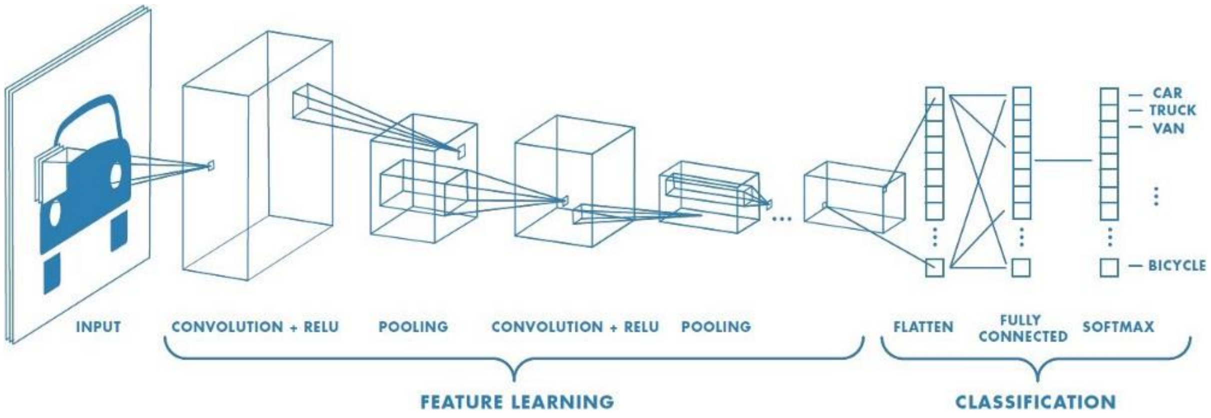
Project Definition

Background

This project is about classify the bird breed by input an image. Classification of data with labeled in supervised learning. Supervised learning is the task of learning a function that maps an input to an output. It infers a function from *labelled* *training data* consisting of a set of *training dataset.*

Problem Statement



The project is to create a Convolutional Neural Networks (CNN) from scratch and leverage the image classification techniques This model can be used as part of a mobile or web app for the real world and user-provided images. CNN is a Deep Learning algorithm which can take in an input image, assign importance to various objects in the image hence to extracts or identifies a feature. CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters or characteristics.

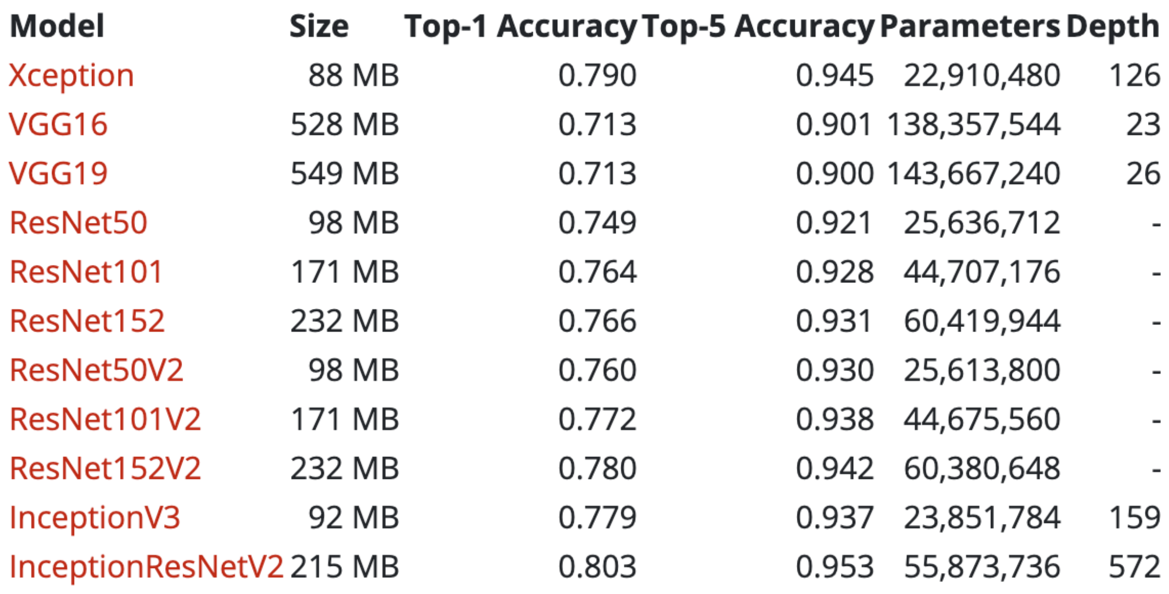
To achieve the goal, the task involved the following step:

1) download the data set and
2) detect the bird object from per trained model.
3) To achieve the high accuracy, using the pre trained model to train the dataset is a must.
4) Classify the image by input simple image.

