Deep Learning Coursework

Topic- CNN implementation on plant dataset using transfer learning.

- Ronit Ganguly (2487190G)

2487190G@student.gla.ac.uk

About the data:

The Data is about Blight, mould in Tomatoes, spider mites, pepper bell bacteria diseases in plants. Also, some normal flowers, and a certain weed have been used to make the model versatile.

- **Blight** is a serious plant illness which is caused by the fungus *Alternaria solani* and runs havoc throughout the United States all round the year. Early detection of Blight is necessary otherwise delayed action mostly goes in vain as major portions of the farmlands are set ablaze by the farmers to control the disease.

  - *https://extension.umn.edu/diseases/early-blight-tomato*

- **Tomato Mould** is a type of fungii which affects the foliage of tomatoes, particularly those grown in greenhouses. Symptoms are sometimes confused with those of other foliar diseases such as grey mould or tomato blight.

  - *https://www.rhs.org.uk/advice/profile?pid=468*

- **Pepper bell bacterium** *Xanthomonas campestris pv. vesicatoria* causes spots on the leaves, fruits and stems. Bacterial spot is the most important disease affecting peppers in New York. A mild case of bacterial spot causes prominent necrotic spots on leaves; a severe case can cause premature leaf drop and spotting of stems and pods that results in **unmarketable fruit**. Like most bacterial diseases, bacterial spot is difficult to control when frequent rains and moist conditions prevail. Hence early detection is needed which provides a scope to the Deep Learning community. **Also, it is confused with blight disease and hence is a challenge for my network.**

  -*http://vegetablemdonline.ppath.cornell.edu/factsheets/Pepper_BactSpot.htm*

- **Spider mites** (*Tetranychus spp*.) can ruin a tomato (*Lycopersicon esculentum*) garden when they colonize the undersides of leaves and suck plant juices. Tiny arachnids, spider mites are in the same family as spiders and ticks and are one of the major pest problems of tomato plants.

  -*https://homeguides.sfgate.com/avoid-spider-mites-tomato-garden-73368.html*

- **Black rot** is a disease of apples that infects fruit, leaves and bark caused by the fungus *Botryosphaeria obtusa*. It can also jump to healthy tissue on pear or quince trees. Fruit infection is the most destructive form of this pathogen and begins with infected flowers, before fruits expand. Once again, ****fruits become unmarketable****.

  -https://www.gardeningknowhow.com/edible/fruits/apples/black-rot-on-apple-trees.htm

Data Distribution:

- There are 12 classes:
  - Apple Black-rot,
  - Apple Healthy,
  - Corn Blight,
  - Corn Healthy,
  - Tomato Blight,
  - Tomato Healthy,
  - Pepper Bell Bacterial Infection,
  - Pepper Healthy,
  - Tomato Mould Infection,
  - Tomato Spider Mites Infection,
  - Flowers (random),
  - Cheek Weed (a common crop weed).

- The data is split into training set, validation set and testing set.
- These 3 sets can be found within the folder named "more plant dataset" on colab.
- **Training** set contains **200 Images per class.**
- Both the **validation** set and the **testing** set contain **50 images per class.**

Objectives:

- **Create a baseline model** - Created a 4-layers CNN (2 convolution layers, 2 dense layers).
- **Fine-Tune state-of-the-art models and compare its performance with baseline** - Implemented **ResNet50** and **VGG16** with batch normalization.

- **Show implementation of regularization methods** - Added dropout layers in both baseline and ResNet50(before last layer) and tuned weight decay for ResNet50. Used only weight decay concept in VGG16.
- **Visualize the data** - Plotted the data before creating tensors and after creating tensors.
- **Add more data augmentation techniques** - Added Affine transformation, random flips, colour jitter, rotation.
- **Optimize hyper-parameters** - Used Bayesian Optimization to tune Learning Rate and Momentum for both Baseline and ResNet50 and VGG16.
- **Implement K-Fold** - Used K-Fold to tune weight decay for ResNet50 and VGG16
- **Plot results** - Plotted Train loss - Validation loss for each architecture, drawn confusion matrix as well.

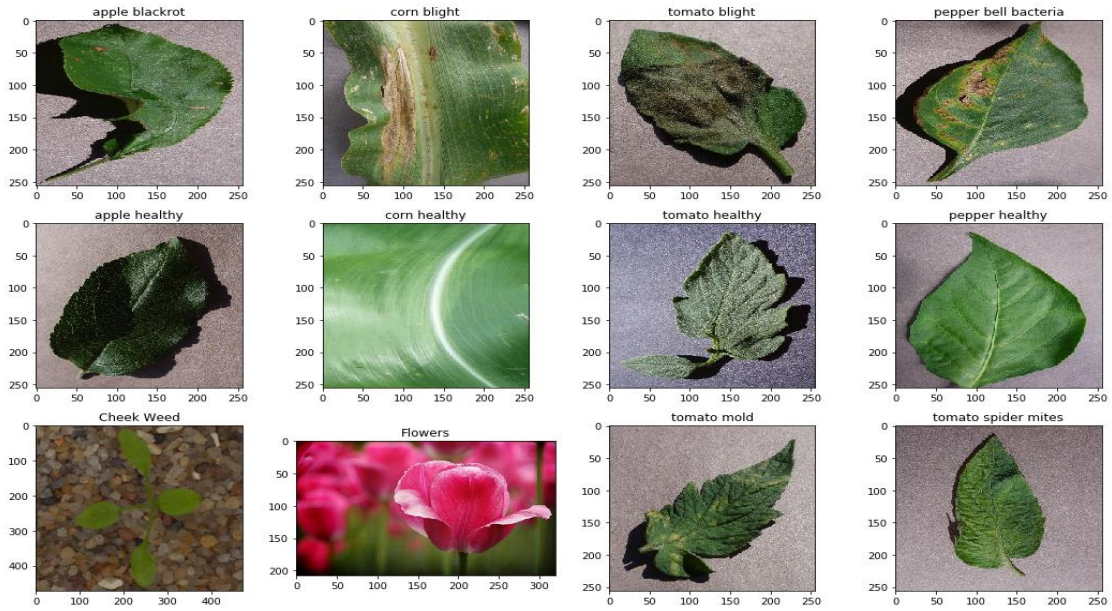
Accuracy Metrics:

- The type 2 error rate: This tells us that out of 100 times on an average how many times the ground truth was not present in the top 2 predictions of a model. This is important because it can be used to tie break between models which have same accuracy.
- The type 1 error accuracy: number of correct predictions/ total number of observations
- Also, it is being aided by confusion matrix as we will see below.

Plan of Attack:

- Visualize the data.
- Create a baseline model and check its performance.
- Tune the baseline model and check its performance.
- Implement Resnet50 architecture and fine-tune the model.
- Compare the result of the tuned Resnet50 and the tuned baseline model.
- Implement another architecture VGG16 with batch normalization and fine-tune it.
- Compare its performance with all the other models.
- Discuss on the best model to use for this task.

Visualization:

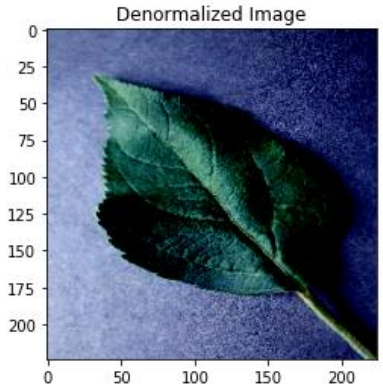


Some basic inferences and challenges:

- All the images have three channels.
- Flowers should be totally distinguishable from the rest.
- Plants that can be confused with such as tomato healthy, tomato mould, tomato blight, tomato spider mites infection because they are the same plants and the data also contains initial stages of symptoms which may be indistinguishable by humans.
- Blight can be confused with pepper bell bacterial infection because the symptoms are similar. Hence, this is a **challenge for the architecture** to properly differentiate between these two diseases.
- Since blight and mould can be very similar, tomato mould and tomato blight can be mixed with and wrong results can be observed. So, this is a **greater challenge for the architecture.**

Baseline model without dropout layers and untuned:

- A de-normalized image :



This image is obtained by reversing the normalization done on the image during pytorch's transform.

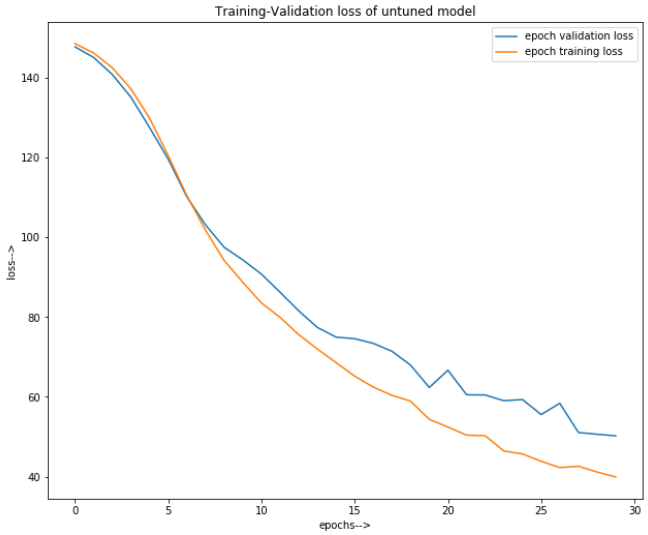
- I have selected a 4-hidden layer Convolution network to set baseline quite high for ResNet50 and VGG16.
- The first out channel is 32 as 32 filters will produce 32 channels which will provide more scope to the architecture to learn more interesting functions about the data in the layer.
- Similarly I increased the filters in the 2nd convolution layer to 64 as more sophisticated functions are learnt in the later layers than in the initial layers.
- Padding is kept as 0, so this helps in reducing the number of output features as the convolution action inherently diminishes the image size.
- Kernel size is 5 throughout the 2 convolution layers. Large kernel size finds general features in the image and since it's a baseline image, I decided to go with a large kernel size.
- Maxpooling is used which further reduces the size of the convolved image and prevents from overfitting. Also, Maxpooling highlights the maximum numbers which means that high pixels will be highlighted more.
- Relu activation function is being used over sigmoid to prevent from **vanishing gradient** and **exploding gradient** because ReLu makes the gradient constant to either 0 or 1.
- Also, the forward definition of my architecture doesn't have a final softmax output because it is being taken care of by nn.CrossEntropyLoss().
- Have chosen CrossEntropyLoss over MSE/MAE because in image classification, any error has to be heavily penalized whih is done by a logarithmic function (cross-entropy)better than a straight line (MAE loss) or a quadratic function (MSE loss).

Operation:

- Data is re-scalled to 224,224 size because we will compare this baseline model with Resnet50 and VGG16 which require the images to be of this size.
- Normalization is also done to scale the data.
- Learning rate is set at 0.05 (randomly selected).
- Batch size for training is 40 and validation is 10.
- n_epochs is 30 to give enough scope for the loss function to converge, if it is possible. This also means that we are giving enough time to the architecture to learn important functions.
- It takes around **20 seconds per epoch** which is 20*30 epochs = 600 seconds.

Baseline untuned architecture observation:

Accuracy is 73.8 % on the test dataset.



- **Both the training loss and the validation loss haven't converged.**

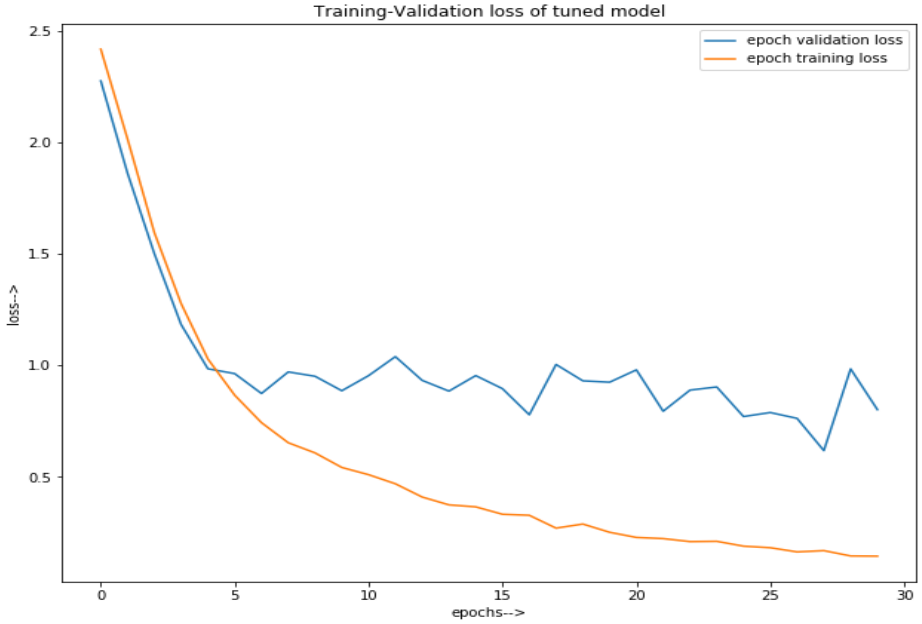
  This suggests that a tuned learning rate which must be higher than the current one, is required.

- **No sign of overfitting though.**

  From the loss graph above, we can see that both validation loss and training loss go together down. For this network to overfit, the training loss should have been lesser than the validation loss.

Baseline model with dropout layers, data augmentation and hyper-parameter tuning:

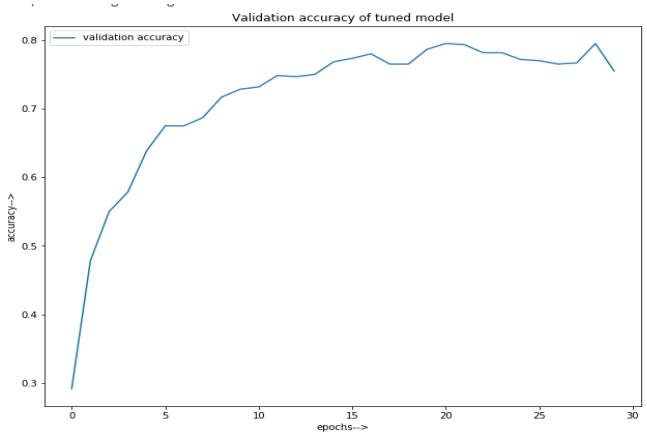
- Dropout layers are added as part of regularization methods to reduce probable overfitting.
- Data Augmentation techniques are as follows:
    - Random horizontal flip - To flip the horizontally.
    - Random Vertical flip - To flip the image vertically because in cheekweed class some images were taken from different sides.
    - Random Rotation - Rotating the image with 20 degree angle.
    - Colour jitters - adding 0.1 brightness, 0.1 contrast, and 0.1 saturation.
    - Random affine - affine transformations like shear effects in the image.

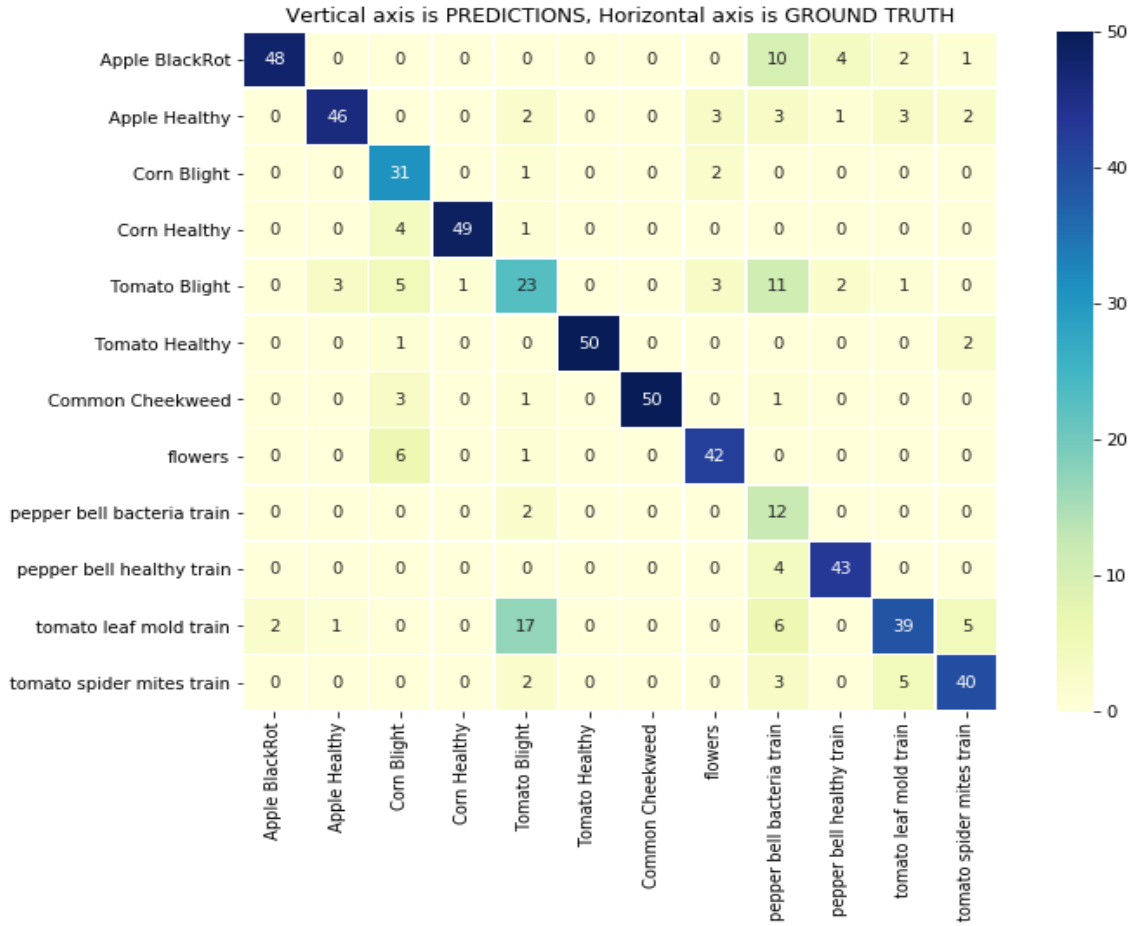
- Have used Ax's Bayesian Optimization to get the best Learning rate and Momentum parameters.
- Best LR found is 0.00203 and best momentum found is 0.6395.
- After training, the **accuracy** on the test dataset is **77.5%** which is way better than the untuned baseline model.
- 

Observations on the tuned-baseline model:

- We can see that the training loss is properly degrading this time.
- Also the validation loss is degrading in a better manner.
- 30 epochs are enough to get a saturated/converged train and validation loss.
- Validation loss is more than the training loss suggesting overfitting. This can be due to the fact that so many transformations are done on a simple network while not enough transformations are present in the validation set to validate against. Perhaps more dropouts probability is needed. But this is the case which is mostly due to the absence of regularization parameter(weight decay).

Validation set accuracy plot:





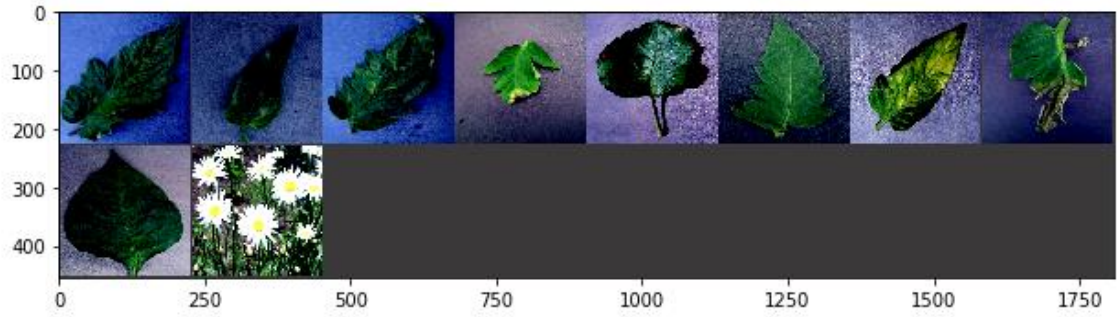
-
- **Confusion Matrix Observations**:

- The pepper bell bacteria is being wrongly identified with Tomato blight(with 11 images out of 50 misclassified) which is a challenge because both the leaves look similar and blight and mould infection patches look similar.
- Also, 10 pepper bell bacteria leaves have been wrongly identified as Apple black-rot.
- I can deduce that it didn't do well on the challenge.
- Also corn blight and flowers have been mixed which should be very orthogonal.
- Also mould infection looks similar to blight infection and we can see that 17 tomato blight images have been misclassified as tomato leaf mould.
- We need a better architecture: such ResNet, VGG16 etc.
- Strangely, some corn blights are being confused with flowers, which could have been at least 100% accurate.
- Top 2 error rate is 9.5 which is lesser than untuned baseline model (13.66)

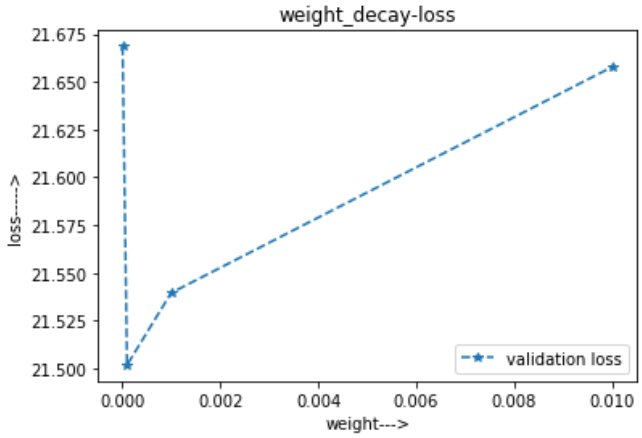
Transfer Learning using Resnet50:

- Resnets are residual networks which use skip connections between layers.
- Skip connections are state-of-the-art techniques to train very deep neural networks effectively.
- In skip connection activated non-linear output from earlier layer can be fed as input to a deeper layer by adding to its linear input. thereby skipping some layers in between.
- This has been done to get around the problems of overfitting where a higher regularization parameter is demanded by the network to compensate for the extreme overfitting (which rises due to very deep architecture). However, high regularization parameter ends up shrinking the weights in the linear input to almost 0, thus the network loses the ability to learn in the depths of its architecture. So, by adding a skip connection, even if the deeper layer is learning very less from its preceding weights, it will get the benefit from the earlier layer's activated values. So, skip connections doesn't harm but only does good or does nothing.
- Resnet50: Each two layer block in the 34 layer architecture is replaced with a 3-layer bottleneck block which results in the formation of Resnet50.

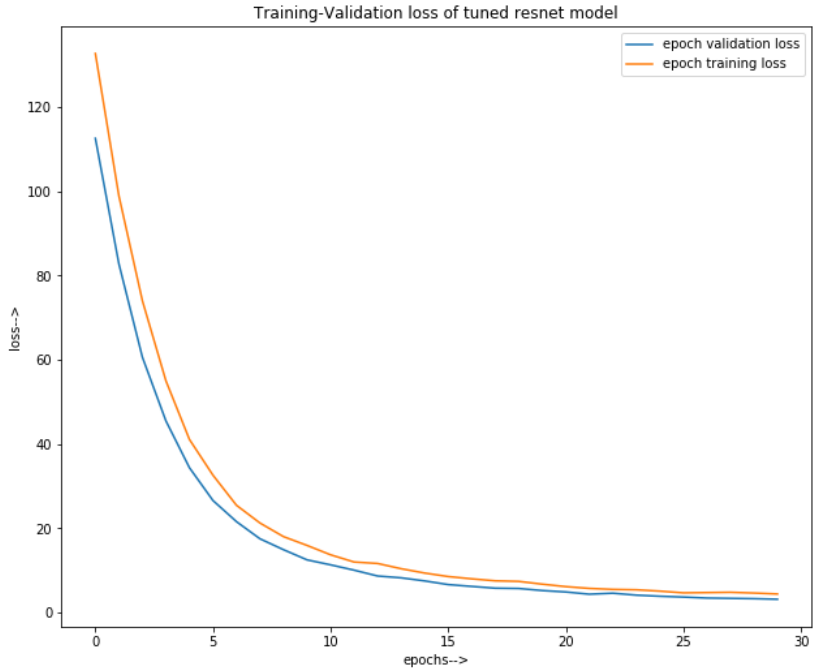
Visualization of the data that is to be fed into the network:



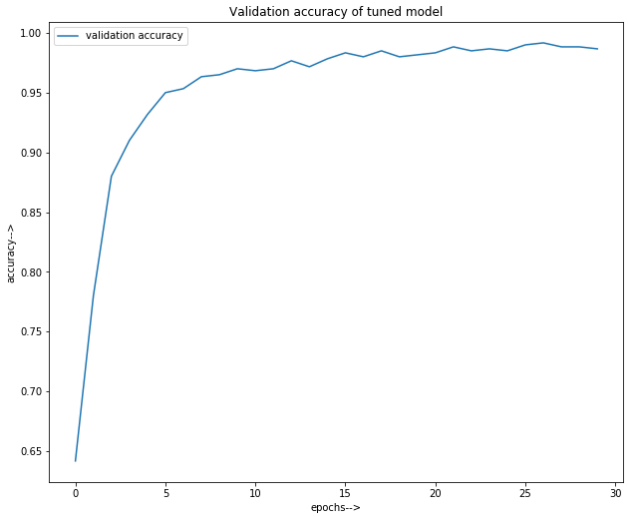
- I have used Ax's Bayesian Optimization to tune the learning rate and momentum.
- Best Learning Rate was found to be 0.002014 and the best momentum was found to be : 0.013033
- After this, I applied **3-Fold cross validation** on the train dataset which is 12classes X 200 =2400 images in total to tune **weight_decay.**
- Thus each held-out validation in a fold will be of size 2400/3 = 800 images and train images will be 1600.
- I have chosen 20 epochs per fold.
- All the weight decay candidates are: [0.00001,0.0001,0.001,0.01]
- Keeping LR and momentum constant which were optimized by Bayesian Optimization above. Since it will take a lot of time to optimize all the 3 hyper-parameters together, I am going to do it serially.
- This is how the best weight decay was chosen from the validation loss of each candidate.



- From Bayesian Optimization, we got Lr = 0.002014, and momentum = 0.0130339
- From 3-Fold cross validation, we got weight decay = 0.0001
- The **accuracy** was found to be **97.5%** after fitting all the tuned parameters and retraining the model.



Accuracy plot of Validation set:



- Accuracy has converged which can be explained from validation loss in the above graphs.

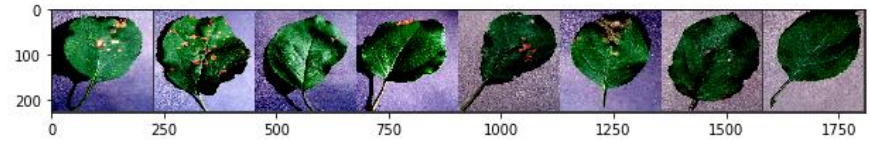
Observation:

- We get state-of-the-art results using fine-tuned Resnet over base model.
- Accuracy is 97.5 % while with baseline tuned model, it was 77.5%.
- Also, since validation loss and train loss go together down, we can say that there is no overfitting.
- Also, the top-2 error rate is 0.0 for Resnet50 this means that every time, the ground truth was present in the top 2 predictions at least.

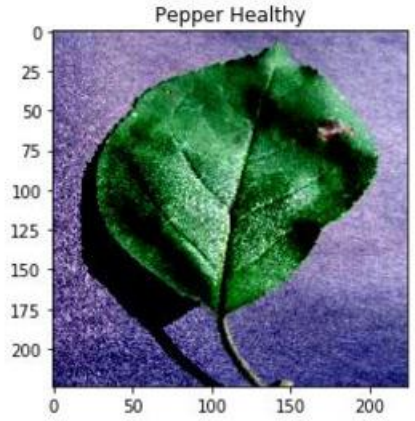
Misclassification by Resnet50:

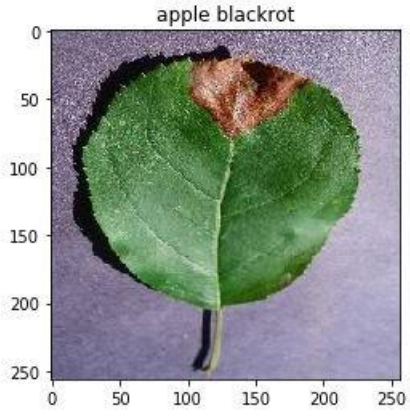
```
{2: ('pepper bell healthy train', 'Apple BlackRot'),
 33: ('Tomato Blight', 'Apple BlackRot'),
 85: ('tomato spider mites train', 'Apple Healthy'),
 86: ('Tomato Blight', 'Apple Healthy'),
 406: ('pepper bell healthy train', 'pepper bell bacteria train'),
 531: ('Tomato Blight', 'tomato leaf mold train'),
 532: ('tomato spider mites train', 'tomato leaf mold train'),
 585: ('Tomato Blight', 'tomato spider mites train')}
```

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



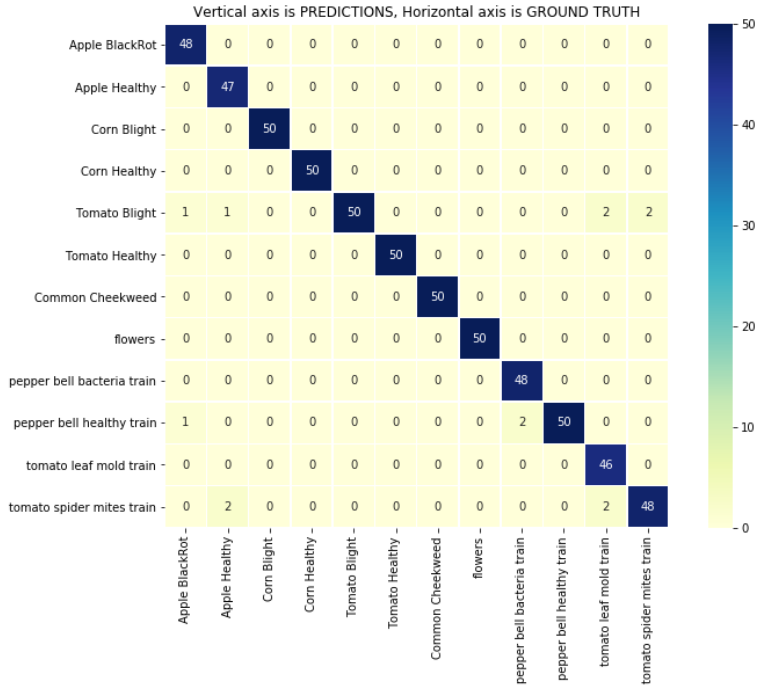
- These are the misclassified images.





- We can see that pepper bell healthy image is being confused with an apple blackrot. This may happen due to both being very similar looking leaves.

