Human Context Recognition:
A Controllable GAN Approach

Joshua DeOliveira, Worcester Polytechnic Institute
Harrison Kim, Northeastern University
MaryClare Martin, College of the Holy Cross

Faculty Advisor: Prof. Elke Rundensteiner
Ph.D. Mentor: Walter Gerych

Funded by NSF Grant #1852498

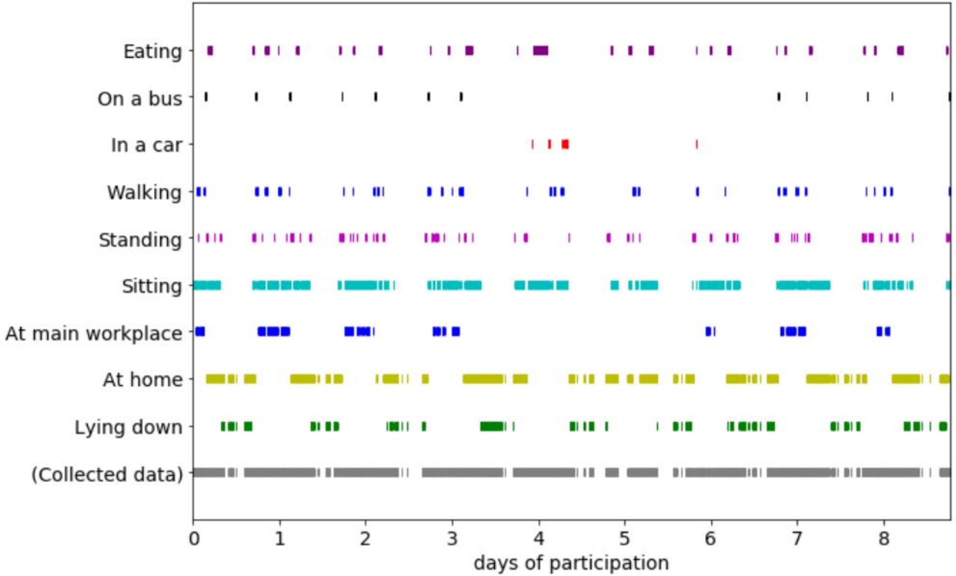
Progress Summary

1. Explored the ExtraSensory dataset through visualizations.
2. Further developed an understanding of PyTorch by building a simple NN classifier and a GAN for digit image generation.
3. Trained a Vanilla and Wasserstein GAN to generate accelerometer data in a non-sequential fashion.
4. Explored a novel training methodology for simple GANs.
5. Read and analyzed the strengths and weaknesses of different GAN-based architectures for time-series data.

ExtraSensory Dataset

- 60 Participants
- 300k+ data points
- 15 different devices
- 10 unique sensors
- 183 features
- 41 discrete device attributes
- 51 labels (not all mutually exclusive)

Accessible at: http://extrasensory.ucsd.edu/



Vaizman, Y., Ellis, K., and Lanckriet, G. "Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches." *IEEE Pervasive Computing,* October-December 2017.

Simple NN for Classification

Using featurized accelerometer data without temporal context from the ExtraSensory dataset, can we classify when users are sitting?

Model:
- 26 accelerometer features
- 4 ReLu layers (40 neurons each)
- 20% dropout
- 1 sigmoid layer to 1 neuron
  - Probability user was sitting

Training:
- e = 0.001
- Epochs = 120
- Training Loss: MSE
- Batch Size = 10,000 datum
- Volume = 264,142 datum

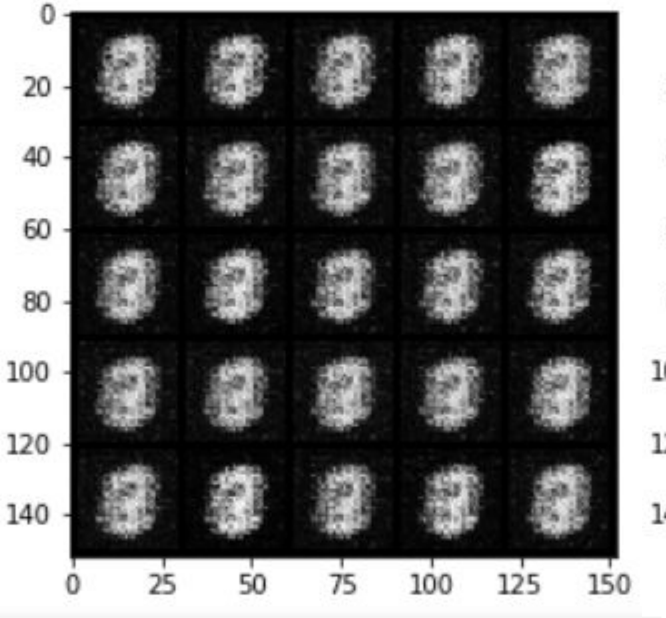
Results:
Naive Accuracy: 63.86%
Test Accuracy: 70.72% (+10.7%)
Loss After Training: 0.1882