Evaluation Metrics

Accuracy is used for evaluation metric and using the CrossEntropy loss function. First, the labels have to be in a categorical format. The target files are the list of encoded bird labels related to the image with this format. This multi-class log loss punishes the classifier if the predicted probability leads to a different label than the actual and cause higher accuracy.

Benchmark Mode



Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572

Figure 1 Pre-trained model information form https://keras.io/api/applications/

The Figure 1 is about deep learning models that are made available alongside pre-trained weights from keras. By using the pre-trained model, the accuracy of image classification is up to 70%, VGG16 will be used for this project because of the dataset image size was 224x224 which suitable for VGG16 input.

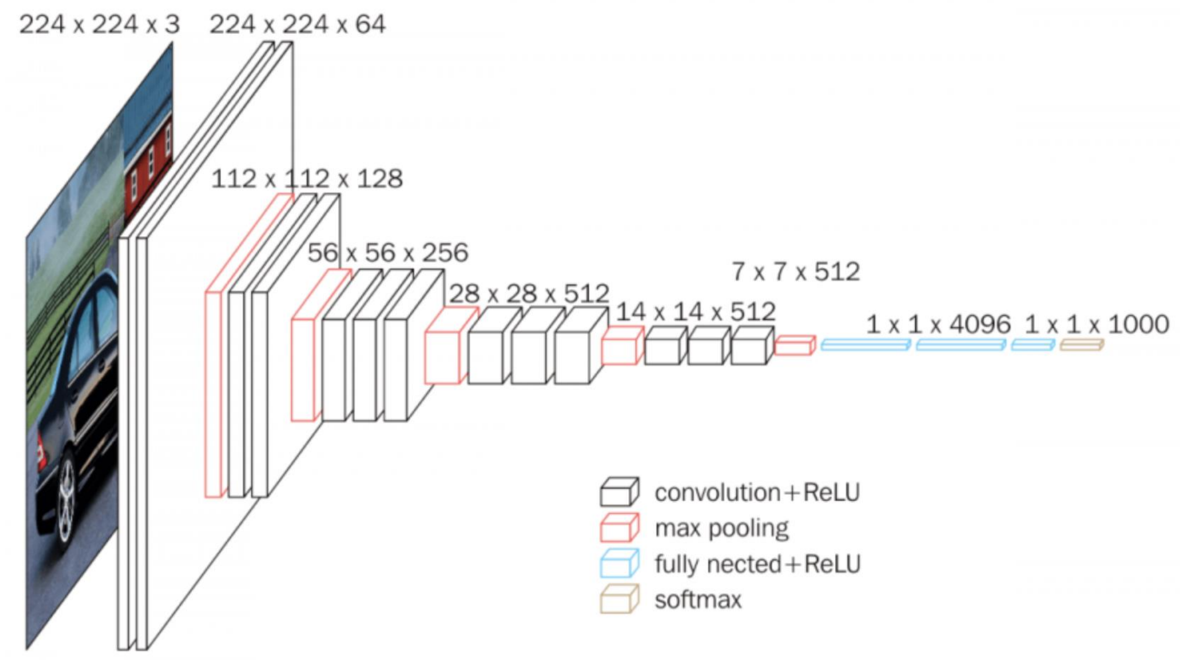


Figure 2 VGG16 image from https://neurohive.io/en/popular-networks/vgg16/

The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It makes the improvement over AlexNet by replacing large kernel-sized filters with multiple 3×3 kernel-sized filters one after another.

