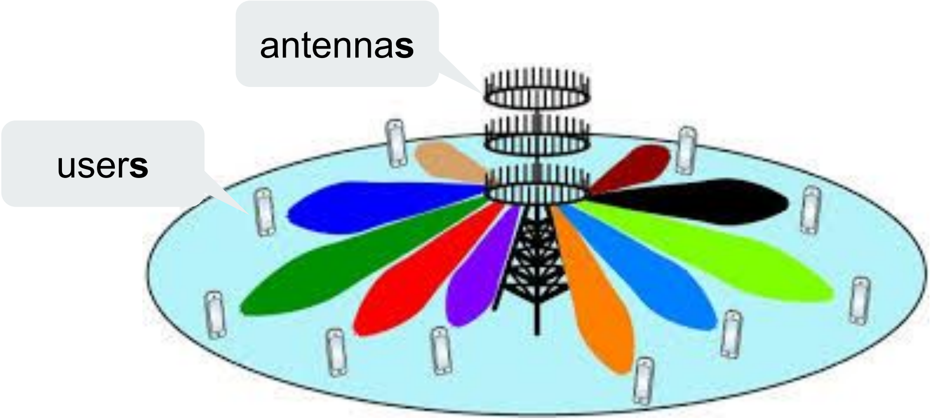
Massive MIMO CSI prediction -- a ML approach

Yaning Hu (Helena) -- Tarence (mentor) -- Robert Chen (co-intern)
August 2020

What is Massive MIMO?

Massive multiple-input, multiple-output, is an extension of MIMO, which essentially groups together antennas at the transmitter and receiver to provide better throughput and better spectrum efficiency.



Abstract

Research Objectives:

Finding the relationship between the Channel State Information (CSI matrix) and the drone's location from a Machine Learning Approach.

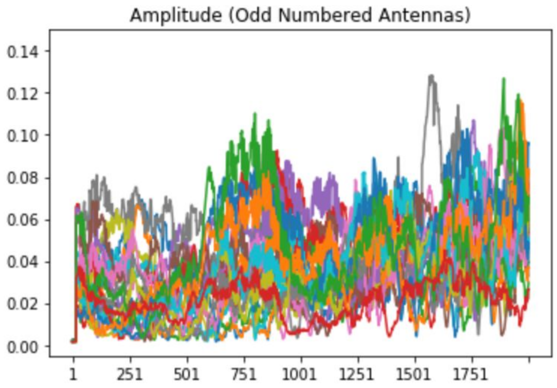
Methodology:

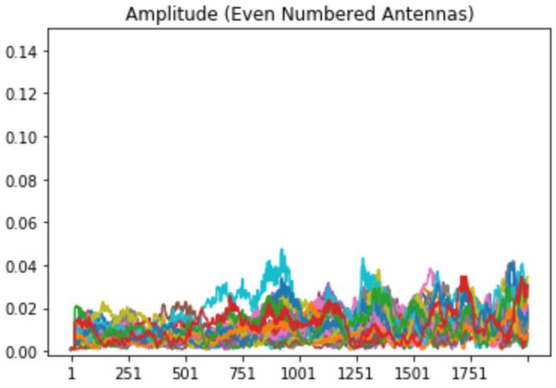
- 1st Approach: **Time Series Forecasting** on CSI (without location labels)
- 2nd Approach: Deep Learning **Neural Network** (with location labels)

