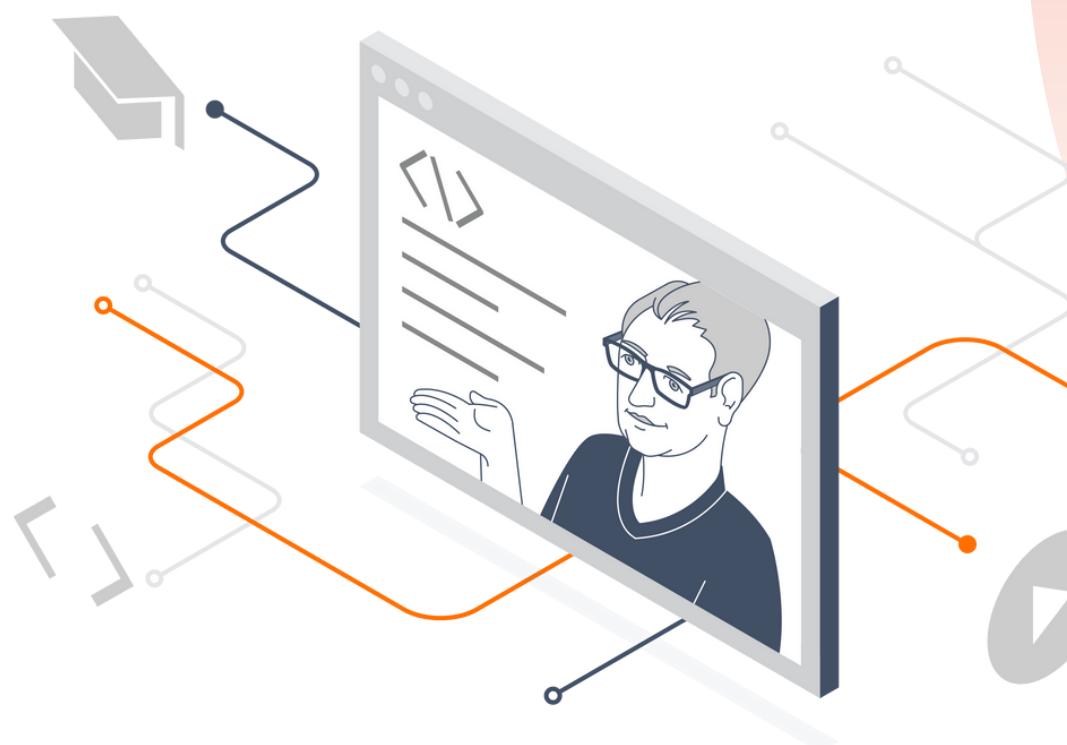


PYTORCH



Bootcamp

01



BY ANSHUMAN MISHRA



About Myself



Anshuman Mishra
Ex SDE Intern at Amazon IN
Warangal, Telangana, India
Joined 3 years ago · last seen in the past day
[Home](#) [Competitions \(4\)](#) [Datasets \(6\)](#) [Code \(33\)](#) [Discussion \(106\)](#) [Followers \(19\)](#) [Notifications](#) [Account](#) [Edit Public Profile](#)

Competitions Contributor		
Unranked		
0	0	0
No completed competitions		

Datasets Expert		
Rank 193 of 73,285		
0	1	5
Quantum Machine 8 aka ... 🕒 - a month ago 22 votes		
Amazon Mintaka 🕒 - 25 days ago 21 votes		
The Mahabharata Summ... 🕒 - a month ago 17 votes		

Notebooks Expert		
Current Rank 578 of 243,056	Highest Rank 569	
0	2	11
EDA : Amazon Mint... 🕒 - 25 days ago 31 votes		
Introduction to Molecule... 🕒 - a month ago 29 votes		
Molecular Fingerprints 🕒 - a month ago 24 votes		

Discussion Expert		
Current Rank 882 of 313,203	Highest Rank 869	
0	3	61
I became Notebooks Exp... 🕒 - a month ago 8 votes		
Internships in AI/Machine... 🕒 - a month ago 6 votes		
Observation T4×2 vs P10... 🕒 - 9 days ago 6 votes		



Anshuman Mishra (He/Him)
3x Expert @Kaggle | Ex-Amazon | Open Sorcerer @DeepMind @Deepchem



Deepchem Open Source Fellow

Deep Forest Sciences · Internship
Jun 2022 - Oct 2022 · 5 mos
United States

The DeepChem project aims to create high quality, open source tools for drug discovery, materials science, quantum chemistry, and biology using Machine Learning.



Literary & Debating Club, NITW

Self-employed · 2 yrs 1 mo
Warangal, Telangana, India

- Joint Secretary
- Executive

Jul 2021 - Sep 2022 · 1 yr 3 mos
Sep 2020 - Jul 2021 · 11 mos



Software Development Intern

Amazon · Internship
May 2022 - Jul 2022 · 3 mos
Hyderabad, Telangana, India

Department - IN Pay, Use Cases Tech
Worked on -



Letter of Completion.pdf



Full Stack Engineer

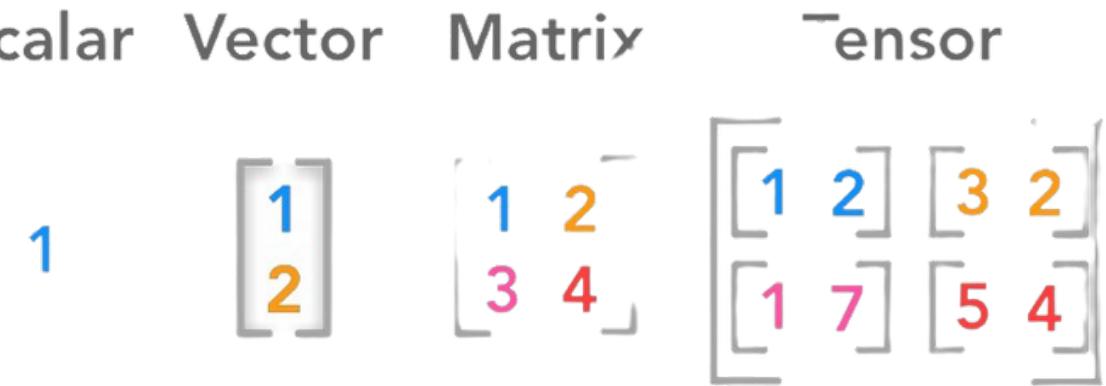
Almach Labs · Internship
Jan 2022 - Mar 2022 · 3 mos
Hyderabad, Telangana, India