Analysis

Data Exploration and Visualization

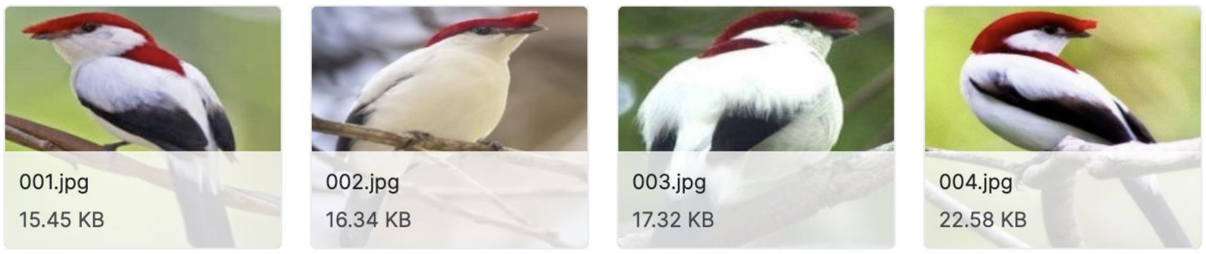
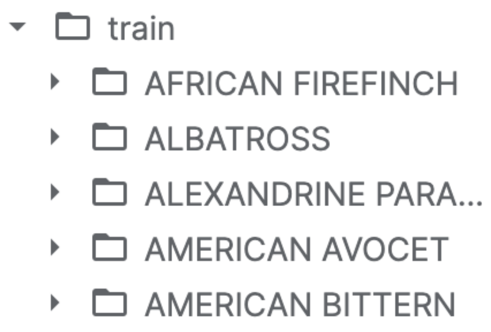


Figure 3 Image captured from Kaggle

By using provide Bird dataset from Kaggle(224x224x3 in jpg format), it provided 30168 training images including 1125 test images(5 per species) and 1125 validation images(5 per species. Images for each species are contained in a separate sub director and all the files were numbered sequential. For example, AMERICAN GOLDFINCH/001.jpg , AMERICAN GOLDFINCH/003.jpg etc.



- train
  - AFRICAN FIREFINCH
  - ALBATROSS
  - ALEXANDRINE PARA...
  - AMERICAN AVOCET
  - AMERICAN BITTERN

Also, the dataset contains imbalance gender ratio, 80% of the images are of the male and 20% of the female. Such that, this project will not recognize the gender of the species.

Methodology

Data Pre-processing

The bird image need to normalize by using data normalization(224px x 224px), that is an important pre-processing step of the images. It ensures that each input comes from a standard distribution. The outcome is makes the model train faster. First, using PIL library to enable the image processing and transforms it to a tensor.

```
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

Implementation

Prediction Index:
When using the pre trained VGG19 model and making prediction to detect the bird existing in the image, it returns an index which representing the object in imagenet1000. It needs to find the correspond bird object within one thousand dictionary keys in the list. Finally, it was between 12 and 146 are related to bird. Such that, the bird detector only limited to 12 to 146, otherwise return false.

```
def bird_detector(img_path):
    index = VGG16_predict(img_path)
    return (13 <= index and index <= 146) # true/false
```

Transform Normalize:

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

This mean and std is based on ImageNet common practice. They are calculated based on millions of images. This is ImageNet pretrained model with its own mean and std is recommended. So, it can be use on training model from scratch and learning transfer model.

Model from Scratch:

In the convolution configuration, I chose 3x3 filter, 32 at the first output layer instead of 16 layer and the final is 128 layer , it turned out CNN learn more deeper but increase the training time.

```
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.norm2d1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
```

Testing Algorithm:

After trained the model, it needs to measure the accuracy by calculating the probabilities. The First model accuracy target is greater than 20%. Lucky, the result is 46%.

```
def test(loaders, model, criterion, use_cuda):

    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target) in enumerate(loaders['test']):
        # move to GPU
        #if use_cuda:
            #data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the loss
        loss = criterion(output, target)
        # update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]
        # compare predictions to true label
        correct +=
np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
        100. * correct / total, correct, total))
```

Resnet50 Model:
By looking at Pytorch document(https://pytorch.org/docs/stable/notes/autograd.html), the pretrained CNN is recommend that the requires_grad is equal to false hence to increase the efficient. This option is an option but recommend.

```
import torchvision.models as models
import torch.nn as nn

model_transfer = models.resnet50(pretrained=True)

for param in model_transfer.parameters():
    param.requires_grad = False

model_transfer.fc = nn.Linear(2048, len(CLASSES), bias=True)
```

The accuracy of learning transfer model is 72% and it was improved comparing with the first model.

Prediction Algorithm:
The function is tying to predict breed by input the image, those of image need to transform to RGB channel and normalizer before loading to model. The result returns an index which is the class name of folder.