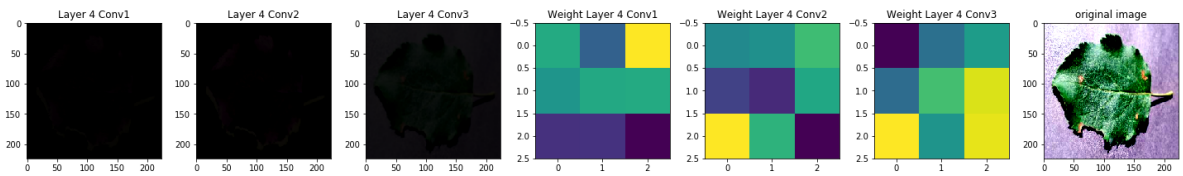
- **Confusion Matrix:** Vertical axis is predictions, horizontal axis is true labels.



Observations:

- Pepper bell bacterial infection is not being confused with tomato blight infection. This passes our challenge where both the infections look similar and both the leaves also look similar.
- Also pepper bell bacterial infection is not being confused with apple black-rot disease.
- Corn blight is not being confused with flowers.
- Also tomato blight images have not been confused with tomato mould images where both the disease symptoms look similar.
- Our model's challenge has been achieved!

Visualizing the weights and the linear operation in the layer 4 in Resnet:

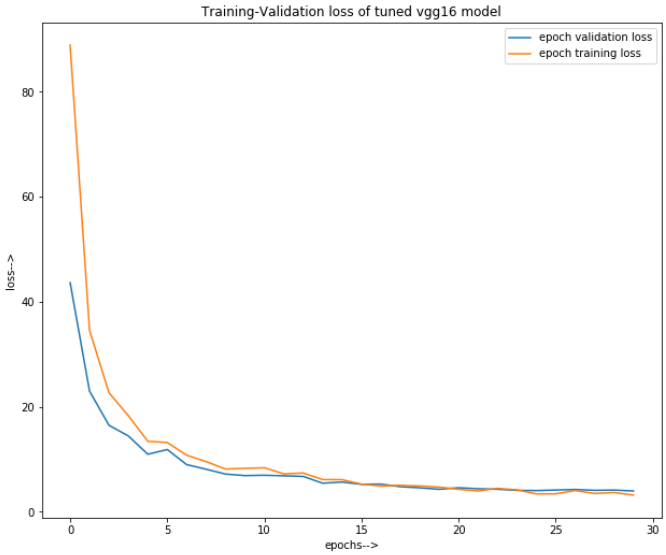


Observations:

- The weights have been pulled from the last layer i.e. layer 4 of Resnet50. All these wights are 3X3 size.
- If we look closely then the action of weights on the leaf image can give us an intuition of what is happening in the layer 4 of Resnet50.
- The last convolution layer (conv3) in the layer 4 with weight size 3X3 gives us a better image. While layer 4 conv1 and conv2 give us the edges.

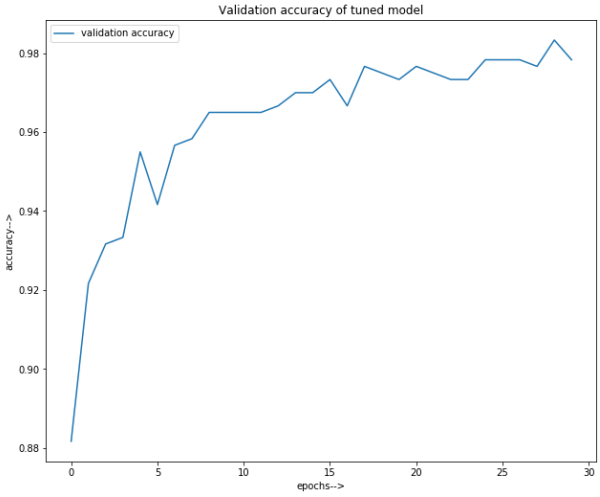
Now, VGG16 with batch normalization:

- Vgg16-batch normalization is a deep neural architecture which uses batch normalization to avoid exploding gradients. There are 16 layers with 3X3 kernels only. Around 140 million parameters exist throughout the network.
- Vgg16 is slower than ResNet50.
- I have used Ax's Bayesian Optimization to tune the learning rate and momentum.
- Best Learning Rate was found to be 0.0054 and the best momentum was found to be : 0.3691
- After this, I applied **3-Fold cross validation** on the train dataset which is 12classes X 200 =2400 images in total to tune **weight_decay.**
- Thus each held-out validation in a fold will be of size 2400/3 = 800 images and train images will be 1600.
- I have chosen 20 epochs per fold.
- All the weight decay candidates are: [0.00001,0.0001,0.001,0.01]
- Keeping LR and momentum constant which were optimized by Bayesian Optimization above. Since it will take a lot of time to optimize all the 3 hyper-parameters together, I am going to do it serially.
- From 3-Fold cross validation, we got weight decay = 0.0001
- The **accuracy** was found to be **98%** after fitting all the tuned parameters and retraining the model.



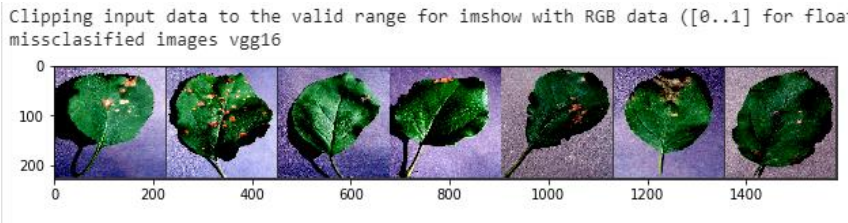
- Very similar to what we have seen in ResNet50. But, the final training loss is even lower in case of VGG16.
- Both VGG16 and Resnet50 have the same accuracy 97.5%
- But the top-2 error rate of VGG16 is 0.5 while Resnet50 has 0.0
- Indeed VGG16 does exceptionally well than the baseline model.
- The starting validation loss is lower in vgg16 than in Resnet50.
- Since both the validation loss and the training loss go down together, it can be said that there is almost no overfitting.

