UDACITY

< Return to "Computer Vision Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Facial Keypoint Detection

REVIEW

CODE REVIEW 1

HISTORY

models.py 1

```
## TODO: define the convolutional neural network architecture

import torch
import torch.nn as nn
import torch.nn.functional as F
# can use the below import should you choose to initialize the weights of your
import torch.nn.init as I


class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()

        ## TODO: Define all the layers of this CNN, the only requirements are:
        ## 1. This network takes in a square (same width and height), grayscale
        ## 2. It ends with a linear layer that represents the keypoints
        ## it's suggested that you make this last layer output 136 values, 2 fo

        # As an example, you've been given a convolutional layer, which you may
        # 1 input image channel (grayscale), 32 output channels/feature maps, !

        self.conv1 = nn.Conv2d(1, 32, 5)
        # output size = (W-F)/S +1 = (224-5)/1 + 1 = 220
        self.pool1 = nn.MaxPool2d(2, 2)
        # 220/2 = 110  the output Tensor for one image, will have the #dimensio
```

```
27          self.conv2 = nn.Conv2d(32,64,3)
28          # output size = (W-F)/S +1 = (110-3)/1 + 1 = 108
29          self.pool2 = nn.MaxPool2d(2, 2)
```

SUGGESTION

Max pooling is a simple down-sampling operation and I am sure you know what it does. And `nn.MaxPool2` do not have to declare multiple instances of it, you can just reuse the same object multiple times. 😉

Same goes for the dropout function (for a given dropping probability, of course).

```
30          #108/2=54   the output Tensor for one image, will have the #dimensions
31          self.conv3 = nn.Conv2d(64,128,3)
32          # output size = (W-F)/S +1 = (54-3)/1 + 1 = 52
33          self.pool3 = nn.MaxPool2d(2, 2)
34          #52/2=26    the output Tensor for one image, will have the #dimensions
35          self.conv4 = nn.Conv2d(128,256,3)
36          # output size = (W-F)/S +1 = (26-3)/1 + 1 = 24
37          self.pool4 = nn.MaxPool2d(2, 2)
38          #24/2=12   the output Tensor for one image, will have the #dimensions:
39          self.conv5 = nn.Conv2d(256,512,1)
40          # output size = (W-F)/S +1 = (12-1)/1 + 1 = 12
41          self.pool5 = nn.MaxPool2d(2, 2)
42          #12/2=6    the output Tensor for one image, will have the #dimensions:
43          #Linear Layer
44          self.fc1 = nn.Linear(512*6*6, 1024)
45          self.fc2 = nn.Linear(1024, 136)
46          #self.fc3 = nn.Linear(136, 136)
47          # dropouts
48          self.drop1 = nn.Dropout(p = 0.3)
49          self.drop2 = nn.Dropout(p = 0.3)
50          self.drop3 = nn.Dropout(p = 0.3)
51          self.drop4 = nn.Dropout(p = 0.4)
52          self.drop5 = nn.Dropout(p = 0.3)
53          self.drop6 = nn.Dropout(p = 0.4)
54          #self.drop7 = nn.Dropout(p = 0.5)
55
56
57          ## Note that among the layers to add, consider including:
58          # maxpooling layers, multiple conv layers, fully-connected layers, and
59                                  # Check out list of layers in pyTorch: batch norm
60
61
62
63      def forward(self, x):
64          ## TODO: Define the feedforward behavior of this model
65          ## x is the input image and, as an example, here you may choose to incl
66          x = self.pool1(F.relu(self.conv1(x)))
67          x = self.drop1(x)
68          x = self.pool2(F.relu(self.conv2(x)))
69          x = self.drop2(x)
70          x = self.pool3(F.relu(self.conv3(x)))
71          x = self.drop3(x)
72          x = self.pool4(F.relu(self.conv4(x)))
73          x = self.drop4(x)
74          x = self.pool5(F.relu(self.conv5(x)))
75          x = self.drop5(x)
76          x = x.view(x.size(0), -1)
```

```
77          x = F.relu(self.fc1(x))
78          x = self.drop6(x)
79          x = self.fc2(x)
80          #x = self.drop7(x)
81          #x = self.fc3(x)
82          # a modified x, having gone through all the layers of your model, shou
83          return x
84
85          # a modified x, having gone through all the layers of your model, shou
86          #return x
87
```

▸ detector_architectures/haarcascade_smile.xml

▸ workspace_utils.py

▸ filelist.txt

▸ data_load.py

RETURN TO PATH