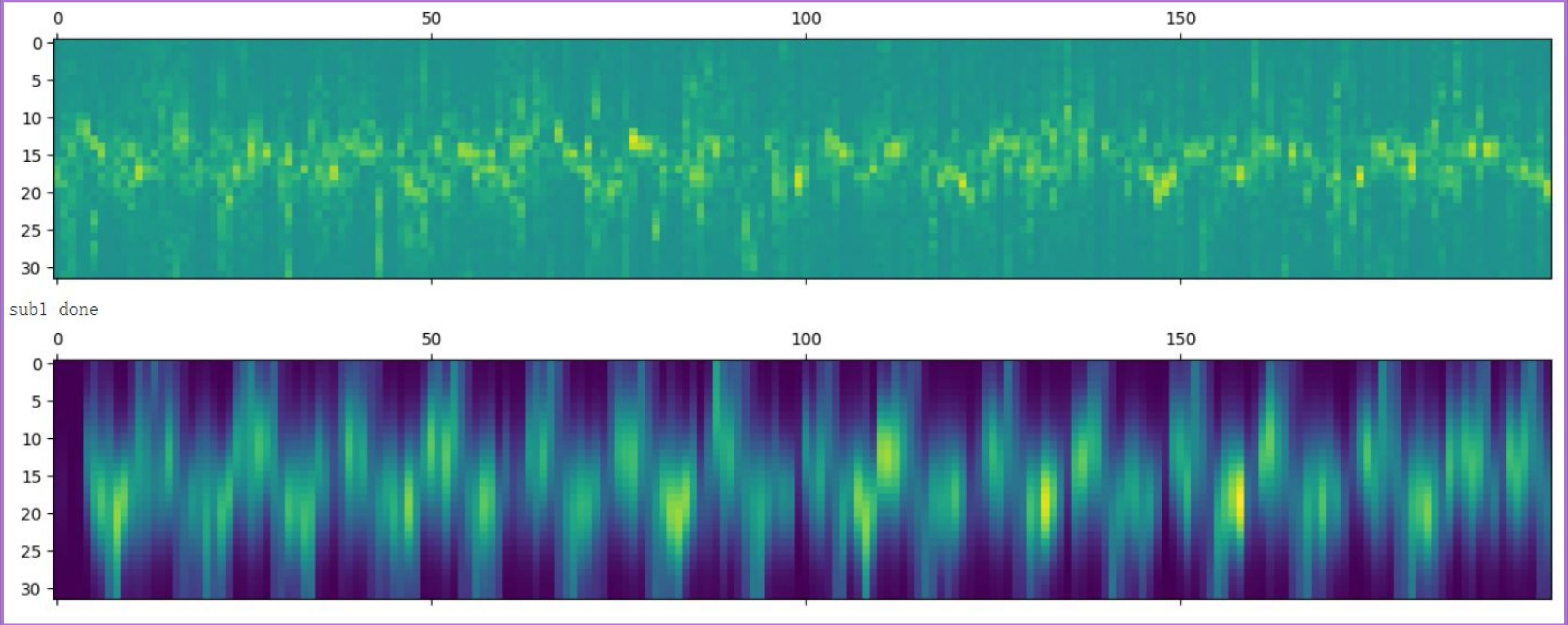
Real Radar Signal

Hand Clipping:

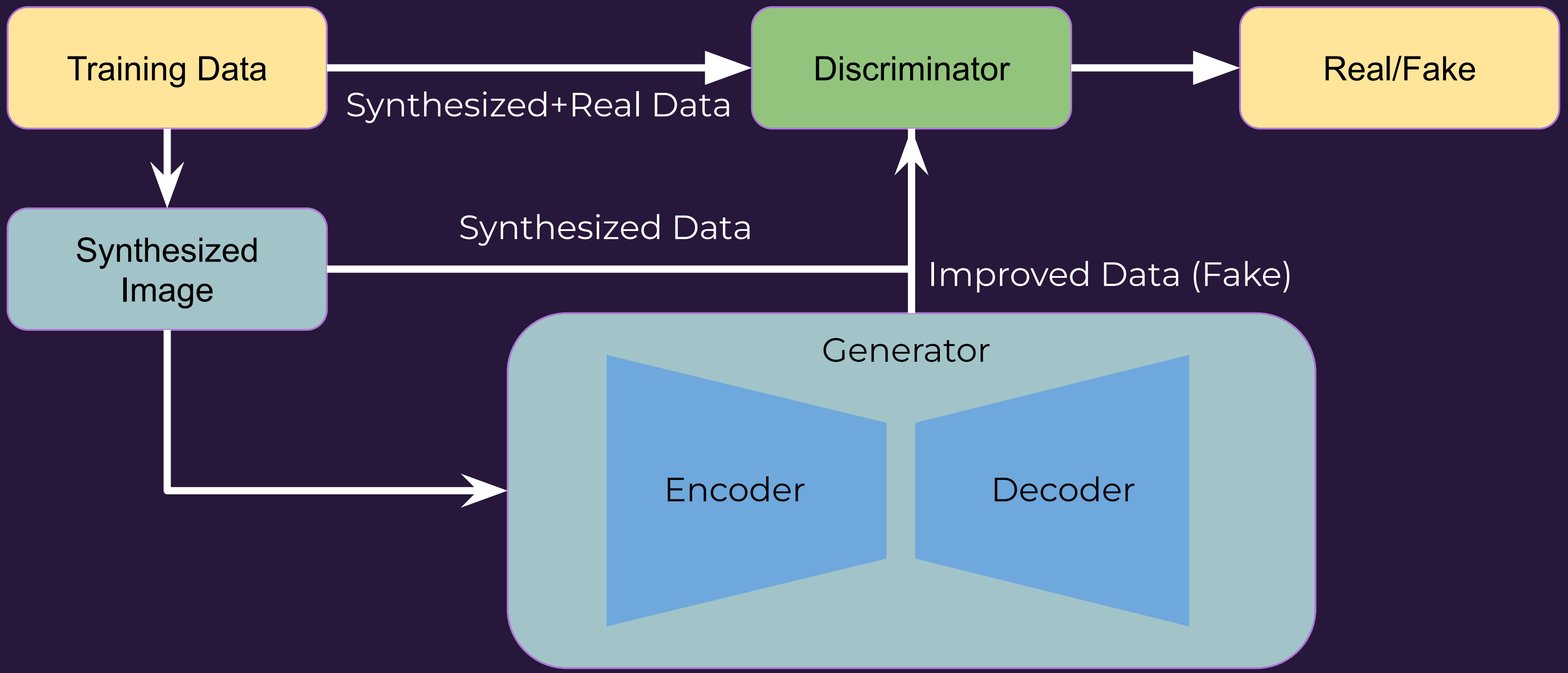


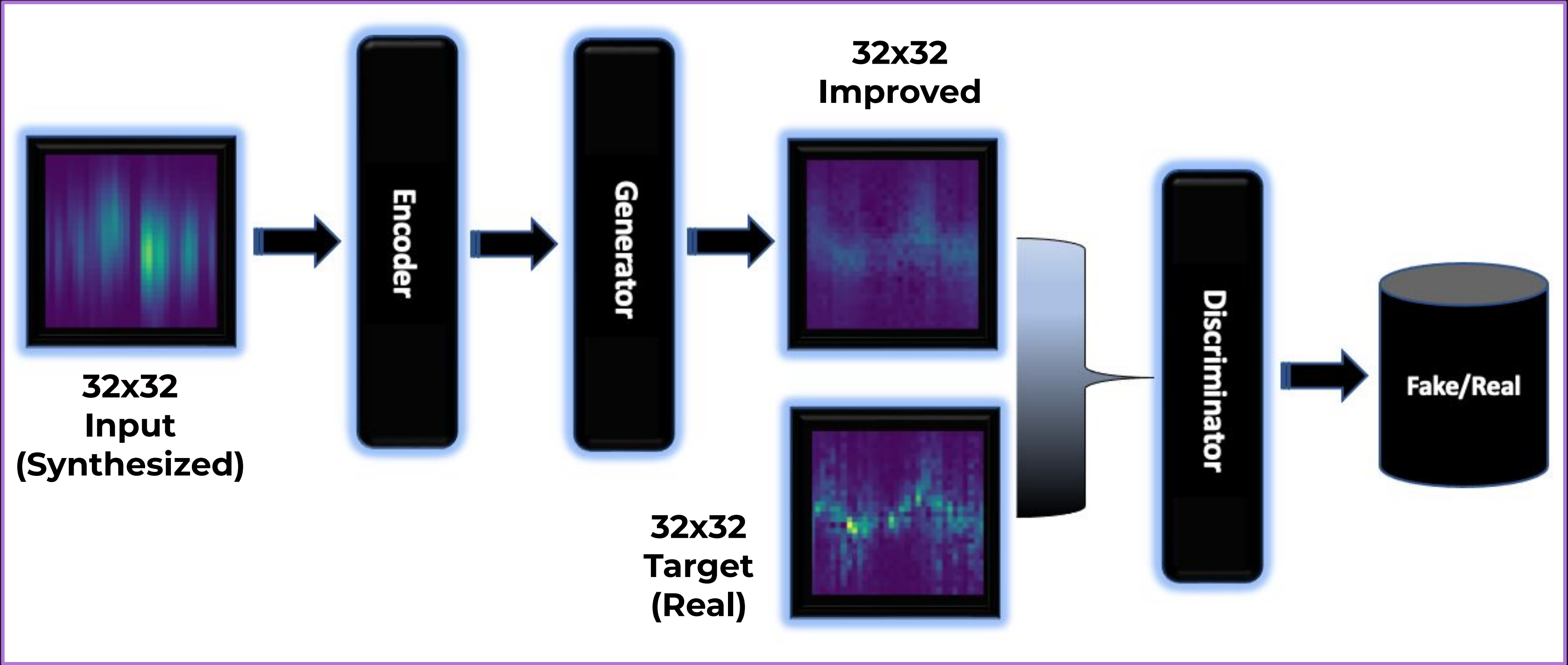
Real Radar Signal

Torso:



Generative Adversarial Network




32x32
Input
(Synthesized)
Encoder
Generator
32x32
Improved
32x32
Target
(Real)
Discriminator
Fake/Real

Generator Design

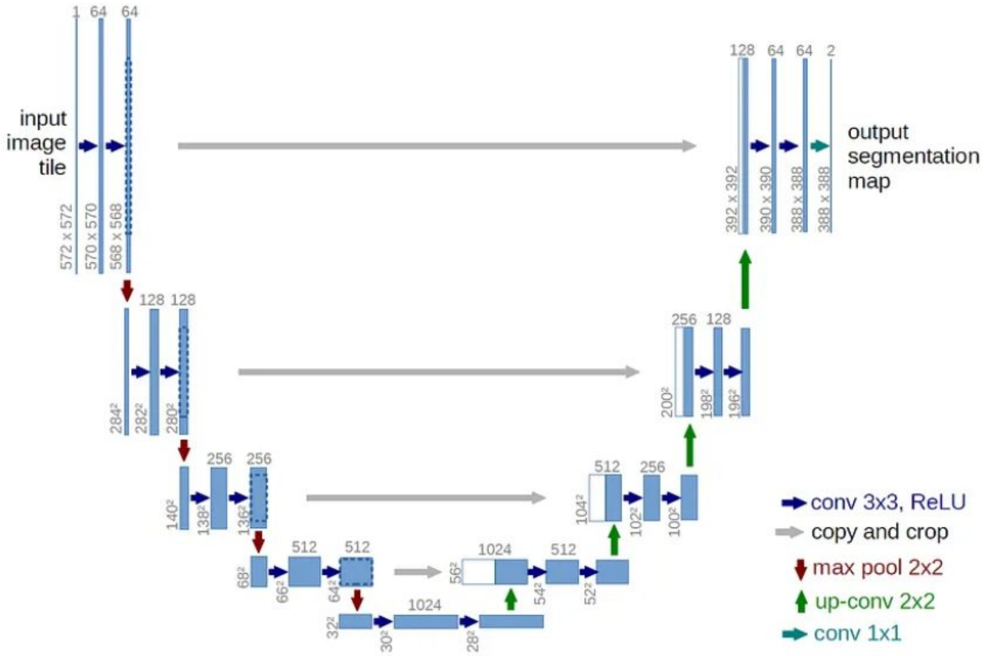
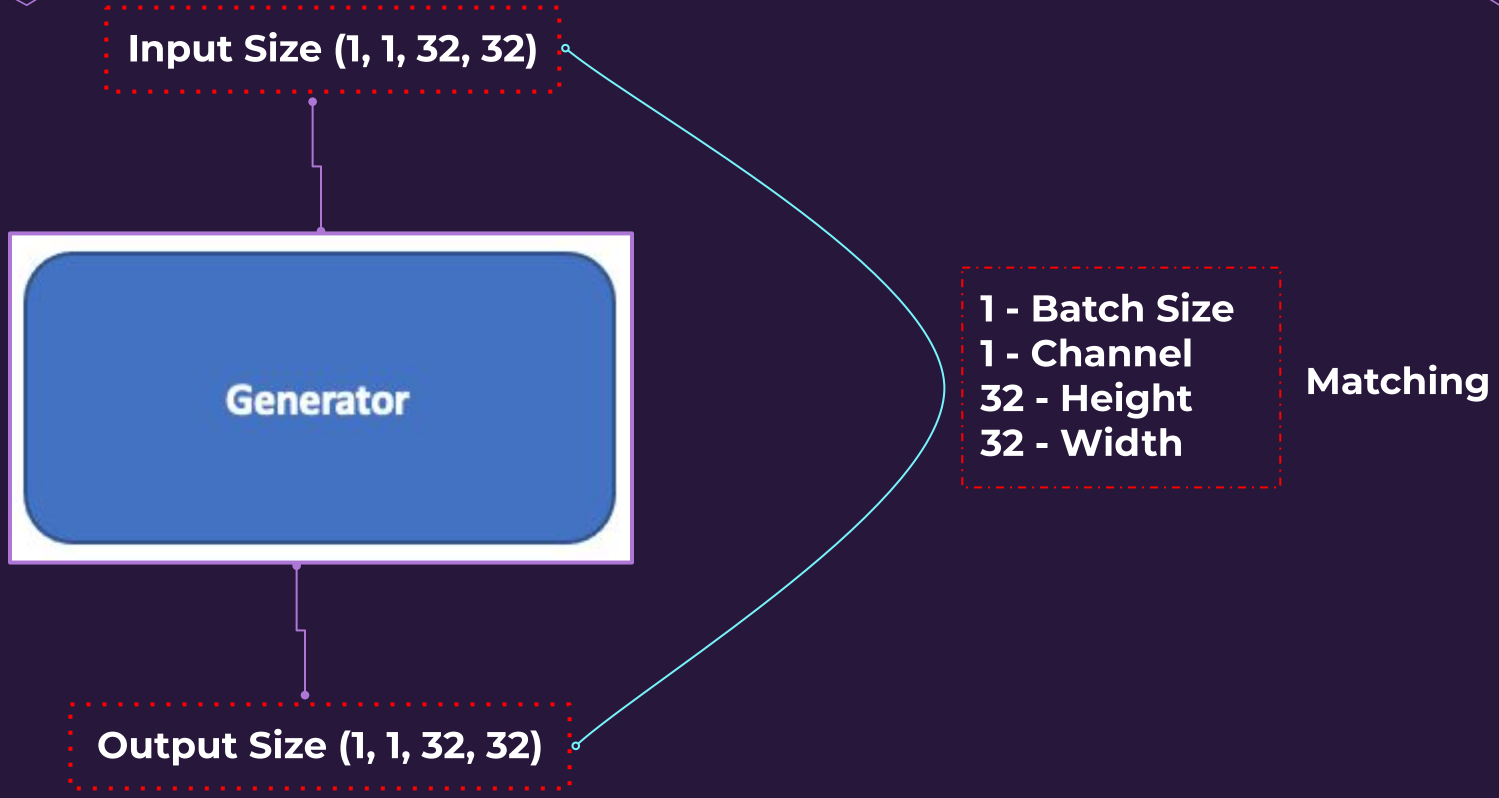
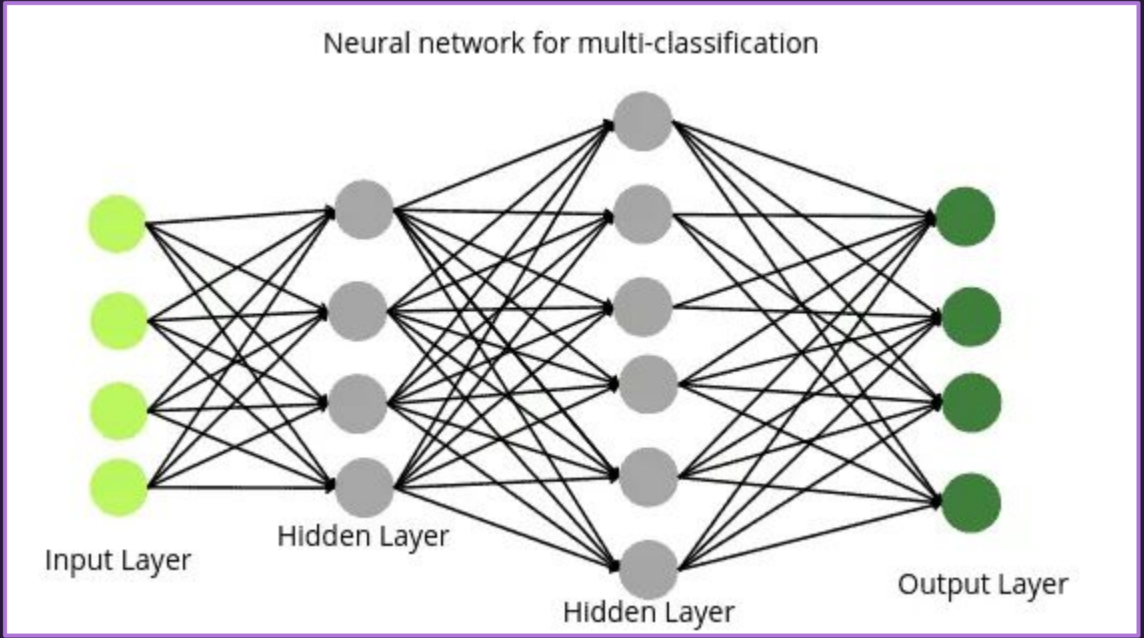


Fig — 4: Example of a Basic U-Net Architecture (Ref:- https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png)

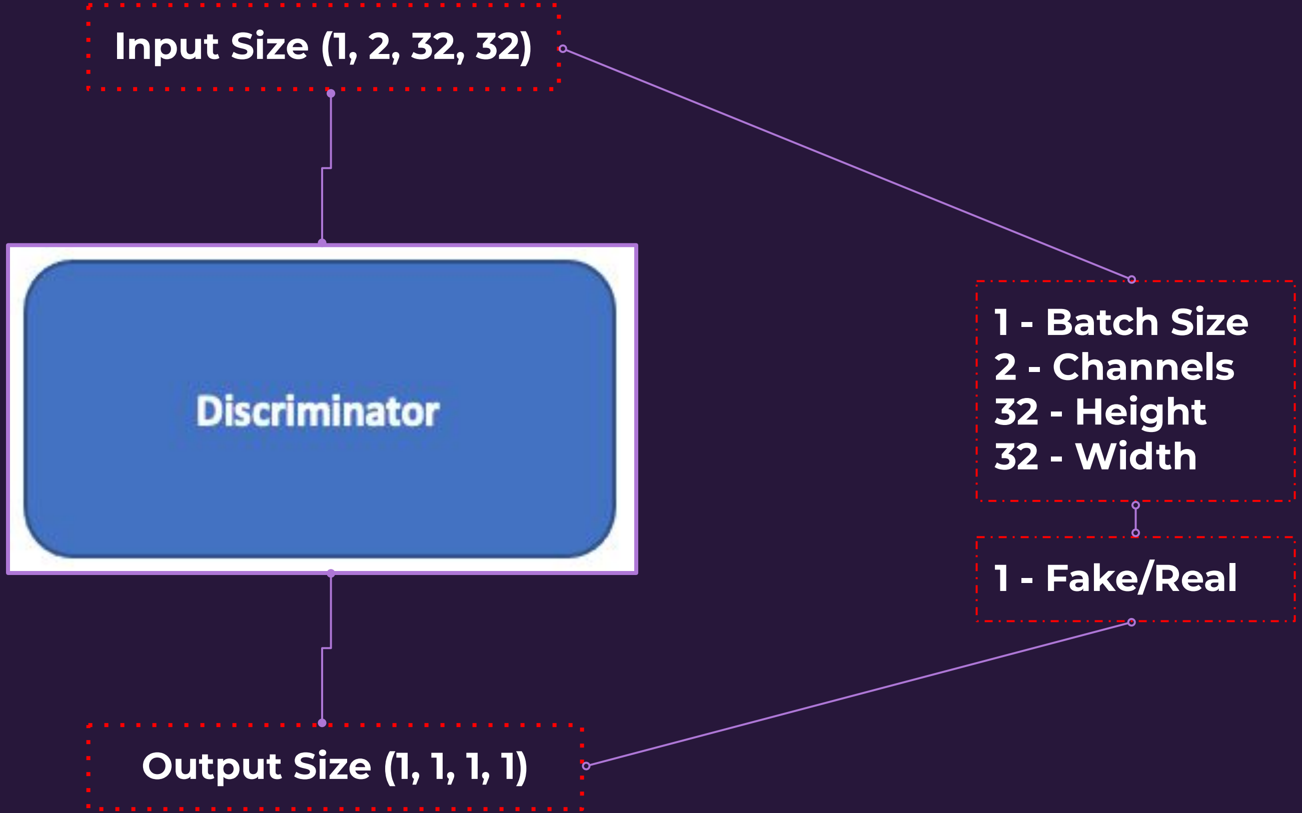
Input_size: (1,1,32,32)
Downsize layer: 5
Upsize layer: 5
# of feature: 4
kernel size: 4
stride: 2
Dropout: 0.5
activation: Leaky relu
Output: (1,1,32,32)


Input Size (1, 1, 32, 32)
Generator
Output Size (1, 1, 32, 32)
1 - Batch Size
1 - Channel
32 - Height
32 - Width
Matching

Discriminator Design



Input_size = (1,2,32,32)
Hidden layer: 4
Layer type: Cov2d
Activation: Leaky Relu
Output: (1,1,1,1)


Input Size (1, 2, 32, 32)
Discriminator
Output Size (1, 1, 1, 1)
1 - Batch Size
2 - Channels
32 - Height
32 - Width
1 - Fake/Real

Dataset and Training Configuration

Training dataset: (5 motion)

- Total data length: 414 (cuts)
- Training and validation data split : (400,14)

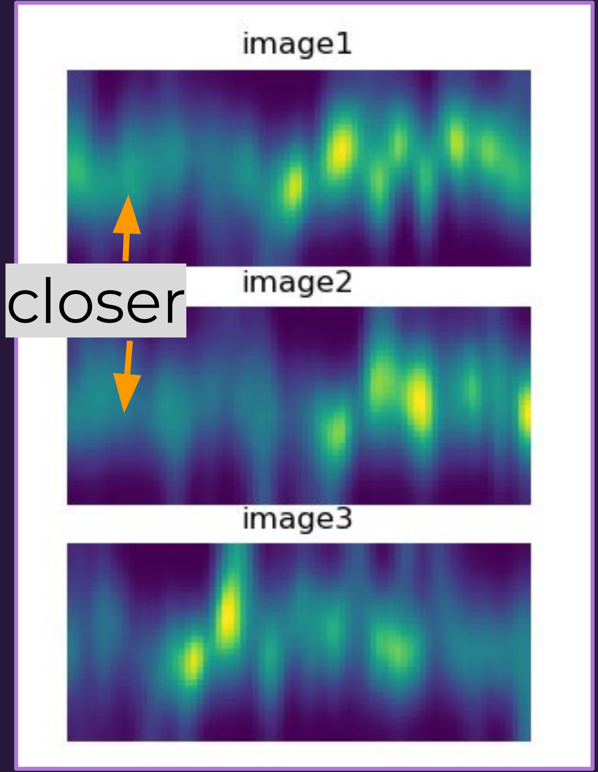
Testing dataset: (2 unseen motion)

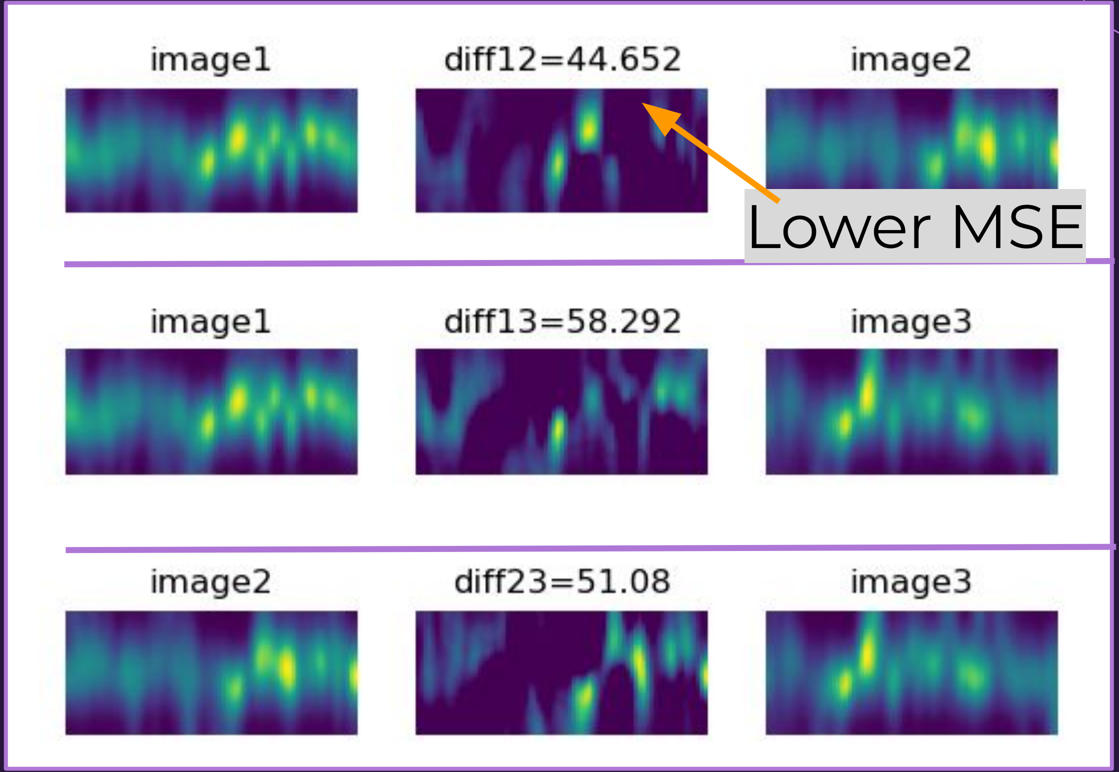
- Total data length: 36 (cuts)

Training config.

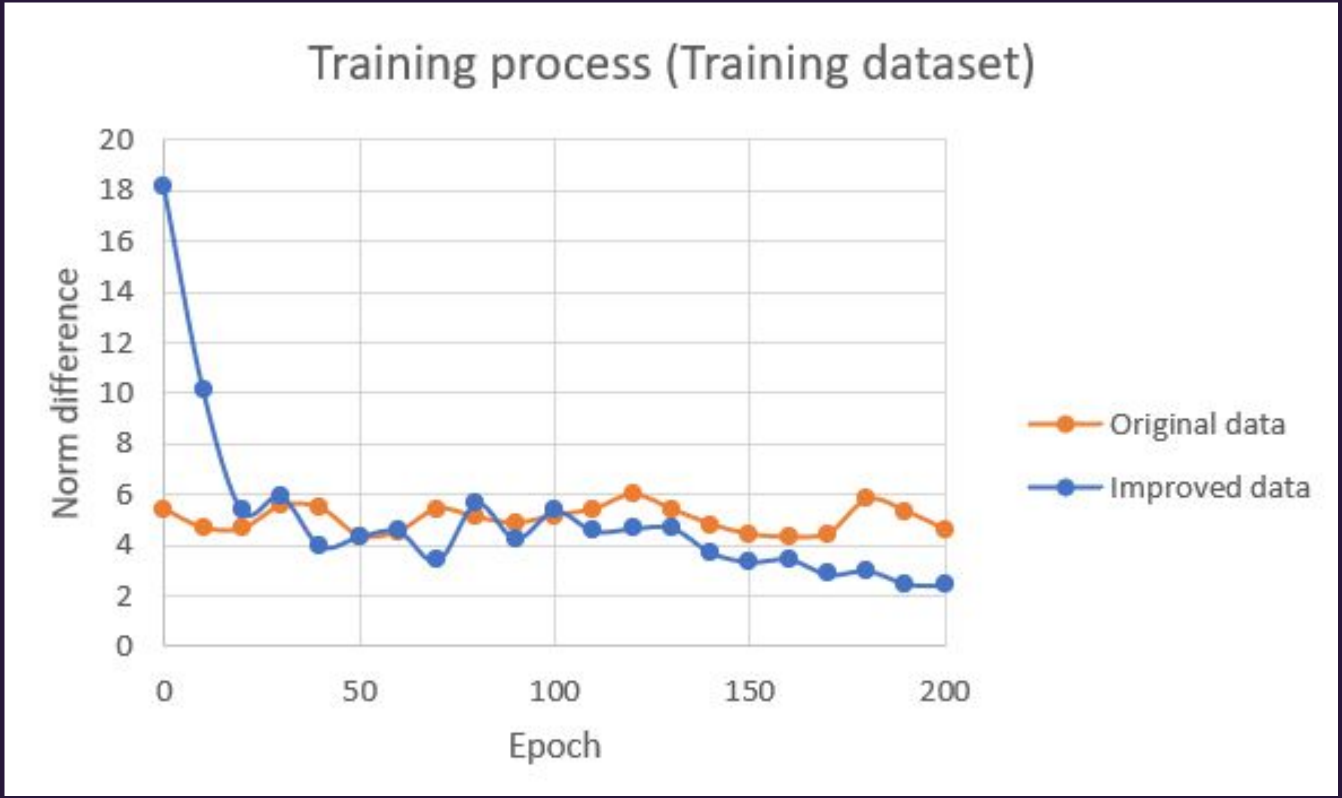
- Batch size: 16
- Epoch: 200
- Platform: Local CPU

Metrics-MSE (L2 Norm)

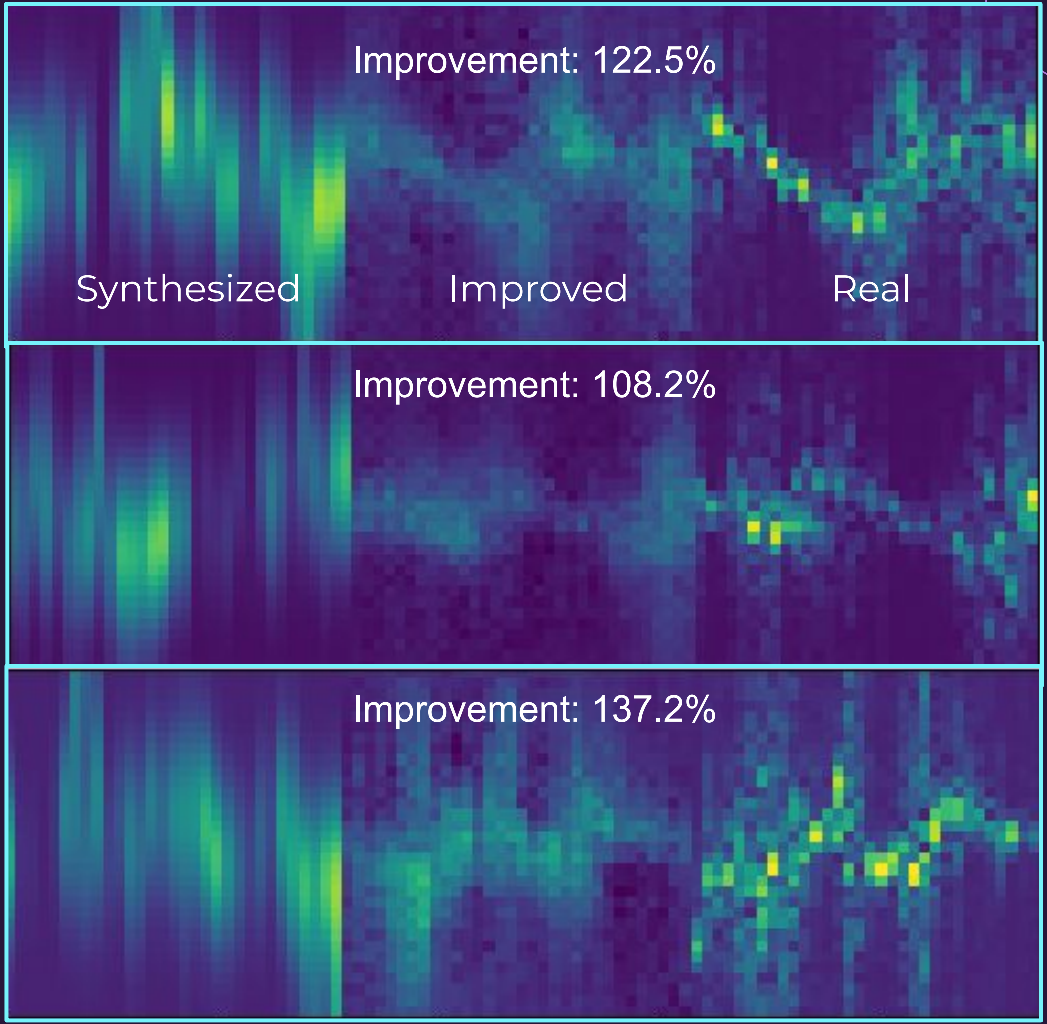




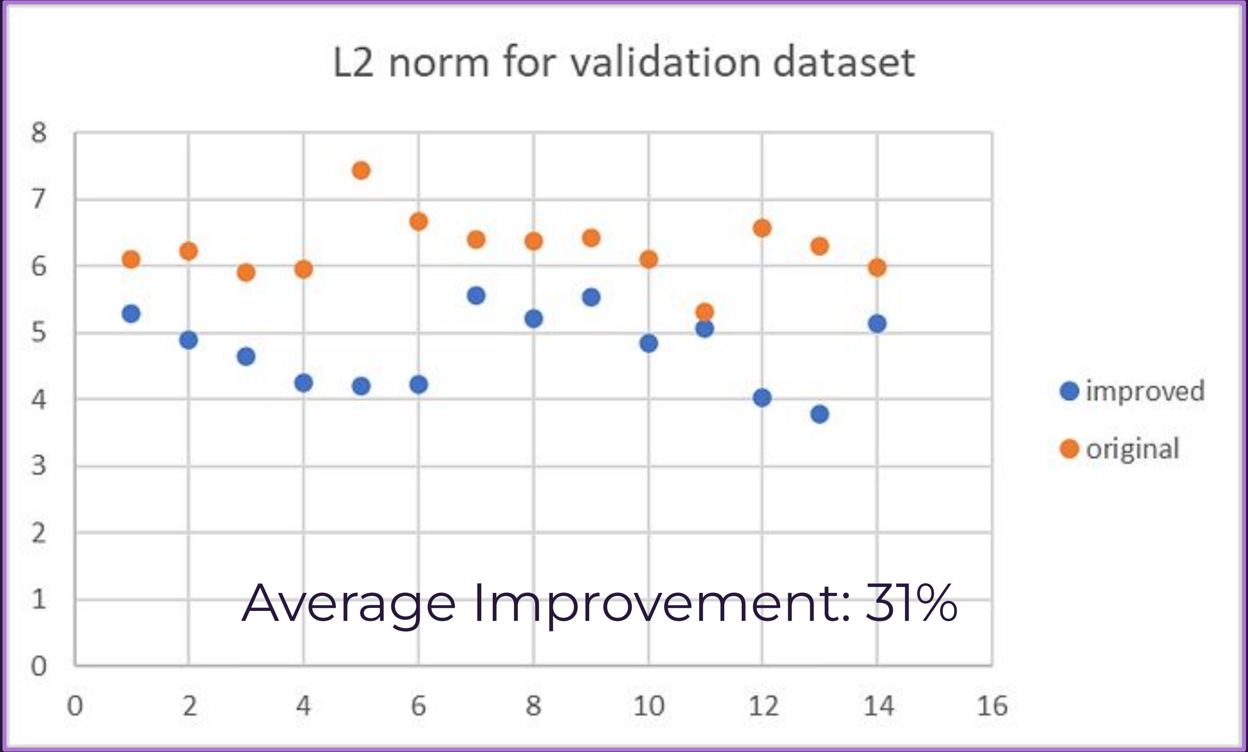
Training Result



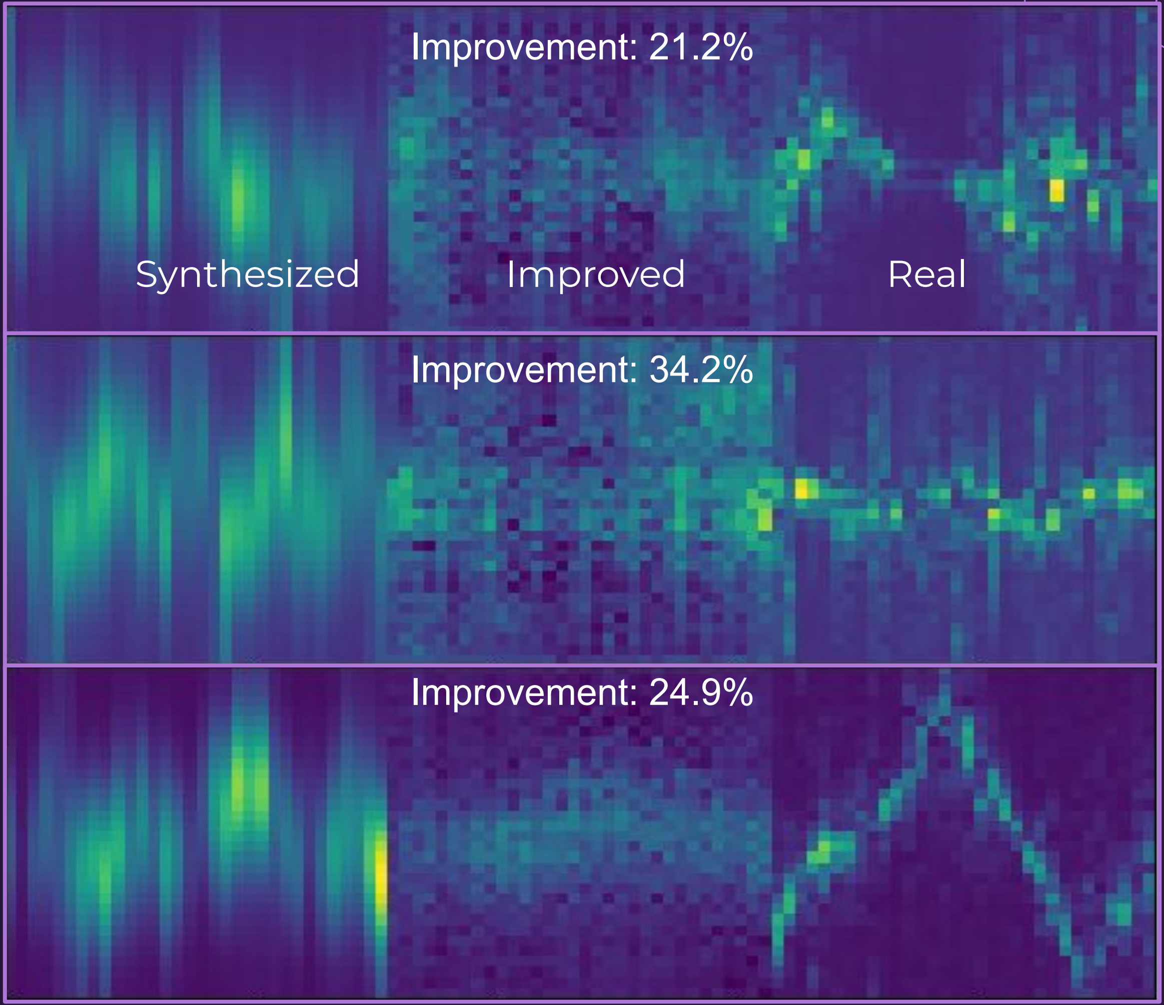
Training Data Result



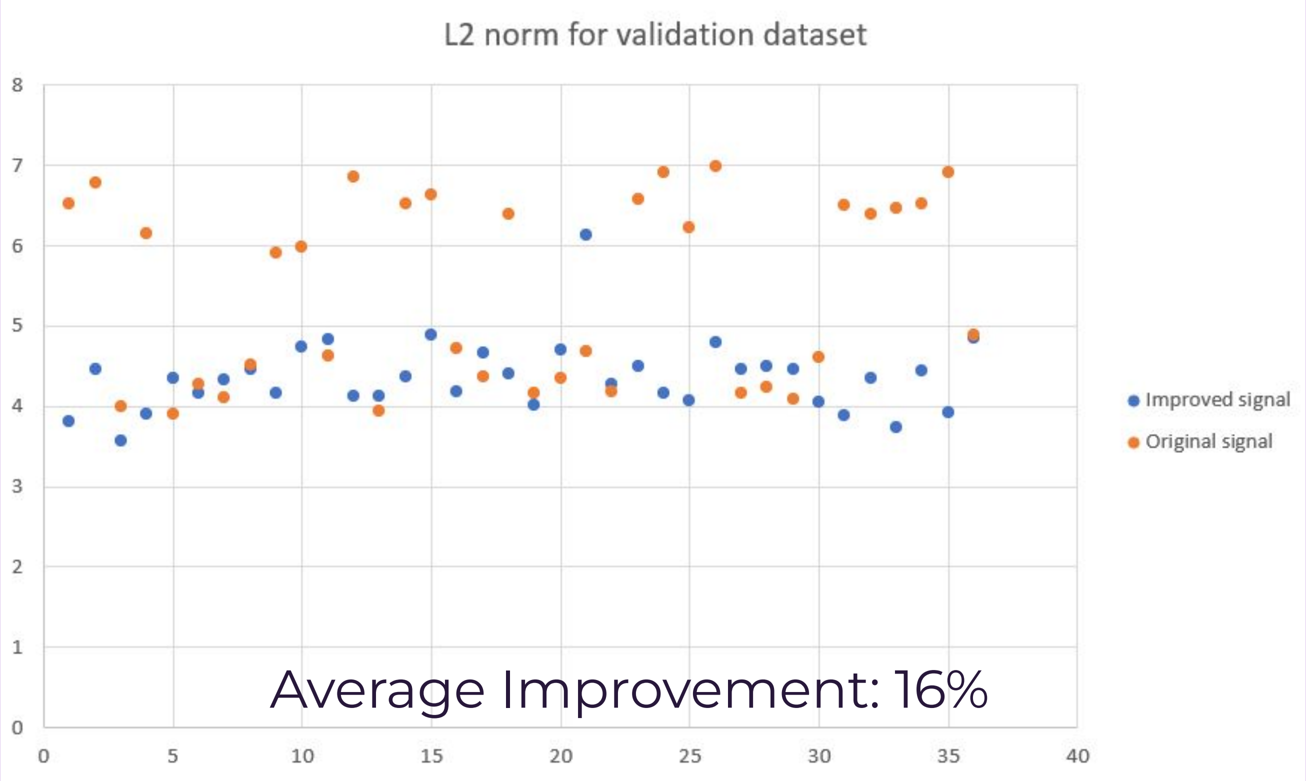
Validation Result



Validation Data Result

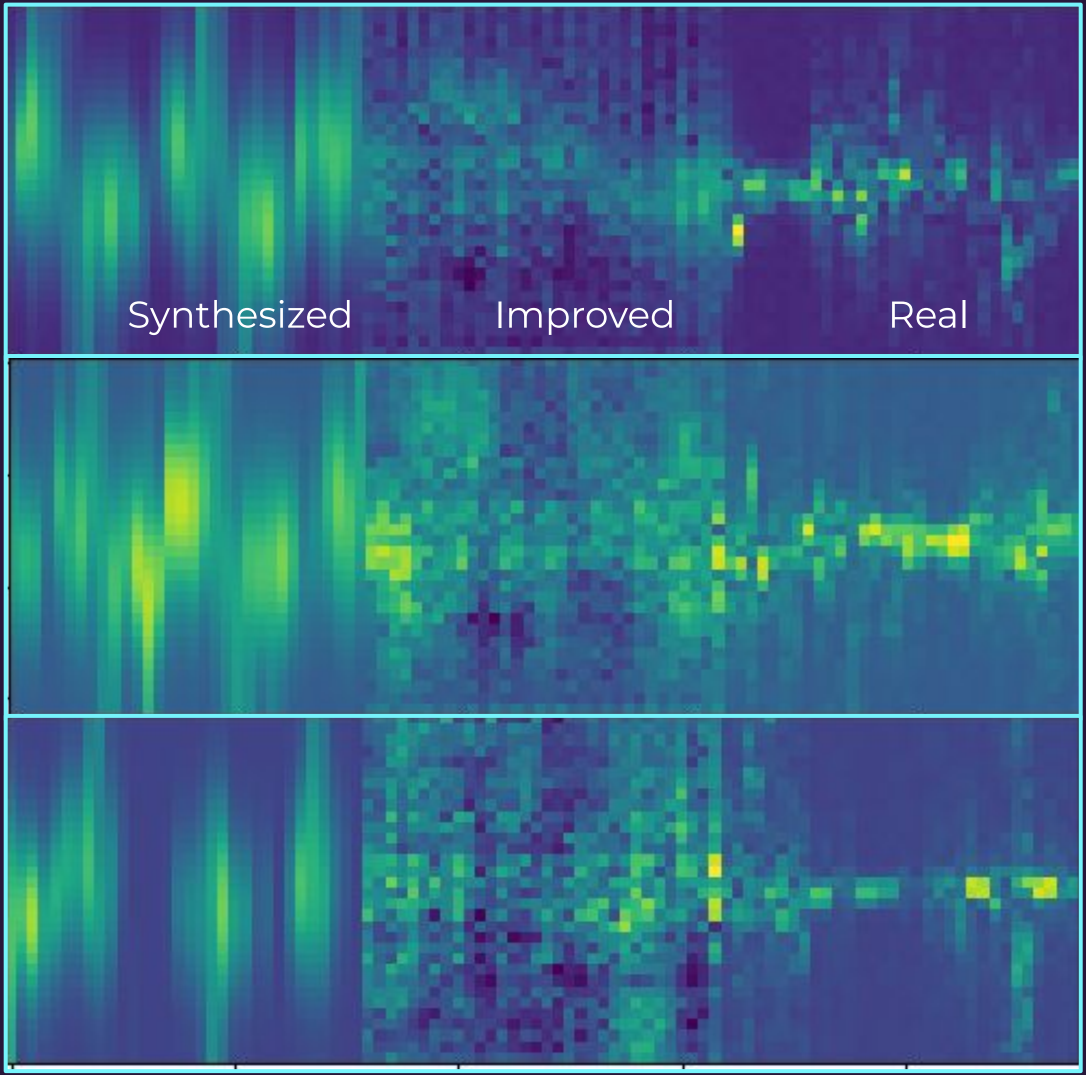


Testing Result



Average Improvement: 16%

Testing Data Result



Conclusion

- We proved GAN can be used for improving signal quality synthesized from video, which make it possible to take advantage of huge activity video dataset online
- For the unseen videos, the quality improvement is not as clear, this can due to lack of training data

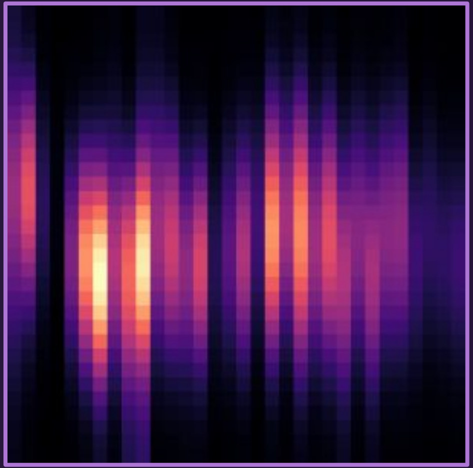
Future Work

- Adapting data agumentation method to increase the training dataset
- Implement time alignment algorithm to avoid time difference between two source
- Test different generative network architecture
- Including “range” dimension
- Implementing HAR network on unseen activity

