Project Report

Definition

Project Overview

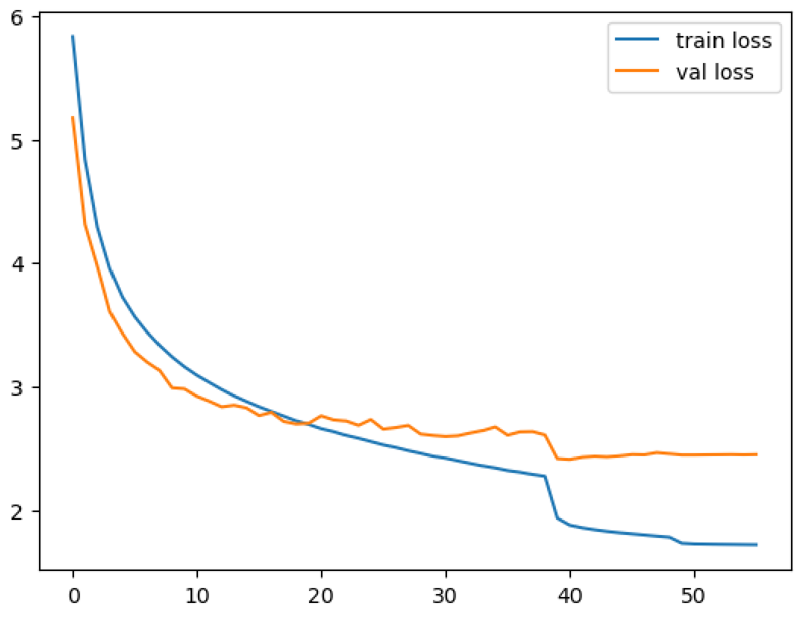
There are governmental forest maintenance organizations in many countries such as IFS in India. These are multi-tier forest Administration whose main job is effective management of forest resources. These organizations would always keep a check on the count of various species of fauna in that specific forest. For this, they will have cameras installed at various locations of the forest, where a personnel would actually sit and watch these camera streams and learn the animal behavior, keep an account on the species living in that forest, count the number of individuals of a species living in that forest. Sometimes these cameras would be damaged by animals. These cameras would be hard-cased. But, that is a different problem. In this proposal we will be proposing an ML method to automate the process of fauna count.

Problem Statement

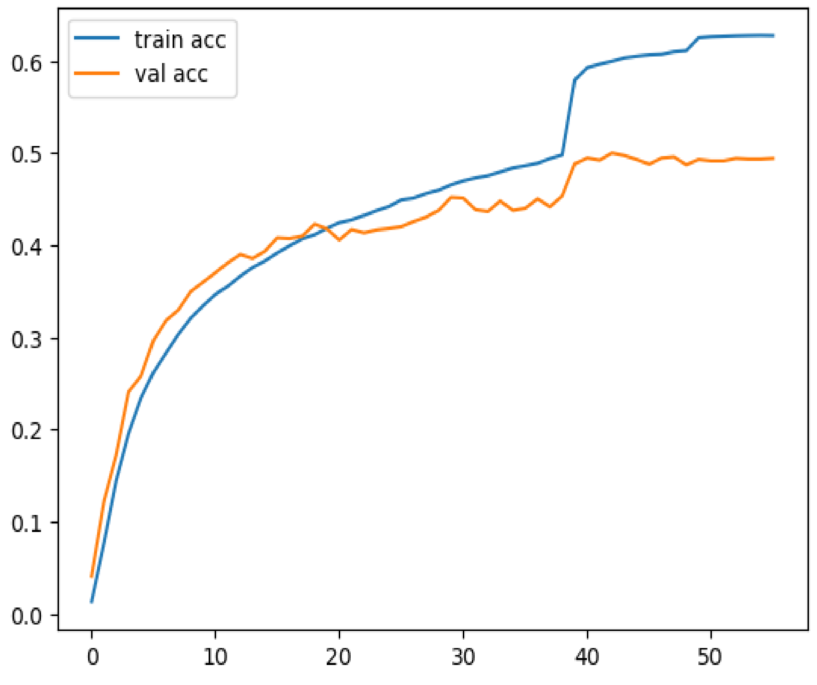
With these domain backgrounds discussed above. We will define the problem statement. Our main job is to automate the process of counting without human interference. But before jumping into the problem of counting. We should be able to differentiate the species more accurately. In this project we have only worked on species identification of birds. As we all are well aware about the animals, we weren't that much aware about the bird species. Also most of the birds being small, it is very much difficult to classify their species.In this project we have tried our best to classify the various species of birds.