PyTorch GAN for Digit Image Generation

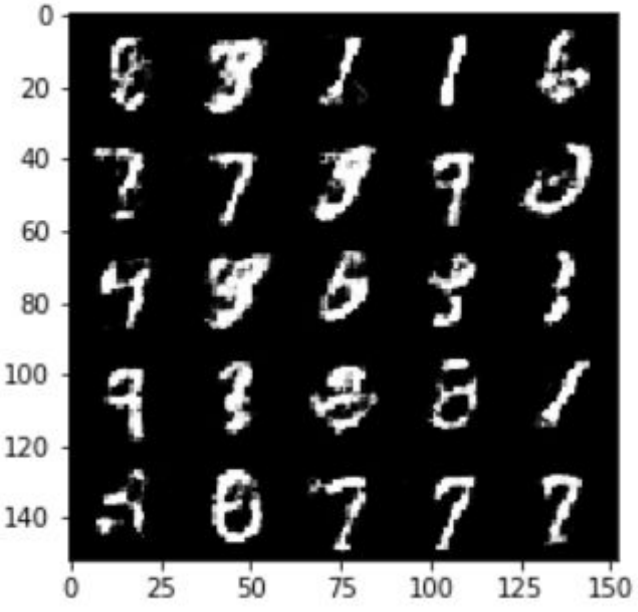
Generator:
- Linear Layers
- 1D Batch Normalization
- ReLU Layers
- Sigmoid Output Layer

Discriminator:
- Linear Layers
- Leaky ReLU Layers
- Linear Output Layer
- BCE with Logits Loss

Epoch 1



Epoch 185



A Dynamic Training Approach

Reasoning:

Conventionally, GANs train by sequentially freezing one machine, ***G*** or ***D***, and training the other, each at a constant number of epochs *g* and *d*. Ideally, $d > g$ and *d* trains first such that ***D*** is always better than ***G*** to provide effective feedback. This can lead to scenarios, after ***G*** begins to generate semi-believable data and D well understands the real data, keeping $d > g$ inefficiently wastes epochs that can lead to overfitting ***D***. We can efficiently train both machine's by enforcing some reason of "fair competition" that requires ***D*** only barely better than ***G*** to maximize improving ***G*** (the purpose of our GAN) as many epochs as possible.

A Dynamic Training Approach

Hyper-parameters:

- Static_threshold: duration of Phase 1 (in epochs)
- D_static, G_static: # of epochs to train each model during static phase
- Pull_threshold: minimum fp Rate to keep ***D*** training
- Push_threshold: maximum fp Rate to keep ***G*** training
- Recall_threshold: minimum recall of ***D*** to begin Phase 3

A Dynamic Training Approach

Training Methodology:

Phase 1: Static Training

- Train conventionally to allow fundamental training to both machines based off of *D_static* and *G_static*

Phase 2: Check for Understanding

- Allow ***D*** to train until the recall of ***D*** is above the *Recall_threshold*

Phase 3: Pull / Push

1. Allow ***G*** to train until it fools ***D*** above the push threshold, then "push" ***G*** away
2. Allow D to train until it no longer is fooled by G according to the pull_threshold

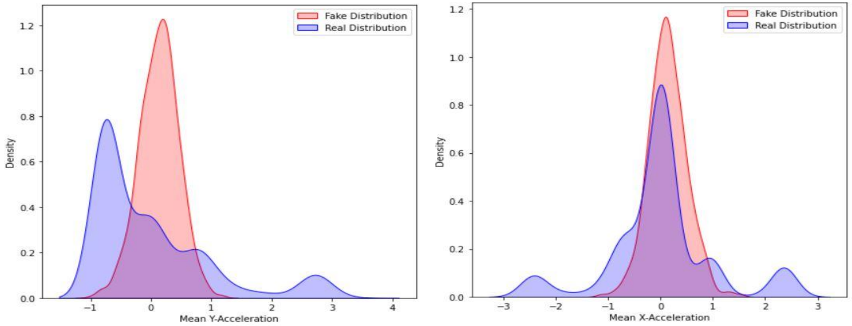
Vanilla GAN for Generating non-Sequential Accelerometer Data

