CS 4347 Introduction to Machine Learning Nov-25-2019 Dr. Yan Yan

Assignment 3 Chichi Christine

Convolution Neural Network

GitHub Repo Link:

https://github.com/mlp12/CS4347_Convolution_Neural_Network.git

https://github.com/mlp12/CS4347_Convolution_Neural_Network/invitations

Architecture

I used Pytorch and a pretrained model ResNet50 to do image classification.

ResNet50 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. We can take a pretrained image classification network such as ResNet50, that has already learned to extract informative features from natural images and use it as a starting point to learn a new task. ResNet, short for Residual Networks is a neural network used as a backbone for computer vision tasks.

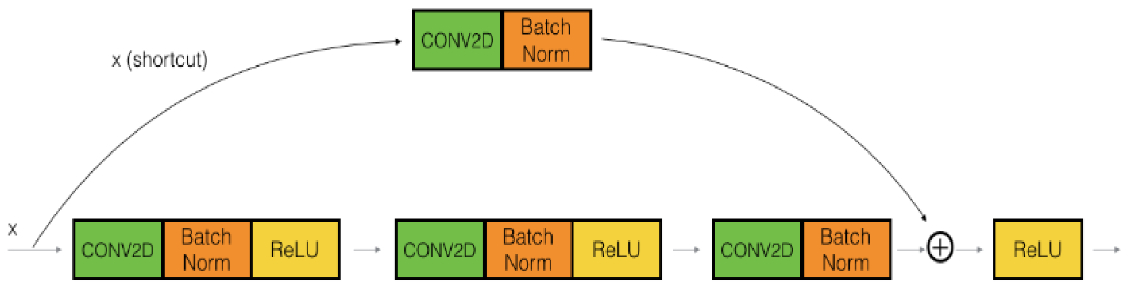
This model was the winner of ImageNet challenge in 2015. The layers learn residual functions with reference to the layer inputs [1]. Increased depth adds to accuracy when used with many training samples. ResNet introduced the concept of skip connection. ResNet uses skip connection to add the output from an earlier layer to a later layer.

Skip connections:

They mitigate the problem of vanishing gradient by allowing this alternate shortcut path for gradient to flow through

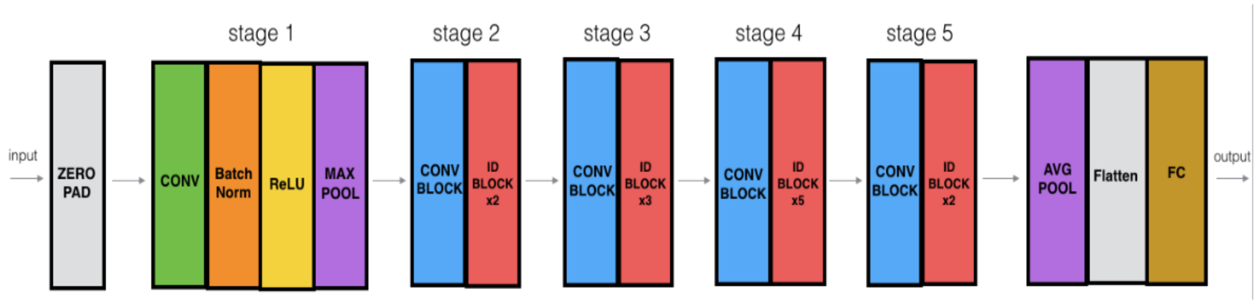
They allow the model to learn an identity function which ensures that the higher layer will perform at least as good as the lower layer, and not worse.

Skip Connection



X_shortcut goes through convolution block

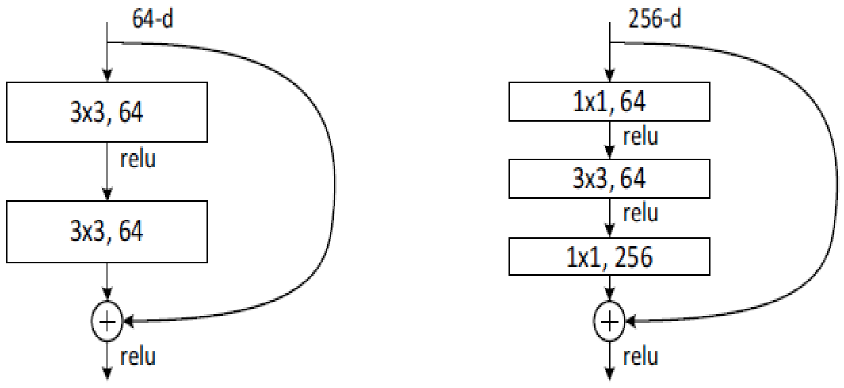
X, X_shortcut above are two matrixes, and they should have the same shape before they are added.



