07

January 23, 2024

```python
[1]: %pylab inline
     import torch
     import sys
     sys.path.append('..')
     sys.path.append('../..')
     from data import load
     train_data, train_label = load.get_dogs_and_cats_data(resize=(32,32),␣
       ↪n_images=10)
     device = torch.device('cuda') if torch.cuda.is_available() else torch.
       ↪device('cpu')
     print('device = ', device)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
device =  cuda
```

```python
[2]: class ConvNet(torch.nn.Module):
         def __init__(self, layers=[], n_input_channels=3, kernel_size=3, stride=2):
             super().__init__()
             L = []
             c = n_input_channels
             for l in layers:
                 L.append(torch.nn.Conv2d(c, l, kernel_size, padding=(kernel_size-1)/
       ↪/2, stride=stride))
                 L.append(torch.nn.ReLU())
                 c = l
             L.append(torch.nn.Conv2d(c, 1, kernel_size=1))
             self.layers = torch.nn.Sequential(*L)

         def forward(self, x):
             return self.layers(x).mean([1,2,3])

     net = ConvNet([32,64])
```

```python
[3]: print( train_data[:1].shape )
     print( net(train_data[:1]).shape )
```

```
torch.Size([1, 3, 32, 32])
```

```
torch.Size([1])
```

[4]:
```python
net2 = ConvNet([32,64,128])
print( net2(train_data[:1]).shape )
```

```
torch.Size([1])
```

[5]:
```python
class ConvNet2(torch.nn.Module):
    def __init__(self, layers=[], n_input_channels=3, kernel_size=3, stride=2):
        super().__init__()
        L = []
        c = n_input_channels
        for l in layers:
            L.append(torch.nn.Conv2d(c, l, kernel_size, padding=(kernel_size-1)/
    /2))
            L.append(torch.nn.ReLU())
            L.append(torch.nn.MaxPool2d(3, padding=1, stride=stride))
            c = l
        L.append(torch.nn.Conv2d(c, 1, kernel_size=1))
        self.layers = torch.nn.Sequential(*L)

    def forward(self, x):
        return self.layers(x).mean([1,2,3])

net3 = ConvNet2([32,64,128])
```

[6]:
```python
print( net3(train_data[:1]).shape )
```

```
torch.Size([1])
```

[7]:
```python
%load_ext tensorboard
import tempfile
log_dir = tempfile.mkdtemp()
%tensorboard --logdir {log_dir} --reload_interval 1
```

```
<IPython.core.display.HTML object>
```

[8]:
```python
from util import train
train.train(net2, batch_size=128, resize=(32,32), log_dir=log_dir+'/net2',
    device=device, n_epochs=100)
```

```
WARNING:root:loading dataset
WARNING:root:loading done

  0%|          | 0/100 [00:00<?, ?it/s]
```

[9]:
```python
from util import train
```

```
train.train(net3, batch_size=128, resize=(32,32), log_dir=log_dir+'/net3',␣
 ↪device=device, n_epochs=100)
```

WARNING:root:loading dataset
WARNING:root:loading done

0%| | 0/100 [00:00<?, ?it/s]

[ ]: