Capstone Project : Dog breed Classification

Sai Aravind Sreeramadas

I. Definition

Project Overview

Dog breed classification is one of the challenging Image Classification problem. In this project I will implement dog breed identification android application and I will be using keras API to build the convolutional neural network. I have used a subset of Stanford dog breed dataset to train my model. I will extract the features of a dog image using the pre-trained models and then pass these features to an artificial neural network which will predict the breed of the dog. I will compare and evaluate the performance of different models that I built using pre-trained models. I will deploy the small model on the android platform.

Problem Statement

Given the dog breed data set of 10,222 images with 120 classes I have to build a cnn network which has a good validation accuracy and minimal log loss on the given dataset. After getting a good accuracy, a trade off between the model size and accuracy has to be made to deploy it on the Android platform.

Metrics

1. Accuracy

It is the ratio of the correct prediction to the total no.of samples from the test set.

2. Loss

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

II. Analysis

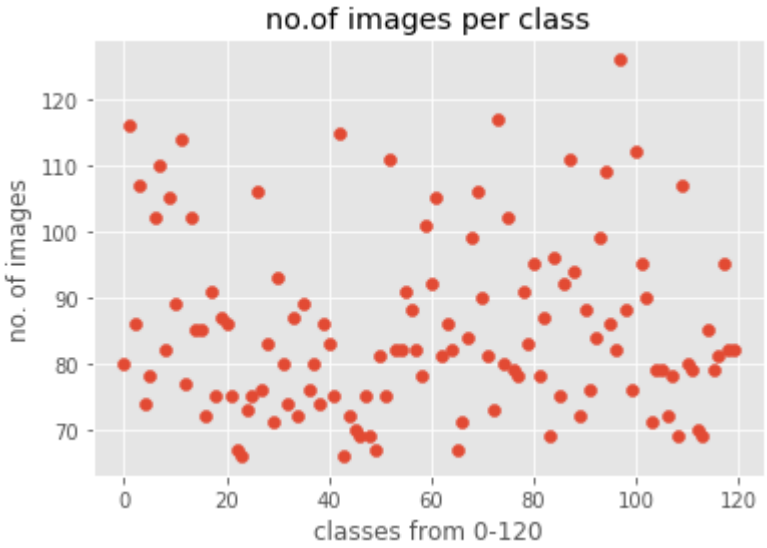
Data Exploration

The dataset to be used is the dog breed dataset which is hosted by the kaggle.This dataset is a subset of Stanford dog dataset.This consists of 10222 images of dogs with 120 classes. The class distribution of images is not uniform data.

This data is relevant as it has dogs which are similar in some aspects .ie, it is not a simple binary classification of two completely different classes , instead it is a fine grained classification.

In this data we use 9,199 samples for training and 1023 samples for validation.The training and validation set have the same class distribution.

Exploratory Visualization



In this we can see that the images in the 120 classes are not equally distributed.So we need to take care of that.

Algorithms and Techniques

1) Build a cnn model from scratch and see its performance

2) Check if the accuracies are good enough? If they aren't good then go for the pretrained models

3) Go for the InceptionV3 ,Xception, combination of VGG and inception to see their performances

4) After seeing those pretrained network performances combine the top three best performing models i.e, concatenate the bottle neck features of these models

5) Build a dense network on these features and tune this dense network

6) After this check for the image size to be loaded.

7) Get the final model after some fine tuning and hyper parameter optimizations

8) Deploy our model on the android platform

Benchmark

The VGG network implemented ont the Stanford dog datasetgave only a validaton accuracy of 0.78 .The inception V3 I implemented could only achieve a val_acc of 92.

These are the big and deep cnns which are mostly used for vison problems.So taking them as bench marks makes sense as it is a good reference check how our model is performing compared to them.

III. Methodology

Data Preprocessing

In the data preprocessing, I am going to load the images of size 400*400 as the mean size of the dataset is 390.

So I am going to load the image of size 400*400 using the keras image loading function and I will scale these images by dividing it with 255 to bring the ranges to 0-1.

Now as we got the class imbalance, I am going to calculate the class_weights using the sklearn function to rectify the class imbalance.

Now we have a nice scaled images of size 400*400 and this data is going to be split in to 9,199 train samples and 1023 validation samples.