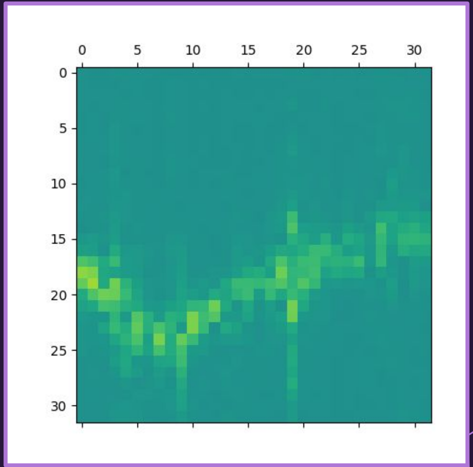
Comparison the Accuracy between trained CNN model with Sythesized Data,

Real Radar Signal and GAN model

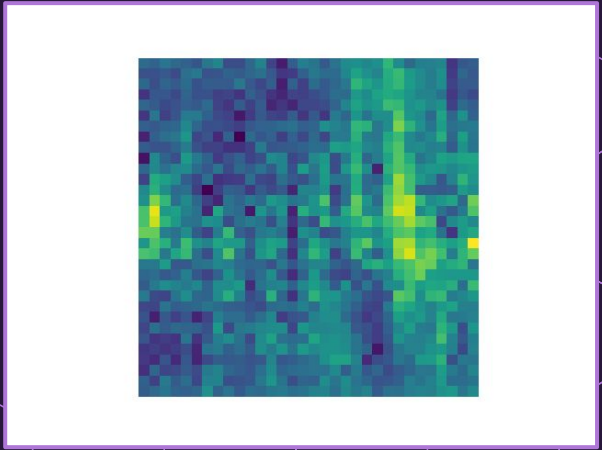
Sythesized Data:



Real Radar Data:



GAN Model Data:



Using the same CNN architecture layers to train these three types of Data Sources:

```python
class CNN_Model(nn.Module):
    #列出需要哪些層
    def __init__(self):
        super(CNN_Model, self).__init__()
        # Convolution 1 , input_shape=(3,224,224)
        self.cnn = nn.Sequential(
            nn.Conv2d(3, 64, 3, 1, 1),  # [64, 128, 128]
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2, 2, 0),  # [64, 64, 64]

            nn.Conv2d(64, 128, 3, 1, 1),  # [128, 64, 64]
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, 2, 0),  # [128, 32, 32]
```

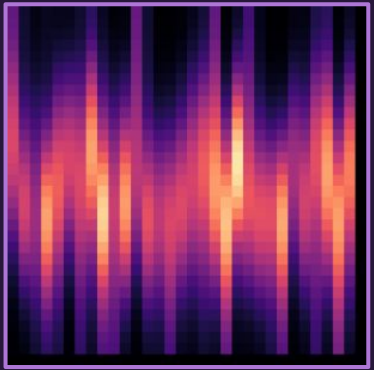
Using the same CNN architecture layers to train these three types of Data Sources:

```
        )
        self.fc = nn.Sequential(
            nn.Linear(512 * 4 * 4, 1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Linear(512, 11)
        )
```

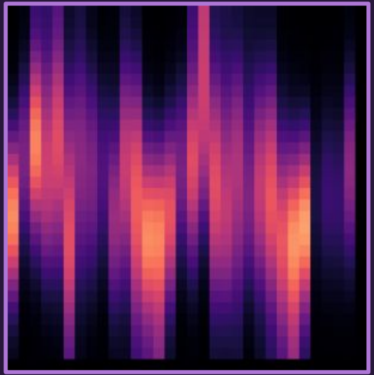
Training Result (using Sythesized Data)

Total five Actions to Recognize:

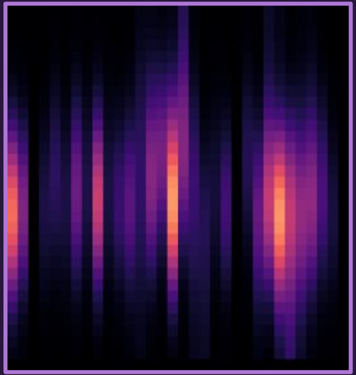
Handmotion:



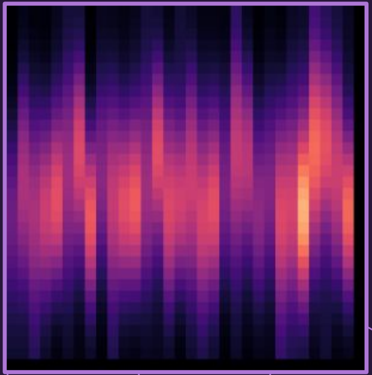
JJ:



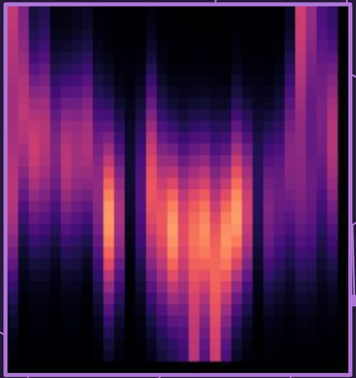
Lunge:



Run:



Squats:



Labeling:

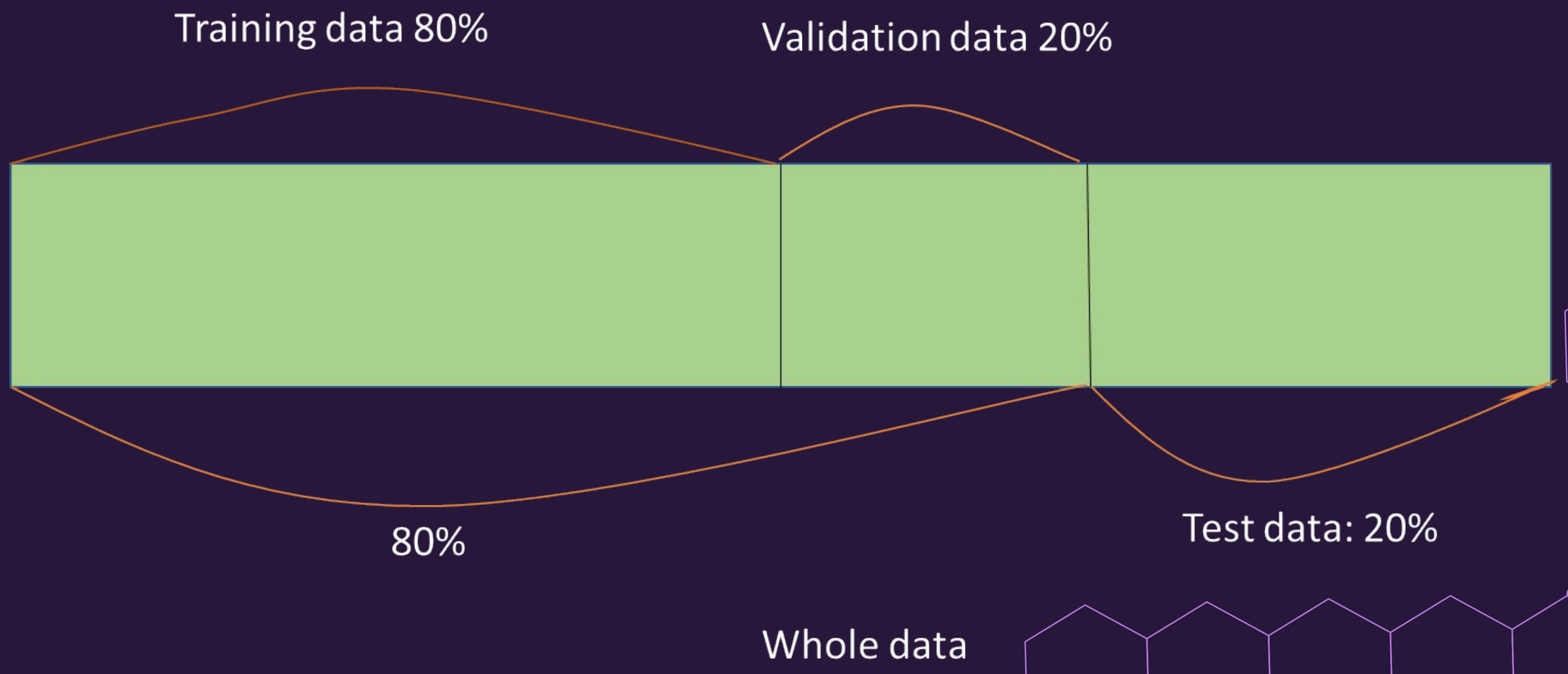
Handmotion: 0 , JJ : 1 , Lunge : 2 , Run : 3, Squats : 4

```
D:\anaconda\envs\pytorch1\python.exe "D:\pycharm project\main6.py"
{'Handmotion': 0, 'JJ': 1, 'Lunge': 2, 'Run': 3, 'Squats': 4}
 10%|█         | 1/10 [00:00<00:02,  3.19it/s]tensor([4, 1, 1, 1, 4, 2, 4, 2])
tensor([1, 2, 1, 4, 1, 3, 2, 0])
 30%|███       | 3/10 [00:00<00:01,  4.54it/s]tensor([4, 4, 4, 3, 2, 1, 0, 4])
tensor([1, 3, 0, 3, 4, 3, 3, 2])
 50%|█████     | 5/10 [00:01<00:00,  5.17it/s]tensor([1, 0, 1, 0, 1, 0, 4, 3])
tensor([4, 0, 1, 2, 0, 0, 3, 2])
 70%|███████   | 7/10 [00:01<00:00,  5.30it/s]tensor([4, 0, 2, 4, 0, 0, 1, 2])
```

