image_classifier_nn-Copy5

November 22, 2020

```
[2]: import torchvision.datasets as dset
```

```
[3]: path2data="COCO/train2017"
     #path2json="./data/annotations/instances_train2017.json"
     path2json="COCO/anno2017/instances_train2017.json"
```

```
[1]: path2data="COCO/val2017"
     #path2json="./data/annotations/instances_train2017.json"
     path2json="COCO/anno2017/instances_val2017.json"
```

```
[4]: coco_train = dset.CocoDetection(root = path2data,
                                     annFile = path2json)
```

```
loading annotations into memory…
Done (t=16.87s)
creating index…
index created!
```

```
[5]: print('Number of samples: ', len(coco_train))
```

```
Number of samples:  118287
```

```
[6]: img, target=coco_train[0]
     print(img.size)
```

```
(640, 480)
```

```
[7]: img
```

```
[7]:
```

0.1 Define Neural Network

[65]:
```
#Loss Scores for Training Iterations
```

```
Epoch: 0/1
Iteration: 1/5000, Loss: 2.9618141651153564
Iteration: 2/5000, Loss: 3.205232620239258
Iteration: 3/5000, Loss: 2.8787338733673096
Iteration: 4/5000, Loss: 2.8303253650665283
Iteration: 5/5000, Loss: 3.2098135948181152
Iteration: 6/5000, Loss: 2.2872207164764404
Iteration: 7/5000, Loss: 2.022399663925171
Iteration: 8/5000, Loss: 1.0841790437698364
Iteration: 9/5000, Loss: 1.27898108959198
Iteration: 10/5000, Loss: 1.2382025718688965
Iteration: 11/5000, Loss: 0.4772915840148926
Iteration: 12/5000, Loss: 5.285684108734131
Iteration: 13/5000, Loss: 1.8661919832229614
Iteration: 14/5000, Loss: 1.6353682279586792
Iteration: 15/5000, Loss: 1.3749465942382812
Iteration: 16/5000, Loss: 0.9085026979446411
```

```
Iteration: 17/5000, Loss: 0.7824200391769409
Iteration: 18/5000, Loss: 0.36797818541526794
Iteration: 19/5000, Loss: 0.35762733221054077
Iteration: 20/5000, Loss: 1.1160602569580078
Epoch: 0, Loss: 0.5877184271812439
Finished Training
```

[102]:
```
#Loss Scores for Larger Training Set Sample
```

```
Epoch: 0/1
Iteration: 1/5000, Loss: 3.1969902515411377
Iteration: 2/5000, Loss: 3.048896312713623
Iteration: 3/5000, Loss: 2.995964765548706
Iteration: 4/5000, Loss: 3.059705972671509
Iteration: 5/5000, Loss: 2.7820475101470947
Iteration: 6/5000, Loss: 2.7798027992248535
Iteration: 7/5000, Loss: 2.496198892593384
Iteration: 8/5000, Loss: 2.158720016479492
Iteration: 9/5000, Loss: 2.002383232116699
Iteration: 10/5000, Loss: 2.2853856086730957
Iteration: 11/5000, Loss: 1.4548333883285522
Iteration: 12/5000, Loss: 0.9626320004463196
Iteration: 13/5000, Loss: 0.665838360786438
Iteration: 14/5000, Loss: 1.7160978317260742
Iteration: 15/5000, Loss: 1.1871893405914307
Iteration: 16/5000, Loss: 0.8262783288955688
Iteration: 17/5000, Loss: 1.5516127347946167
Iteration: 18/5000, Loss: 0.9631065130233765
Iteration: 19/5000, Loss: 0.9639790058135986
Iteration: 20/5000, Loss: 0.6573706269264221
Iteration: 21/5000, Loss: 0.574190616607666
Iteration: 22/5000, Loss: 4.347209930419922
Iteration: 23/5000, Loss: 0.8141676783561707
Iteration: 24/5000, Loss: 2.0040512084960938
Iteration: 25/5000, Loss: 0.939065694808960
Iteration: 26/5000, Loss: 0.9551538228988647
Iteration: 27/5000, Loss: 2.797743082046509
Iteration: 28/5000, Loss: 1.6325498819351196
Iteration: 29/5000, Loss: 1.8107563257217407
Iteration: 30/5000, Loss: 1.4131027460098267
Iteration: 31/5000, Loss: 1.2638797760009766
Iteration: 32/5000, Loss: 1.7885191440582275
Iteration: 33/5000, Loss: 1.0561878681182861