Evaluation Metrics

The evaluation metric we have used is accuracy and loss. These metrics are enough to check the reliability of the model. We have also plotted the graph against the training loss and validation loss and training accuracy and validation accuracy. Below is the graph



(a) Training loss vs Validation loss



(b) Training accuracy and Validation accuracy

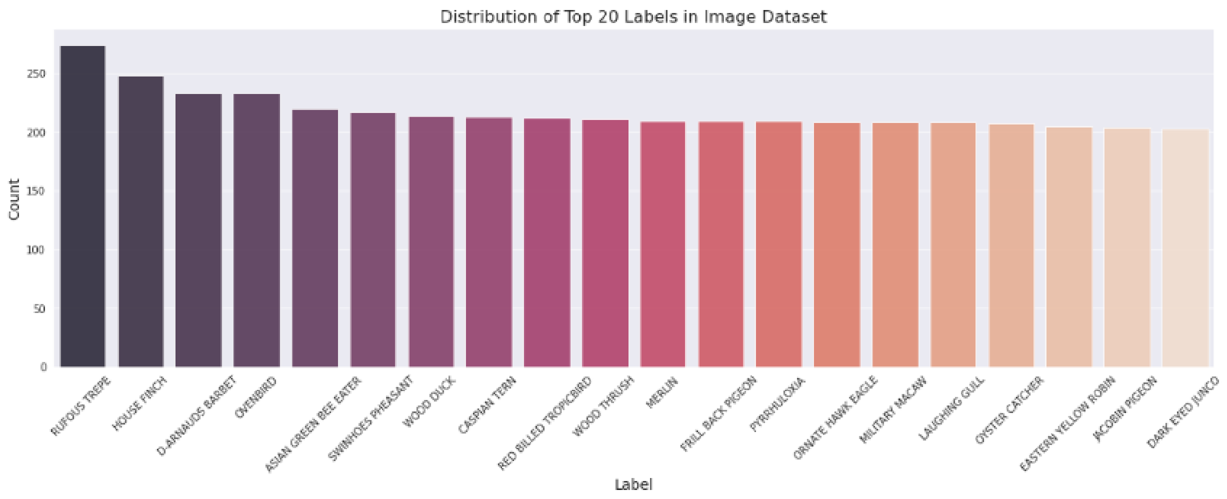
These graphs would help us check for the overfitting. From the graphs, we could say that there isn't much of an overfitting. But at around the 19th epoch the validation accuracy reduces than the training accuracy.

