Animal_Classification-transfer-VGG16

June 10, 2024

1 Changing to transfer learning with VGG16

From the previous model, we have swapped out the model for VGG16 that has been pretrained and tweaking it to use our data and classes.

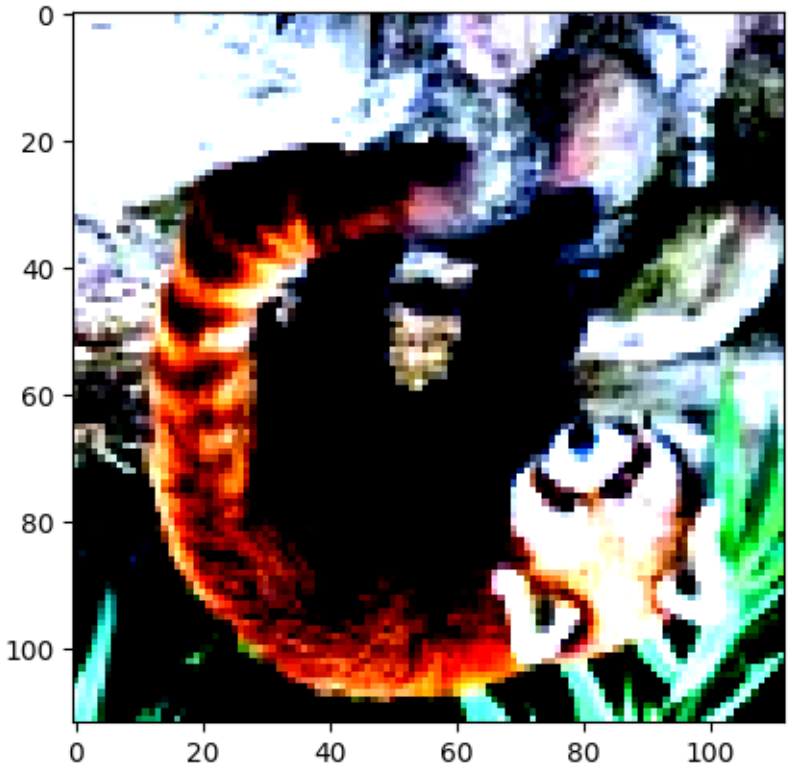
Validation Loss: 1.177 Validation Accuracy: 85.00% FLOPS: 7.89G

```
Size of training dataset : 6270
```

```
torch.Size([3, 112, 112])
```