Iteration: 34/5000, Loss: 1.3286726474761963
Iteration: 35/5000, Loss: 1.8243672847747803
Iteration: 36/5000, Loss: 1.619247317314148
Iteration: 37/5000, Loss: 0.9300145506858826
Iteration: 38/5000, Loss: 1.0475451946258545
```

```
Iteration: 39/5000, Loss: 2.481318712234497
Iteration: 40/5000, Loss: 0.6941351890563965
Iteration: 41/5000, Loss: 1.1117371320724487
Iteration: 42/5000, Loss: 0.5555919408798218
Iteration: 43/5000, Loss: 0.3468267023563385
Iteration: 44/5000, Loss: 0.6712538599967957
Iteration: 45/5000, Loss: 0.27915430068969727
Iteration: 46/5000, Loss: 2.6284914016723633
Iteration: 47/5000, Loss: 0.3973613679409027
Iteration: 48/5000, Loss: 0.7288219332695007
Iteration: 49/5000, Loss: 0.9074833393096924
Iteration: 50/5000, Loss: 0.5650018453598022
Iteration: 51/5000, Loss: 0.9912083148956299
Iteration: 52/5000, Loss: 3.017937421798706
Iteration: 53/5000, Loss: 1.7051355838775635
Iteration: 54/5000, Loss: 0.7771546244621277
Iteration: 55/5000, Loss: 2.2815308570861816
Iteration: 56/5000, Loss: 1.18510103225708
Iteration: 57/5000, Loss: 1.3771591186523438
Iteration: 58/5000, Loss: 0.8936006426811218
Iteration: 59/5000, Loss: 0.8441560864448547
Iteration: 60/5000, Loss: 0.5482197999954224
Iteration: 61/5000, Loss: 1.444069504737854
Iteration: 62/5000, Loss: 1.6406787633895874
Iteration: 63/5000, Loss: 0.6339568495750427
Iteration: 64/5000, Loss: 0.9337686896324158
Iteration: 65/5000, Loss: 0.5671324729919434
Iteration: 66/5000, Loss: 0.4313022494316101
Iteration: 67/5000, Loss: 0.4254782497882843
Iteration: 68/5000, Loss: 3.4131579399108887
Iteration: 69/5000, Loss: 0.7743933796882629
Iteration: 70/5000, Loss: 0.5037552714347839
Iteration: 71/5000, Loss: 0.46221423149108887
Iteration: 72/5000, Loss: 1.1179466247558594
Iteration: 73/5000, Loss: 1.2796998023986816
Iteration: 74/5000, Loss: 0.6452022194862366
Iteration: 75/5000, Loss: 0.6985913515090942
Iteration: 76/5000, Loss: 0.7451249361038208
Iteration: 77/5000, Loss: 0.43105465173721313
Iteration: 78/5000, Loss: 1.0797446966171265
Iteration: 79/5000, Loss: 0.6977296471595764
Iteration: 80/5000, Loss: 1.3430132865905762
Iteration: 81/5000, Loss: 2.5854692459106445
Iteration: 82/5000, Loss: 0.9883027672767639
Iteration: 83/5000, Loss: 1.039892554283142
Iteration: 84/5000, Loss: 1.1419799327850342
Iteration: 85/5000, Loss: 0.5268951058387756
Iteration: 86/5000, Loss: 0.4134770929813385
```

```
Iteration: 87/5000, Loss: 0.9205060601234436
Iteration: 88/5000, Loss: 0.4309234619140625
Iteration: 89/5000, Loss: 0.48836788535118103
Iteration: 90/5000, Loss: 0.14097820222377777
Iteration: 91/5000, Loss: 1.4567723274230957
Iteration: 92/5000, Loss: 0.488818496465683
Iteration: 93/5000, Loss: 0.7694180011749268
Iteration: 94/5000, Loss: 1.4862366914749146
Iteration: 95/5000, Loss: 1.7529606819152832
Iteration: 96/5000, Loss: 0.9300702214241028
Iteration: 97/5000, Loss: 0.8990601301193237
Iteration: 98/5000, Loss: 0.46202704310417175
Iteration: 99/5000, Loss: 1.820399284362793
Iteration: 100/5000, Loss: 1.1438050270080566
Iteration: 101/5000, Loss: 0.3605959117412567
Iteration: 102/5000, Loss: 0.368846595287323
Iteration: 103/5000, Loss: 0.4403396248817444
Iteration: 104/5000, Loss: 4.811018466949463
Iteration: 105/5000, Loss: 0.37660080194473267
Iteration: 106/5000, Loss: 0.7324421405792236
Iteration: 107/5000, Loss: 1.6390907764434814
Iteration: 108/5000, Loss: 0.5505133271217346
Iteration: 109/5000, Loss: 0.6105875968933105