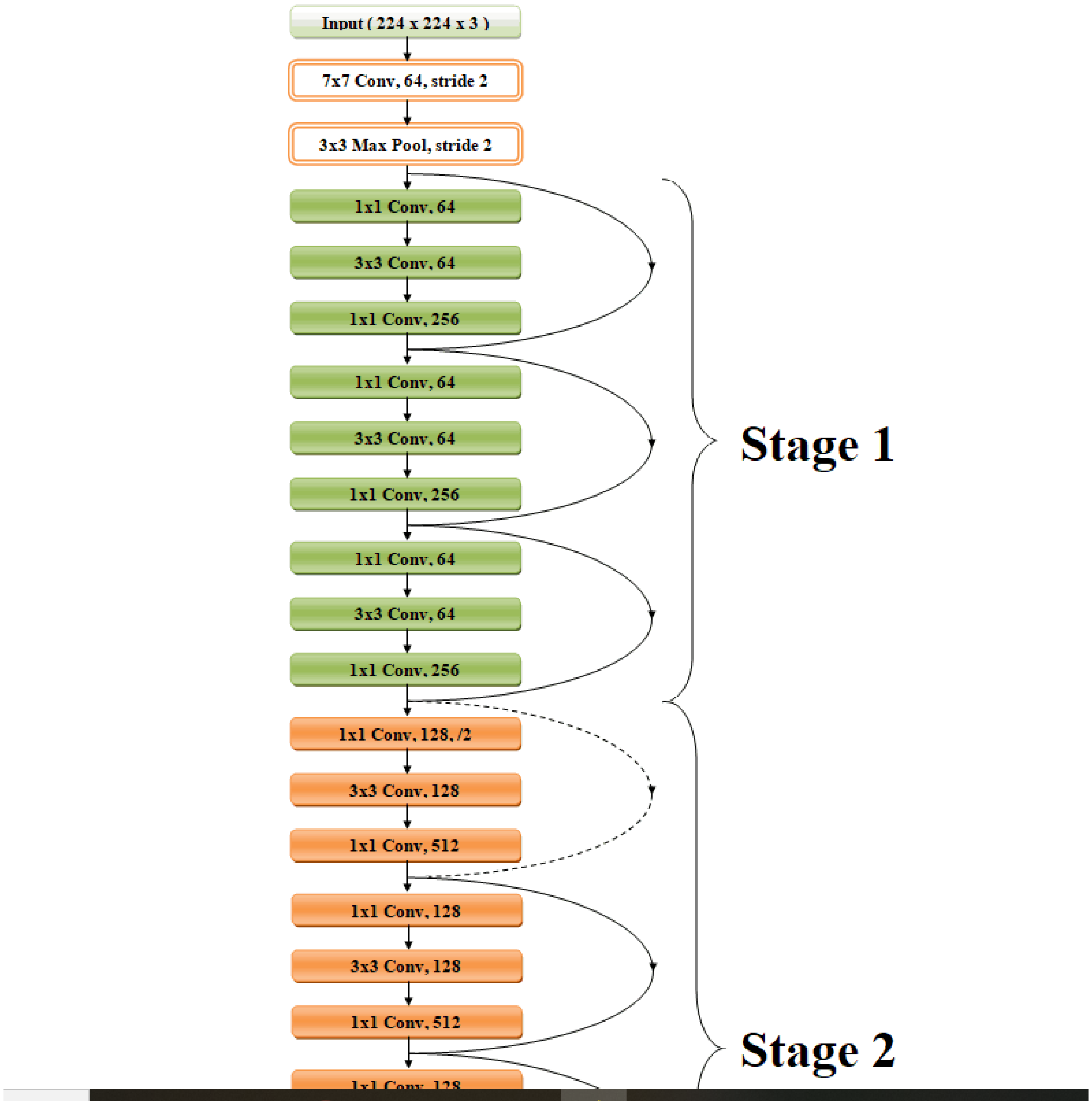
ResNet-50 Model

ResNet-50 Model Structure [2].

