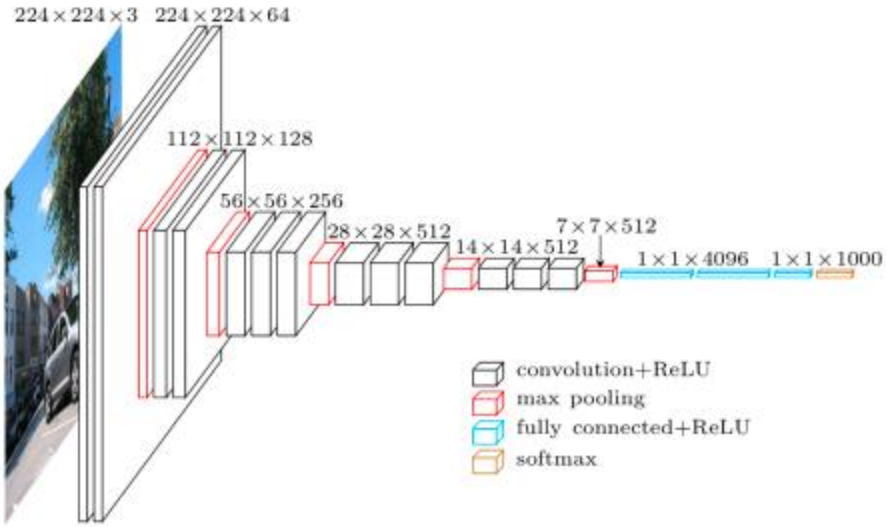
Architecture



Architecture of VGG16

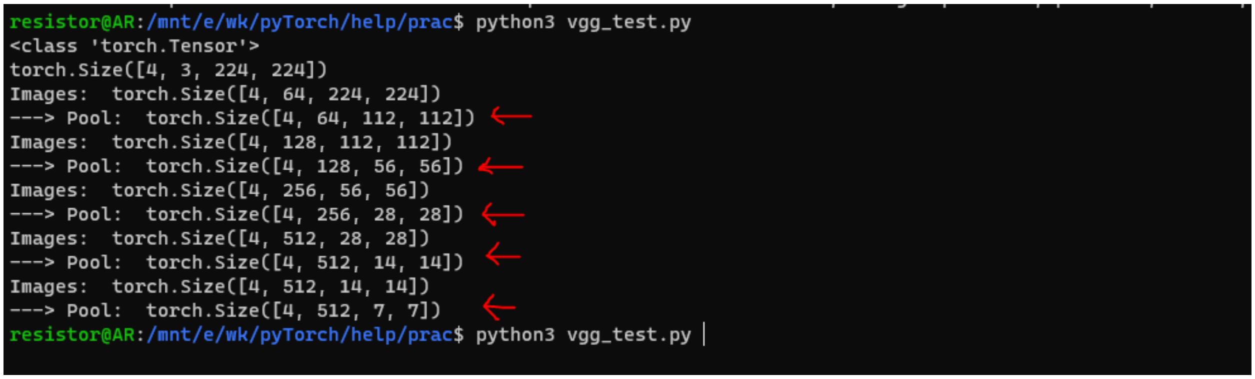
Dataset

- Cat & Dog Dataset -> https://www.kaggle.com/datasets/biaiscience/dogs-vs-cats
- Original Dataset -> (25k, 15k) (train, test)
- Preprocessed Dataset (20k) -> 5k even distributed (2.5k cats, 2.5dogs)
- P.Dataset split -> (80%, 20%) (train, test) => ([4k cats, 4k dogs], [1kcats, 1kdogs])

Files

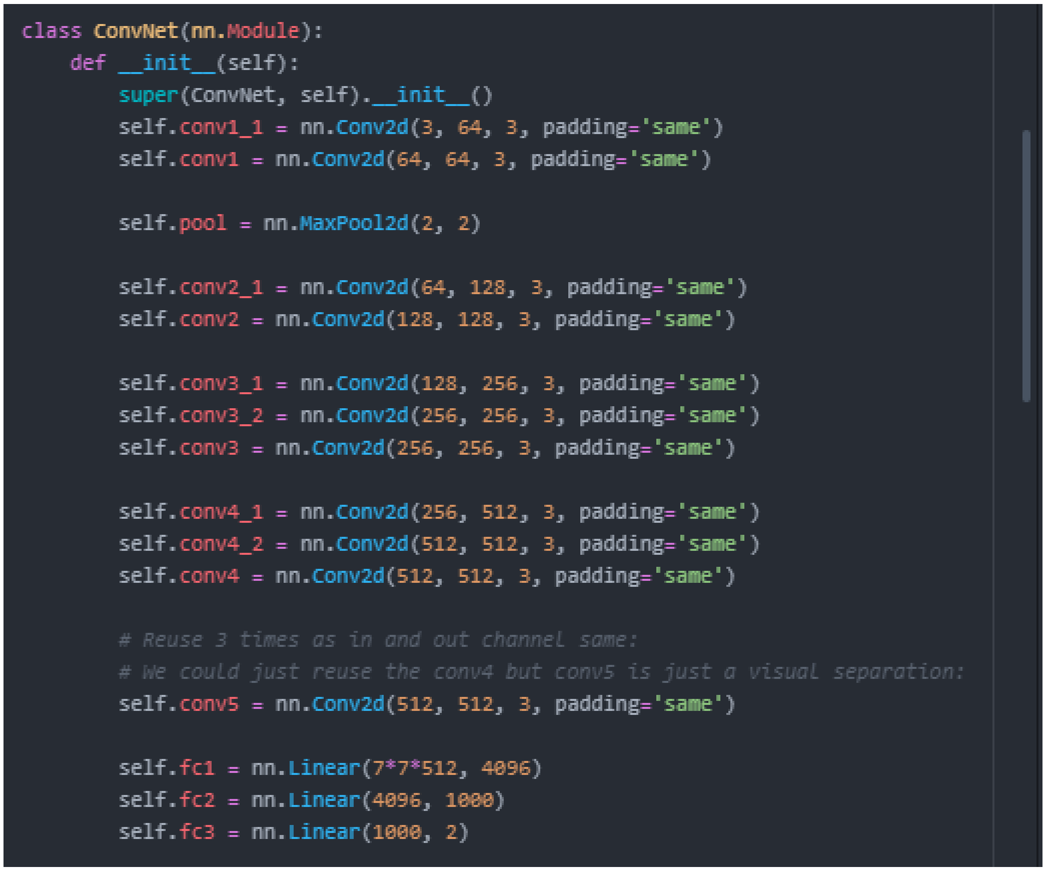
Vgg.py	(actual model file)
Vgg_test.py	(to check/test img demission)
vgg_scriptData	(to preprocess dataset) [ONE TIME RUN]
○ divided the cat and dogs in separate folders	dataSeparator()
○ train and test split them	splitCustomDataset()