Iteration: 110/5000, Loss: 1.000941276550293
Iteration: 111/5000, Loss: 0.5455852746963501
Iteration: 112/5000, Loss: 0.3428376317024231
Iteration: 113/5000, Loss: 2.64473557472229
Iteration: 114/5000, Loss: 0.47712835669517517
Iteration: 115/5000, Loss: 0.8245223760604858
Iteration: 116/5000, Loss: 1.0733468532562256
Iteration: 117/5000, Loss: 0.7143657803535461
Iteration: 118/5000, Loss: 1.732537865638733
Iteration: 119/5000, Loss: 1.4315664768218994
Iteration: 120/5000, Loss: 0.9352352023124695
Iteration: 121/5000, Loss: 0.7465447783470154
Iteration: 122/5000, Loss: 0.9684396982192993
Iteration: 123/5000, Loss: 1.098712682723999
Iteration: 124/5000, Loss: 0.4168340265750885
Iteration: 125/5000, Loss: 0.9253132939338684
Iteration: 126/5000, Loss: 0.4972649812698364
Iteration: 127/5000, Loss: 0.7378990650177002
Iteration: 128/5000, Loss: 1.005717158317566
Iteration: 129/5000, Loss: 1.3192970752716064
Iteration: 130/5000, Loss: 0.5566829442977905
Iteration: 131/5000, Loss: 1.298157811164856
Iteration: 132/5000, Loss: 0.7775055766105652
Iteration: 133/5000, Loss: 0.4205437898635864
Iteration: 134/5000, Loss: 1.2194243669509888
```

```
Iteration: 135/5000, Loss: 0.2486470341682434
Iteration: 136/5000, Loss: 0.37752822041511536
Iteration: 137/5000, Loss: 0.28125983476638794
Iteration: 138/5000, Loss: 1.3293715715408325
Iteration: 139/5000, Loss: 0.3680635690689087
Iteration: 140/5000, Loss: 0.2395775020122528
Iteration: 141/5000, Loss: 1.5623799562454224
Iteration: 142/5000, Loss: 0.8662793040275574
Iteration: 143/5000, Loss: 0.8578730821609497
Iteration: 144/5000, Loss: 0.5697879791259766
Iteration: 145/5000, Loss: 1.21336829662323
Iteration: 146/5000, Loss: 0.7401672601699829
Iteration: 147/5000, Loss: 0.598829984664917
Iteration: 148/5000, Loss: 0.3513275384902954
Iteration: 149/5000, Loss: 0.4866957366466522
Iteration: 150/5000, Loss: 1.6218905448913574
Iteration: 151/5000, Loss: 0.30741575360298157
Iteration: 152/5000, Loss: 2.2383437156677246
Iteration: 153/5000, Loss: 0.6841162443161011
Iteration: 154/5000, Loss: 0.7329118251800537
Iteration: 155/5000, Loss: 2.1721625328063965
Iteration: 156/5000, Loss: 1.090996503829956
Iteration: 157/5000, Loss: 0.5818138718605042
Iteration: 158/5000, Loss: 0.5925453305244446
Iteration: 159/5000, Loss: 1.3169050216674805
Iteration: 160/5000, Loss: 0.47384002804756165
Iteration: 161/5000, Loss: 0.8505755662918091
Iteration: 162/5000, Loss: 0.24840518832206726
Iteration: 163/5000, Loss: 0.8944939374923706
Iteration: 164/5000, Loss: 0.5884906649589539
Iteration: 165/5000, Loss: 0.9131523370742798
Iteration: 166/5000, Loss: 0.3995893597602844
Iteration: 167/5000, Loss: 0.2503143548965454
Iteration: 168/5000, Loss: 1.6455016136169434
Iteration: 169/5000, Loss: 0.5786951184272766
Iteration: 170/5000, Loss: 0.7218962907791138
Iteration: 171/5000, Loss: 2.4155566692352295
Iteration: 172/5000, Loss: 2.048872470855713
Iteration: 173/5000, Loss: 1.9327430725097656
Iteration: 174/5000, Loss: 1.0244345664978027
Iteration: 175/5000, Loss: 0.8392701745033264
Iteration: 176/5000, Loss: 1.995056390762329
Iteration: 177/5000, Loss: 0.8739548325538635
Iteration: 178/5000, Loss: 1.5399049520492554
Iteration: 179/5000, Loss: 0.5741201043128967
Iteration: 180/5000, Loss: 0.8402286767959595
Iteration: 181/5000, Loss: 1.4925413131713867
Iteration: 182/5000, Loss: 0.4397652745246887
```

```
Iteration: 183/5000, Loss: 1.760629415512085