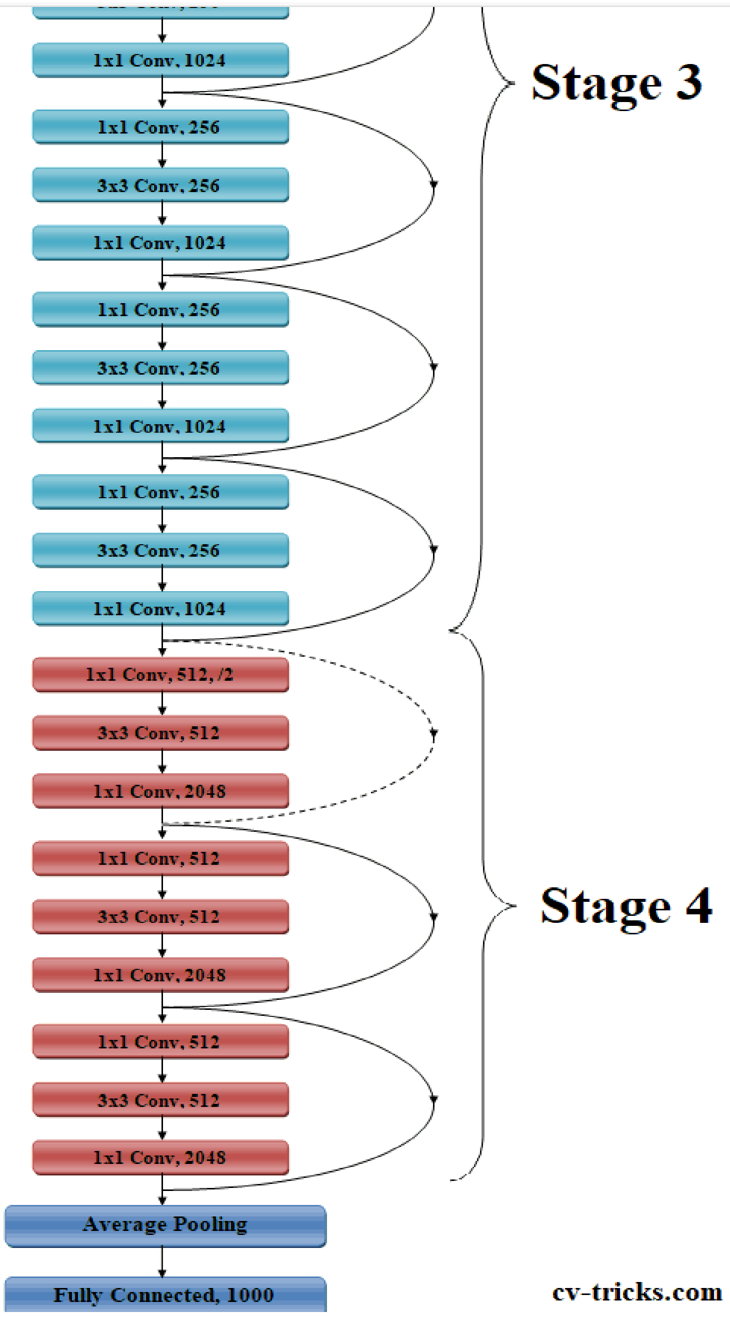
In ResNet50 a bottleneck design improves time complexity. The 1×1 conv layers are added to the start and end of network. 1×1 conv can reduce the number of connections or parameters while not degrading the performance of the network so much.



The Basic Block (Left) and The Proposed Bottleneck Design (Right)



Details of the ResNet50 layers [3].



Architecture diagram of ResNet50

Details of the ResNet50 layers [3].

ResNet uses batch normalization to adjust the input layer and increase the performance of the network.

Some layers in the model from the program:

```
ResNet( (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True) (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1,
dilation=1, ceil_mode=False) (layer1): Sequential(
```

Etc. Please see the submitted ipynb file.

Convolution applies filters to the images so that numbers can be generated for max pooling. It helps us identify features present in an image. Normalizing the inputs brings all the input features to the same scale which helps the algorithm find the global minimum faster. Batch normalization means we compute the mean and standard deviation from a single batch as opposed to computing it from the entire data. Batch normalization is done individually at hidden neurons in the network. Images have four dimensions — batch_size x channels x height x width. We normalize the batch for images over each channel by applying BatchNorm2d.