Generator:
- 4 ReLU/Batch Normalization layers
- 10% dropout
- Linear output layer (26-dimensional)

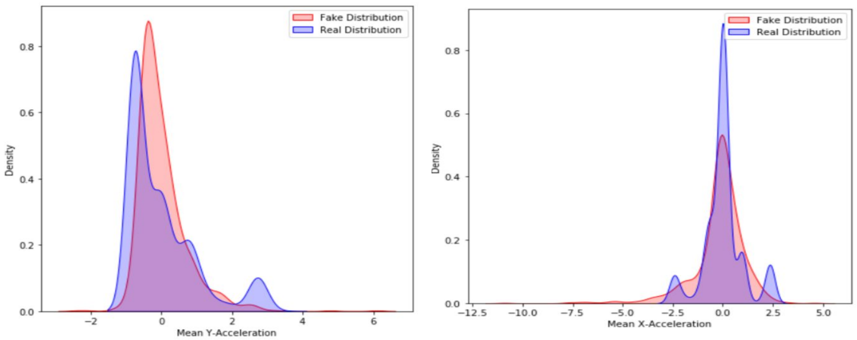
Discriminator:
- 3 Leaky ReLU layers
- 10% dropout
- Linear output layer (1-dimensional) with Sigmoid activation function
    - Probability of real/fake features

Training:
- Generator e = 0.001
- Discriminator e = 0.0001
- Epochs = 2000
- Training Loss = BCE Loss
- Batch Size = 1000 datum
- Latent Vector Dimension = 100

