

“Monkey Breed Classification”

1. Project Definition

1.1 Project Overview:

Image classification is one of the most trending field in recent years. It helps people to get correct information about the image. E.g. Lots of breeds of flowers are available in the world. So human finds very difficult to remember each name of species of flower. Here image classification plays important role. By capturing photo , new techniques will directly tell you the correct name of flower species. Similarly it is applicable to all types of animals, fruits etc.

Here in this project , I am going to discuss breed of monkeys. At the end of this project, we can classify any monkey to its breeds.

1.2 Problem Statement:

In this project, as mentioned above image classification of monkeys images is to be done. Mainly there are 10 types of monkey breeds are available. Our model will specify given monkey image to its correct breeds.

1.3 Metrics:

Performance of this project is measured in terms of models accuracy to predict correct classification. Accuracy is measured by measuring correct classification divided by total no of images given.

2. Analysis:

2.1 Data Exploration:

For implementing monkey breed classification, I consider dataset available on kaggle. I consist of 10 species of monkey. Following are the actual dataset details :











Label	Latin Name	Common Name	Train Images	Validation Images
n0	alouatta_palliata	mantled_howler	131	26
n1	erythrocebus_patas	patas_monkey	139	28
n2	cacajao_calvus	bald_uakari	137	27
n3	macaca_fuscata	japanese_macaque	152	30
n4	cebuella_pygmea	pygmy_marmoset	131	26
n5	cebus_capucinus	white_headed_capuchin	141	28
n6	mico_argentatus	silvery_marmoset	132	26
n7	saimiri_sciureus	common_squirrel_monkey	142	28
n8	aotus_nigriceps	black_headed_night_monkey	133	27
n9	chypithecus_johnii	nilgiri_langur	132	26

Dataset consist of total 1642 images. These images is divided into two caregories. Training Images and validation images. Training folder consist of 1370 images and validation images consist of 272 images. Images inside testing folder are taken randomly from internet. Click [here](#) to check dataset at kaggle.

2.2 Data Visualization:

Data consist of total 1642 images with 10 species. All are colored images with different size. Also images consist of various backgrounds, orientation. Following are the details

mantled_howler	patas_monkey	bald_uakari	bald_uakari
----------------	--------------	-------------	-------------

			
pygmy_marmoset	white_headed_capuchin	silvery_marmoset	common_squirrel_monkey
			
black_headed_night_monkey	nilgiri_langur		
			

3. Methodology

3.1 Data Preprocessing:

Our dataset consist of three folders.

- Training
- Testing

- Validation

Each images in folder is preprocessed before applying to neural network, it is preprocessed in following ways:

1. Random Rotation: Only images in training dataset are preprocessed by this method. Here images from training dataset are rotated to 30 degree. Reason to rotate images is to make train network in various way.
2. Random Resize Crop: All images from each dataset are preprocessed by this step. It crops the given images to mentioned size. I cropped each images to 224 size.
3. Random Horizontal Flip: All images from each dataset are preprocessed by this step. It flip randomly given images with a given probability. I am not mentioned any value for probability but given function takes value of 0.5 as a default value.
4. Transform to tensor: All images from each dataset are preprocessed by this step. Each images are transformed to tensor.
5. Transform Normalise: All images from each dataset are preprocessed by this step. Now by applying step no. 4 in preprocessing, each image is in tensor format. This step normalizes given images to mentioned mean and standard deviation.

Each image is preprocessed by step 2 to step 5. Only training image is goes through all steps.

[illegible]

```
transforms.Normalize([0.485, 0.456, 0.406],
                    [0.229, 0.224, 0.225]))

train_data = datasets.ImageFolder(train_dir, transform=data_transforms_train)
validation_data=datasets.ImageFolder(valid_dir, transform=data_transforms_validation)
test_data = datasets.ImageFolder(test_dir, transform=data_transforms_test)

trainloader = torch.utils.data.DataLoader(train_data, batch_size=64, shuffle=True)
testloader = torch.utils.data.DataLoader(test_data, batch_size=64)
validloader=torch.utils.data.DataLoader(validation_data, batch_size=64)
```

3.2 Implementation:

Project mainly consists of two parts:

1. Training: For doing classification, it is necessary to train the model. Following diagram show the flowchart of training procedure.



2. Testing: Any images which we want to classify is applied in following ways:



3.3Refinement:

There is no standard procedure for defining parameters of model of neural network. So the result is tested by taking some trial and error procedure for finding best match parameters of neural network.

Initially I trained model with following parameters. But I am not getting result in terms of accuracy of validation and testing dataset.

Initial Parameters of Neural Network

```
classifier = nn.Sequential(OrderedDict([
    ('fc1', nn.Linear(1024, 500)),
    ('relu1', nn.ReLU()),
    ('fc3', nn.Linear(500, 10)),
    ('output', nn.LogSoftmax(dim=1))
]))
criterion = nn.NLLLoss()
model.classifier = classifier
optimizer = optim.Adam(model.classifier.parameters(), lr=0.01)
```

As mentioned earlier, it doesn't get satisfying result, after some trial and error I Choose following parameters. Dropout layer is added in final model to reduce time and complexity during training.