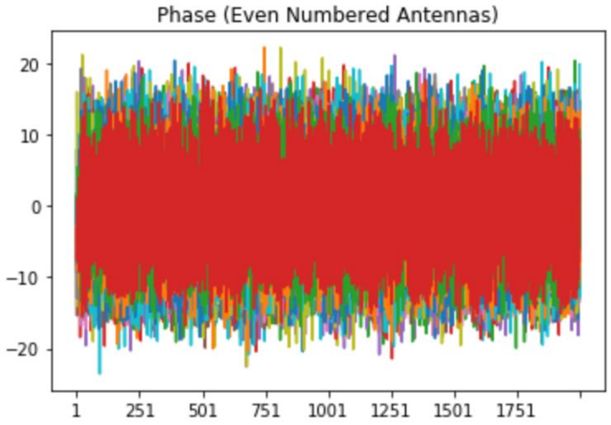
Time Series Forecasting

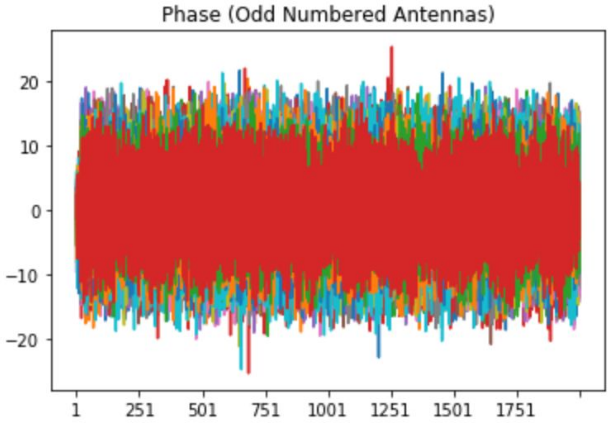
STEP 1: Investigating the dataset

amplitude & phase









Time Series Forecasting

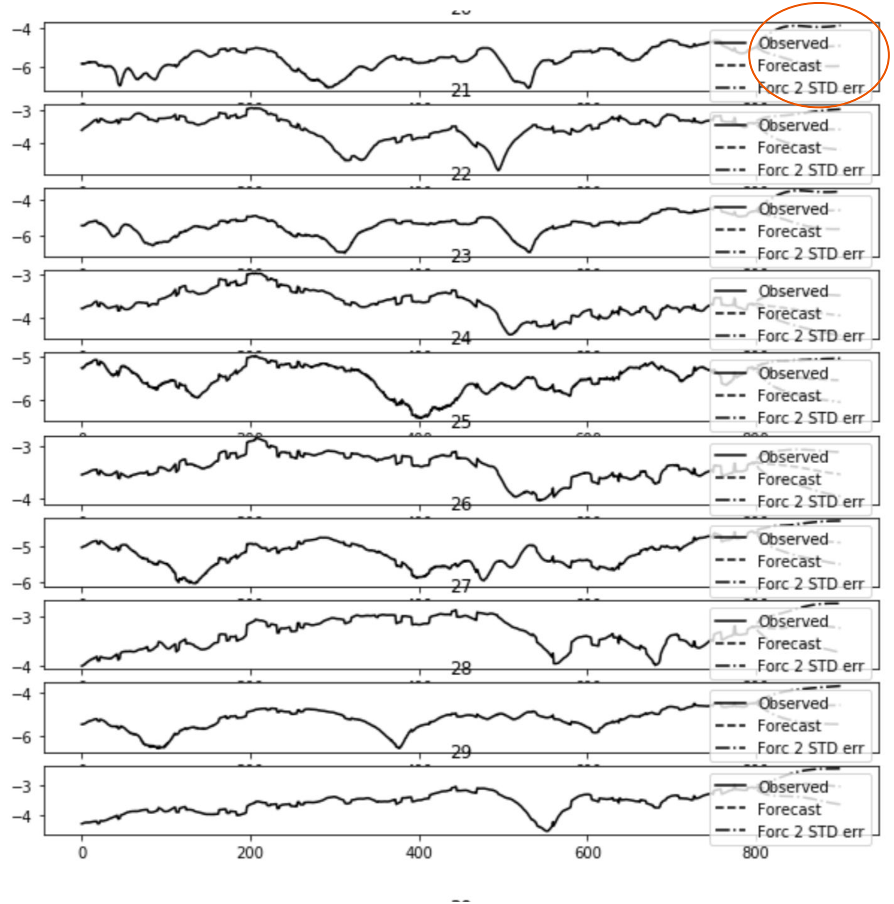
STEP 2:

Apply VAR model (Vector Autoregression) on phase

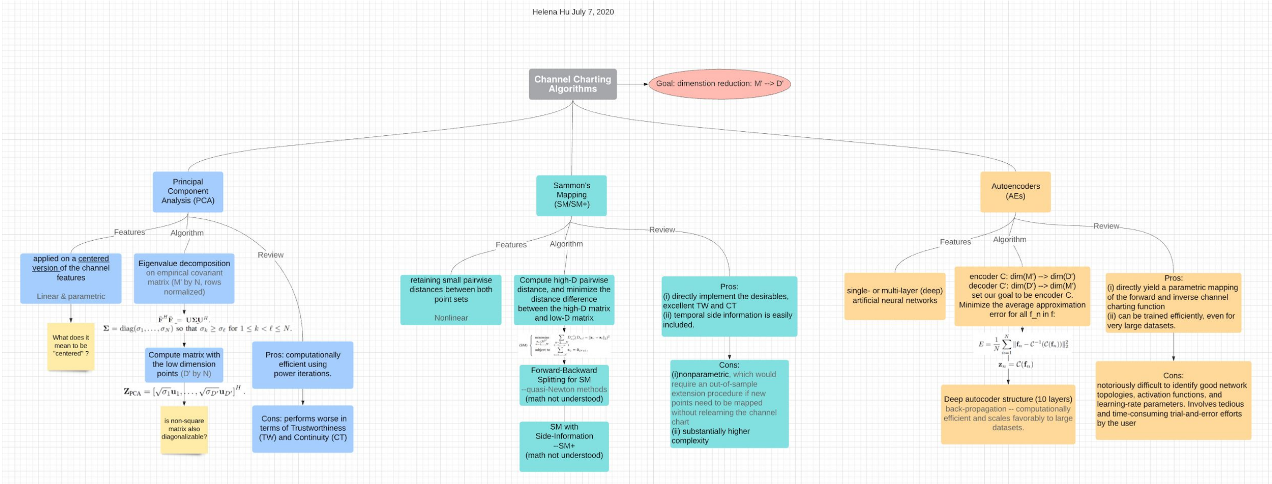
Why VAR:

Multivariable Forecasting Method

(consider influences between variables)



Relationship between CSI and Location



STEP 1: Literature Review

Channel Charting Algorithms (3)

Overall Idea:

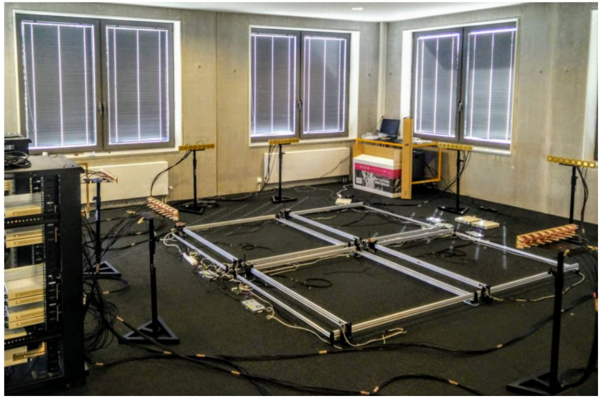
CSI → distill properties → reduce dimension → Channel Chart that preserves spatial information → approximation of location data

Next Step: View the problem from pure Machine Learning Angle

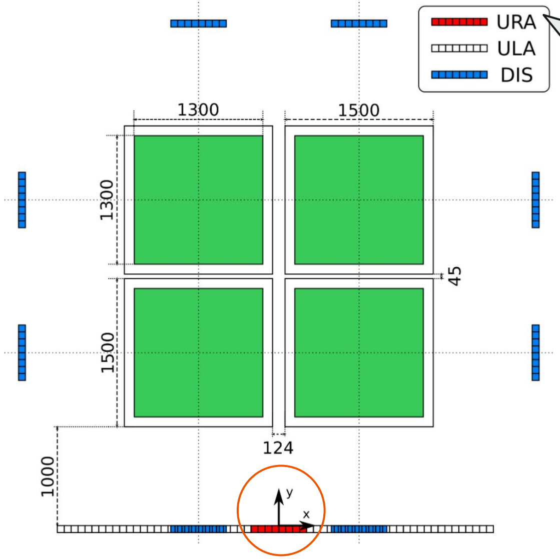
Data Setup:

1. Download Data Set:

   Indoor Spatially Labelled Massive MIMO CSI Measurements (~25000 location labels)
   (https://homes.esat.kuleuven.be/~sdebast/measurements/measurements_boardroom.html)



2. The Base Station (BS) is equipped with 64 antennas, each receiving a predefined pilot signal from each position. Using these pilot signals, the CSI is estimated for 100 subcarriers, evenly spaced in frequency.