For a pretrained model, it is necessary to freeze some of its layers and parameters and not train those. Code:

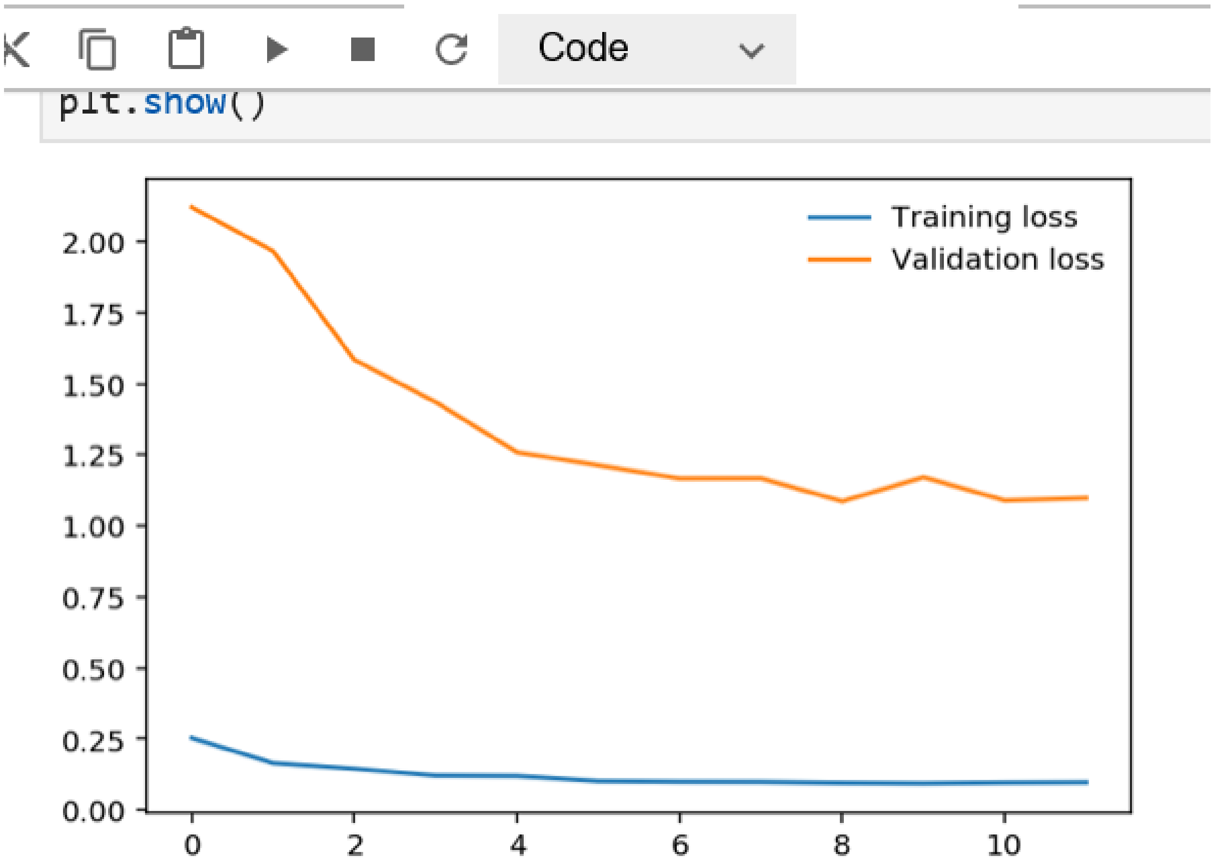
```
for param in model.parameters():
    param.requires_grad = False


model = models.resnet50(pretrained=True)
model.fc = nn.Sequential(nn.Linear(2048, 512),
                         nn.ReLU(),
                         nn.Dropout(0.2),
                         nn.Linear(512, 10),
                         nn.LogSoftmax(dim=1))
criterion = nn.NLLLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.003)
```

Fully connected neural network involves a forward pass and backward pass. Forward pass applies the activation function to the (dot product of inputs and weights + bias). Backward pass involves backpropagation and the weights and biases update. This involves calculating loss, partial differential of loss with respect to input variables, gradient descent and updating weights. Sequential model with Linear layers and input and output dimensions. ReLU activation function for hidden layers and output layer had LogSoftmax function.