Final Parameters of Neural Network

```
from collections import OrderedDict
classifier = nn.Sequential(OrderedDict([
    ('fc1', nn.Linear(1024, 500)),
    ('relu1', nn.ReLU()),
    ('drpot1', nn.Dropout(0.2)) ,
    ('fc2', nn.Linear(500, 250)),
    ('relu2', nn.ReLU()),
    ('drpot2', nn.Dropout(0.2)) ,
    ('fc3', nn.Linear(250, 10)),
    ('output', nn.LogSoftmax(dim=1))
]))
```

```
criterion = nn.NLLLoss()
model.classifier = classifier
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

4. Results:

4.1 Model Evaluation and Validation:

As mentioned earlier in refinement section of methodologies, final parameters for neural network are chosen after finding best models.

```
model = models.densenet121(pretrained=True)
```

I used pretrained densenet neural network for implementation. Densenet is a network architecture where each layer is directly connected to other layer in forward fashion.

```
# Training the network
epochs = 2
steps = 0
running_loss = 0
print_every = 5
model.to('cpu')
for epoch in range(epochs):
    for inputs, labels in trainloader:
        steps += 1
        # Move input and label tensors to the default device
        inputs, labels = inputs.to('cpu'), labels.to('cpu')
        optimizer.zero_grad()

        logps = model.forward(inputs)
        loss = criterion(logps, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if steps % print_every == 0:
```

```

valid_loss = 0
accuracy = 0
model.eval()
with torch.no_grad():
    for inputs, labels in validloader:
        inputs, labels = inputs.to('cpu:0'), labels.to('cpu:0')
        model.to('cpu:0')
        logps = model.forward(inputs)
        batch_loss = criterion(logps, labels)

        valid_loss += batch_loss.item()

    # Calculate accuracy
    ps = torch.exp(logps)
    top_p, top_class = ps.topk(1, dim=1)
    equals = top_class == labels.view(*top_class.shape)
    accuracy += torch.mean(equals.type(torch.FloatTensor)).item()

print(f"Epoch {epoch+1}/{epochs}.. "
      f"Train loss: {running_loss/print_every:.3f}.. "
      f"Valid loss: {valid_loss/len(validloader):.3f}.. "
      f"Valid accuracy: {accuracy/len(validloader):.3f}")
running_loss = 0
model.train()

```

Testing Neural Network:

Above models is tested on testing dataset. It find that it accuracy is close to 90%.

```

Test_loss = 0
accuracy = 0
model.eval()
with torch.no_grad():
    for inputs, labels in testloader:

```



```
inputs, labels = inputs.to(device), labels.to(device)
logps = model.forward(inputs)
batch_loss = criterion(logps, labels)

test_loss += batch_loss.item()

    # Calculate accuracy
ps = torch.exp(logps)
top_p, top_class = ps.topk(1, dim=1)
equals = top_class == labels.view(*top_class.shape)
accuracy += torch.mean(equals.type(torch.FloatTensor)).item()

print(f"Test loss: {test_loss/len(testloader):.3f}.. "
      f"Test accuracy: {accuracy/len(testloader):.3f}")
running_loss = 0
```

4.2 Justification:

To check the classification of monkey species, I tested trained model on images from internet [\[link\]](#).

I found following results:

1. Image 1

Original Image:



Output from model:

```

first Five Probabilities are
[0.9464812  0.04054244 0.00631886 0.00316347 0.00133768]
first Five Classes are
['n8', 'n7', 'n4', 'n5', 'n6']

```

<Figure size 432x288 with 0 Axes>

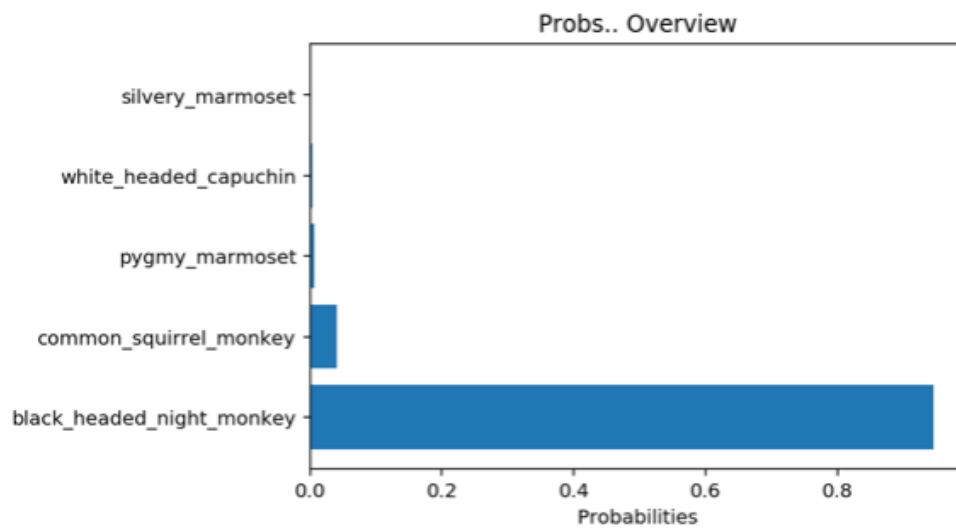


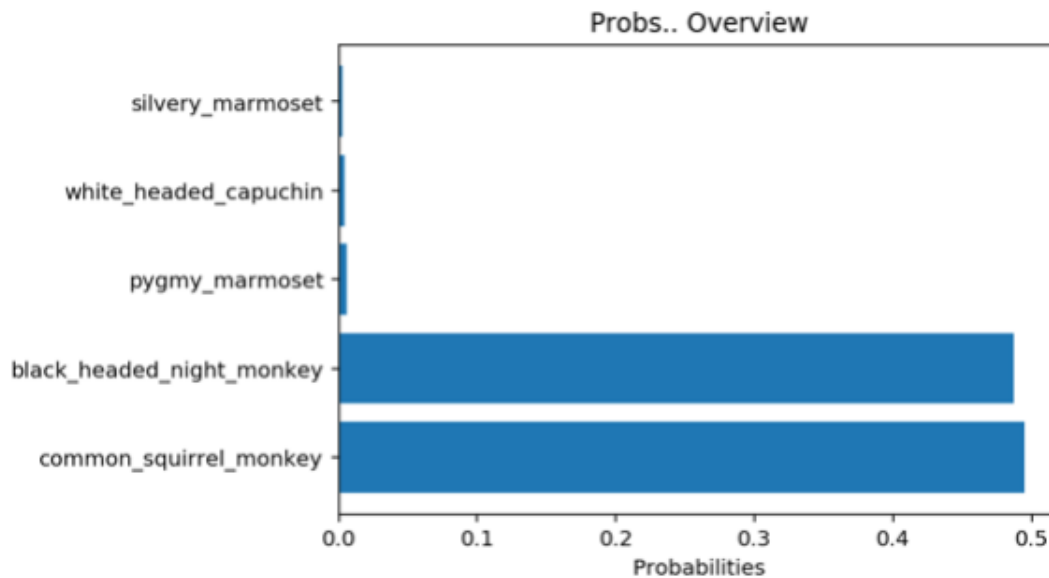
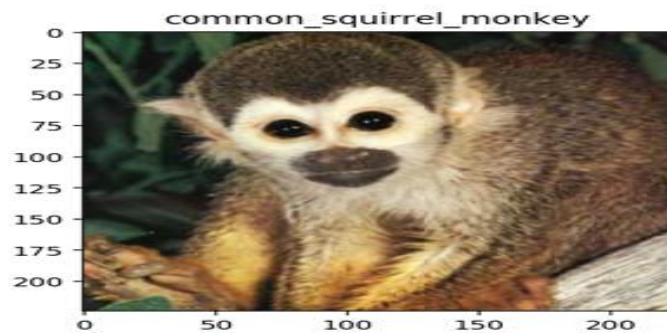
Image 2:



Results:

```
first Five Probabilities are  
[0.4950699  0.4872082  0.0057679  0.00458308  0.00256334]  
first Five Classes are  
['n7', 'n8', 'n4', 'n5', 'n6']
```

<Figure size 432x288 with 0 Axes>



5. Conclusion:

5.1 Reflection:

Implementation of this project consists of following steps:

1. Initially a monkey species classification is defined, dataset is taken from kaggle.
2. Dataset is divided into three parts. Testing, Training, Validation
3. Preprocessing algorithm is applied to all dataset.
4. Pretrained 'densenet121' model is defined from torch

5. Model parameters are defined
6. Model is trained with defined parameters
7. Model is checked for validation and testing dataset.
8. If satisfactory results are found, then model is saved.
9. A random image from testing dataset or from internet is checked.
10. First that image is preprocessed.
11. After preprocessing, image is passed through network.
12. Top most 5 categories is printed.
13. Also top most categories with its probability is printed.
14. Top most probability along with its class is labeled to tested image.

Most challenging part of this project is to set parameters for learning network. As there is no rule to define this parameters, I played with each and best parameters setting. Next interesting part is to be patient when the network is trained. As it decide performance of system.

5.2 Improvements:

Following are possible improvements:

1. Here only 10 breeds of monkeys are classified. Need to consider more breeds to improve application of system.
2. More number of epoch will be great solution for improving performance of system.
3. May be some image pre processing is required to remove background of images. So that network will learn interesting part of image very clearly.

Reference:

1. <https://pytorch.org/docs/stable/index.html>
2. <https://www.factzoo.com/mammals/types-of-monkeys.html>
3. <https://classroom.udacity.com/nanodegrees/nd025/dashboard/overview>
4. Dataset [<https://www.kaggle.com/slothkong/10-monkey-species>]