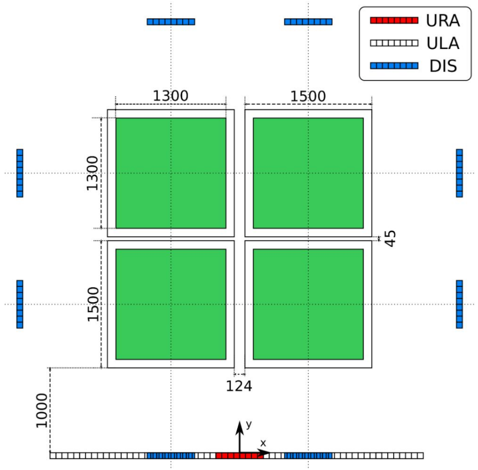


Uniform Rectangular Array

How much data do we have?

3 Array Types

-- each has 64 antennas

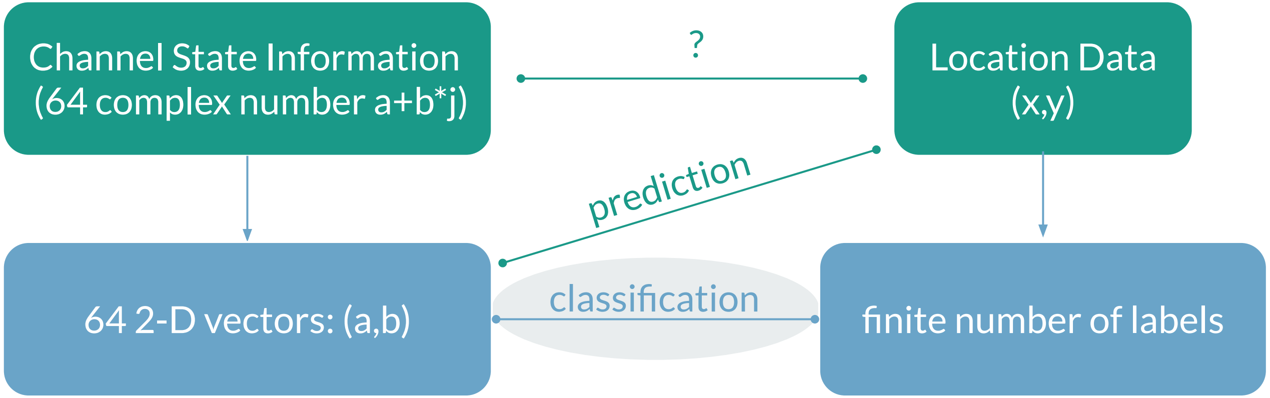


--each antenna has 100 subcarriers (frequencies)

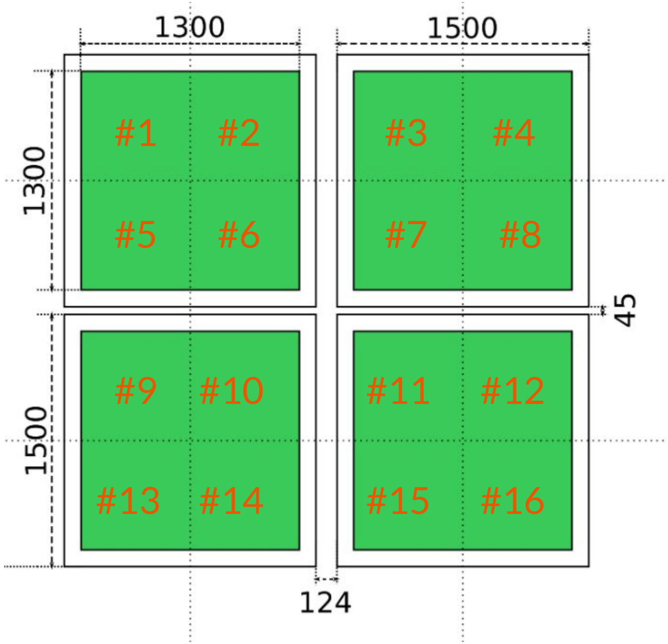
--at each frequency, we have ~250,000 corresponding location data & CSI data