Vgg_test.py

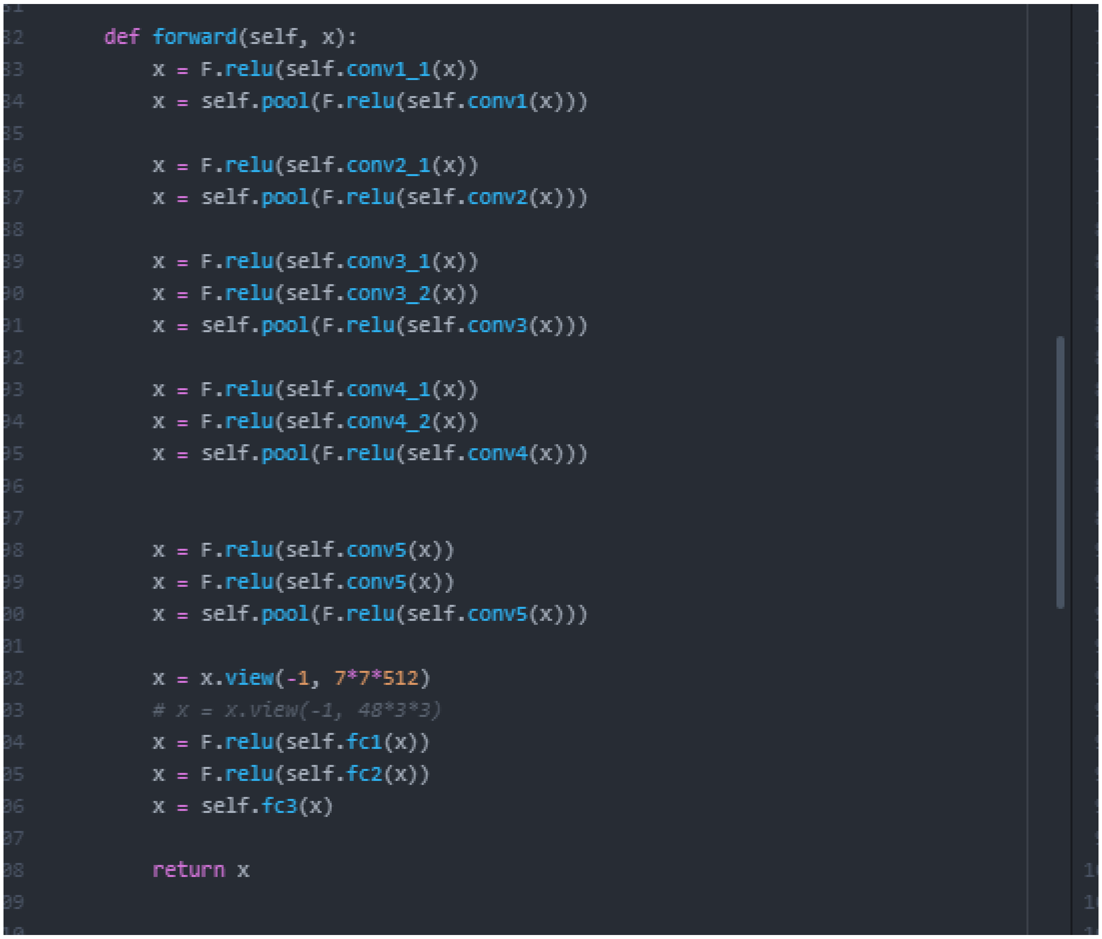


```
resistor@AR:/mnt/e/wk/pyTorch/help/prac$ python3 vgg_test.py
<class 'torch.Tensor'>
torch.Size([4, 3, 224, 224])
Images:  torch.Size([4, 64, 224, 224])
---> Pool:  torch.Size([4, 64, 112, 112])
Images:  torch.Size([4, 128, 112, 112])
---> Pool:  torch.Size([4, 128, 56, 56])
Images:  torch.Size([4, 256, 56, 56])
---> Pool:  torch.Size([4, 256, 28, 28])
Images:  torch.Size([4, 512, 28, 28])
---> Pool:  torch.Size([4, 512, 14, 14])
Images:  torch.Size([4, 512, 14, 14])
---> Pool:  torch.Size([4, 512, 7, 7])
resistor@AR:/mnt/e/wk/pyTorch/help/prac$ python3 vgg_test.py
```