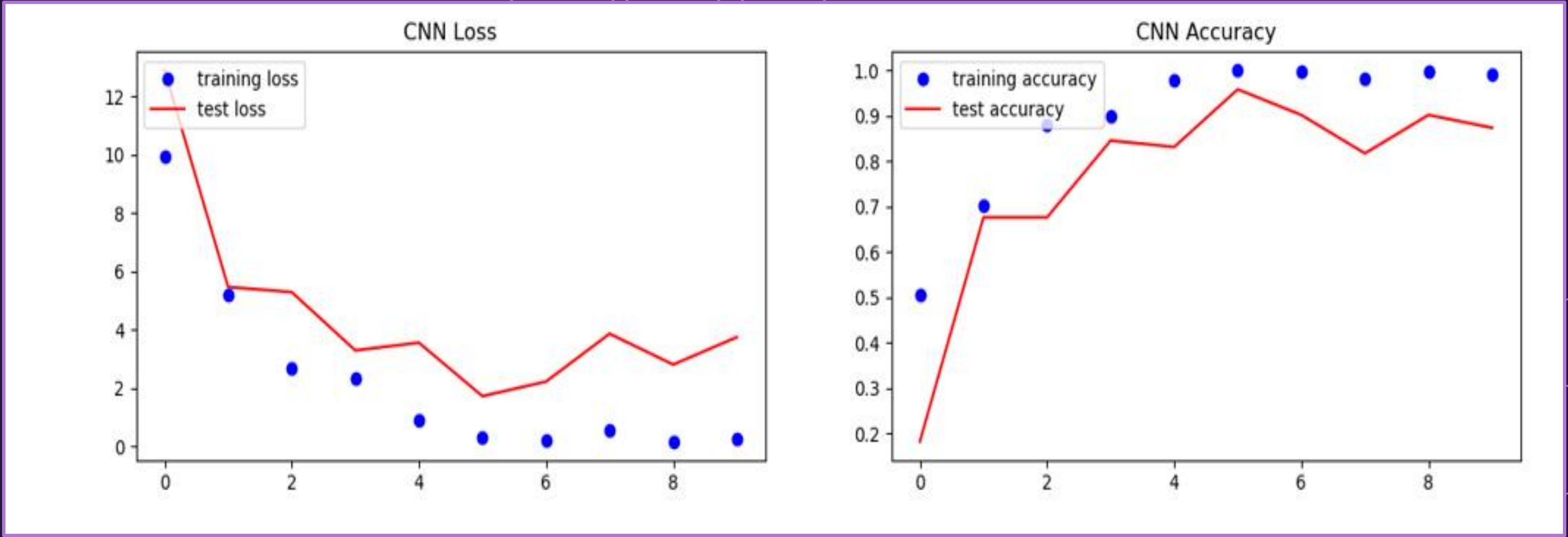
Training Result:

We trained the sythesized CNN model using 20% of the data as test data. For the remaining 80% data sets, we split 20% of those training data into validation data and use the remaining 80% for training.

Splitting the Data:



Training Result:



Testing Accuracy:

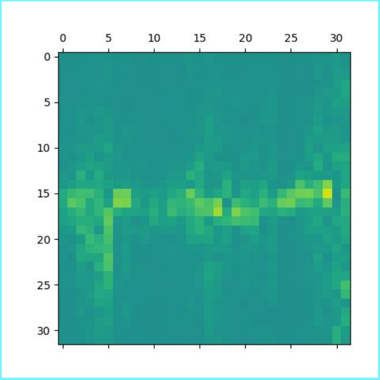
Testing accuracy about: 80.5 %

```
 78%|███████   | 7/9 [00:01<00:00,  5.20it/s]tensor([4, 0, 3, 3, 4, 0, 0, 3])
tensor([0, 2, 2, 1, 2, 2, 3, 0])
100%|██████████| 9/9 [00:01<00:00,  5.22it/s]
tensor([4, 1, 2, 3, 2, 3, 1, 3])
58
0.8055555555555556
[1, 4, 0, 4, 2, 0, 3, 0, 2, 2, 2, 2, 1, 0, 0, 2, 3, 0, 4, 3, 2, 4, 1, 4, 4, 3,
```

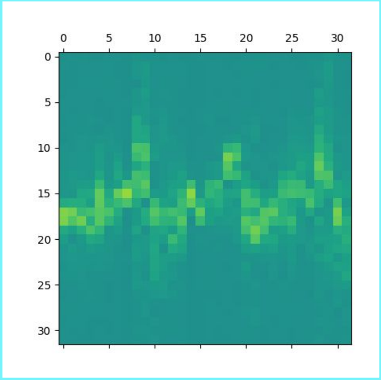
Training Result (using Real Radar Signal Data)

Total five Actions to Recognize:

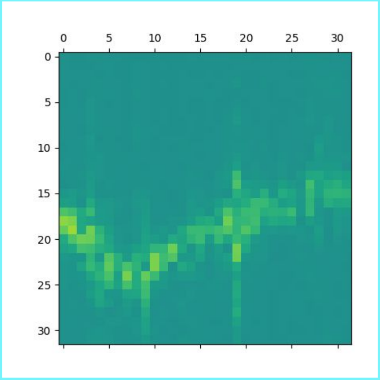
Handmotion:



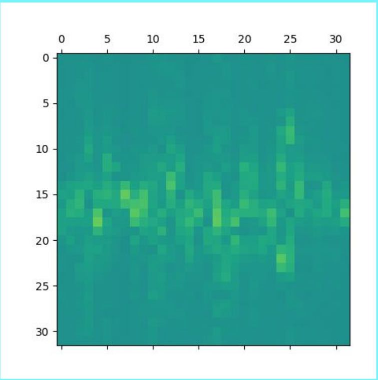
JJ:



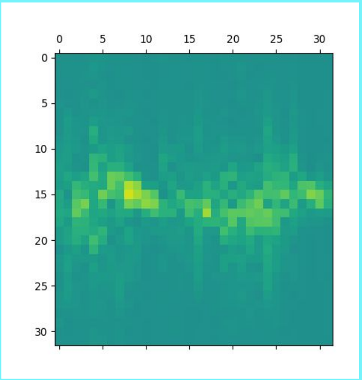
Lunge:



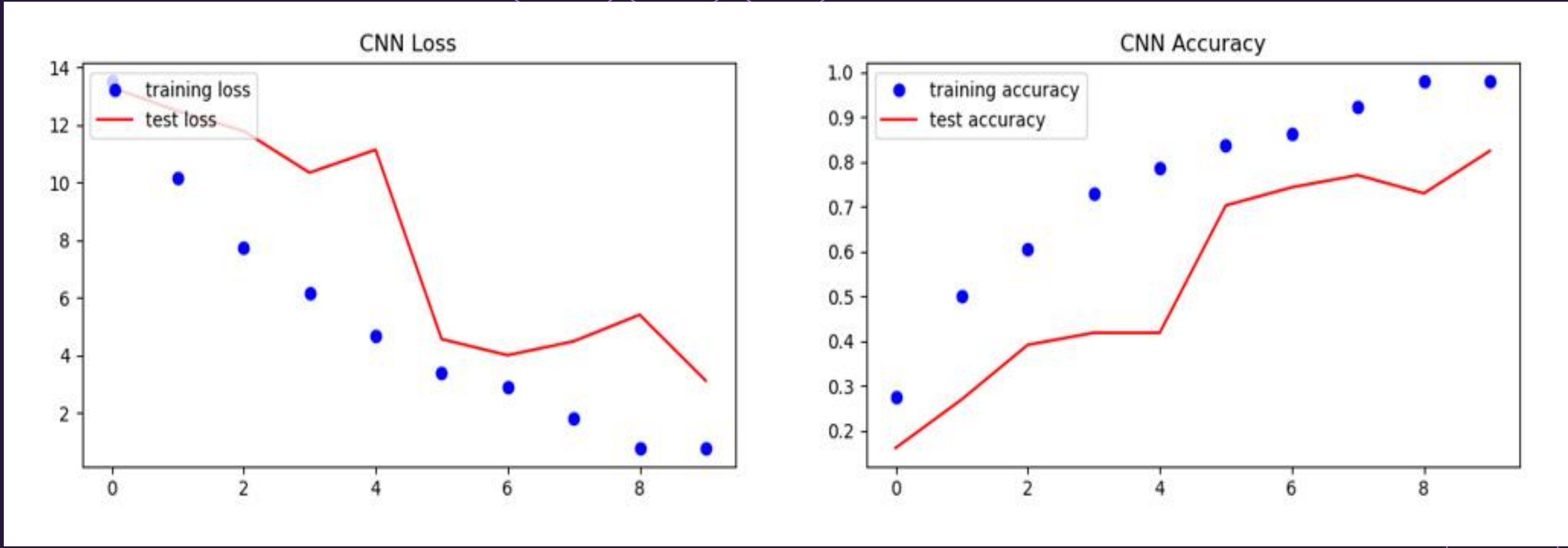
Run:



Squats:



Training Result:



Testing Accuracy (Real Radar):