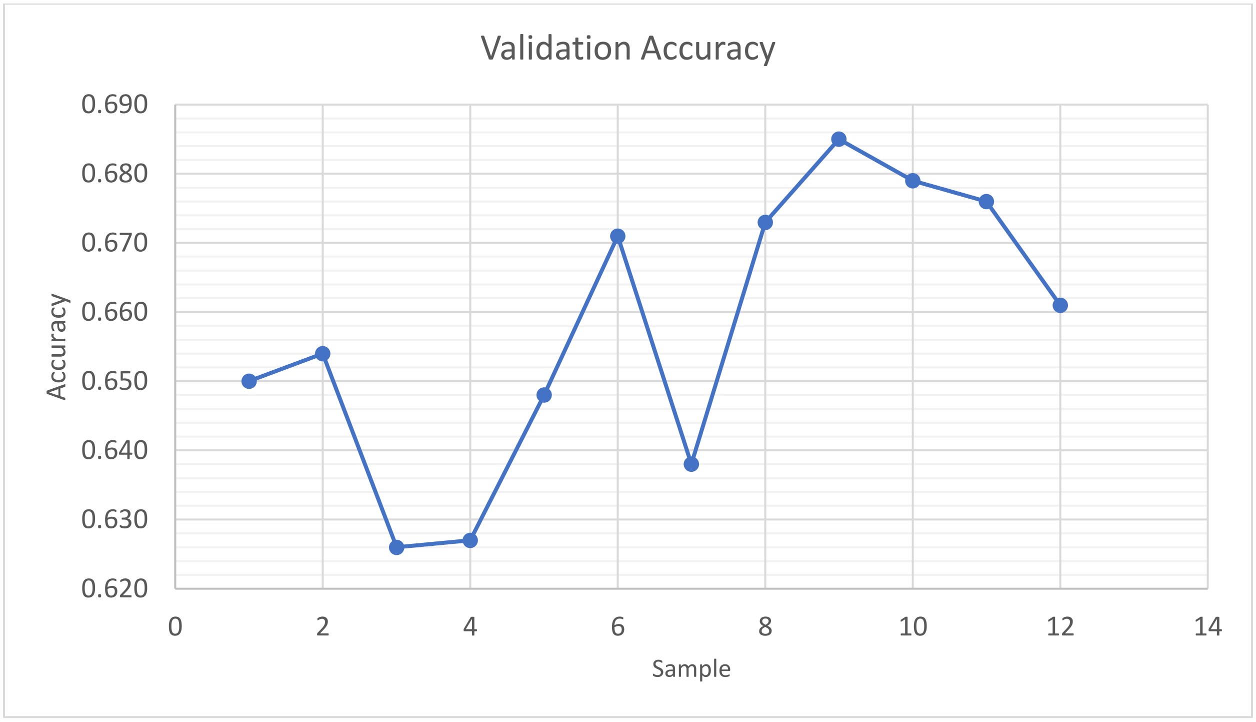
<u>Results</u>

```
Epoch 1/1.. Train loss: 1.184.. Test loss: 1.022.. Test accuracy: 0.650
Epoch 1/1.. Train loss: 1.041.. Test loss: 0.989.. Test accuracy: 0.654
Epoch 1/1.. Train loss: 1.053.. Test loss: 1.065.. Test accuracy: 0.626
Epoch 1/1.. Train loss: 1.046.. Test loss: 1.049.. Test accuracy: 0.627
Epoch 1/1.. Train loss: 1.194.. Test loss: 1.024.. Test accuracy: 0.648
Epoch 1/1.. Train loss: 1.111.. Test loss: 0.956.. Test accuracy: 0.671
Epoch 1/1.. Train loss: 1.032.. Test loss: 1.034.. Test accuracy: 0.638
Epoch 1/1.. Train loss: 1.065.. Test loss: 0.916.. Test accuracy: 0.673
Epoch 1/1.. Train loss: 1.018.. Test loss: 0.915.. Test accuracy: 0.685
Epoch 1/1.. Train loss: 1.002.. Test loss: 0.912.. Test accuracy: 0.679
Epoch 1/1.. Train loss: 1.050.. Test loss: 0.944.. Test accuracy: 0.676
Epoch 1/1.. Train loss: 1.035.. Test loss: 0.938.. Test accuracy: 0.661
```



```
plt.show()
```

Graph of Loss



Testing the model on a few unlabeled samples: Index returned was 5 for first sample.