Vgg.py (Layer and Feedforward)



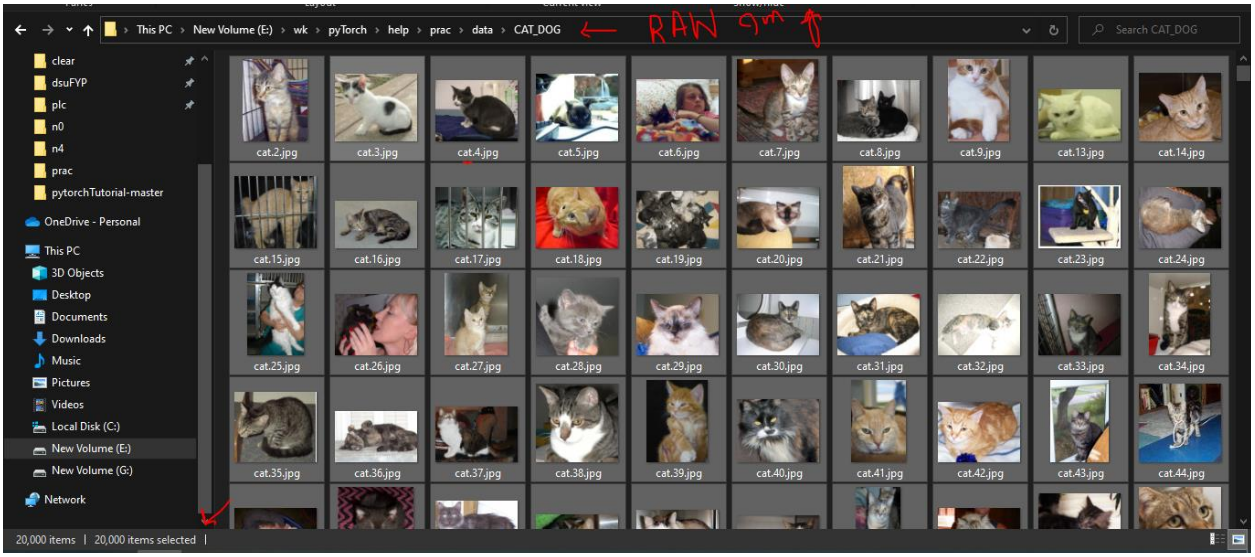
```python
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1_1 = nn.Conv2d(3, 64, 3, padding='same')
        self.conv1 = nn.Conv2d(64, 64, 3, padding='same')

        self.pool = nn.MaxPool2d(2, 2)

        self.conv2_1 = nn.Conv2d(64, 128, 3, padding='same')
        self.conv2 = nn.Conv2d(128, 128, 3, padding='same')

        self.conv3_1 = nn.Conv2d(128, 256, 3, padding='same')
        self.conv3_2 = nn.Conv2d(256, 256, 3, padding='same')
        self.conv3 = nn.Conv2d(256, 256, 3, padding='same')

        self.conv4_1 = nn.Conv2d(256, 512, 3, padding='same')
        self.conv4_2 = nn.Conv2d(512, 512, 3, padding='same')
        self.conv4 = nn.Conv2d(512, 512, 3, padding='same')

        # Reuse 3 times as in and out channel same:
        # We could just reuse the conv4 but conv5 is just a visual separation:
        self.conv5 = nn.Conv2d(512, 512, 3, padding='same')

        self.fc1 = nn.Linear(7*7*512, 4096)
        self.fc2 = nn.Linear(4096, 1000)
        self.fc3 = nn.Linear(1000, 2)
```