Iteration: 184/5000, Loss: 0.43117761611938477
Iteration: 185/5000, Loss: 0.467487096786499
Iteration: 186/5000, Loss: 1.8101074695587158
Iteration: 187/5000, Loss: 0.3547060191631317
Iteration: 188/5000, Loss: 1.6447556018829346
Iteration: 189/5000, Loss: 0.9380011558532715
Iteration: 190/5000, Loss: 0.38763517141342163
Iteration: 191/5000, Loss: 1.418509602546692
Iteration: 192/5000, Loss: 0.4318119287490845
Iteration: 193/5000, Loss: 0.5983543992042542
Iteration: 194/5000, Loss: 1.2837105989456177
Iteration: 195/5000, Loss: 0.17891466617584229
Iteration: 196/5000, Loss: 0.31739941239356995
Iteration: 197/5000, Loss: 2.0140693187713623
Iteration: 198/5000, Loss: 0.7401397228240967
Iteration: 199/5000, Loss: 0.3275834619998932
Iteration: 200/5000, Loss: 0.9757547974586487
Iteration: 201/5000, Loss: 1.0293619632720947
Iteration: 202/5000, Loss: 1.2379769086837769
Iteration: 203/5000, Loss: 0.9408031702041626
Iteration: 204/5000, Loss: 1.7971158027648926
Iteration: 205/5000, Loss: 0.7073979377746582
Iteration: 206/5000, Loss: 0.2784920334815979
Iteration: 207/5000, Loss: 1.6068793535232544
Iteration: 208/5000, Loss: 0.6217333674430847
Iteration: 209/5000, Loss: 1.368154525756836
Iteration: 210/5000, Loss: 0.08461061120033264
Iteration: 211/5000, Loss: 0.3515865206718445
Iteration: 212/5000, Loss: 0.3239373564720154
Iteration: 213/5000, Loss: 0.4573105275630951
Iteration: 214/5000, Loss: 0.3754955530166626
Iteration: 215/5000, Loss: 0.7892888188362122
Iteration: 216/5000, Loss: 0.3250851631164551
Iteration: 217/5000, Loss: 0.4248100817203522
Iteration: 218/5000, Loss: 1.7559183835983276
Iteration: 219/5000, Loss: 1.7762466669082642
Iteration: 220/5000, Loss: 1.405247688293457
Iteration: 221/5000, Loss: 0.8996542096138
Iteration: 222/5000, Loss: 0.8474833369255066
Iteration: 223/5000, Loss: 0.8354241251945496
Iteration: 224/5000, Loss: 0.5235930681228638
Iteration: 225/5000, Loss: 1.0635870695114136
Iteration: 226/5000, Loss: 0.6069650650024414
Iteration: 227/5000, Loss: 0.5066357851028442
Iteration: 228/5000, Loss: 0.5552139282226562
Iteration: 229/5000, Loss: 0.3285568952560425
Iteration: 230/5000, Loss: 1.5950329303741455
```

```
Iteration: 231/5000, Loss: 1.1449296474456787
Iteration: 232/5000, Loss: 0.24141094088554382
Iteration: 233/5000, Loss: 0.3925676941871643
Iteration: 234/5000, Loss: 1.6167649030685425
Iteration: 235/5000, Loss: 2.1152639389038086
Iteration: 236/5000, Loss: 0.675682783126831
Iteration: 237/5000, Loss: 0.8690586686134338
Iteration: 238/5000, Loss: 0.8388773798942566
Iteration: 239/5000, Loss: 0.48444804549217224
Iteration: 240/5000, Loss: 0.7934463620185852
Iteration: 241/5000, Loss: 0.8583378791809082
Iteration: 242/5000, Loss: 1.1751596927642822
Iteration: 243/5000, Loss: 0.7382664680480957
Iteration: 244/5000, Loss: 0.5953572988510132
Iteration: 245/5000, Loss: 0.43042901158332825
Iteration: 246/5000, Loss: 1.6510801315307617
Iteration: 247/5000, Loss: 1.750913143157959
Iteration: 248/5000, Loss: 0.8653919696807861
Iteration: 249/5000, Loss: 0.9110537767410278
Iteration: 250/5000, Loss: 0.9046513438224792
```

```
       ␣
 ↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
 ↪last)

       <ipython-input-102-ced10993cb5a> in <module>
        37             continue
        38
    ---> 39         loss_dict = model(imgs, annotations)
        40         losses = sum(loss for loss in loss_dict.values())
        41


       ~/.local/lib/python3.7/site-packages/torch/nn/modules/module.py in␣