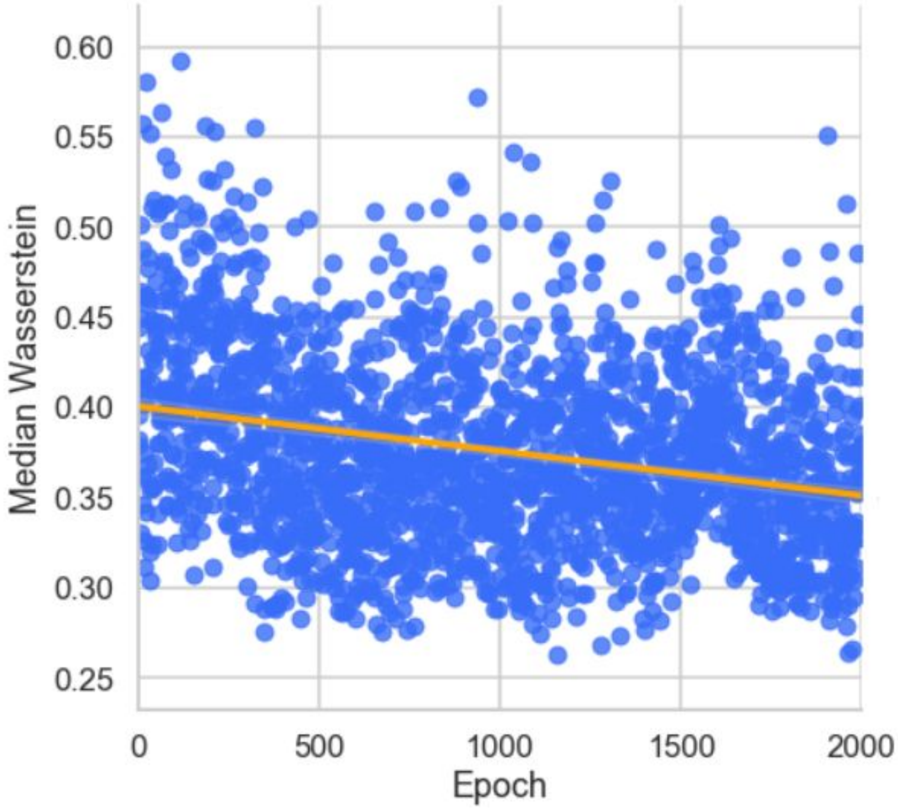
Before Training

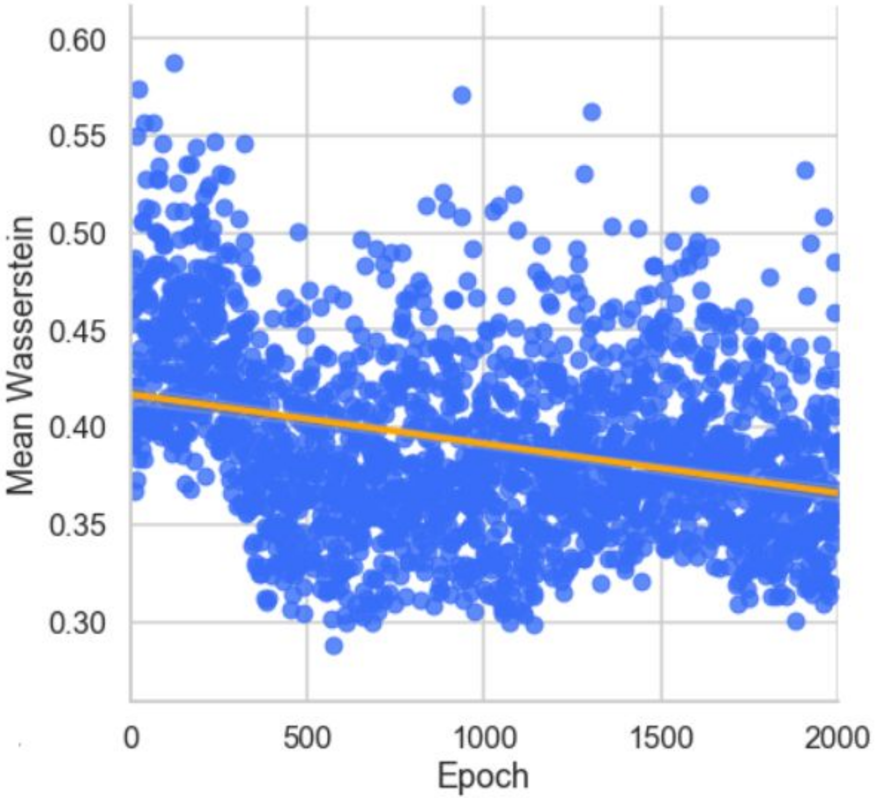


2000 Epochs



Wasserstein Distance on Vanilla GAN





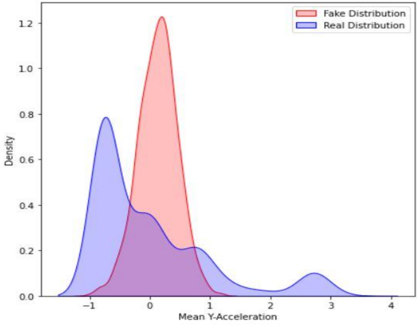
Wasserstein GAN for Generating non-Sequential Accelerometer Data

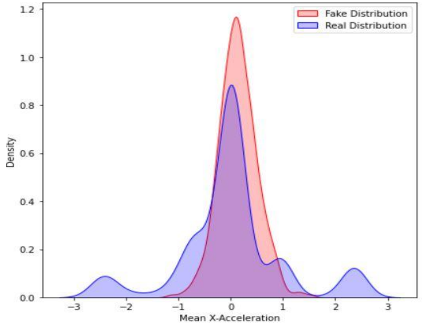
Generator:
- 4 ReLU/Batch Normalization layers
- 10% dropout
- Linear output layer (26-dimensional)

Discriminator:
- 3 Leaky ReLU layers
- 10% dropout
- Linear output layer (1-dimensional)

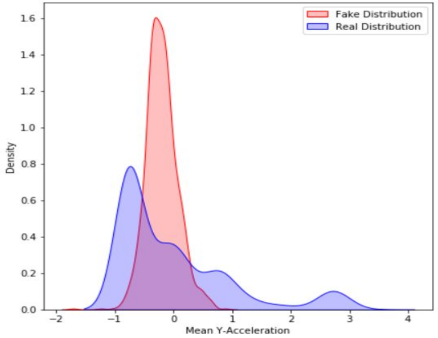
Training:
- Generator e = 0.00001
- Discriminator e = 0.00001
- Epochs = 800
- Training Loss = Wasserstein Distance
- Batch Size = 1000 datum
- Latent Vector Dimension = 100