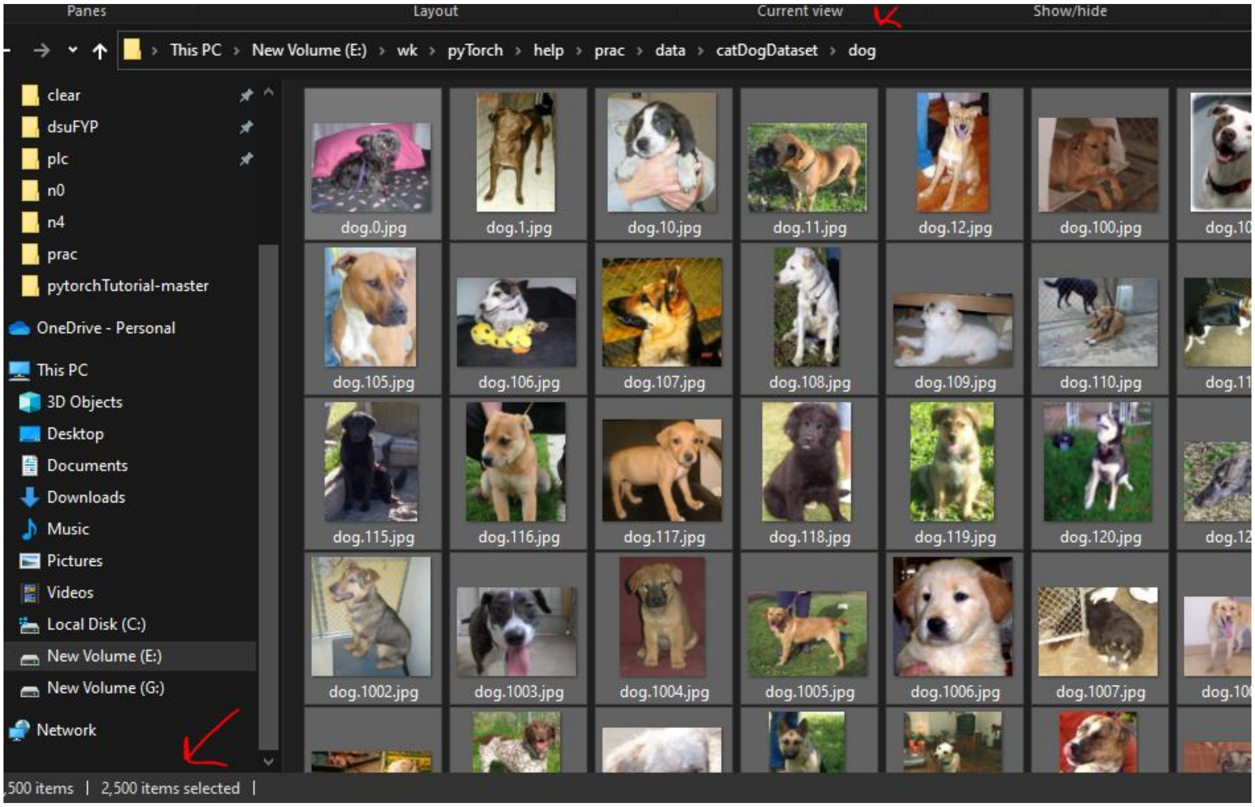
```
    def forward(self, x):
        x = F.relu(self.conv1_1(x))
        x = self.pool(F.relu(self.conv1(x)))

        x = F.relu(self.conv2_1(x))
        x = self.pool(F.relu(self.conv2(x)))

        x = F.relu(self.conv3_1(x))
        x = F.relu(self.conv3_2(x))
        x = self.pool(F.relu(self.conv3(x)))

        x = F.relu(self.conv4_1(x))
        x = F.relu(self.conv4_2(x))
        x = self.pool(F.relu(self.conv4(x)))


        x = F.relu(self.conv5(x))
        x = F.relu(self.conv5(x))
        x = self.pool(F.relu(self.conv5(x)))

        x = x.view(-1, 7*7*512)
        # x = x.view(-1, 48*3*3)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)

        return x
```

Dataset View

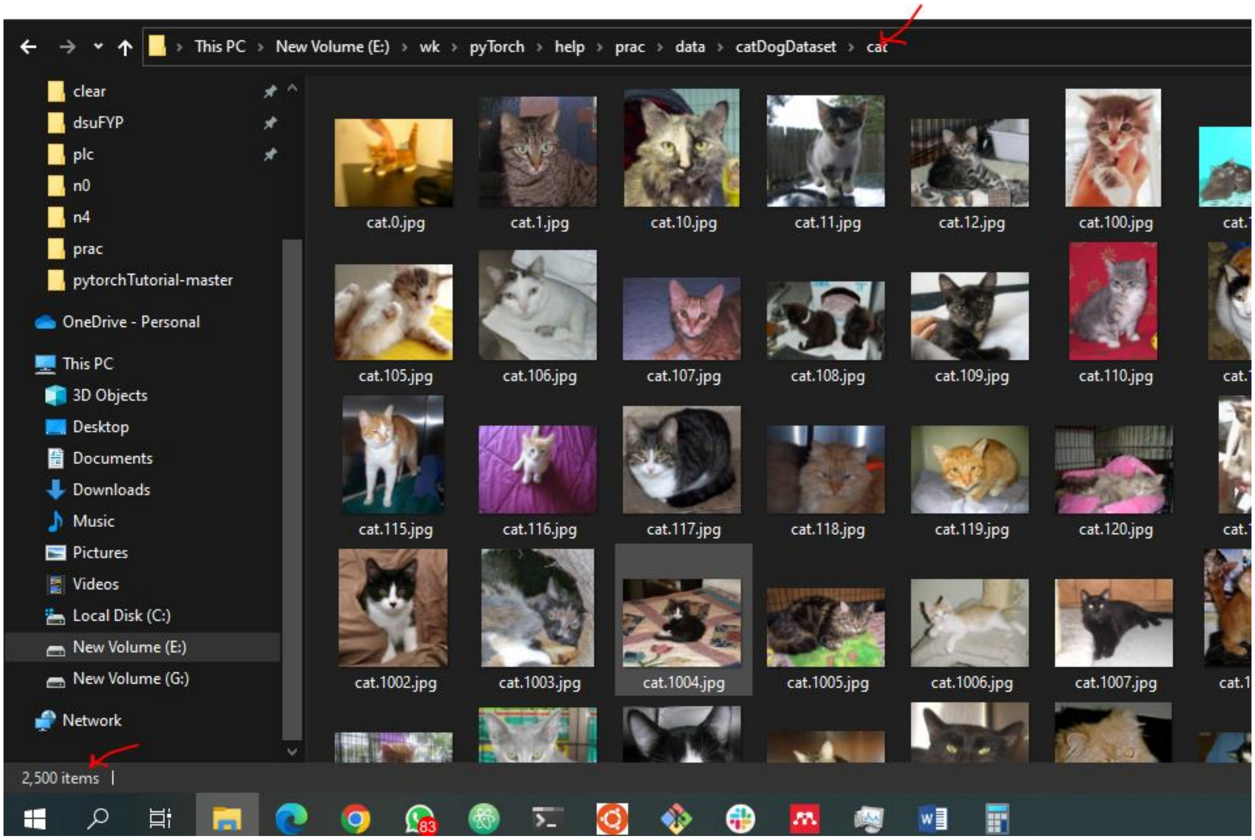
<u>Raw Dataset</u>



<u>Preprocessed</u> (5k)

