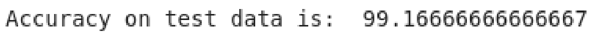
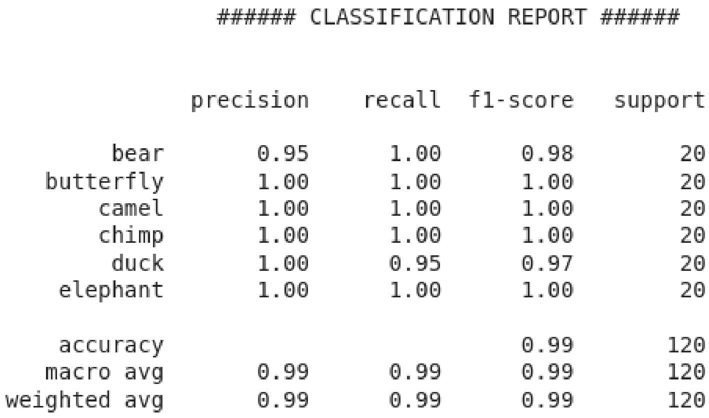
Q2) IMAGE CLASSIFICATION

(a) k-NN:

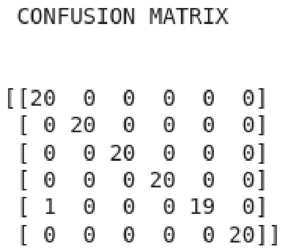
- The given data is normalized using the mean and standarad deviation of ImageNet data and appropriate transforms are applied on train and test datasets
- Pretrained ResNet18 model is used with pretrained weights and the gradients are freezed for evaluation
- Feature vectors of both train and test data are extracted from the inputs of last fully connected layer
- Now we have feature vectors and labels of both the train and the test data
- k-NN classifier is used from scikit-learn library with k = 50 and obtained an accuracy of over **95%** over multiple runs on the test data
- To plot the decision regions, PCA is used to reduce the dimensionality to 2 and k-NN is again used to fit the newly created 2 dimensional feature vectors of the train data
- **Results**
    - Accuracy is good because we are extracting features from the pretrained model and using them for classification
    - Different values of k are used for k-NN classifier and best accuracy is obtained when k = 50 i.e **99%**
    - Observed that the decision regions and accuracy are changing over multiple runs for the same data