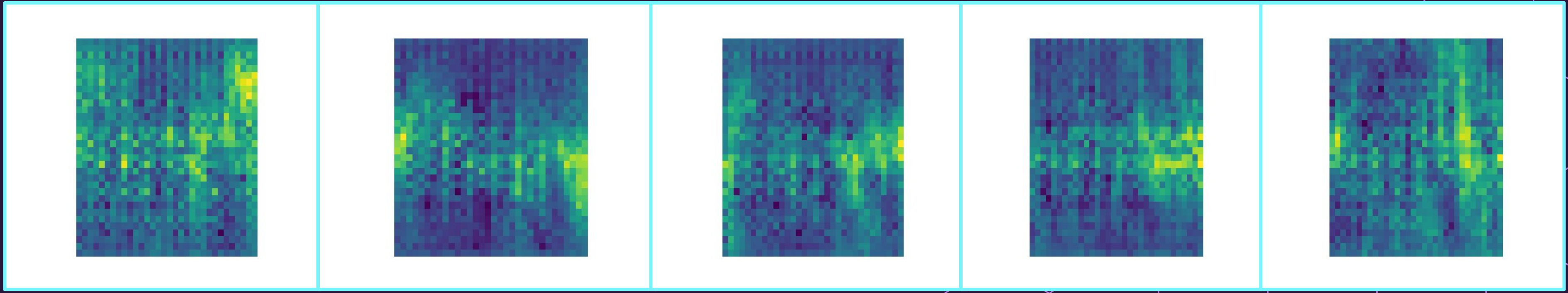
Testing accuracy about: 91.25 %

```
tensor([0, 4, 3, 3, 0, 0, 2, 4])
 90%|█████████ | 9/10 [00:01<00:00,  5.76it/s]tensor([4, 1, 0, 2, 2, 4, 1, 0])
tensor([2, 2, 1, 3, 4, 1, 0, 1])
73
0.9125
[3, 0, 1, 3, 4, 3, 3, 0, 4, 1, 0, 1, 0, 4, 0, 2, 1, 4, 1, 3, 0, 4, 0, 1, 1, 1, 3, 1,
100%|██████████| 10/10 [00:01<00:00,  5.79it/s]
```

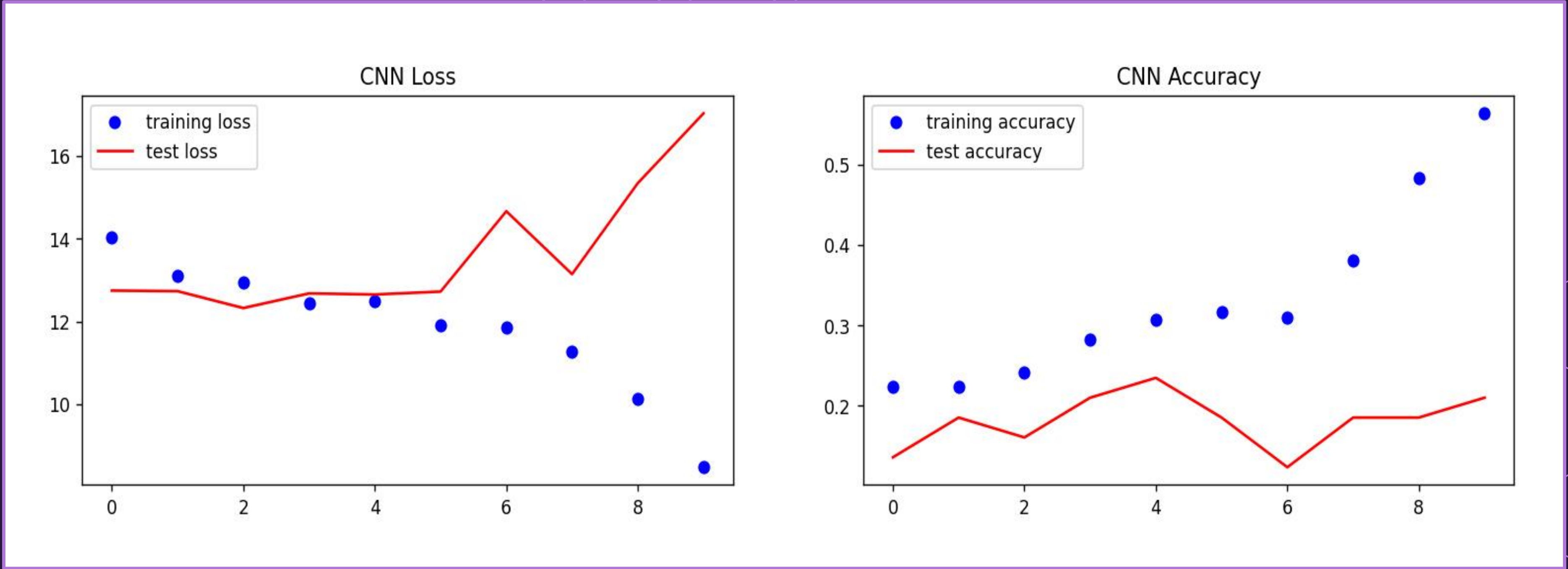
Training Result (using GAN Data)

Total five Actions to Recognize:

Handmotion: JJ: Lunge: Run: Squats:



Training Result:



Testing Accuracy (GAN):

Testing accuracy about: 20 %

```
    Training Loss: 9.261424     Validation Loss: 14.065243
    Training Accuracy: 0.531915     Validation Accuracy: 0.188525
 50%|█████     | 1/2 [00:00<00:00,  3.95it/s]tensor([3, 2, 1, 0, 2, 4, 3, 1])
100%|██████████| 2/2 [00:00<00:00,  5.64it/s]
tensor([4, 0])
2
0.2
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Process finished with exit code 0
```

Comparison Accuracy:

CNN model trained by Sythesized Data: 80.5%

CNN model trained by Real Radar Signal Data: 91.25%

CNN model trained by GAN Data: 20%

Using trained CNN model with Synthesized Data to test the Real Radar Signal

Accuracy: 20%

```
 96%|██████████ | 45/47 [00:10<00:00,  4.48it/s]tensor([0, 1, 1, 3, 0, 2, 3, 4])
 98%|██████████| 46/47 [00:10<00:00,  4.51it/s]tensor([0, 3, 0, 3, 2, 4, 1, 1])
100%|██████████| 47/47 [00:10<00:00,  4.49it/s]
tensor([1, 4])
74
0.2
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

Process finished with exit code 0
```

Using trained CNN model with GAN Data to test the Real Radar Signal

Accuracy: 15%

```
tensor([2, 2, 2, 3, 4, 1, 1, 1])
 90%|█████████ | 9/10 [00:01<00:00,  5.48it/s]tensor([0, 4, 0, 1, 2, 0, 3, 4])
tensor([3, 3, 4, 1, 2, 0, 4, 1])
12
0.15
[0, 0, 3, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 3, 0, 0, 0, 0, 3, 0, 3,
100%|██████████| 10/10 [00:01<00:00,  5.37it/s]

Process finished with exit code 0
```

Conclusion:

When training our CNN model with GAN data, our CNN model overfits. The reason maybe lack of training data.

Another reason maybe that the data generated by GAN has more features added to the PNG image, which makes it harder to distinguish the different actions compared to the data generated by the other two sources.

Future work: to improve and adjust our CNN model layers to train the data produced by GAN successfully.

Issues/Challenges/Roadblocks

- Channel mismatch during Training
- Operands Broadcast Error with Shapes
- Matching Tensor Size in accordance with Input-Output Combinations including Batch Size, Channels, Height and Width for Discriminator-Generator Blocks
- Adjusting Conv2D layers for efficient results

D. Project Status Update

Update of Team Member’s Technical Responsibilities

- **Upanshu Srivastava**
  - Collect and Label Video and Radar Data
  - Build and Train Generative Model (Pix2Pix)
- **Yinchien Haung**
  - Construct Mesh from Video and Simulate Radar Signal
  - Collect and Process the Signals from the Real Radar
  - Build and Train Generative Model
- **Ruijie Song**
  - Process the Signals from the Real Radar
- **Jin-Hau Yeh**
  - Build and Train CNN Model for Human Activity Recognition
- **All group members**
  - Study Radar Signal Formulation
  - Visualizing Neural Network and Training Process

Update of Team Member’s Management Responsibilities

- **Technical Project Manager (TPM) — Yin chien Huang**
  - Divide the whole project into specific tasks and assign to every team members
  - Track the progress of each tasks
  - Coordinate meetings
- **Document Coordinator (DC) — Upanshu Srivastava**
  - Restore and manage the video and radar data
  - Edit and proofread materials for reports/presentations
- **Business Office Liaison (BOL) — Ruijie Song**
  - Help purchasing and managing hardware equipments

Update of Project Plan (Milestones)

1. Collecting videos of motion and real-world radar signals, label those data for training. ✔
Milestone 1: Obtain labelled motion videos and real-world radar signals from motions. ✔

2a. Constructing mesh and Simulate radar signal ✔
Milestone 2a: Obtain simulated velocity Histogram plot. ✔

2b. Process the real signals from radar ✔
Milestone 2b: Obtain real velocity Histogram plot. ✔

3. Building basic generative model ✔
Milestone 3: Finish the code for the generative model structure and loss function validation. ✔

Update of Project Plan (Milestones)

4. Training and cross validation of the generativemodel. ✔
Milestone 4: Obtain the accuracy of our generative model. ✔

5. Testing, debugging and improving our generative model.
Milestone 5: Obtain refined radar signals which are ready for the motion recognition.

6. Apply the motions recognition model to the refined signals.
Milestone 6: Observe correct rate of each NN

Update of Project Plan (GANTT Chart)

Issues/Challenges/Roadblocks

1. Synchronize real and synthesized radar signal
   Solution: We start the camera recording and radar recording at the same time
2. Adjust the input size for both real and synthesized signal to be the same
   Solution: We format the processed real and synthesized signals to 32*32 matrices.
3. ~~Estimate camera angle from video~~
4. ~~Create a cascade training dataset of 32x32~~
5. ~~Design Pix2Pix GAN model in accordance to input-output of 32x32 matrix~~

E. Contributions

- Upanshu Srivastava (PPT Section - A, B, C, E; Developing and Training Pix2Pix GAN Model)
- Ruijie Song (PPT Section - B, C, D; Project Status Update)
- Yinchien Huang (PPT Section - B, C; Operating Radar system, Convert Video to Radar Signal, Signal Processing of Radar Signal, Training the GAN Model)
- Jin-Hau Yeh (PPT Section - B, C; Developing Human Activity Machine Learning Model)