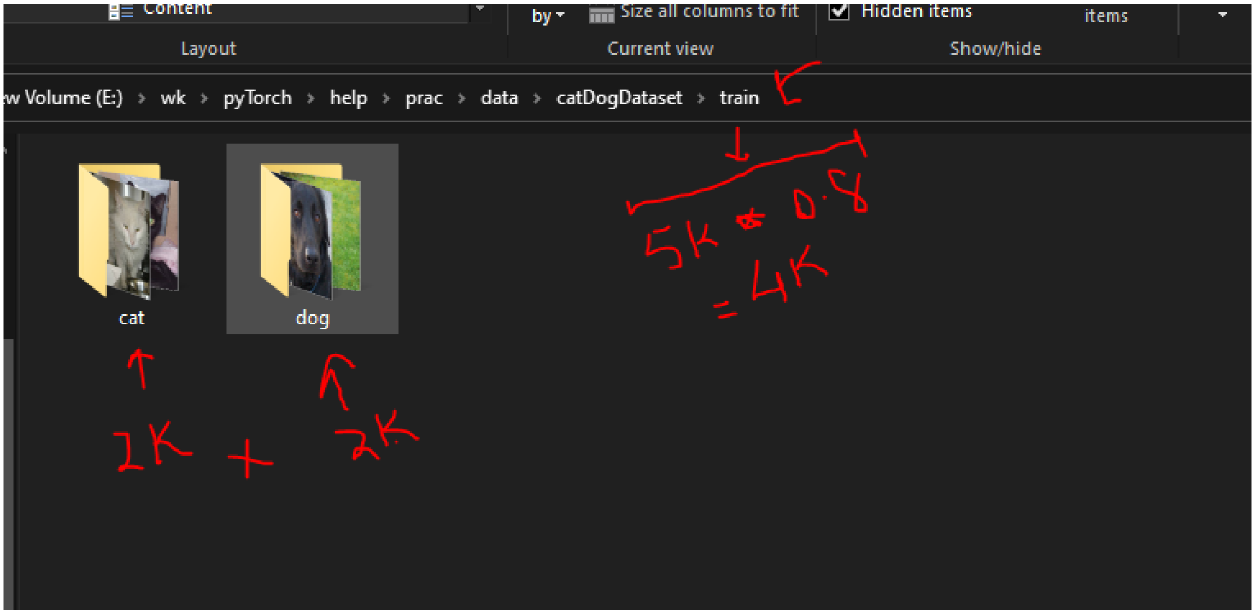


Even Distribution (2.5k) {DOG]

