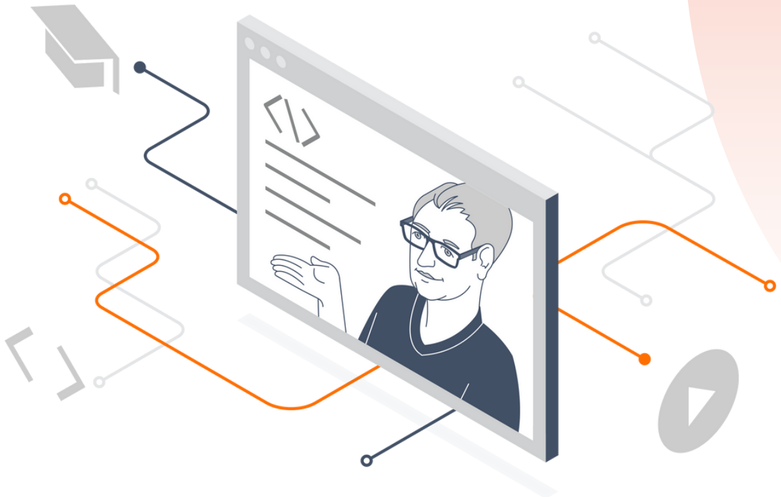
PYTORCH

Bootcamp

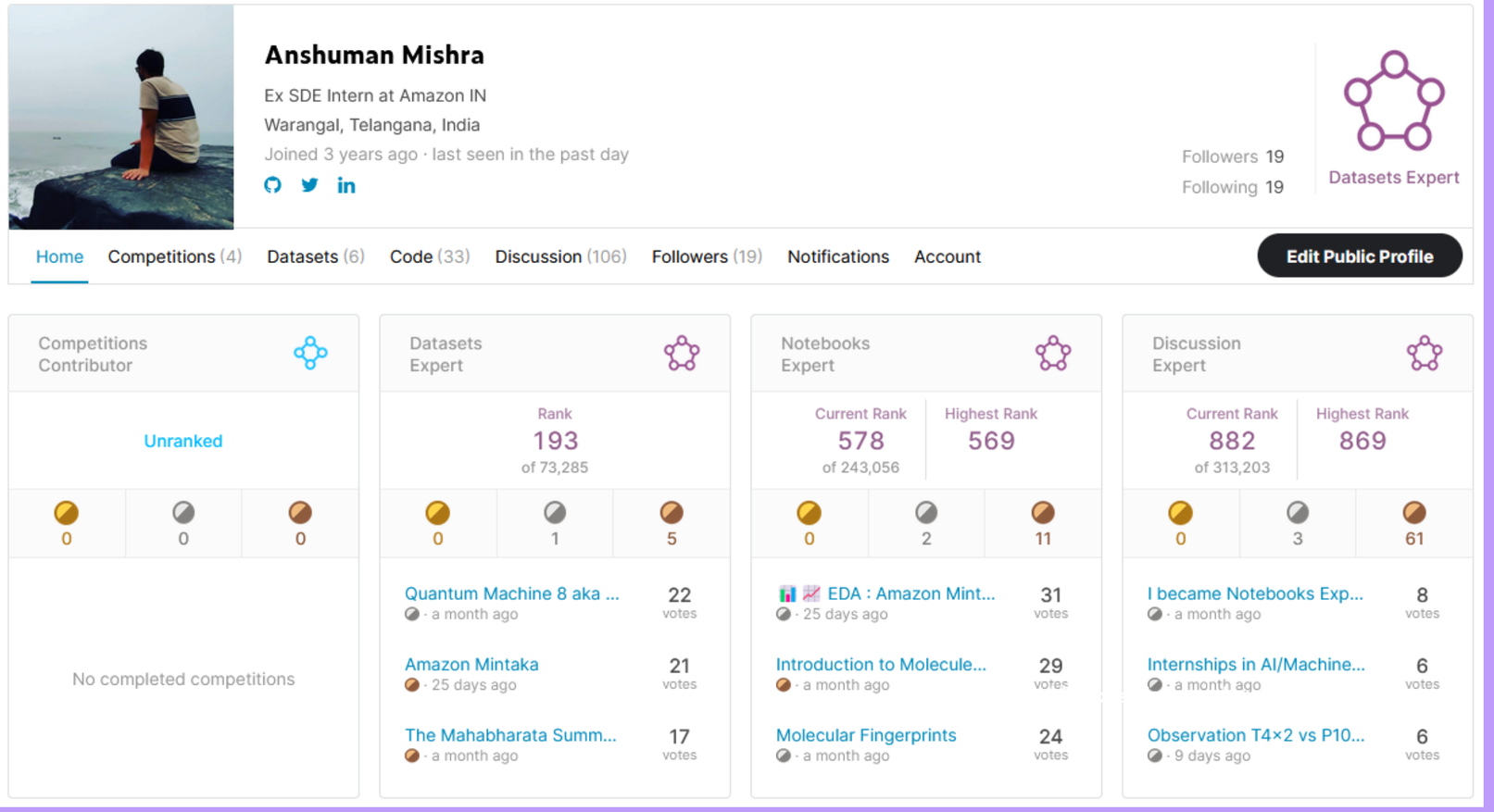
01

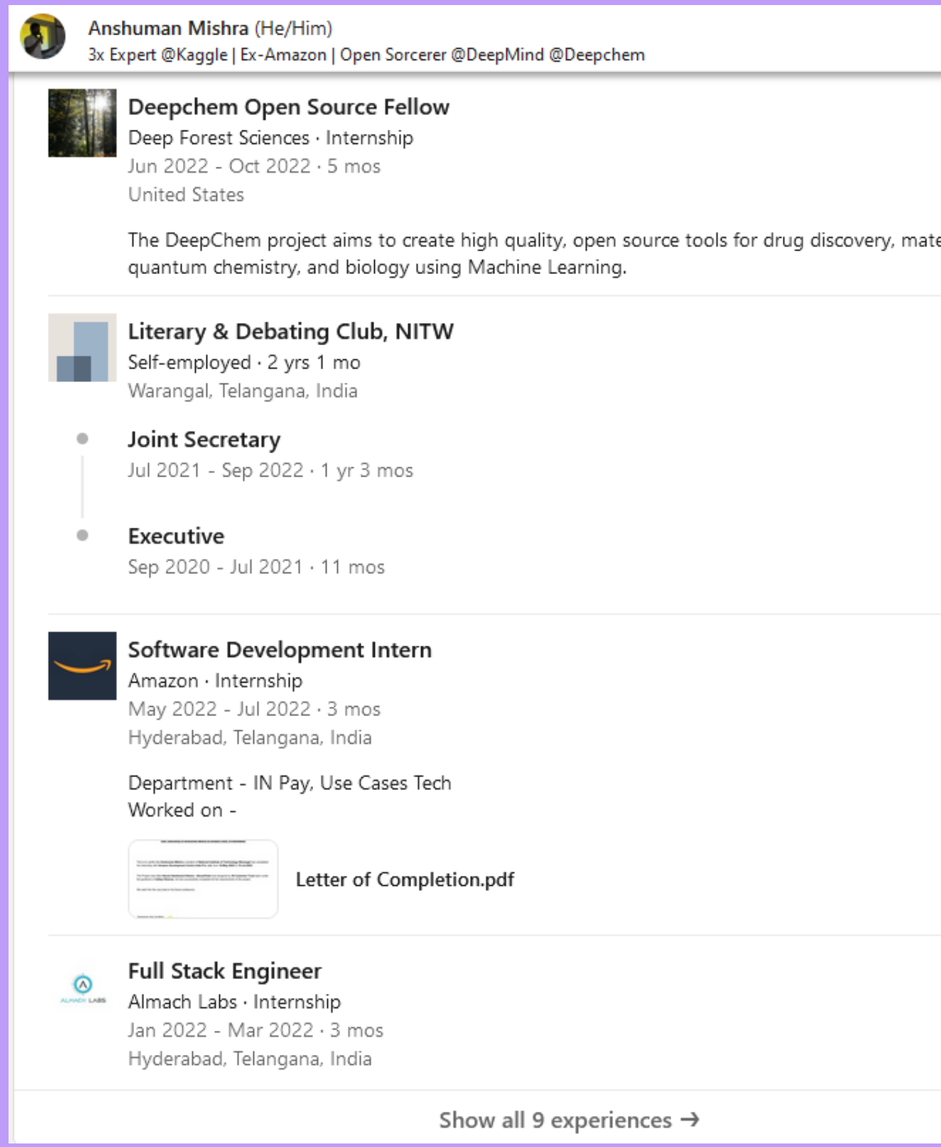


BY ANSHUMAN MISHRA

BIG DATA, ANALYTICS & CONSULTING CELL

About Myself