5

beet_salad

Actual label from training: beet_salad

True

3

beef_carpaccio

Actual label from training: beef_carpaccio

True

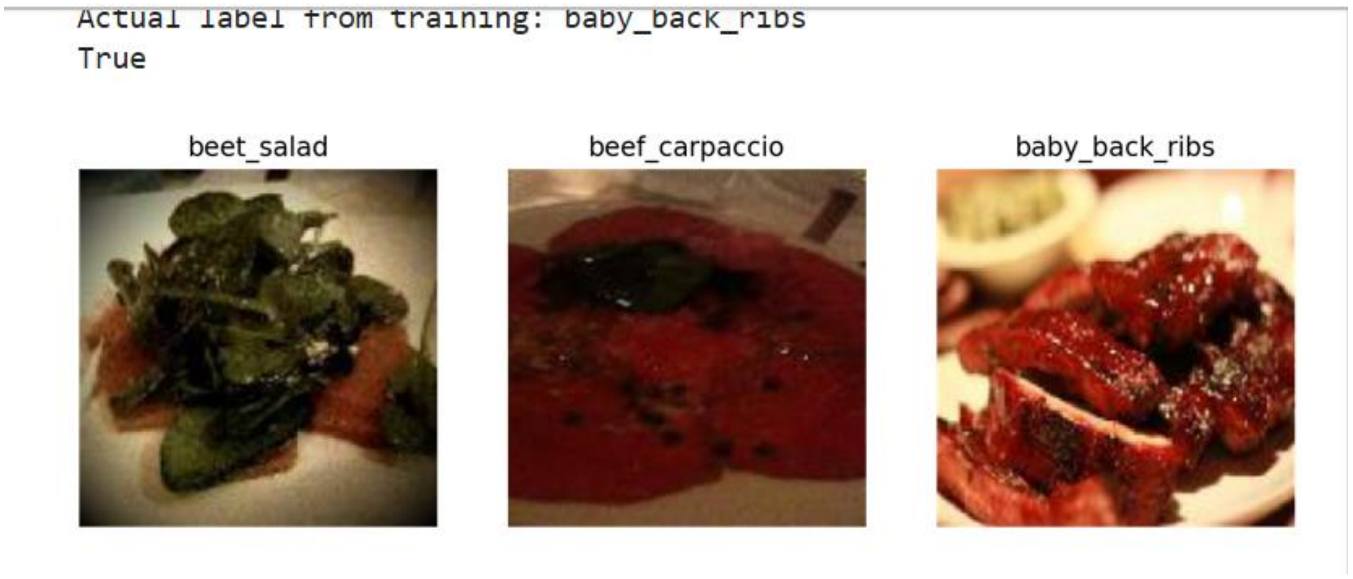
1

baby_back_ribs

Actual label from training: baby_back_ribs

True

Accuracy = 1



```
Actual label from training: baby_back_ribs
True
```

Testing 1000 random samples from test folder

```
beef_tartare
Actual label from training: beef_tartare
True
1
beignets
Actual label from training: beignets
True
2
breakfast_burrito
Actual label from training: breakfast_burrito
True
3
……..
beignets
Actual label from training: beignets
True
1000

True 1000
Total 1000
Accuracy 1.0
```

Conclusion: Using a pretrained model increased accuracy.

Parameters

Loss function: Negative log likelihood loss criterion = nn.NLLLoss()

Learning rate: 0.003

Optimizer: Adam

Epoch: 1

Train size: 0.8 * 10000 = 8000 samples

Validation size: 0.2 * 10000 = 2000 samples

Test size: 1000 random samples from test folder

Data dimension (height * width): 128 * 128

Works Cited

[1] K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” Microsoft Research, December 10, 2015. [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf

[2] P. Dwivedi, “Understanding and Coding a Resnet in Keras,” Towards Data Science, January 4, 2019. [Online]. Available: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

[3] U. Gupta, “Detailed Guide to Understand and Implement ResNets,” CV-Tricks.com, 2017.

[Online]. Available: https://cv-tricks.com/keras/understand-implement-resnets/