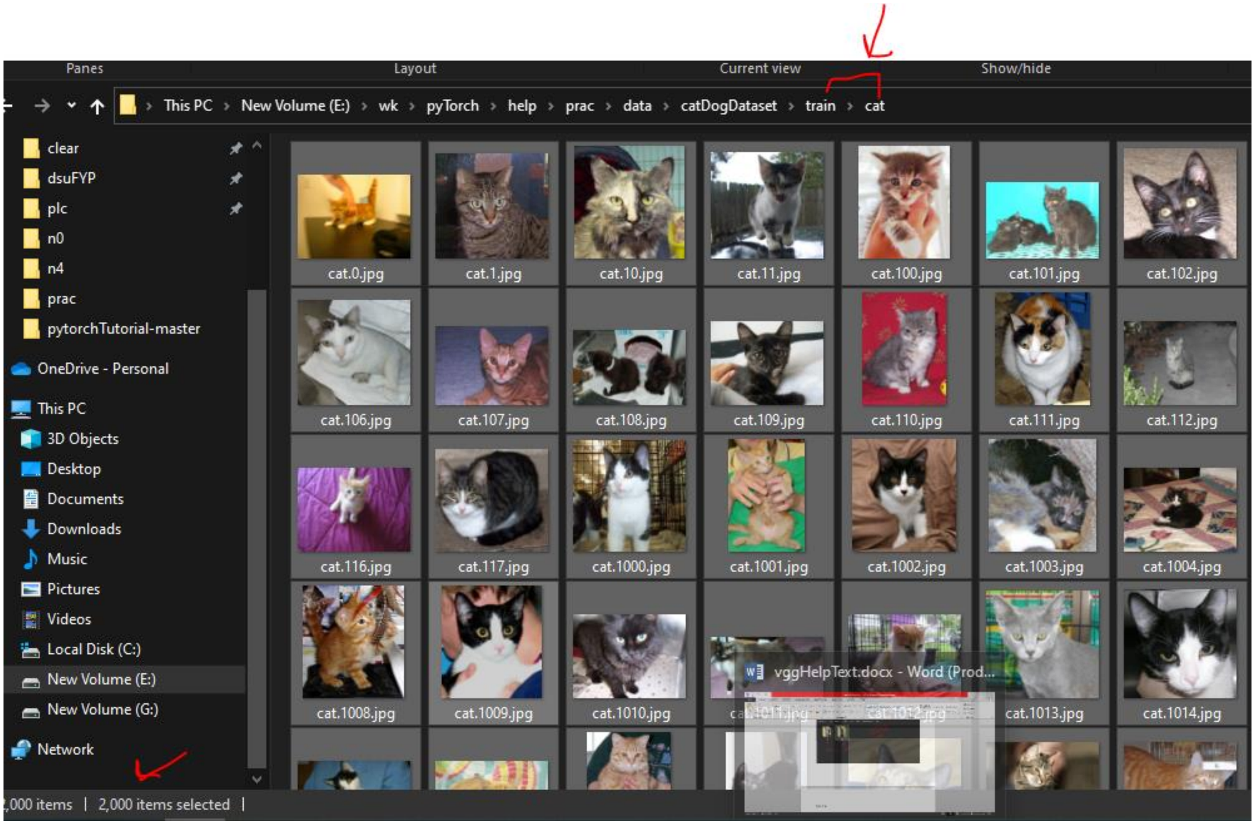


Even Distribution (2.5k) {CAT]

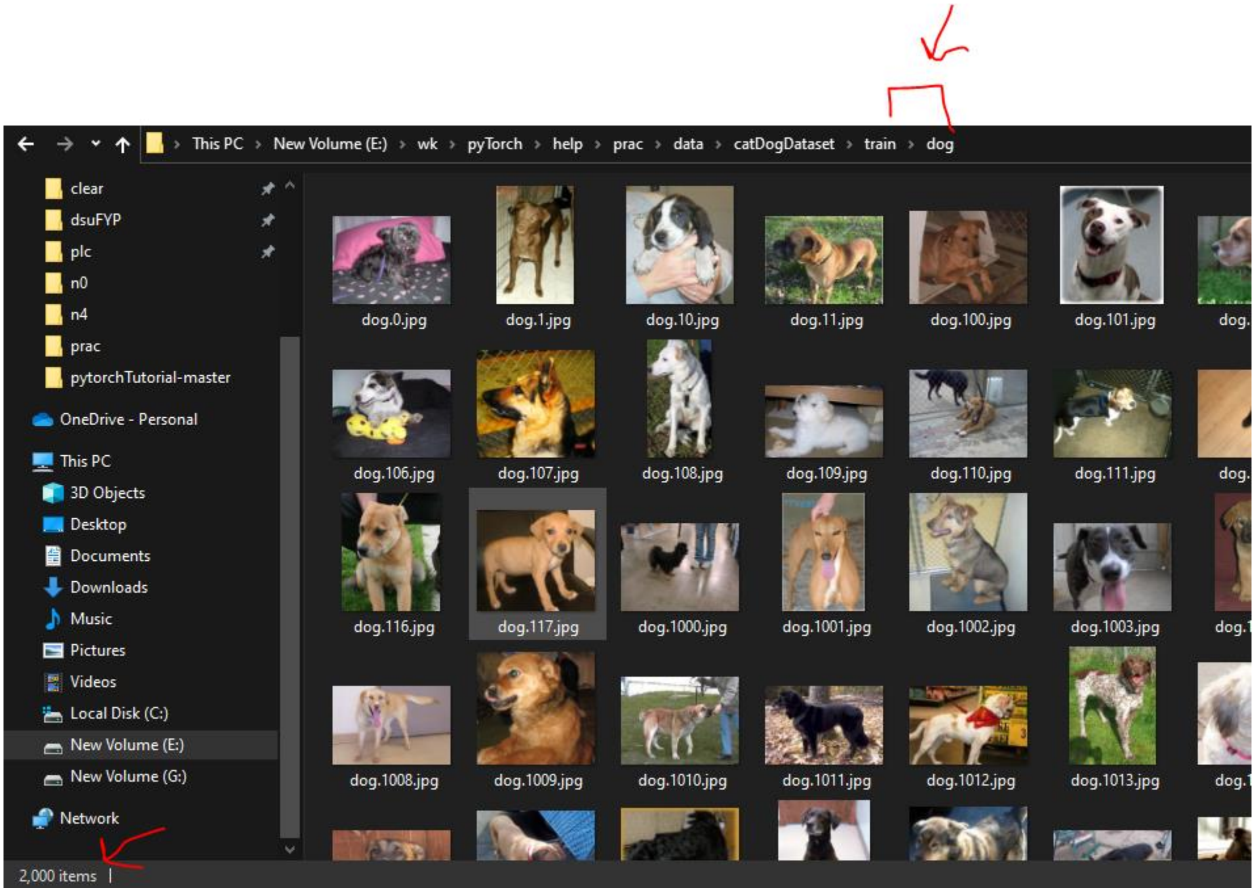
Train & Test Distribution



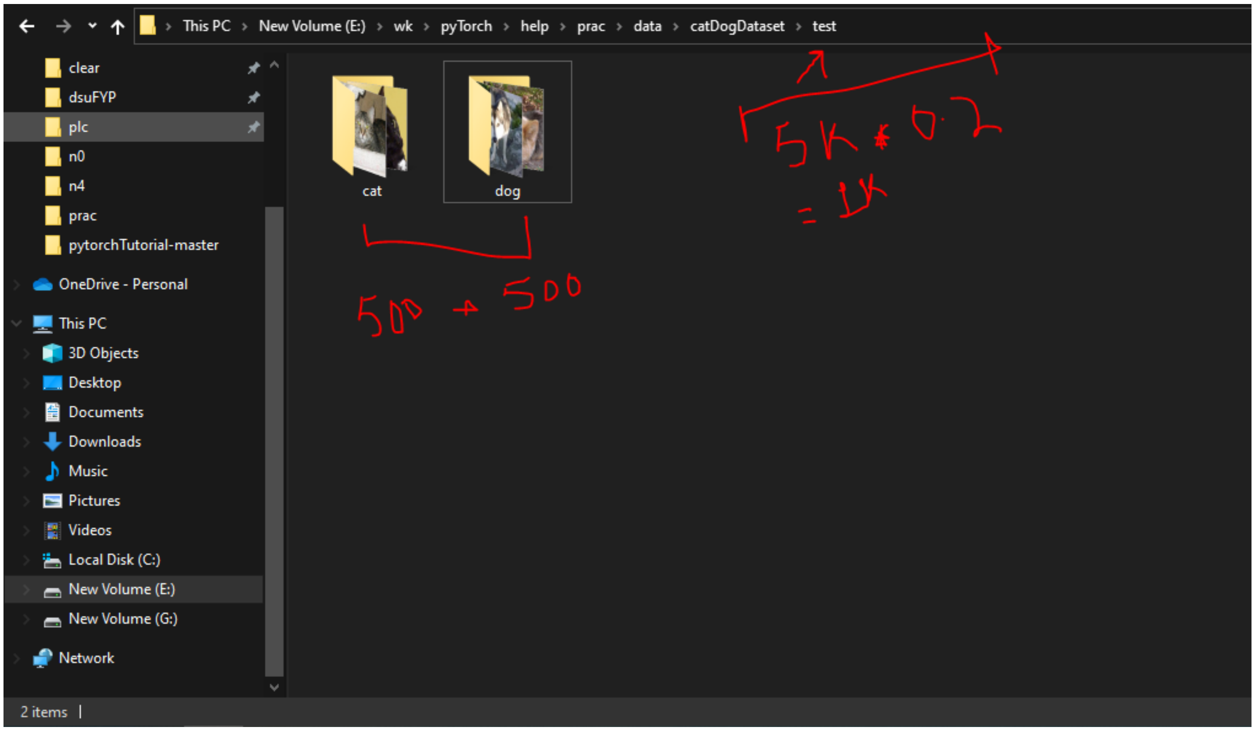
Cat Train



Dog Train


This PC › New Volume (E:) › wk › pyTorch › help › prac › data › catDogDataset › train › dog
clear
dsuFYP
plc
n0
n4
prac
pytorchTutorial-master
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (E:)
New Volume (G:)
Network
dog.0.jpg
dog.1.jpg
dog.10.jpg
dog.11.jpg
dog.100.jpg
dog.101.jpg
dog.
dog.106.jpg
dog.107.jpg
dog.108.jpg
dog.109.jpg
dog.110.jpg
dog.111.jpg
dog.
dog.116.jpg
dog.117.jpg
dog.1000.jpg
dog.1001.jpg
dog.1002.jpg
dog.1003.jpg
dog.1
dog.1008.jpg
dog.1009.jpg
dog.1010.jpg
dog.1011.jpg
dog.1012.jpg
dog.1013.jpg
dog.1
2,000 items

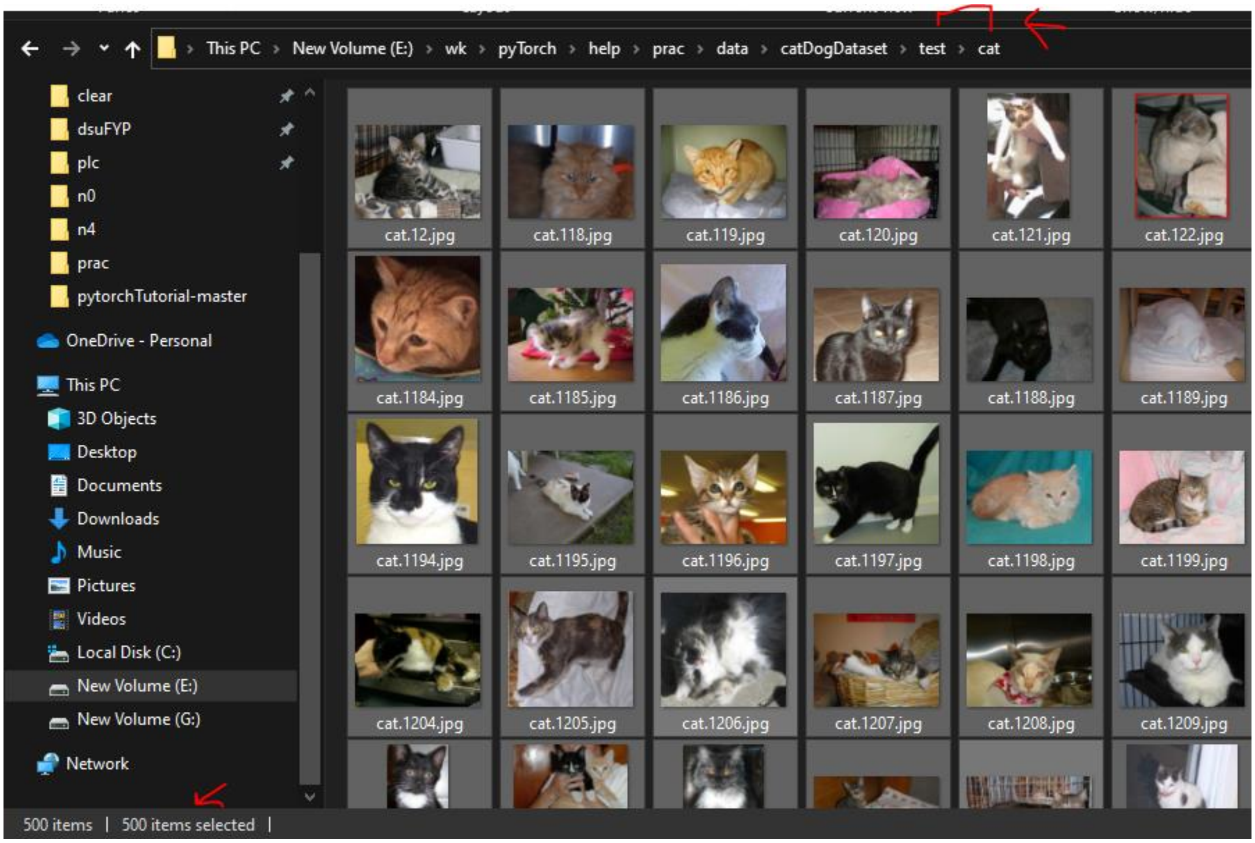
Test



5k * 0.2 = 1k

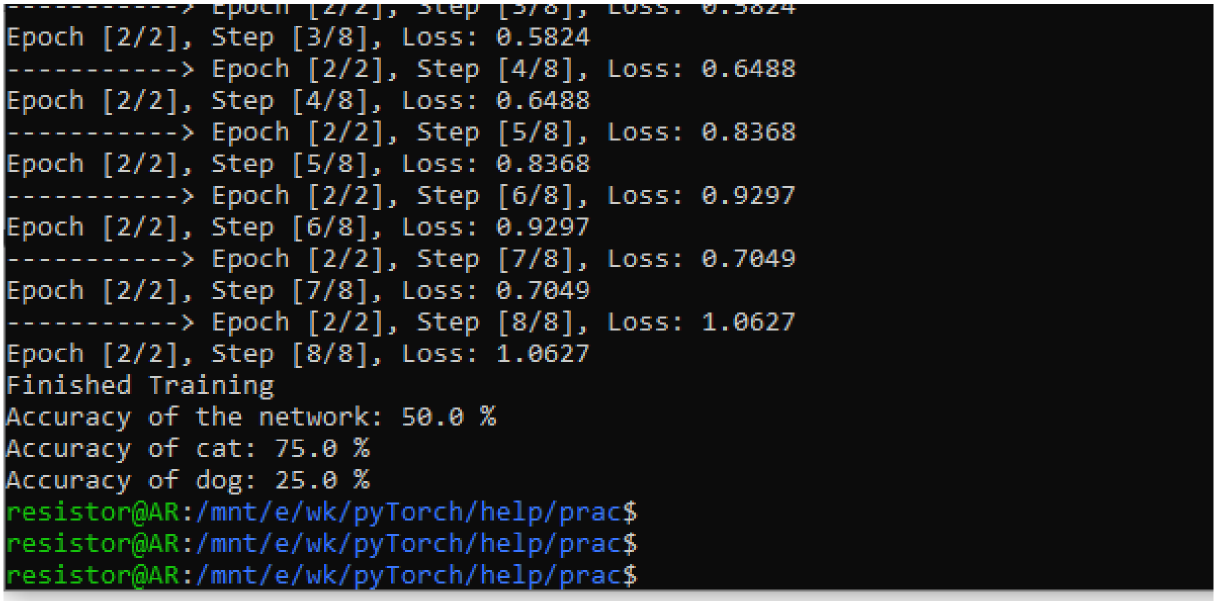
500 + 500

2 items



500 items | 500 items selected |

Model Output



```
-----------> Epoch [2/2], Step [3/8], Loss: 0.5824
Epoch [2/2], Step [3/8], Loss: 0.5824
-----------> Epoch [2/2], Step [4/8], Loss: 0.6488
Epoch [2/2], Step [4/8], Loss: 0.6488
-----------> Epoch [2/2], Step [5/8], Loss: 0.8368
Epoch [2/2], Step [5/8], Loss: 0.8368
-----------> Epoch [2/2], Step [6/8], Loss: 0.9297
Epoch [2/2], Step [6/8], Loss: 0.9297
-----------> Epoch [2/2], Step [7/8], Loss: 0.7049
Epoch [2/2], Step [7/8], Loss: 0.7049
-----------> Epoch [2/2], Step [8/8], Loss: 1.0627
Epoch [2/2], Step [8/8], Loss: 1.0627
Finished Training
Accuracy of the network: 50.0 %
Accuracy of cat: 75.0 %
Accuracy of dog: 25.0 %
resistor@AR:/mnt/e/wk/pyTorch/help/prac$
resistor@AR:/mnt/e/wk/pyTorch/help/prac$
resistor@AR:/mnt/e/wk/pyTorch/help/prac$
```