<u>Analysis</u>

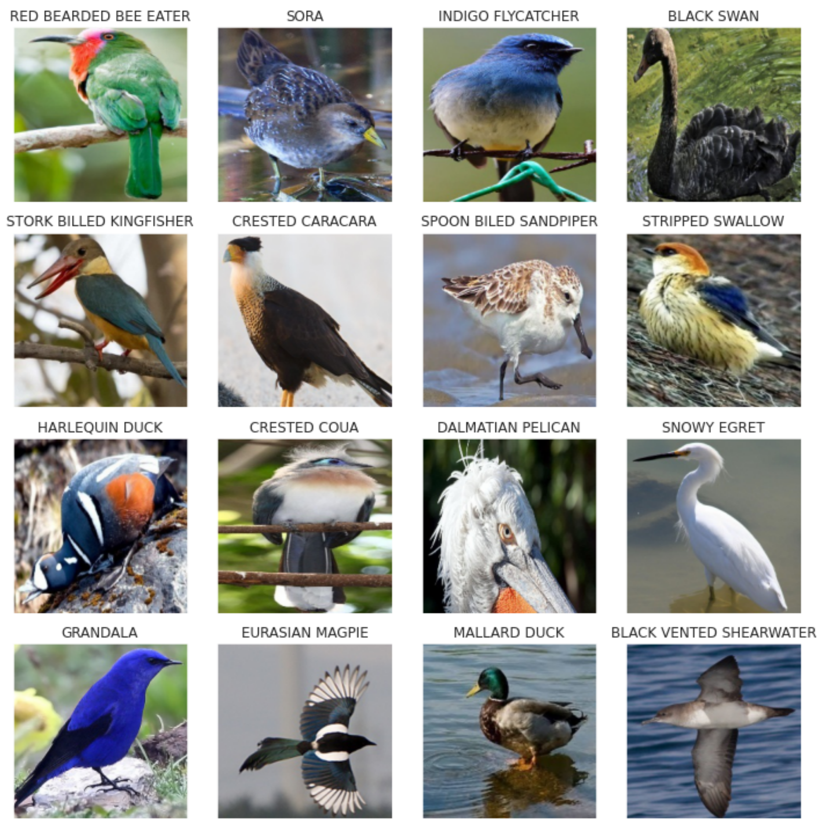
Data Exploration

The datasets for this project were taken from the kaggle datasets. This dataset is really a very large dataset and bringing up a very good accuracy for this dataset is really a challenging task. This dataset consists of images of 510 species of birds. This is a well cleaned dataset. It consists of 81,950 train images, 2550 test images and 2550 validation images. We have managed to bring out a validation accuracy of 45%. This isn't bad, but should definitely be improved. Each image is composed of 224X224 pixels containing three layers of R, G and B. It would be really difficult if we use this image as such. So, I first tried reducing it to 150X150, but that took really a great time for each epoch, around 50 mins for each single epoch. Then I still reduced to around 28X28, this took around 3 hours to complete the entire training of around 56 epochs.

Exploratory Visualization



(c) Distribution of Top 20 labels in Image Dataset



(d) *A sample showcase of images*

Algorithms and Techniques

First we used the SVM algorithm from sklearn library. SVM will only require data to be in linear format. First we converted each image to tensor object of size 28X28 using TensorFlow's flow_from_directory API, then we converted each of these images to numpy array of shape 28X28X3, so feed it into the SVM, we need to condense them to linear format. We have done that using reshape() from np. Then we fed into the SVM. The validation accuracy came to be around 20%. This shows that SVM performs really badly for this dataset. So, let us move to CNN.

For CNN, what we do is we directly feed the tensor object into a couple of CNN layers and a single Dense layer. Then we split out these arrays to a length equal to the number of classes of our datasets. This took time equal to the SVM but for around 56 epochs it took 3 hours to train.

The model is compiled with The train accuracy came to be around 62% but the validation accuracy reached only 49%.

Benchmark Model

The benchmark model that we are using is “CNN + MobileNetV2”. This notebook briefly explains the structure of this model. MobileNetV2 is a pre-trained model and on top of it they are using CNN to reduce the image size. This model gives a very good train accuracy of 92% and test accuracy of 84%. As this is a pre-trained model, the result should be pretty good as expected, but our model is a custom made model and it still requires good refining to reach that level of accuracy.