250,000
data points

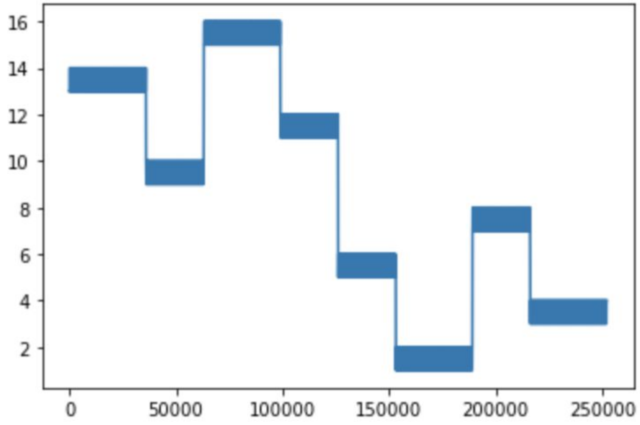
Make use of this data set



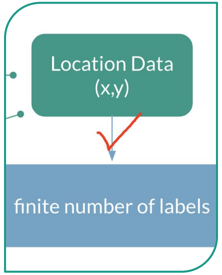
Step to a classification Task: Fitting Grids



label



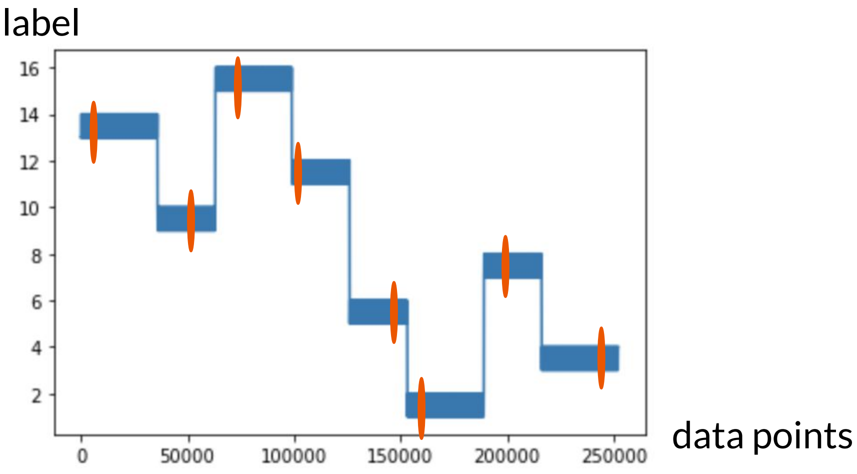
data points



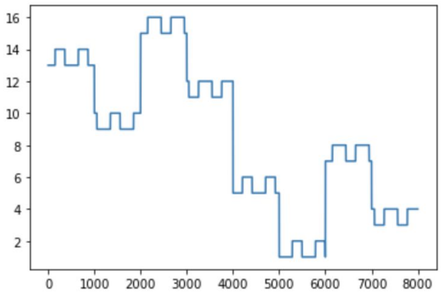
Select data for our model

Desired feature: including all labels

```
#taking data that covers all labeled locations:
data_points = [[1000,2000],[50000,51000],[75000,76000],[100000,101000],[140000,141000],[160000,161000],[200000,201000],
data_loc = label[1000:2000]+label[50000:51000]+label[75000:76000]+label[100000:101000]+label[140000:141000]+label[16000
plt.plot(data_loc)
```



Total: 8000 data points

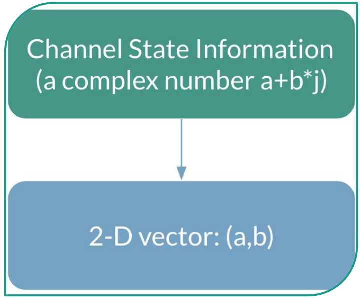


A different way to look at CSI

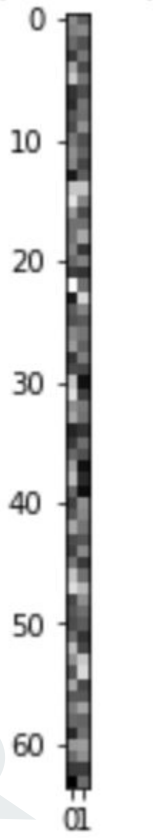
For each location, we have: 64 antennas, each having 100 frequencies

1) Fix the frequency for all antennas → select the 10th subcarrier
2) CSI datashape: 64*2*1 → view this as an IMAGE!