```
from PIL import Image
import torchvision.transforms as transforms

data_transfer = loaders_transfer.copy()

# list of class names by index
class_names = [item[:] for item in data_transfer['train'].dataset.classes]
def predict_breed_transfer(img_path):
    global model_transfer
    global transform

    # load the image
    image = Image.open(img_path).convert('RGB')

    # Removing transparent, alpha
    image = transform(image)[:3,:,:].unsqueeze(0)

    #if use_cuda:
        #model_transfer = model_transfer.cuda()
        #image = image.cuda()

    model_transfer.eval()

    #return the predicted breed by using transfered model
    idx = torch.argmax(model_transfer(image))
    return class_names[idx]
```

Input image file Algorithm:
This algorithm is looking at the folder and search any image file, then run the prediction function and return the breed. For example, ‘dataset/int/’ as the image folder and contain some of image, then run this function.

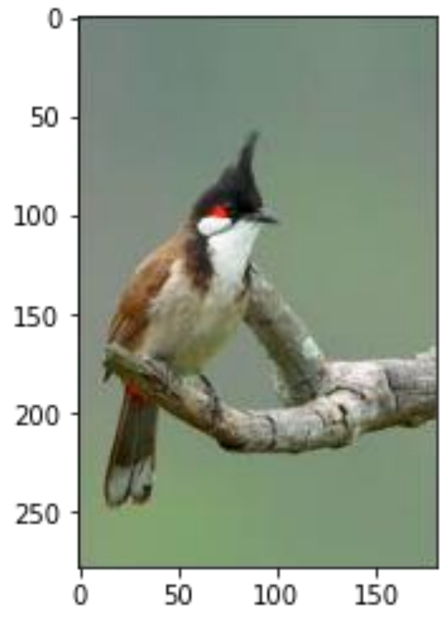
```
import os

import matplotlib.pyplot as plt

from PIL import Image



for img_file in os.listdir('dataset/int/'):
```

```
img_path = os.path.join('dataset/int/', img_file)

run_app(img_path)

print(img_path)

img = Image.open(img_path)

plt.imshow(img)

plt.show()
```

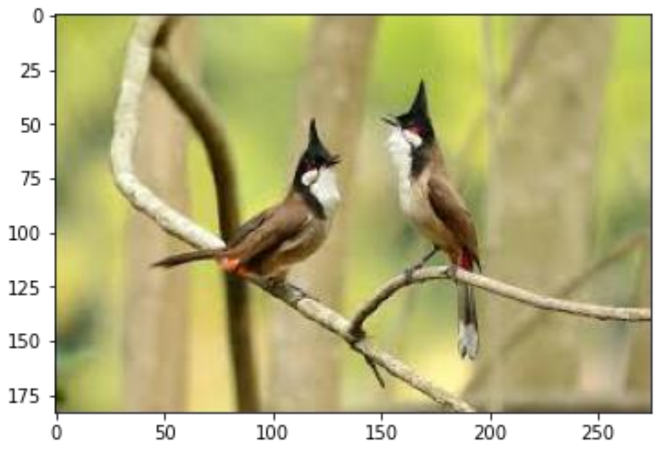
Result

As the result, The test accuracy is 46% which produced by step 2 model from Scratch. After using transfer learning Resnet50 model in step 3, the accuracy was increase from 46% to 72%.
By Entering 'Bird' in google, there are thousand of bird pictures with unknow species. Such that I select the image randomly and put those of photo in testing folder, also include non-bird image for valid the model. As the below result, the model returns the result is encouraging. The model can specify the bird breed precisely and return false if there are not bird image.

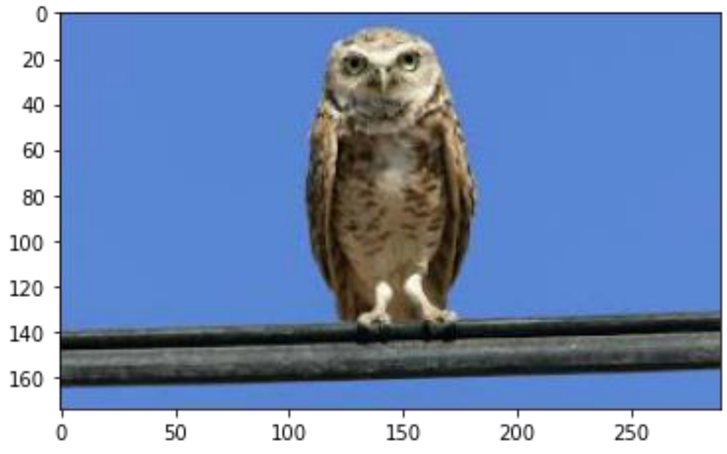
```
Bird breed is RED WISKERED BULBUL
dataset/int/1.jpg
```



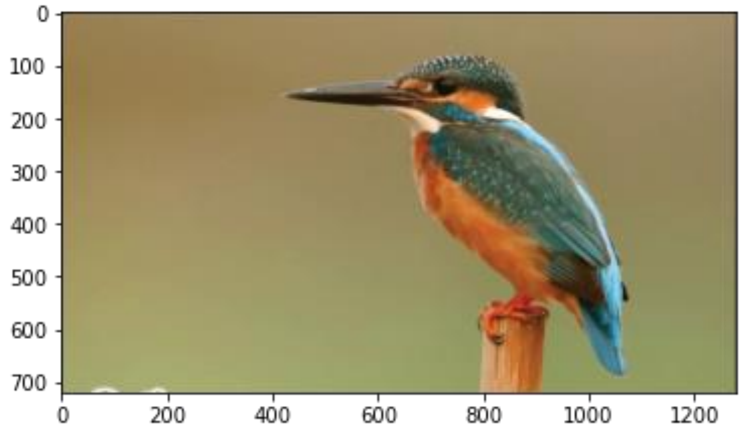
```
Bird breed is RED WISKERED BULBUL
dataset/int/2.jpg
```