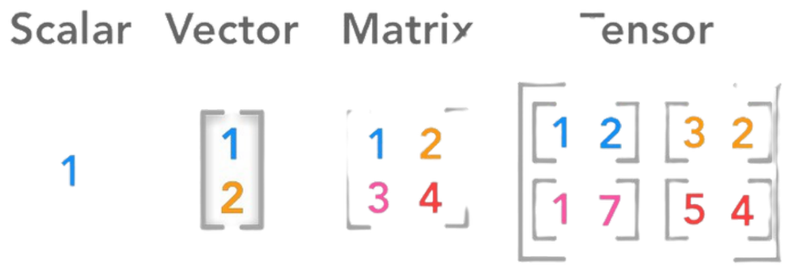




BASICS

Tensors

torch.Tensor
torch.ones()
torch.zeros()



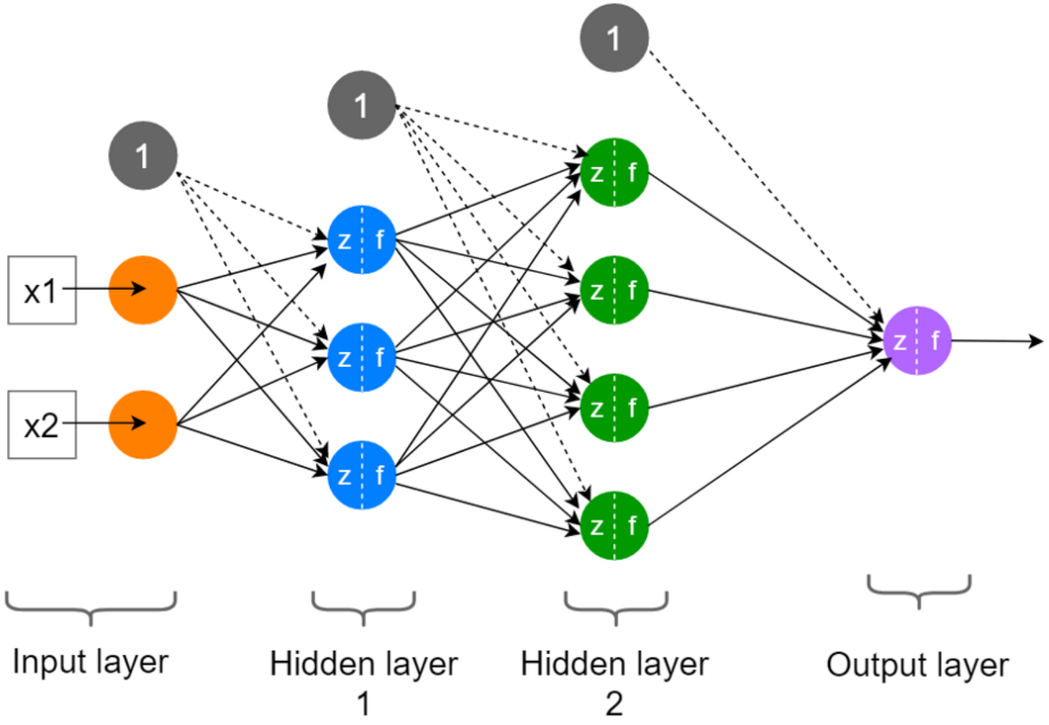
Object Oriented Programming

```
class Vehicle:
    def __init__(self, n_wheels):
        self.wheels = n_wheels
        self.petrol = 0

    def fill_petrol(volume):
        self.petrol += volume
        return self.petrol
```

```
>>> car = Vehicle(4)
>>> car.petrol
0
```