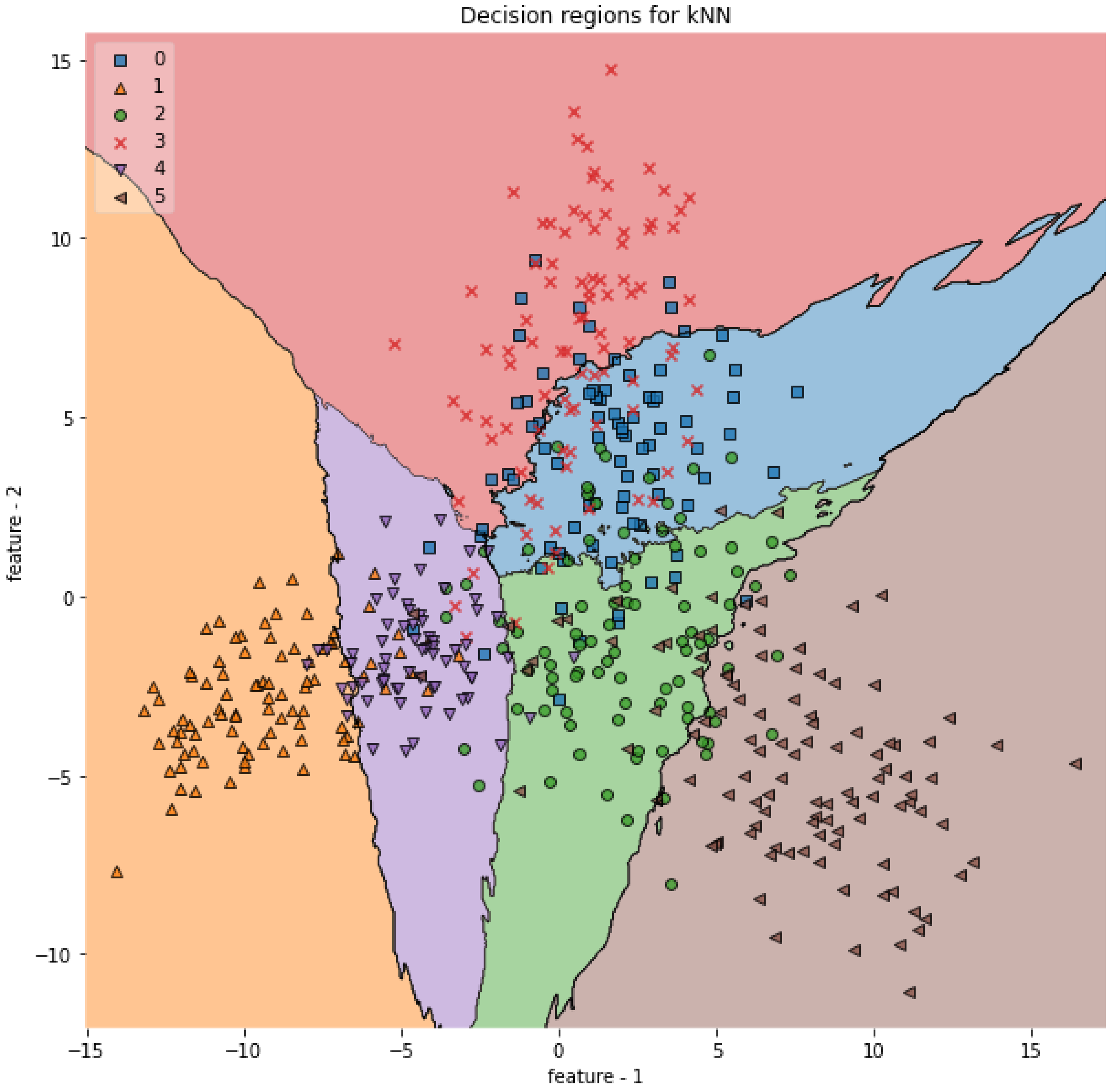
```
Accuracy on test data is:  99.16666666666667
```



```
                ###### CLASSIFICATION REPORT ######


              precision    recall  f1-score   support

        bear       0.95      1.00      0.98        20
   butterfly       1.00      1.00      1.00        20
       camel       1.00      1.00      1.00        20
       chimp       1.00      1.00      1.00        20
        duck       1.00      0.95      0.97        20
    elephant       1.00      1.00      1.00        20

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```



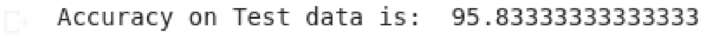
```
 CONFUSION MATRIX