```
Bird breed is PEREGRINE FALCON
dataset/int/3.jpg
```

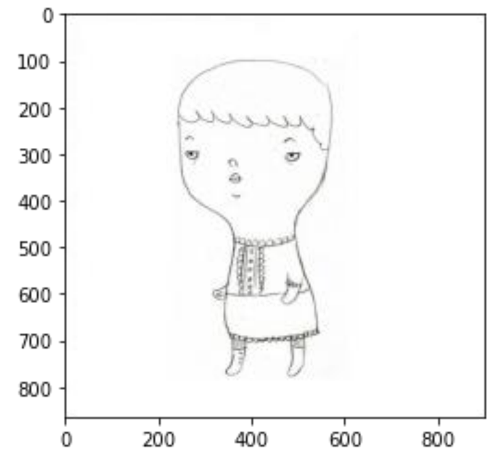


```
Bird breed is VIOLET GREEN SWALLOW
dataset/int/4.jpg
```

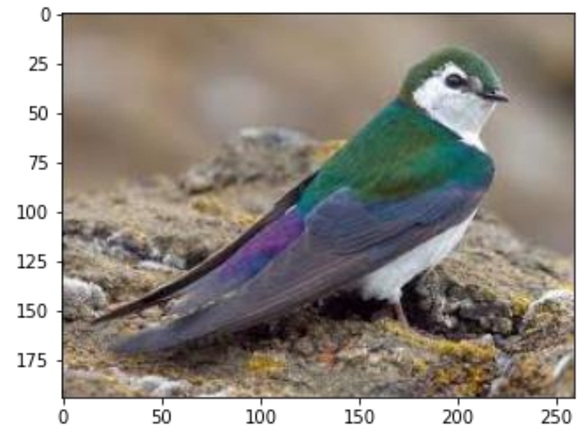


```
Did not find any bird!
dataset/int/5.jpg
```

```
Did not find any bird!
dataset/int/6.JPG
```



```
Bird breed is VIOLET GREEN SWALLOW
dataset/int/7.jpg
```



Conclusion

The bird recognition problem basically solved by this model. During the development, I need to search many information about the coding skill, CNN principle. In the coding skill, I'm not familiar in python and Pytorch.   Meanwhile, I get confuse will keras and pytorch because there are many resources about using keras compare with pytorch. Also, I spend some time in Linear regression for learning the relationship between lost function and criterion, that remine my mathematics knowledge on my high school.

Improvement

There is serval way to improve the accuracy. First, increase the number of Conv2d, it can help the neural network learn more deeper in the image. Increase the epochs and add drop out option to prevent over-fitting. Number of training sample or different shooting angle of object also is the way to improve the accuracy.
Also, ResNet50 is not a final solution of CNN model, also there are kind of ResNet model or 'Inception v3' we can choose. Figure 1 is a benchmark of Top-5 accuracy which we can reference, but it must cost a lot of GPU power to try those of model and it is not recommend running on CPU only and the reason of using (nvidia) GPU is, they was optimized the GPU and accolated the deep learning framework.

References

nvidia. (n.d.). Retrieved from https://developer.nvidia.com/deep-learning