Methodology

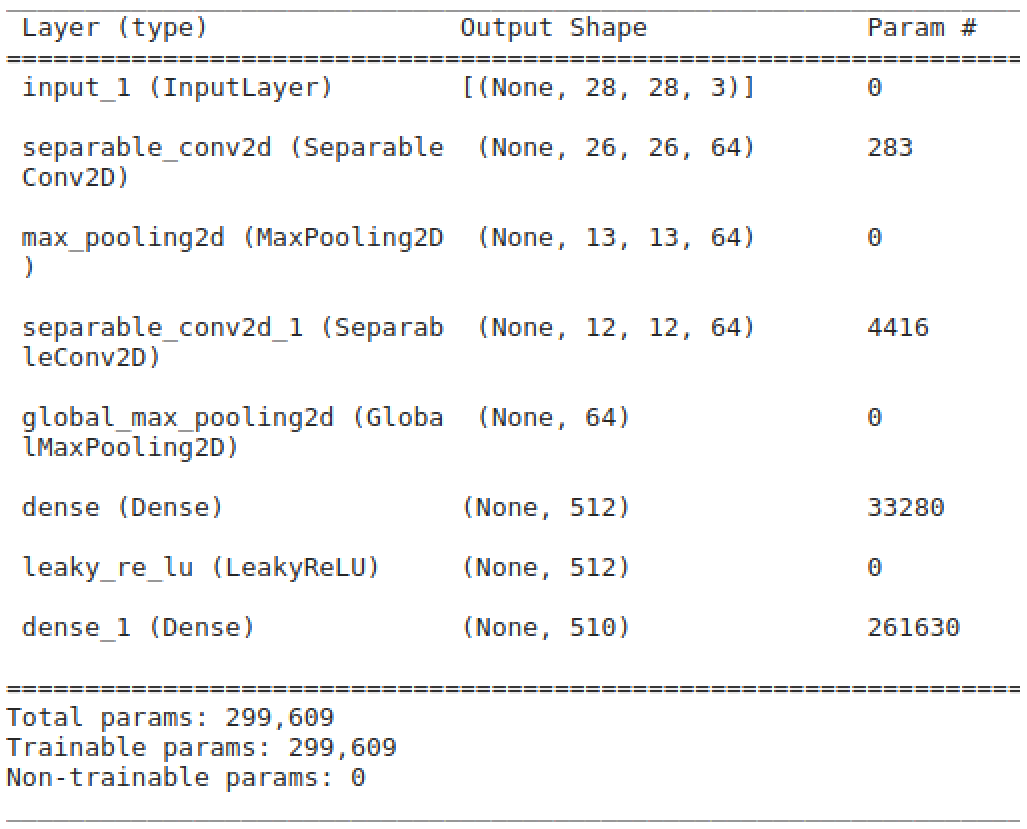
Data Preprocessing

We haven’t used the raw image data as such, because it was at 224X224 pixels, training that would take days of training with the dataset of our scale, so I first reduced to 150X150 but that too didn’t give good training time, so we further reduced to 28X28, this took a reasonable time of around 3 hrs to train with better accuracy.

Implementation

The project was implemented using a custom made model using two layers of CNN, the first layer has a filter size of 3X3 and the next layer has a filter size of 2X2. Then we transferred the resultant arrays into a dense layer of 512 neurons and we directly took an output of 510 dense neurons equal to the number of classes we have.

Below is a detailed implementation of the model stacks



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 3)]       0

 separable_conv2d (Separable  (None, 26, 26, 64)       283
 Conv2D)

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 separable_conv2d_1 (Separab  (None, 12, 12, 64)       4416
 leConv2D)

 global_max_pooling2d (Globa  (None, 64)               0
 lMaxPooling2D)

 dense (Dense)               (None, 512)               33280

 leaky_re_lu (LeakyReLU)     (None, 512)               0

 dense_1 (Dense)             (None, 510)               261630

=================================================================
Total params: 299,609
Trainable params: 299,609
Non-trainable params: 0
_________________________________________________________________
```

Refinement

First the data was trained for around 30 epochs reducing the images to 28X28. But that gave only 30% accuracy, I could notice that there is further scope for training accuracy to increase if we train it still further, so I set it to 70 epochs, but the callback has stopped at around 56 epochs with an increment of double the accuracy.

Results

Model evaluation and Validation

As we used accuracy as our major parameter of evaluation. The training accuracy is checked epoch by epoch, there is a callback feature enabled, that stops training if these are 5 consecutive epochs with the same accuracy. We have actually set it for about 70 epochs but early stopping feature stopped it at around 56th epoch and the training accuracy came to about 62% and validation accuracy came to about 49%

To increase it still further we need to work on the image preprocessing and model design.

Justification

The result of this work can be justified to some extent, because we couldn’t reach as better accuracy as the benchmark model, we differ about 30% accuracy in training accuracy and 34% accuracy in validation accuracy. But if we could spend better time on training, I think we would be able to bring better justification to the model. Also to further out the accuracy, we need to not reduce the image as small as 28X28, something at around 150X150 would also give better accuracy.