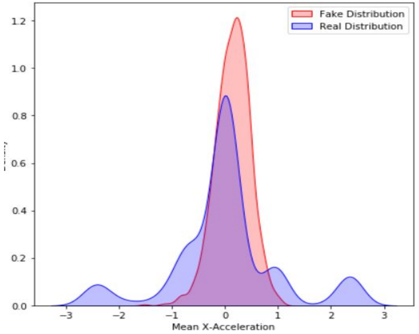
Before Training



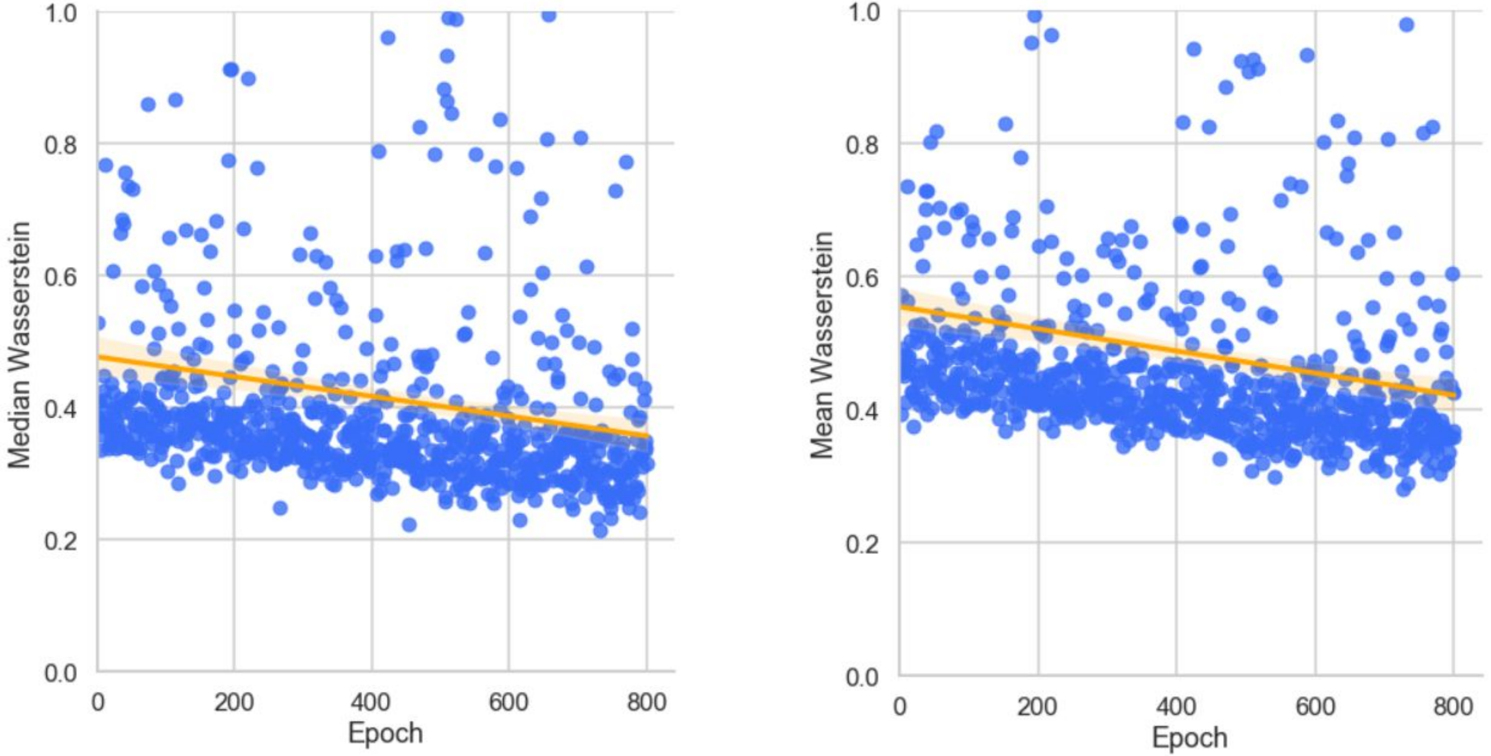


800 Epochs



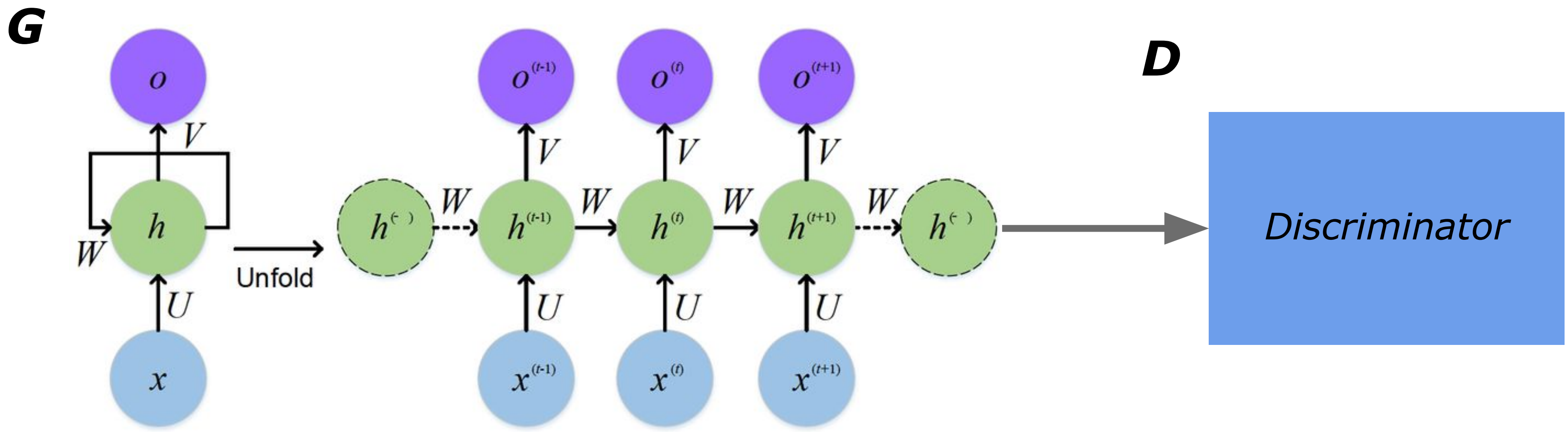


Wasserstein Distance on WGAN



GANs for Sequential Data

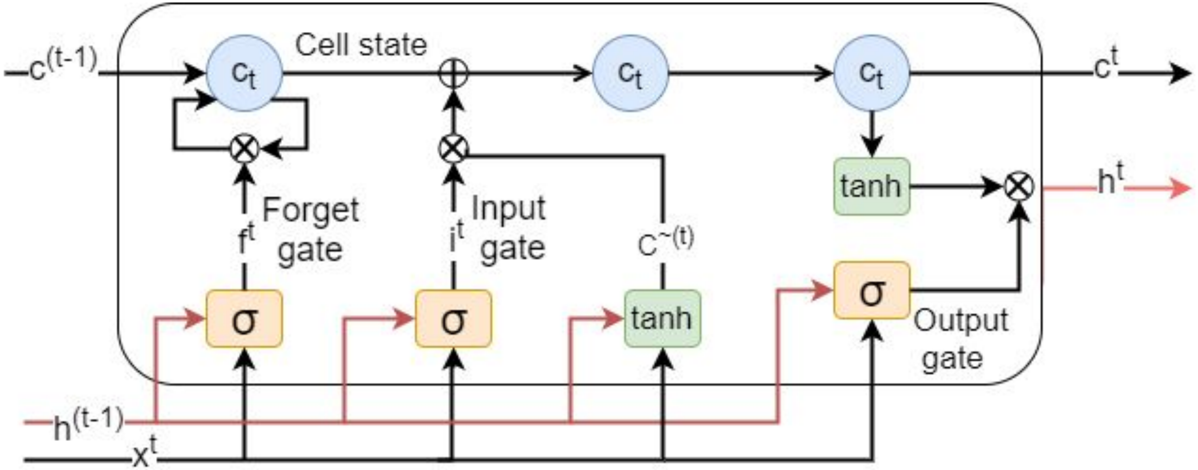
Generators with recurrent architectures allow for developing realistic time-series data for an iteration t by using previous iterations for generation (t-1, t-2, t-3, … t-n)



Feng et. al. (2017). Audio visual speech recognition with multimodal recurrent neural networks. 681-688. 10.1109/IJCNN.2017.7965918.

GANs for Sequential Data

Generators with recurrent architectures allow for developing realistic time-series data for an iteration t by using previous iterations for generation (t-1, t-2, t-3, … t-n)



Jenkins et. al. (2018). Accident Scenario Generation with Recurrent Neural Networks. 3340-3345. 10.1109/ITSC.2018.8569661.

Next Steps

1. Implement third GAN evaluation metric (compare accuracy of a classifier trained on fake & real features to a classifier trained on only real features).

2. Continue hyperparameter tuning/experimentation of the Vanilla/Wasserstein GAN architectures.

3. Begin developing a Controllable GAN for generating accelerometer data for select classes.