Show all 9 experiences →

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):
```

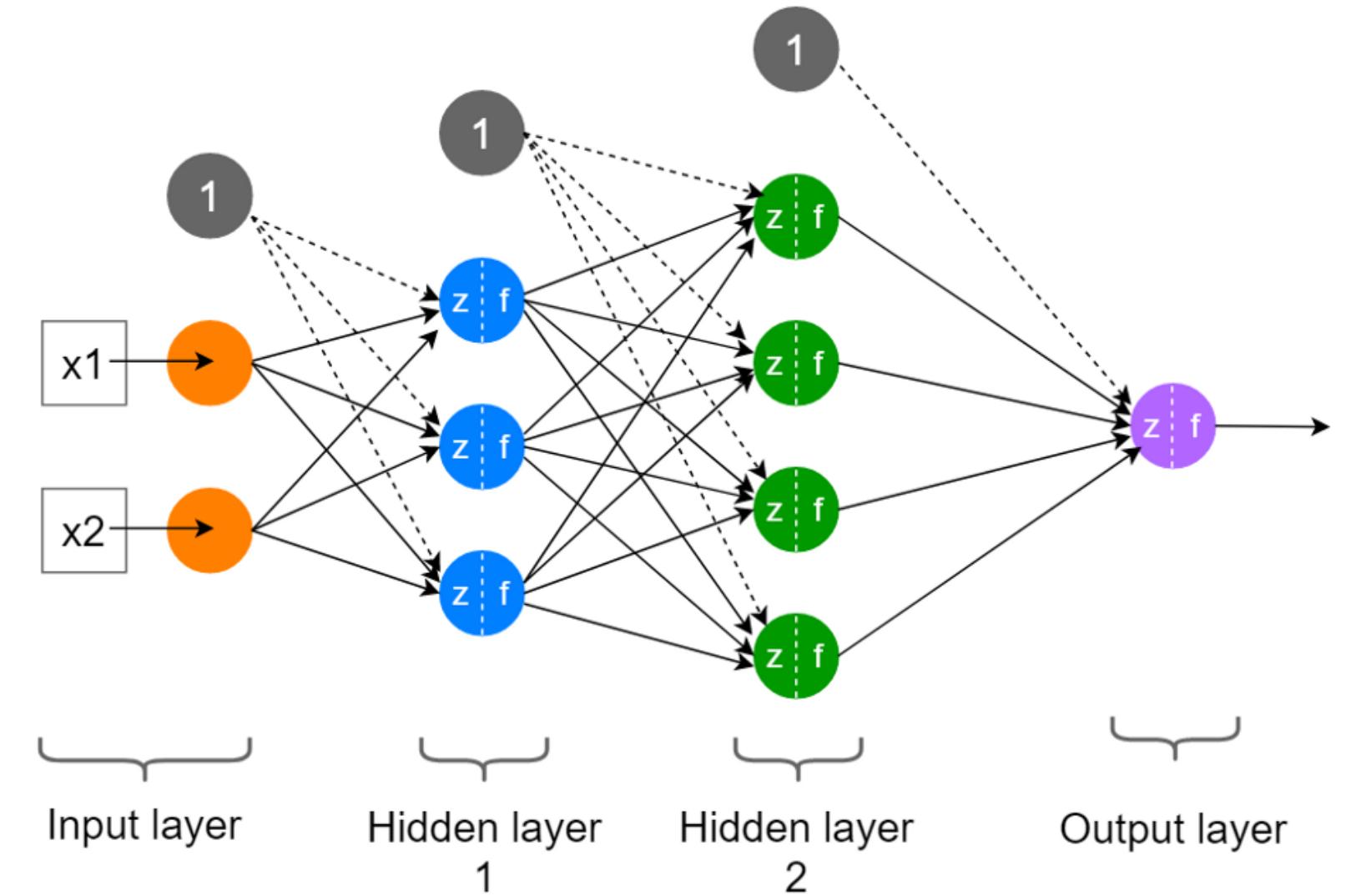
```
    self.petrol += volume  
    return self.petrol
```

```
>>> car = Vehicle(4)
```

```
>>> car.petrol
```

```
0
```

Neural Network



Link

```
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);  
}  
}
```

C++

```
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
```

PyTorch

Let's run it on kaggle

MODEL

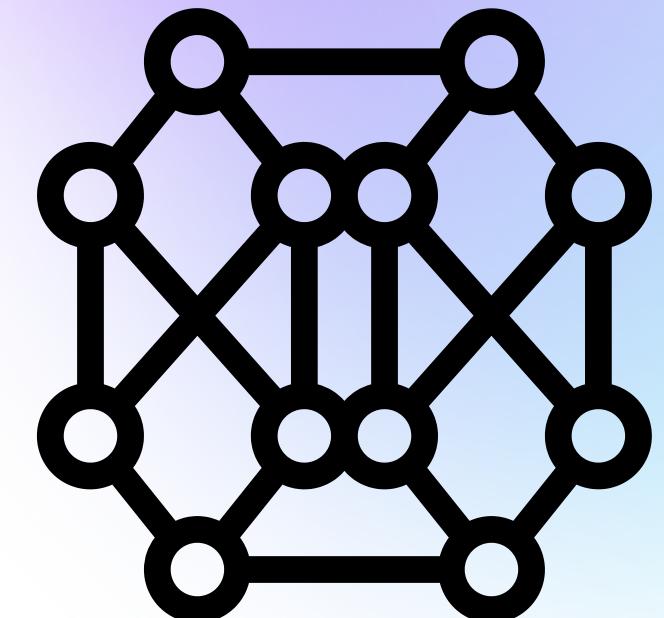
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        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 (cuda or 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 logs 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