Accuracy plot of validation data:



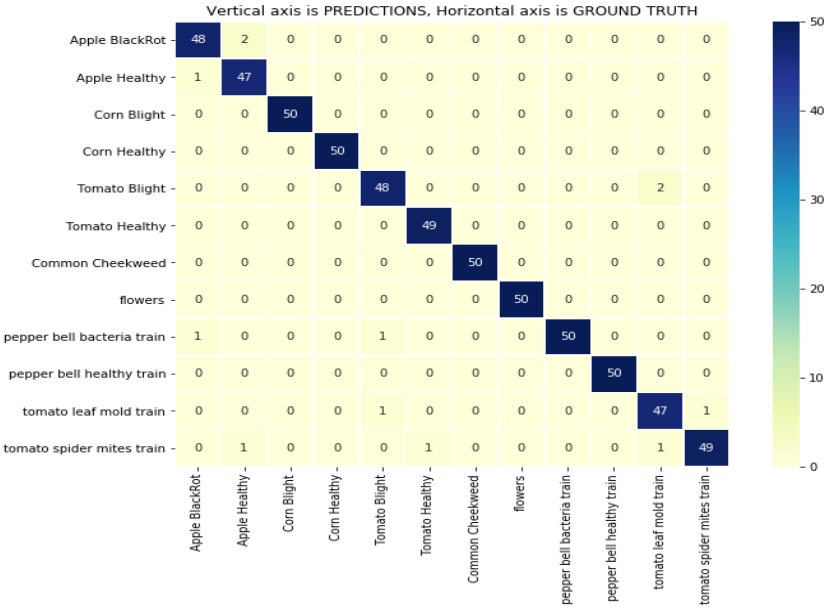
Misclassifications:

```
{86: ('Apple BlackRot', 'Apple Healthy'),
 93: ('tomato spider mites train', 'Apple Healthy'),
 227: ('Tomato Healthy', 'Tomato Blight'),
 238: ('tomato spider mites train', 'Tomato Blight'),
 504: ('Tomato Blight', 'tomato leaf mold train'),
 519: ('tomato spider mites train', 'tomato leaf mold train'),
 563: ('Tomato Healthy', 'tomato spider mites train')}
```



- For example: It confuses Apple Healthy with Apple black-rot.
- Again, It misclassifies Apple Healthy with Tomato spider mites. etc.

Confusion Matrix:



Observations:

- Leaf Mould infection and blight infection have similar symptoms, so it's a challenge for the architecture to properly distinguish these two. 5 images of Tomato Leaf Mould have been confused with Tomato Blight compared to Resnet50 which got 2 images of tomato leaf mould confused with tomato blight. While our baseline model got only one tomato leaf mould wrongly identified as tomato blight. Surprisingly, our baseline is doing better on this challenge.
- Pepper Bell bacterial infection also resembles blight disease. VGG16 got all the pepper bell bacterium images correct while ResNet50 got 2 confused with pepper bell healthy. And the baseline model got 5 of them confused with tomato blight disease. In this challenge, VGG16 is the winner.
- VGG16 has got 13 images wrong, ResNet50 also has got 13 images wrong while Baseline tuned model has got 135 images wrong.
- Final training loss of VGG16 is the lowest while baseline model's loss is the highest

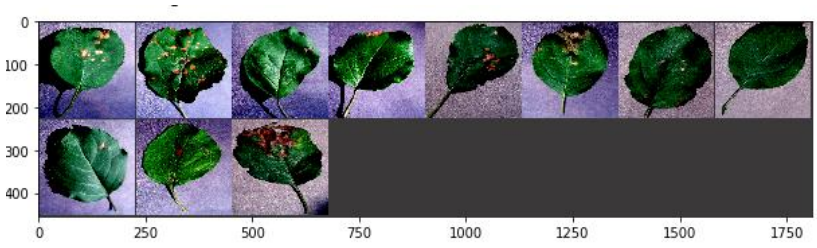
- Test set accuracy for VGG16 is 97.83%, resnet50 is 97.83% and for baseline model is 77.5%
- Also, looking at the curve, baseline is overfitting as the validation loss is more than the training loss and saturates/converges early while training loss keeps decreasing. In case of VGG16 and Resnet50, there is no overfitting as the curves go hand-in-hand.

Alexnet:

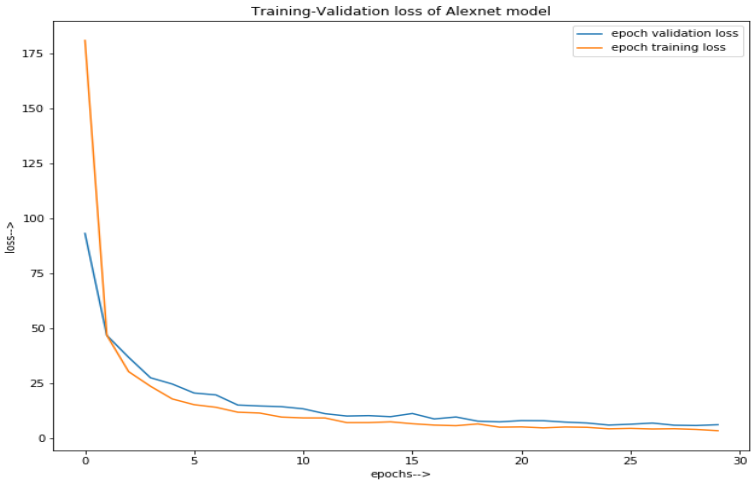
- I have chosen Alexnet here because it was the first among all the deeplearning models in the imagenet competition. So, it should provide as a benchmark for the other models.
- Best values for parameters learning rate and momentum (as given by Bayesian Optimization) : 0.00204939 and 0.0048511 .