[[20  0  0  0  0  0]
 [ 0 20  0  0  0  0]
 [ 0  0 20  0  0  0]
 [ 0  0  0 20  0  0]
 [ 1  0  0  0 19  0]
 [ 0  0  0  0  0 20]]
```



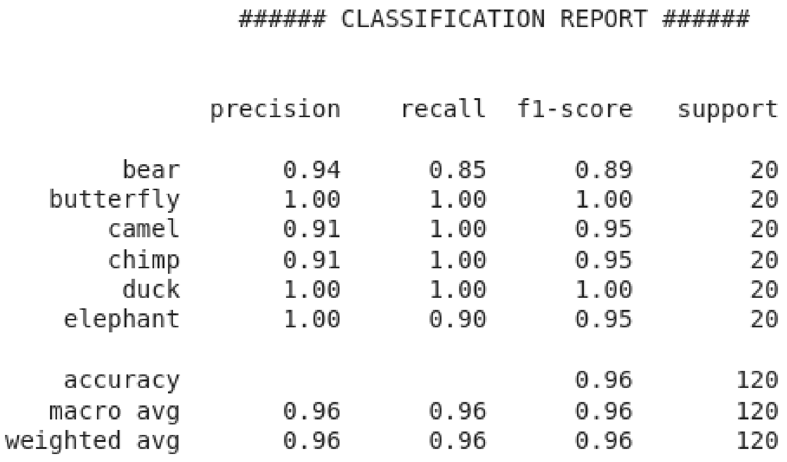
(b) Finetuning:

- Validation dataset is created from train data by splitting it in 80:20 ratio. Rest of the process is done using train, validation and test datasets thus obtained
- The given data is normalized using the mean and standarad deviation of ImageNet data and appropriate transforms are applied on train, validation and test datasets
- ResNet18 model is designed for 1000 classes. Last fully connected layer of ResNet18 model is removed and a new fully connected layer is added with 6 classes
- Pretrained weights are used and gradients are freezed except for the last newly added layer
- CrossEntropy loss is evaluated and is optimized using SGD optimizer
- Accuracy on test data is over **95%** over multiple runs on the test data. Best valiation data accuracy is **98%** in 25 epochs

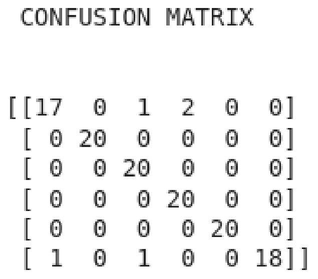
- **Results**
  - Accuracy is good because we have just finetuned the pretrained model



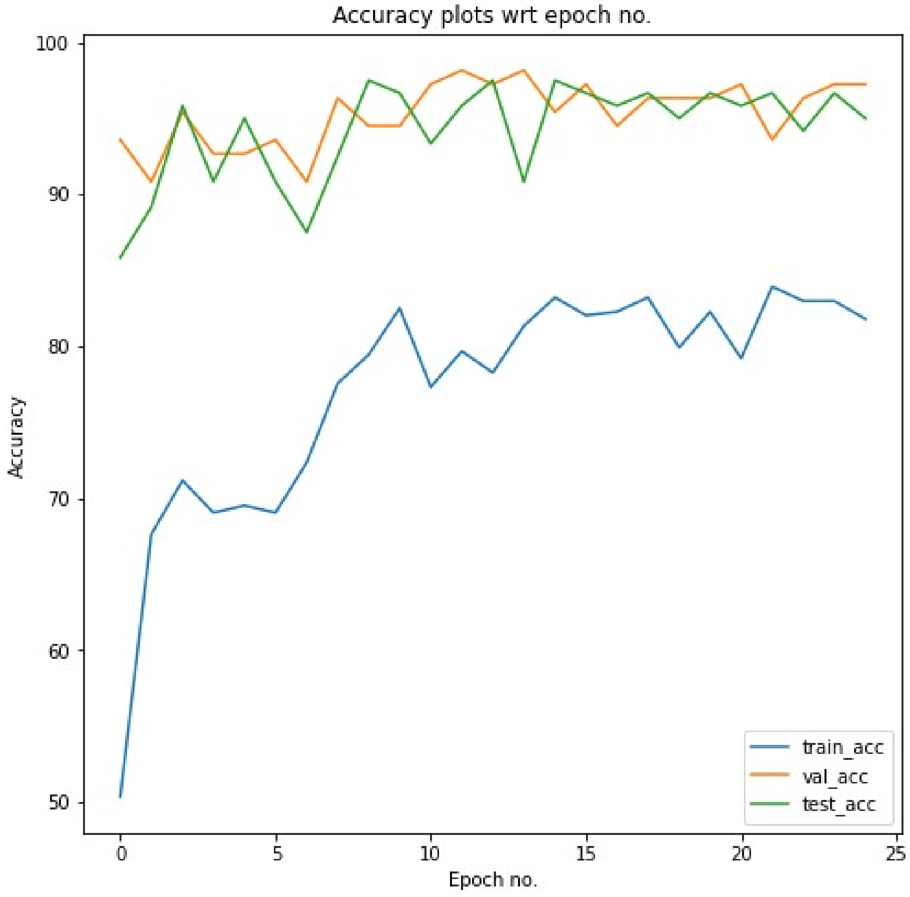
```
Accuracy on Test data is:  95.83333333333333
```



```
                ###### CLASSIFICATION REPORT ######

              precision    recall  f1-score   support

        bear       0.94      0.85      0.89        20
   butterfly       1.00      1.00      1.00        20
       camel       0.91      1.00      0.95        20
       chimp       0.91      1.00      0.95        20
        duck       1.00      1.00      1.00        20
    elephant       1.00      0.90      0.95        20

    accuracy                           0.96       120
   macro avg       0.96      0.96      0.96       120
weighted avg       0.96      0.96      0.96       120
```