 ↪_call_impl(self, *input, **kwargs)
       725             result = self._slow_forward(*input, **kwargs)
       726         else:
    --> 727             result = self.forward(*input, **kwargs)
       728         for hook in itertools.chain(
       729                 _global_forward_hooks.values(),


       ~/.local/lib/python3.7/site-packages/torchvision/models/detection/
 ↪generalized_rcnn.py in forward(self, images, targets)
        97         if isinstance(features, torch.Tensor):
```

```
      98             features = OrderedDict([('0', features)])
  ---> 99         proposals, proposal_losses = self.rpn(images, features,␣
↪targets)
     100         detections, detector_losses = self.roi_heads(features,␣
↪proposals, images.image_sizes, targets)
     101         detections = self.transform.postprocess(detections, images.
↪image_sizes, original_image_sizes)


     ~/.local/lib/python3.7/site-packages/torch/nn/modules/module.py in␣
↪_call_impl(self, *input, **kwargs)
     725             result = self._slow_forward(*input, **kwargs)
     726         else:
  --> 727             result = self.forward(*input, **kwargs)
     728         for hook in itertools.chain(
     729                 _global_forward_hooks.values(),


     ~/.local/lib/python3.7/site-packages/torchvision/models/detection/rpn.py␣
↪in forward(self, images, features, targets)
     329         # RPN uses all feature maps that are available
     330         features = list(features.values())
  --> 331         objectness, pred_bbox_deltas = self.head(features)
     332         anchors = self.anchor_generator(images, features)
     333


     ~/.local/lib/python3.7/site-packages/torch/nn/modules/module.py in␣
↪_call_impl(self, *input, **kwargs)
     725             result = self._slow_forward(*input, **kwargs)
     726         else:
  --> 727             result = self.forward(*input, **kwargs)
     728         for hook in itertools.chain(
     729                 _global_forward_hooks.values(),


     ~/.local/lib/python3.7/site-packages/torchvision/models/detection/rpn.py␣
↪in forward(self, x)
      57         for feature in x:
      58             t = F.relu(self.conv(feature))
  ---> 59             logits.append(self.cls_logits(t))
      60             bbox_reg.append(self.bbox_pred(t))
      61         return logits, bbox_reg


     ~/.local/lib/python3.7/site-packages/torch/nn/modules/module.py in␣
↪_call_impl(self, *input, **kwargs)
```

```
    725                 result = self._slow_forward(*input, **kwargs)
    726             else:
--> 727                 result = self.forward(*input, **kwargs)
    728             for hook in itertools.chain(
    729                     _global_forward_hooks.values(),


    ~/.local/lib/python3.7/site-packages/torch/nn/modules/conv.py in␣
↪forward(self, input)
    421
    422     def forward(self, input: Tensor) -> Tensor:
--> 423         return self._conv_forward(input, self.weight)
    424
    425 class Conv3d(_ConvNd):


    ~/.local/lib/python3.7/site-packages/torch/nn/modules/conv.py in␣
↪_conv_forward(self, input, weight)
    418                             _pair(0), self.dilation, self.groups)
    419         return F.conv2d(input, weight, self.bias, self.stride,
--> 420                         self.padding, self.dilation, self.groups)
    421
    422     def forward(self, input: Tensor) -> Tensor:


    KeyboardInterrupt:
```