Security in AI

general formatting for slides

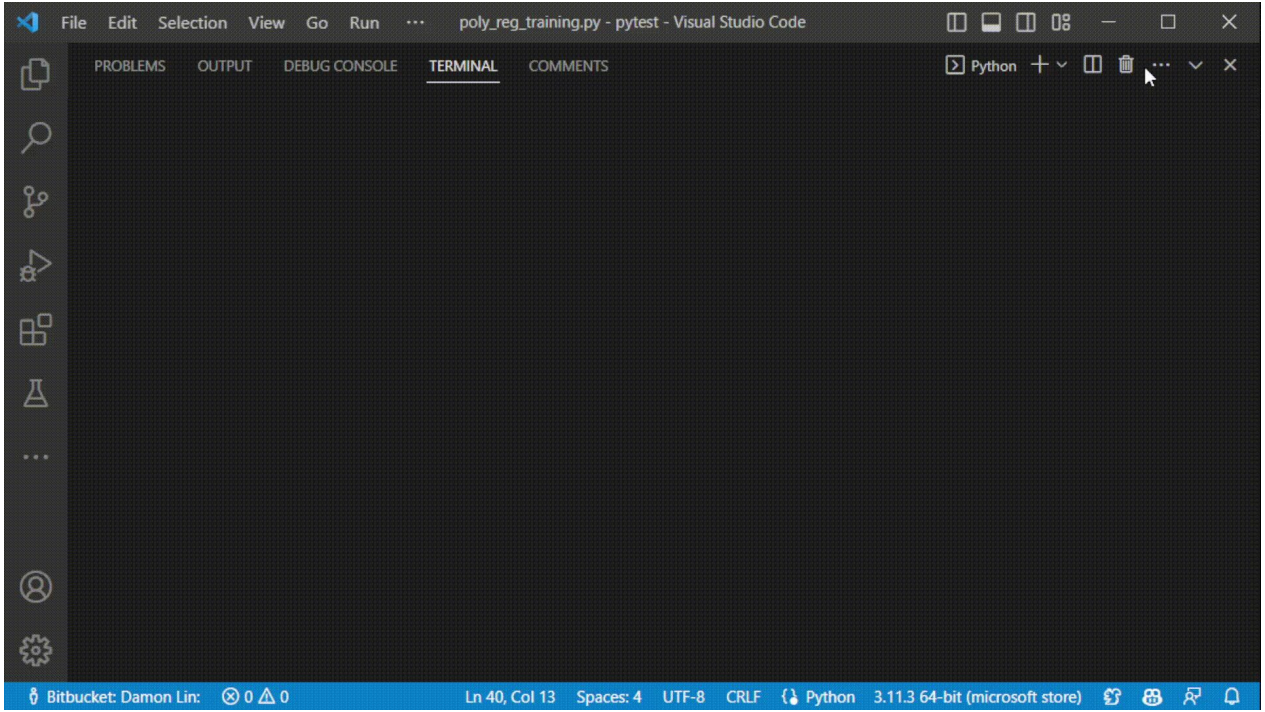
- Direction of the project/research changed slightly
  - instead of targeting voice assistance systems, it will be WiFi-enabled IoT devices
- What we worked on this week:
  - Familiarizing ourselves with PyTorch
  - Reading paper (check Slack for this if haven't seen already)
  - (Maybe?) exploration/research into attack mitigation

Exploring PyTorch

```python
import torch
import torch.nn as nn
import numpy as np

def data_generator(data_size=1000):
    inputs = []
    labels = []

    # loop data_size times to generate the data
    for i in range(data_size):
        # generate a random number between 0 and 1000
        x = np.random.randint(2000) / 1000

        # calculate the y value using the function
        y = x**3 + 2*(x**2) - 2*x - 3                          # change

        # append the values to our input and labels lists
        inputs.append([x**3, 2*x*x, -2*x])                      # change
        labels.append([y])

    return inputs, labels

# Cubic regression model
class PolyRegression(nn.Module):

    # init method
    def __init__(self):
        super(PolyRegression, self).__init__()
        # define layers
        self.fc1 = nn.Linear(3, 1)                            # change

    # forward method
    def forward(self, x):
        return self.fc1(x)

model = PolyRegression()
```

```python
# Training
learning_rate = 0.01
n_iters = 2000

loss = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

for epoch in range(n_iters):
    # generate data
    X, y = data_generator(data_size=1000)
    X = torch.tensor(X, dtype=torch.float32)
    Y = torch.tensor(y, dtype=torch.float32)

    # prediction = forward pass
    y_pred = model(X)

    # loss
    l = loss(Y, y_pred)

    # gradients
    l.backward() # dl/dw

    # update weights
    optimizer.step()

    # zero gradients
    optimizer.zero_grad()

    if epoch % 100 == 0:
        [w, b] = model.parameters()
        print(f'epoch {epoch + 1}: w = {w[0][0].item():.3f}, loss = {l:.8f}')

model.eval()
test_data = data_generator(1)
prediction = model(torch.autograd.Variable(torch.Tensor(test_data[0][0])))
print("Input: x = {}".format(test_data[0][0][2]/(-2)))                # change based on polynomial
print("Prediction: {}".format(prediction.data[0]))
print("Expected: {}".format(test_data[1][0][0]))
```

Exploring PyTorch (Cont.)



Results

```
Input: x = 1.329
Prediction: 0.19943690299987793
Expected: 0.22181628899999994
```

IoT Devices

Internet of Things (IoT) Devices

- Connected to internet
- Gather information
- Send and receive data through wireless channels

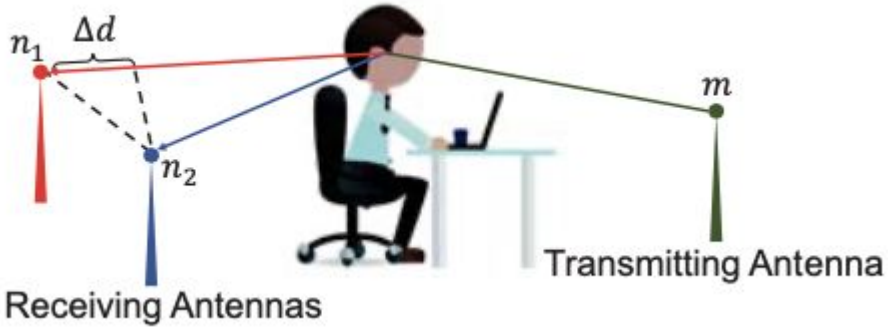
- **Channel State Information** (CSI) has the communications between devices

Using IoT Devices for Authentication

IoT Devices can pick up interference from:

- behavioral characteristics
  - body movements

Gives **unique biometric signature** of each user from **behavioral** and **physiological** characteristics



User-authentication can be deployed using deep neural networks on user activity information

- Without devices
- No password/fingerprint/facial recognition

Goals for Next Week

Continue learning PyTorch with more advanced mathematical functions to apply to IoT interference data

Set up experiments to collect interference data from mobile devices