Figure: "CSI image" of the 1st selected datapoint



0: real part
1: imaginary part



Why image? -- Convolutional Neural Network

1

8000 data points

Train: 5360(70%)

Test: 2640(30%)

```
print(X_train.shape,X_test.shape)
print(y_train.shape,y_test.shape)
```

```
(5360, 64, 2, 1) (2640, 64, 2, 1)
(5360, 16) (2640, 16)
```

2

Set Parameters and build up CNN model

(following the MNIST dataset tutorial) -- need some changes

```
training_iters = 10
learning_rate = 0.001
batch_size = 64

#MNIST data input (img shape:64*2)
n_input = 64

#MNIST total classes
n_classes = 16

#both placeholders are of type float
x = tf.placeholder("float", [None, 64, 2, 1])
y = tf.placeholder("float", [None, n_classes])

def conv2d(x, W, b, strides=1):
    # Conv2D wrapper, with bias and relu activation
    x = tf.nn.conv2d(x, W, strides=[1, strides, strides, 1], padding='SAME')
    x = tf.nn.bias_add(x, b)
    return tf.nn.relu(x)

def maxpool2d(x, k=2):
    return tf.nn.max_pool(x, ksize=[1, k, k, 1], strides=[1, k, k, 1],padding='SAME')
```

Result interpretation:

Potential Problems:

1) Not fully making use of the subcarriers properly
2) Model is used for a 28*28 image in the original problem, and utilizes max_pooling algorithm which downsized the image to 4*4.
3) In our situation, the CSI image has the 2nd dimension as small as 2, so downsizing might not yield desirable results

```
Iter 0, Loss= 2.416927, Training Accuracy= 0.09375
Optimization Finished!
Testing Accuracy: 0.06705
Iter 1, Loss= 12.204795, Training Accuracy= 0.03125
Optimization Finished!
Testing Accuracy: 0.05227
Iter 2, Loss= 27.742737, Training Accuracy= 0.06250
Optimization Finished!
Testing Accuracy: 0.07348
Iter 3, Loss= 74.773407, Training Accuracy= 0.01562
Optimization Finished!
Testing Accuracy: 0.05492
Iter 4, Loss= 100.262283, Training Accuracy= 0.04688
Optimization Finished!
Testing Accuracy: 0.07083
Iter 5, Loss= 108.614357, Training Accuracy= 0.03125
Optimization Finished!
Testing Accuracy: 0.05606
Iter 6, Loss= 482.886902, Training Accuracy= 0.04688
Optimization Finished!
Testing Accuracy: 0.07083
Iter 7, Loss= 995.252747, Training Accuracy= 0.07812
Optimization Finished!
Testing Accuracy: 0.05038
Iter 8, Loss= 1490.359253, Training Accuracy= 0.09375
Optimization Finished!
Testing Accuracy: 0.10076
Iter 9, Loss= 5262.278320, Training Accuracy= 0.03125
Optimization Finished!
Testing Accuracy: 0.04735
```

Future investigations

1) Find another classification model that is more suitable for our data feature
2) Divide the location into smaller grids for more intricate classification
3) Utilize prediction models to predict from **64 2-D vectors** (CSI) to get its most likely corresponding **2-D vector** (location).
4) Our wish from the super high level: reverse the above prediction -- from a 2-D vector to predict all these 64 2-D vectors!

Acknowledgement

Dr. Knightly

Robert Chen

Tarence

Major References:

[1] Christoph Studer, Sa, ıd Medjkouh, Emre Gon, ultas ̧ Tom Goldstein, and Olav Tirkkonen, “Channel Charting: Locating Users within the Radio Environment using Channel State Information”

[2] Sibren De Bast, Andrea P. Guevara, and Sofie Pollin, *"CSI-based Positioning in Massive MIMO systems using Convolutional Neural Networks"* accepted at IEEE 91st Vehicular Technology Conference: VTC2020-Spring, Antwerp, May 2020

[3] Convolutional Neural Networks with TensorFlow -- DataCamp Tutorial https://www.datacamp.com/community/tutorials/cnn-tensorflow-python

Thank you for listening!