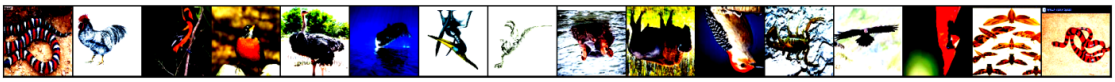
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```

```
Label:  ailurus-fulgens (5)
```



```
(5330, 313, 627)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```



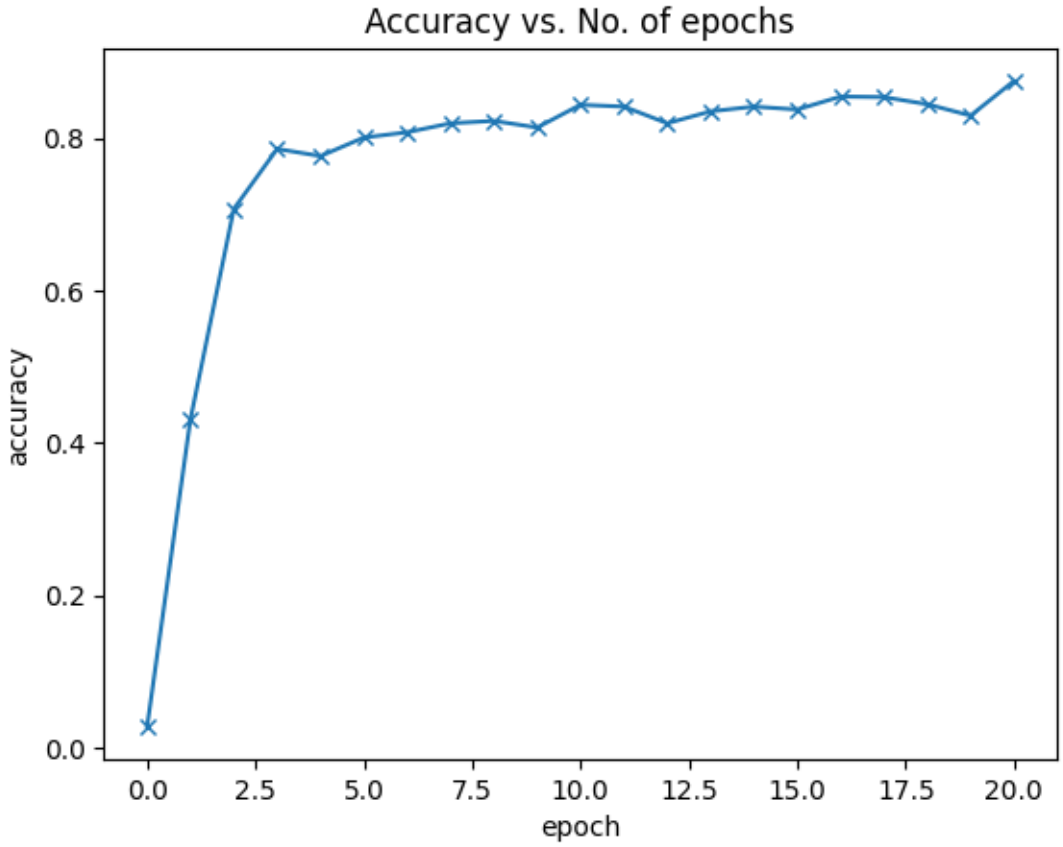
```
ConvolutionalNetwork(
  (model): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU(inplace=True)
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): ReLU(inplace=True)
      (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (13): ReLU(inplace=True)
      (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): ReLU(inplace=True)
      (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): ReLU(inplace=True)
      (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (20): ReLU(inplace=True)
      (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): ReLU(inplace=True)
      (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (27): ReLU(inplace=True)
      (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (29): ReLU(inplace=True)
```

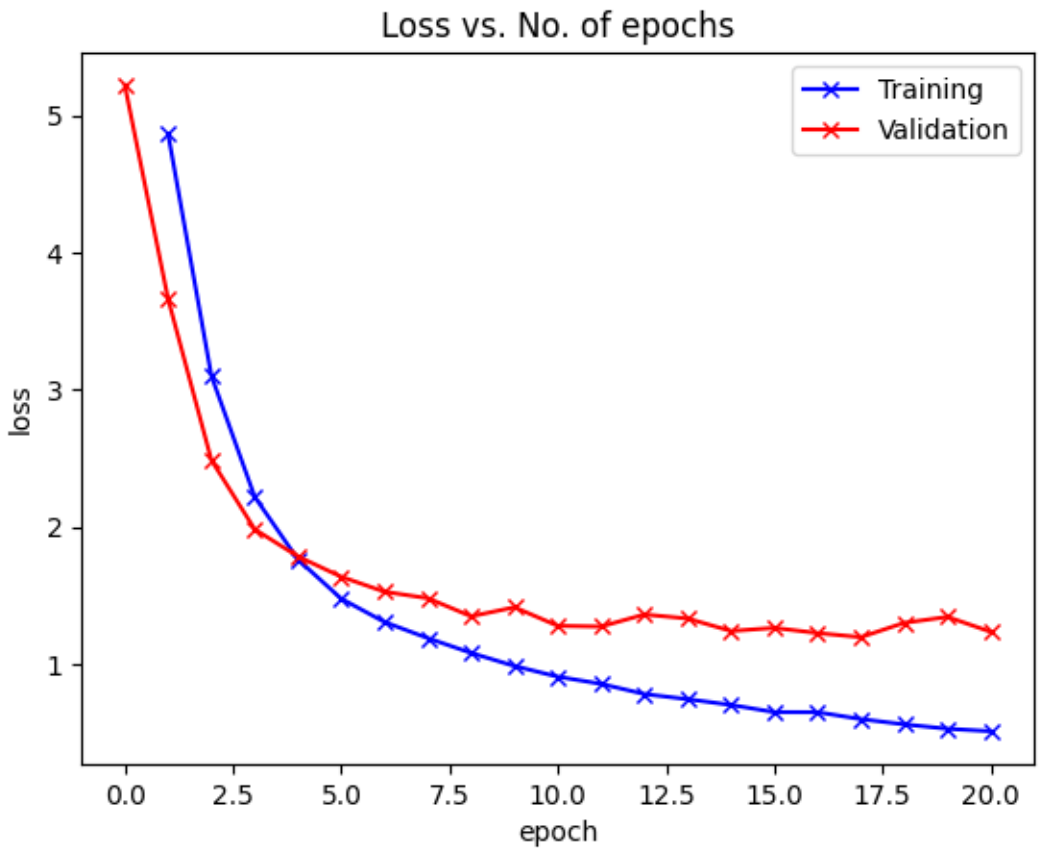
```
      (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
    (classifier): Sequential(
      (0): Linear(in_features=25088, out_features=4096, bias=True)
      (1): ReLU(inplace=True)
      (2): Dropout(p=0.5, inplace=False)
      (3): Linear(in_features=4096, out_features=4096, bias=True)
      (4): ReLU(inplace=True)
      (5): Dropout(p=0.5, inplace=False)
      (6): Linear(in_features=4096, out_features=151, bias=True)
    )
  )
)