Neural Network



Link

C++

```
class Layer{
 //implementation
};

class network{
public:
 Layer l1, l2;

 network(){
  l1 = new Layer(*params);
  l2 = new layer(*params);
 }

 vector<int> forward(vector<int>& input){
  vector<int> out;
  out = this->l1(input)
  out = this->l2(out);
 }
}
```

PyTorch

```
import torch
import torch.nn as nn

class network(nn.Module):

def __init__(self):
      super().__init__()
      self.layer1 = nn.Linear(64, 32)
      self.layer2 = nn.Linear(32, 8)

def forward(self, x):
      out = self.layer1(x)
      out = self.layer2(out)

      return out
```

Let's run it on kaggle

MODEL

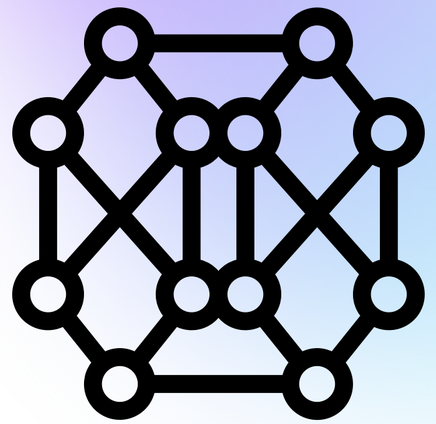
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(1, 32, 3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(32, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2, 2),
            nn.Dropout(0.25))

        self.fc = nn.Sequential(
            nn.Linear(32, 10),
        )
```

```
def forward(self, x):
    x = self.conv1(x)
    x = self.conv2(x)
    x = self.conv3(x)

x = x.view(x.size(0), -1)
    return self.fc(x)
```



Loading Data, Devices and CUDA

Numpy Arrays to PyTorch Tensors
- **torch.from_numpy(x_train)**
- **Returns a cpu tensor!**

PyTorch Tensor to Numpy
- **t.numpy()**

Using GPU acceleration
- t.to()
- Sends to whatever device (cudaor cpu)

- CPU - torch.cpu.FloatTensor
- GPU - torch.cuda.FloatTensor

Fallback to cpu if gpu is unavailable:
- **torch.cuda.is_available()**

Check cpu/gpu tensor OR numpyarray ?
- **type(t)or t.type()returns**
- **numpy.ndarray**
- **torch.Tensor**

Let's run things on kaggle

TRAINING LOOP

```
for e in range(1, epochs+1):
    running_loss = 0

    for images, labels in train_loader:
        if train_on_gpu:
            images, labels = images.cuda(), labels.cuda()
        # Clear the gradients, do this because gradients are accumulated.
        optimizer.zero_grad()

        # Forward pass, get our log-probabilities.
        ps = model(images)

        # Calculate the loss with the logps and the labels.
        loss = criterion(ps, labels)

        # Turning loss back.
        loss.backward()

        # Take an update step and few the new weights.
        optimizer.step()

        running_loss += loss.item()
```


Q
A