Misclassified images:

```
{52: ('Tomato Healthy', 'Apple Healthy'),
 65: ('Apple BlackRot', 'Apple Healthy'),
 75: ('tomato leaf mold train', 'Apple Healthy'),
 211: ('tomato leaf mold train', 'Tomato Blight'),
 216: ('tomato spider mites train', 'Tomato Blight'),
 247: ('pepper bell bacteria train', 'Tomato Blight'),
 406: ('Apple BlackRot', 'pepper bell bacteria train'),
 407: ('pepper bell healthy train', 'pepper bell bacteria train'),
 466: ('pepper bell bacteria train', 'pepper bell healthy train'),
 519: ('tomato spider mites train', 'tomato leaf mold train'),
 544: ('Tomato Healthy', 'tomato leaf mold train')}
```



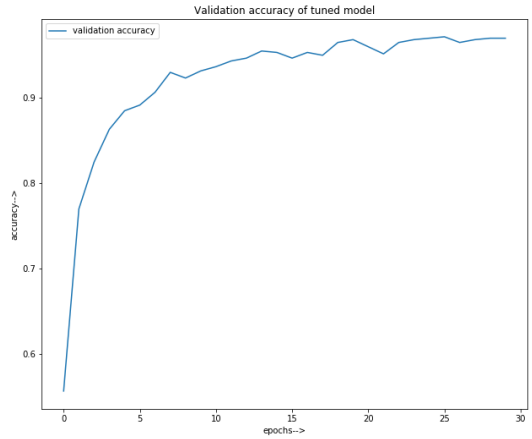
Training loss and validation loss:



Observations:

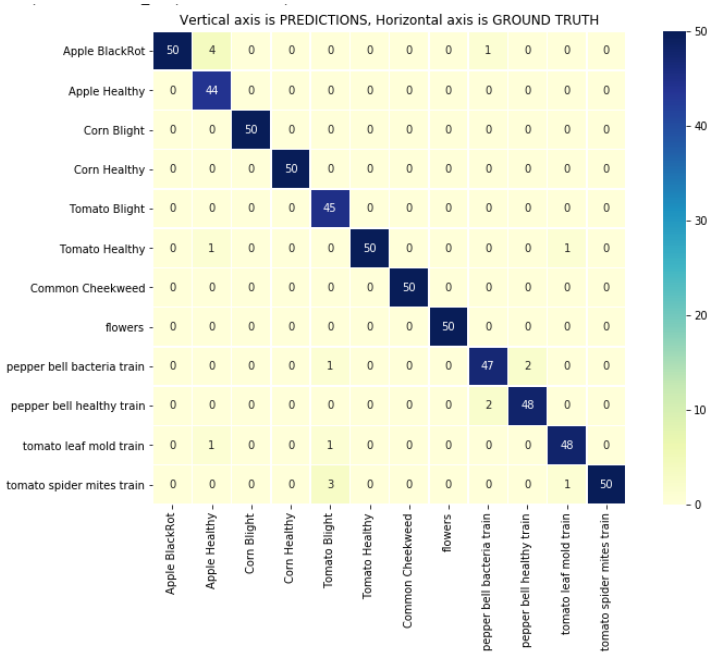
- Validation loss starts at a smaller number than the training loss**.**
- At around 17-18 epochs, both the curve converge.
- No sign, of overfitting because both the curves go down together.

Accuracy:



- The accuracy on the test set is found to be 96.33%
- The top 2 error rate is 0.833 which means that out of 100 times, on an average, 0.833 times the ground truth is absent in the top 2 predictions.

Confusion matrix:



Observations:

- Alexnet doesn't confuse Pepper Bell bacterial infection with tomato blight. But, tomato blight is being confused with pepper bell infection.
- It also doesn't confuse tomato leaf mould with tomato blight disease.
- Accuracy is slightly less than Resnet50 and Vgg16.
- With just 1.2% less accuracy but faster training time, it can be considered as a better model than VGG16. However, VGG16 uses batch normalization which is required to normalize within the network.

Final Comparison:

Model	Accuracy	Top-2 error rate	Final Epoch loss : Training	Final Epoch loss: Validation	Average Epoch time (in seconds)
Baseline untuned	0.738	13.66	40.72	51.22	20.5
Baseline tuned	0.775	9.5	9.2790	48.19	20.5
Resnet50	0.975	0.0	4.628	3.856	23.5
VGG16	0.975	0.5	4.0882	3.884	27
Alexnet	0.963	0.833	3.61	6.23	18.5

Performance of the models on the challenges:

1) Challenge 1: Confusing pepper bell bacterial infection with Tomato Blight and vice-versa:

- Baseline tuned model:
    - 5 pepper bell infection have been wrongly classified as tomato blight disease.
    - 4 tomato blight disease have been wrongly classified as pepper bell infection.
- Resnet50 :
    - None of the pepper bell infections has been misclassified as tomato blight.
    - None of the tomato blight disease has been wrongly classified as tomato blight disease.
- VGG16:
    - None of the pepper bell infections has been misclassified as tomato blight.
    - None of the tomato blight disease has been wrongly classified as tomato blight disease.
- Alexnet:
    - None of the pepper bell infections has been misclassified as tomato blight.
    - Only 1 tomato blight disease has been misclassified as pepper bell infection.

2) Challenge 2: Confusing Tomato Blight disease with Tomato Mould infection and vice-versa:

- Baseline tuned model:
    - 7 Tomato blight disease have been misclassified as tomato mould infection.
    - 1 Tomato mould has been misclassified as tomato mould infection.
- Resnet50:
    - None of the tomato blight has been wrongly classified as tomato mould disease.
    - 2 of the tomato mould disease have been misclassified as tomato blight disease.
- VGG16:
    - None of the tomato blight has been wrongly classified as tomato mould disease.
    - 5 of the tomato mould have been misclassified as tomato blight disease.

- Alexnet:
    - 1 tomato blight has been misclassified as tomato mould infection.
    - None of the tomato mould has been misclassified as tomato blight disease.

Best Model and conclusion:

- As compared to vgg16, although both give the same accuracy, the time taken in each epoch of Resnet50 is 23.5 seconds and in case of vgg16 it is 27seconds.
- Also, the type 2 error of Resnet50 is 0 this means that every time, the ground truth was present in the top 2 predictions at least.
- Also, vgg16 is a bulky model.
- If compared to Alexnet, it doesn't have batch normalization. This can be a problem with different datasets. In our case, there is perhaps no scope of exploding gradients, and as a result, Alexnet does pretty well on the data with 97% accuracy.
- Theoretically, with increasing layers, performance should increase in deep learning models. But, vanishing gradient makes it practically impossible to create very deep models. With the advent of Restnet where skip connections are used, the deeper layers can get the input from earlier layers and continue to work without worrying about vanishing gradients. So, for deeper network tasks, Resnet is the ideal choice.
- Type 2 error rate has helped to break tie between VGG16 and Resnet accuracy scores because both have the same numbers.

References:

Prof. Roderick Murray-Smith, Deep Learning, Lab 3, University of Glasgow.

Pytorch tutorial: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

Alexnet: https://www.learnopencv.com/understanding-alexnet/

VGG16: https://neurohive.io/en/popular-networks/vgg16/

Resnet: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4