```
 CONFUSION MATRIX

[[17  0  1  2  0  0]
 [ 0 20  0  0  0  0]
 [ 0  0 20  0  0  0]
 [ 0  0  0 20  0  0]
 [ 0  0  0  0 20  0]
 [ 1  0  1  0  0 18]]
```



Some of the few test images with predicted labels are as follows:

predicted: camel

predicted: bear

predicted: bear

predicted: butterfly

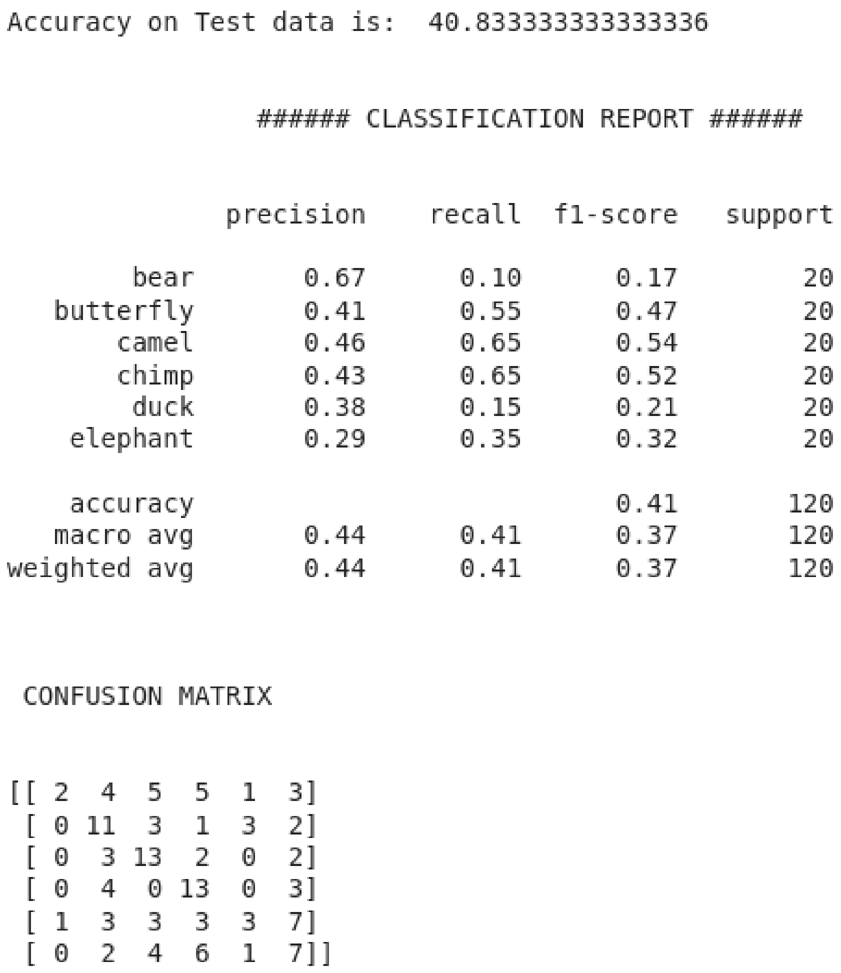
predicted: chimp

predicted: bear

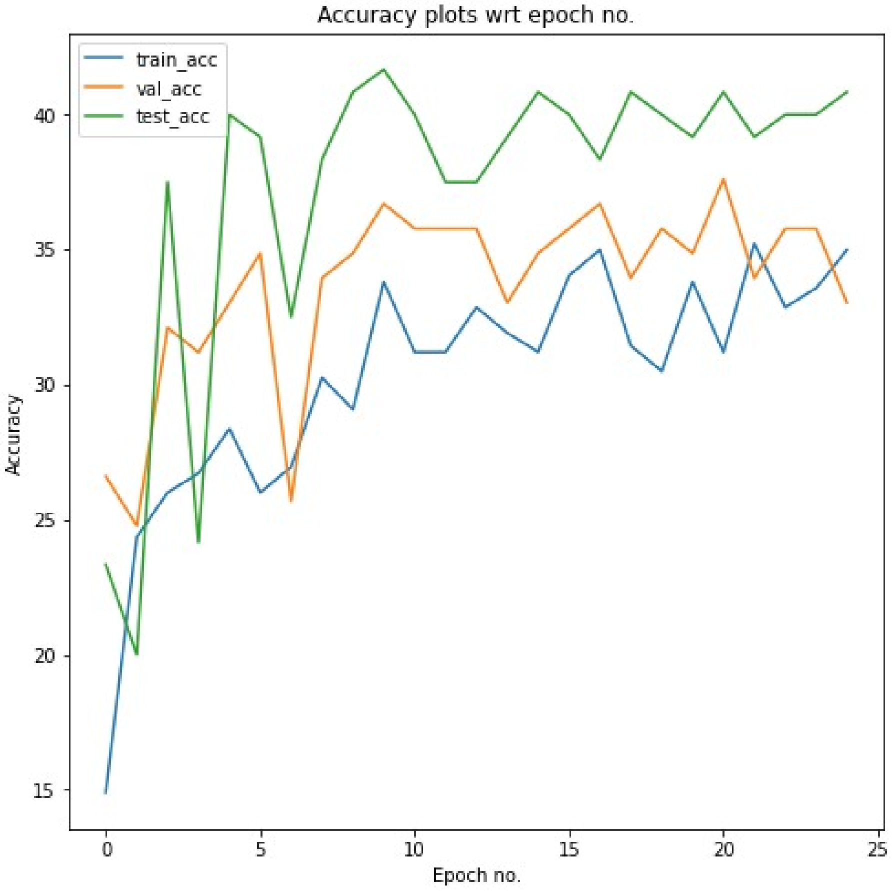
(c)Simple Neural Network:

- Validation dataset is created from train data by splitting it in 80:20 ratio. Rest of the process is done using train, validation and test datasets thus obtained
- Mean and standard deviation of the data are calculated to be [0.47949, 0.4665, 0.4086] and [0.2849, 0.2770, 0.2849]. Using these values, Data normalization is done and transforms are applied to train, validation and test datasets

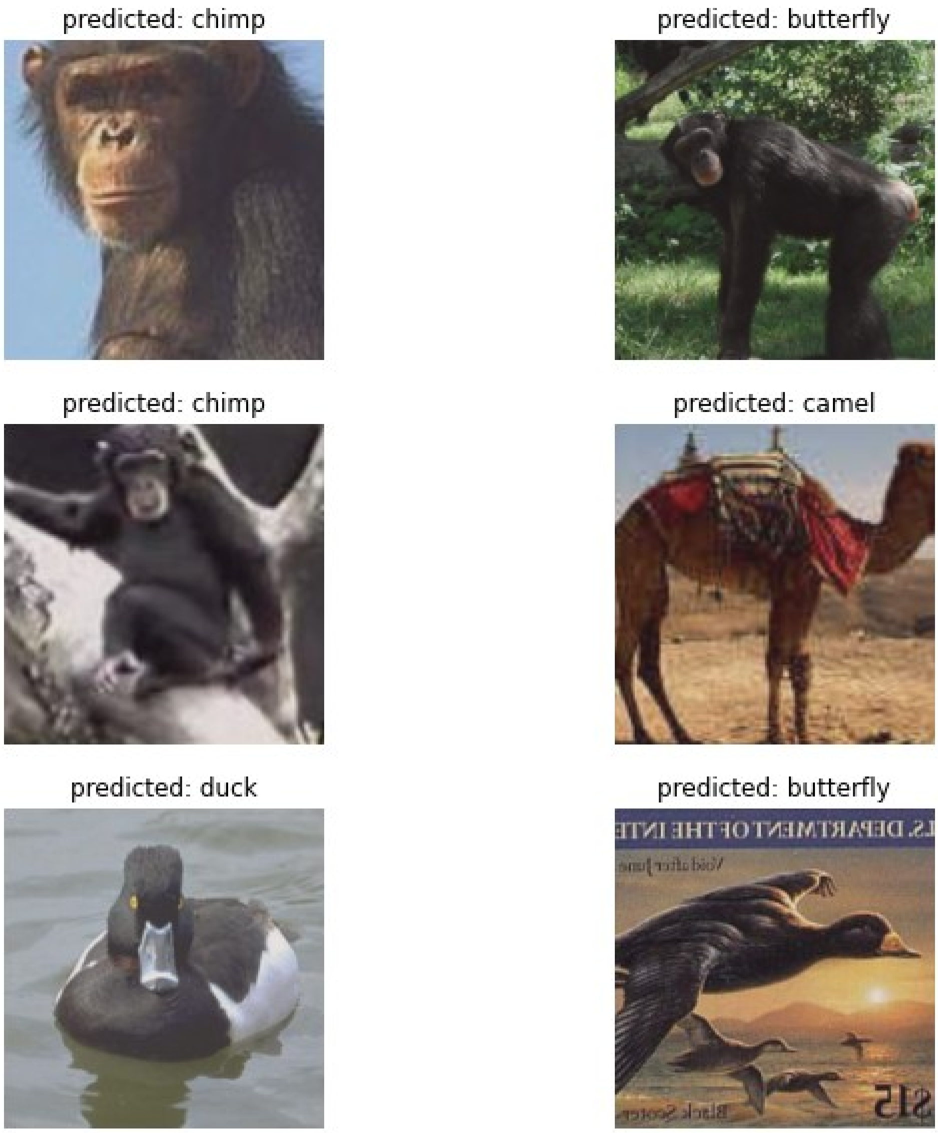
- A simple Neural Network is build using the Module class of pytorch
    - Layer 1 :
        - Convolution : 224 * 224 * 3 with 5 * 5 kernels (16 kernels) with stride=1, padding = 2
        - Batchnormalization: 16 kernals
        - ReLU Activation Function
        - Max - pooling (kernel : 5 *5 , stride : 2)
    - Layer 2 :
        - Convolution : 112 * 112 * 16 with 5 * 5 kernels (32 kernels) with stride=1, padding = 2
        - Batchnormalization: 32 kernals
        - ReLU Activation Function
        - Max - pooling (kernel : 5 *5 , stride : 2)
    - Layer 3 :
        - Convolution : 56 * 56 * 32 with 5 * 5 kernels (64 kernels) with stride=1, padding = 2
        - Batchnormalization: 64 kernals
        - ReLU Activation Function
        - AdaptiveAvgPool : output-size (1,1)
    - Layer 4 :
        - Fully Connected layer with Linear classifer (input : 64, output: 6 (classes are 6))
- CrossEntropy loss is evaluated and is optimized using SGD optimizer
- **Results:**
    - Accuracy on test data is over **35%** over multiple runs on the test data. Best validation data accuracy is **41%** in 25 epochs
    - We can see from the images that the accuracy is not very good and this is because the no. of layers is less and the data is insufficient to properly train the model
    - The training accuracy is less initially and is increasing as the no. of epochs increase. This follows the same even when pretrained model is finetuned which indicates that the model is being trained as the no. of epochs increase