images.shape: torch.Size([16, 3, 112, 112])
out.shape: torch.Size([16, 151])
out[0]: tensor([-5.0155, -4.8244, -6.4500, -5.2882, -5.6761, -5.2723, -8.1427,
-6.9423,
        -6.1285, -3.6968, -4.2547, -7.4687, -4.7189, -4.3489, -5.8139, -3.6071,
        -6.2837, -5.6112, -6.3119, -6.4560, -5.2747, -6.3577, -5.3859, -6.2020,
        -6.2545, -6.1579, -5.7563, -6.2114, -6.6234, -4.3613, -5.2160, -4.1965,
        -6.5862, -5.6447, -6.0422, -3.9137, -5.6003, -7.0435, -5.8277, -5.6527,
        -5.5874, -4.9604, -4.8894, -3.4264, -5.5862, -6.0568, -6.7974, -5.1861,
        -5.1507, -4.9369, -6.1578, -3.7941, -5.7377, -6.2363, -7.3798, -5.4343,
        -5.1280, -5.1943, -6.9009, -4.4677, -5.4308, -4.5975, -4.8791, -7.1803,
        -3.5253, -6.1256, -6.0084, -8.1544, -7.0820, -5.6563, -6.5758, -4.9197,
        -5.3123, -6.0809, -5.0418, -5.8404, -2.6726, -2.7620, -6.7194, -6.2637,
        -5.9573, -6.1571, -7.9727, -4.3419, -5.5299, -5.2522, -4.0395, -5.0515,
        -4.6345, -5.2396, -6.7284, -6.3737, -5.1581, -5.6935, -6.9317, -4.8901,
        -6.6558, -4.2015, -6.1732, -3.9284, -5.0238, -5.1499, -6.3977, -4.0645,
        -5.7820, -4.9934, -5.7307, -4.7646, -6.9689, -7.1588, -5.8584, -8.3142,
        -5.6302, -4.2608, -4.5197, -5.6761, -6.1620, -5.3299, -5.2093, -5.0674,
        -5.9995, -7.9803, -6.6908, -4.9153, -7.8850, -7.6099, -6.4608, -7.0354,
        -3.6432, -5.3385, -6.1655, -4.8455, -5.6094, -5.3136, -3.9510, -6.8466,
        -4.6274, -4.3193, -5.6198, -5.2383, -6.9402, -7.3852, -3.6571, -6.4589,
        -5.5141, -7.1587, -5.3708, -5.7735, -5.2962, -4.1481, -6.7642],
       device='cuda:0', grad_fn=<SelectBackward0>)

ConvolutionalNetwork(
  (model): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU(inplace=True)
```

```
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): ReLU(inplace=True)
      (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (13): ReLU(inplace=True)
      (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): ReLU(inplace=True)
      (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): ReLU(inplace=True)
      (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (20): ReLU(inplace=True)
      (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): ReLU(inplace=True)
      (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (27): ReLU(inplace=True)
      (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (29): ReLU(inplace=True)
      (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
    (classifier): Sequential(
      (0): Linear(in_features=25088, out_features=4096, bias=True)
      (1): ReLU(inplace=True)
      (2): Dropout(p=0.5, inplace=False)
      (3): Linear(in_features=4096, out_features=4096, bias=True)
      (4): ReLU(inplace=True)
      (5): Dropout(p=0.5, inplace=False)
      (6): Linear(in_features=4096, out_features=151, bias=True)
    )
  )
)

[{'val_loss': 5.219048500061035, 'val_acc': 0.02743055485188961}]
```

```
627
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [0], train_loss: 4.8645, val_loss: 3.6620, val_acc: 0.4309
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [1], train_loss: 3.1019, val_loss: 2.4879, val_acc: 0.7069
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [2], train_loss: 2.2207, val_loss: 1.9842, val_acc: 0.7858
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [3], train_loss: 1.7636, val_loss: 1.7860, val_acc: 0.7764
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [4], train_loss: 1.4785, val_loss: 1.6382, val_acc: 0.8007
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [5], train_loss: 1.3082, val_loss: 1.5291, val_acc: 0.8076
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [6], train_loss: 1.1912, val_loss: 1.4818, val_acc: 0.8194
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [7], train_loss: 1.0833, val_loss: 1.3501, val_acc: 0.8226
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [8], train_loss: 0.9882, val_loss: 1.4161, val_acc: 0.8139
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [9], train_loss: 0.9103, val_loss: 1.2807, val_acc: 0.8438
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [10], train_loss: 0.8582, val_loss: 1.2756, val_acc: 0.8413
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [11], train_loss: 0.7840, val_loss: 1.3637, val_acc: 0.8194
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [12], train_loss: 0.7465, val_loss: 1.3341, val_acc: 0.8351
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [13], train_loss: 0.7050, val_loss: 1.2432, val_acc: 0.8413
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [14], train_loss: 0.6524, val_loss: 1.2659, val_acc: 0.8375
  0%|          | 0/334 [00:00<?, ?it/s]
```

```
Epoch [15], train_loss: 0.6518, val_loss: 1.2282, val_acc: 0.8545
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [16], train_loss: 0.6015, val_loss: 1.1983, val_acc: 0.8538
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [17], train_loss: 0.5634, val_loss: 1.3028, val_acc: 0.8444
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [18], train_loss: 0.5312, val_loss: 1.3459, val_acc: 0.8295
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [19], train_loss: 0.5133, val_loss: 1.2373, val_acc: 0.8750
```






```
{'val_loss': 1.1766126155853271, 'val_acc': 0.8500000238418579}
```

1.1 FLOPs

```
+ Number of FLOPs: 7.89G
```