```
Accuracy on Test data is:  40.833333333333336

                ###### CLASSIFICATION REPORT ######

              precision    recall  f1-score   support

        bear       0.67      0.10      0.17        20
   butterfly       0.41      0.55      0.47        20
       camel       0.46      0.65      0.54        20
       chimp       0.43      0.65      0.52        20
        duck       0.38      0.15      0.21        20
    elephant       0.29      0.35      0.32        20

    accuracy                           0.41       120
   macro avg       0.44      0.41      0.37       120
weighted avg       0.44      0.41      0.37       120


 CONFUSION MATRIX

[[ 2  4  5  5  1  3]
 [ 0 11  3  1  3  2]
 [ 0  3 13  2  0  2]
 [ 0  4  0 13  0  3]
 [ 1  3  3  3  3  7]
 [ 0  2  4  6  1  7]]
```



Some of the test images with the predicted labels are as follows:



Final Conclusions:

- Observed that Finetuning a pretrained model and extracting features from a pretrained model work better than building a new simple Neural Network and training it using our data
- Finetuning showed an accuray of ~ **95%** and using k-NN on the features extracted from a pretrained model showed an accuracy of ~**99%** but the Neural Network that is build from scratch showed an accuray of ~**40%**
- Adam Optimizer was also used to review the performance but didnt find much of a difference from SGD Optimizer (in this case)
- Observed that the accuracy is changing everytime the models are run from scratch which implies that the training is different even with the same data (probably due to shuffling and random split)

References:

1.https://pytorch.org/vision/stable/generated/torchvision.models.feature_extraction.create_feature_extractor.html#torchvision.models.feature_extraction.create_feature_extractor

2.https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

3. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

4. AIP tutorial

5. scikit-learn documentation

6. pytorch documentation