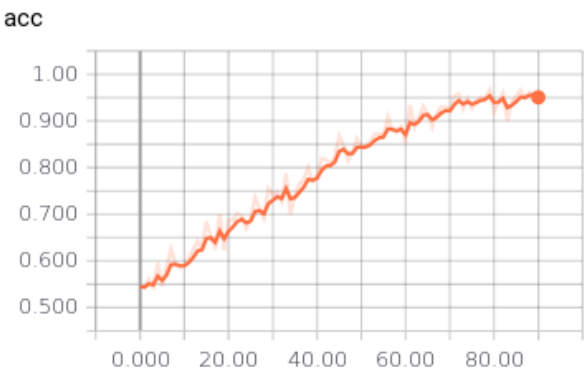
These images are going to flipped and added to the dataset. So this implies that we have 18398 training samples and 2046 validation samples.

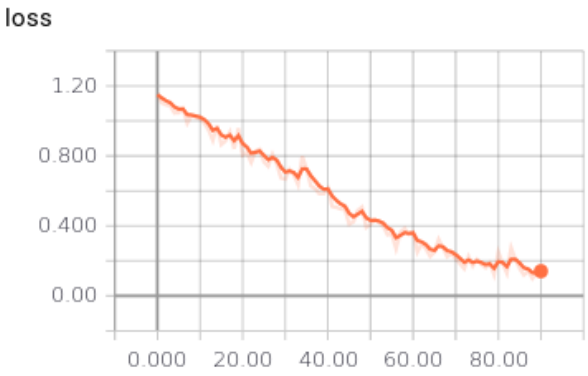
Implementation(Taken from the github link of my project *https://github.com/Eximius-Design/DogBreedIdentification/wiki, same analysis was done.)*

First Model:

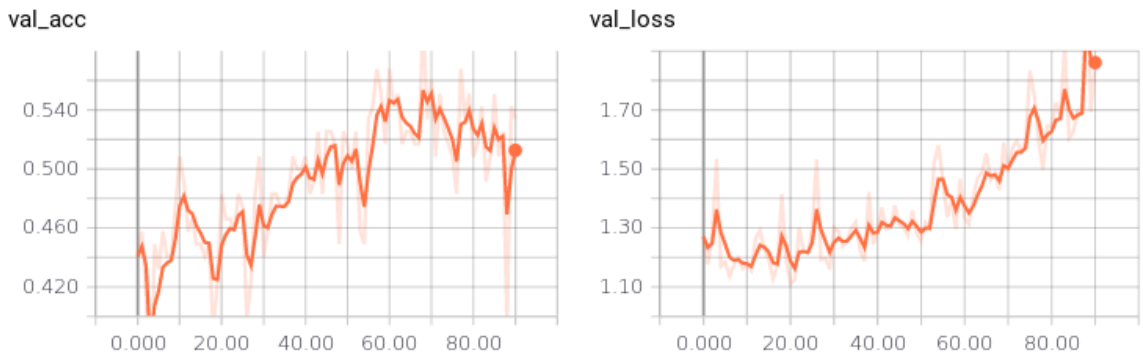
- We have trained that data using a model from scratch, we got an accuracy of 50%. Variance is very high then we introduced the drop-out in the model, now the accuracy has increased to 54% but still the variance is high

- In our model we have used Adam optimizer, categorical cross entropy as loss function, 6 convolutional layers,3 max pooling layers and a fully connected layer.





Training data accuracy and loss



Validation data accuracy and loss

- As we can see above training accuracy increased as we increased the number of epochs but after a certain point validation accuracy started decreasing and validation loss is increasing. Both validation accuracy and validation loss are noisy which clearly states that our model is over fitting the data.
- To overcome over-fitting, we have to increase the size of the dataset or use ensemble techniques or use pre-trained models.

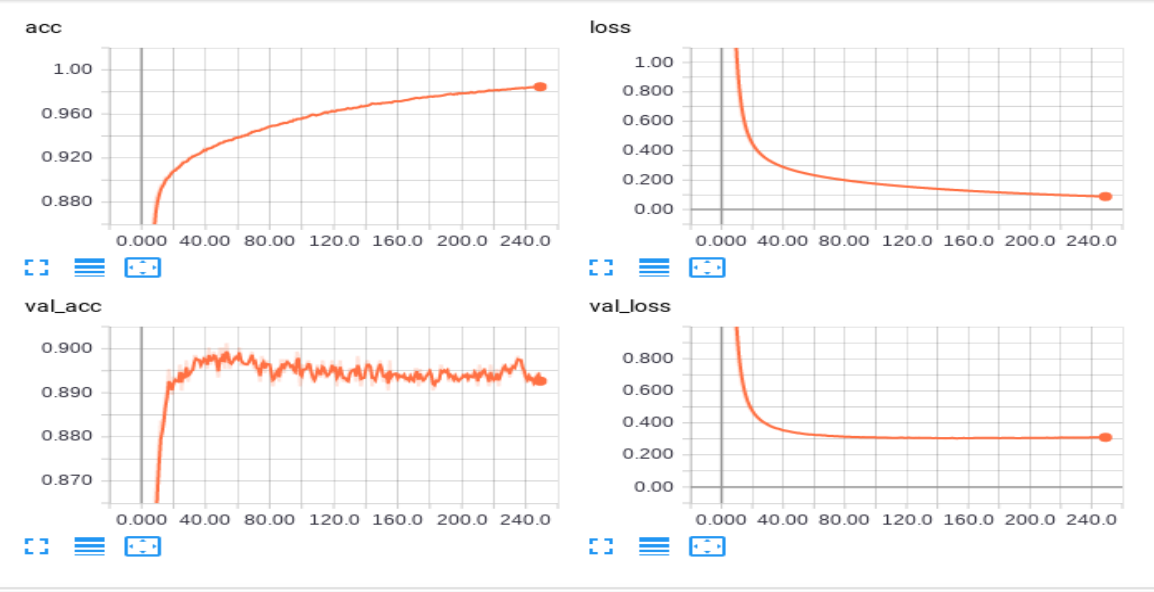
Transfer Learning:

1. Inception_v3:

- So, the next plan was to use transfer learning, we have extracted the features from inception_v3 model and passed them to our model then the accuracy has jumped to 89%.
    - Images of size 300*300*3 were used
    - We have used ReLU activations in the hidden layers for the classifier and softmax on output layer(ReLU-->softmax).
    - 1024-->120 are the neurons configuration in each layer.
    - Using the above model 98% training accuracy and 89% validation accuracy and

0.08, 0.30 losses for training and validation data respectively were achieved.

- For the same configuration with the image size of 224*224*3, 77% validation accuracy was achieved.
- Bias-variance-iterations graphs are shown in Figure 4.2.



Accuracy and loss of training and validation data

2 Xception:

- Next, we trained with xception model separately with a simple classifier at the end and obtained a log loss of 0.32 with 0.89 validation accuracy, 0.18 training loss and accuracy of 0.95.
  - Better results were obtained when image size 300*300*3 was used than 224*224*3.
  - In this only one hidden layer was used with 1024 neurons.
  - relu-->softmax

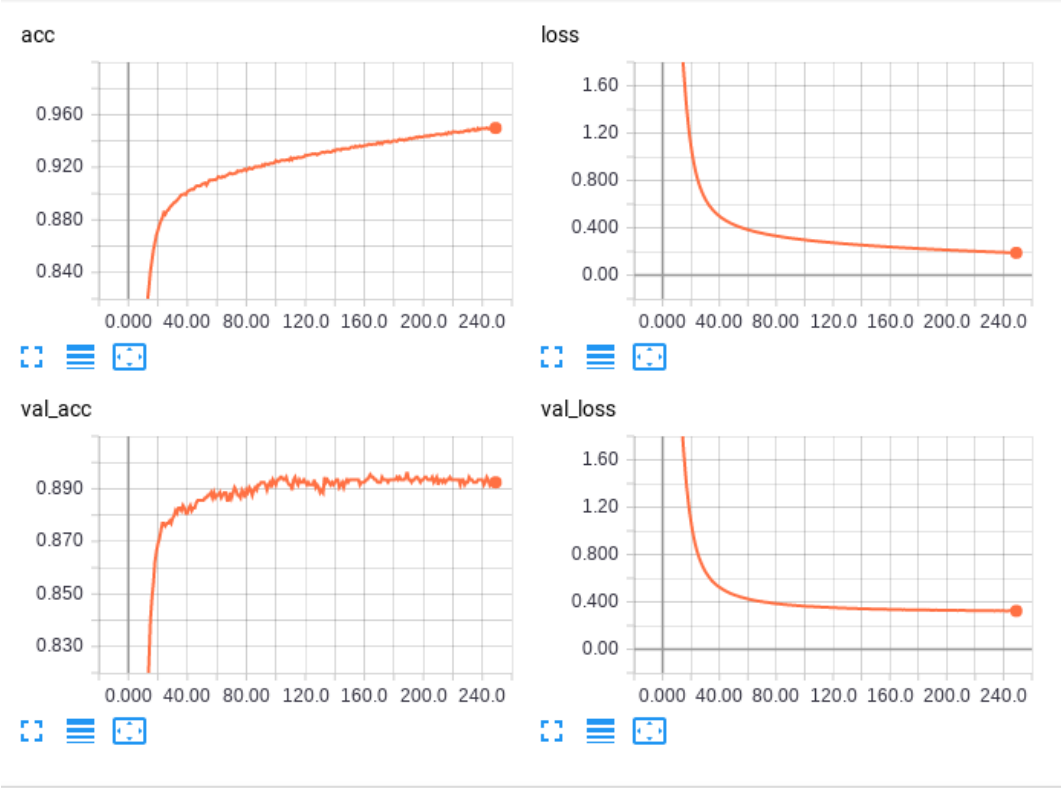
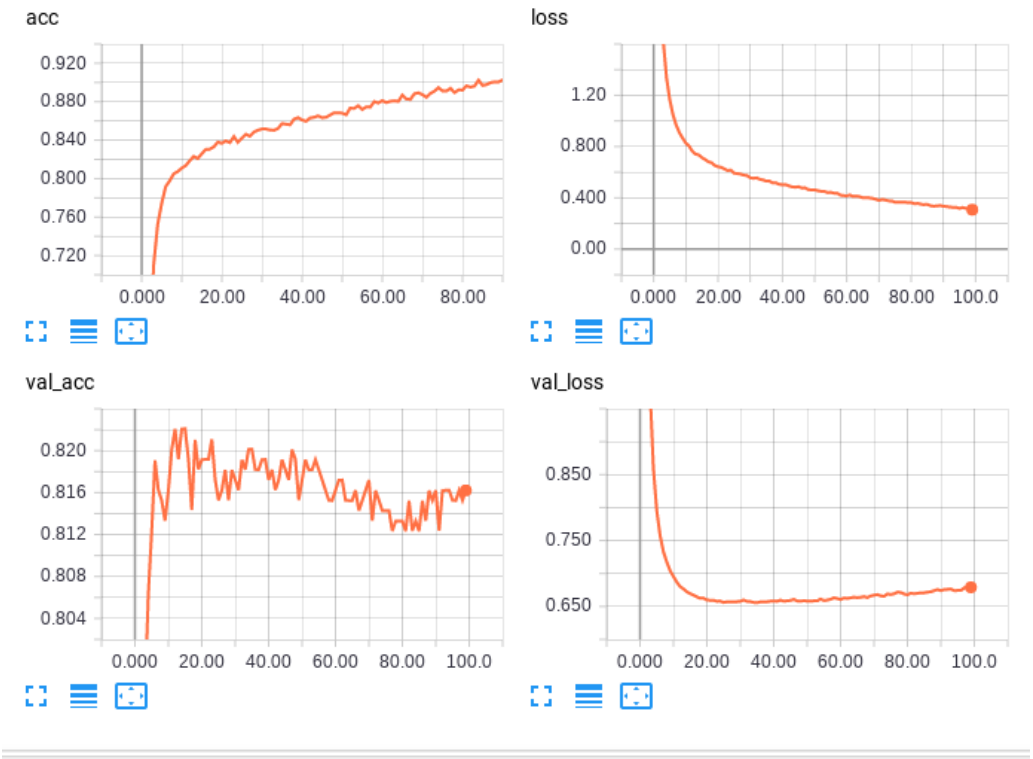


Fig 6.5.2. Accuracy and loss of training and validation data

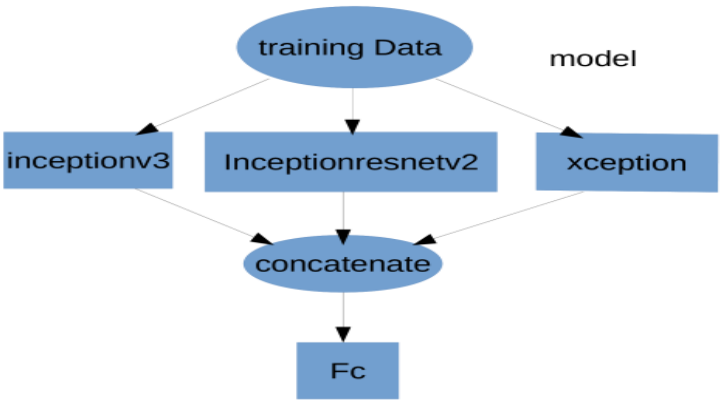
3. VGG and InceptionV3:

- Now, the features from the models VGG and inceptionv3 were extracted and concatenated to get more features This input was given to 2 dense layers with 1024,512 neurons to get a log_loss of 0.55 and the validation accuracy was 0.81
- For getting VGG and Inceptionv3 features. We have used relu activations in the two hidden layers for the classifier and softmax on output layer(relu-->relu-->softmax), sgd optimizer with learning rate=0.001.
- Using the above model 88% training accuracy and 82% validation accuracy and 0.1,0.65 losses for training and validation data respectively were achieved. (over-fitting can be observed at epoch 60 in the graph)



Accuracy and loss of training and validation data

4. Hybrid model:

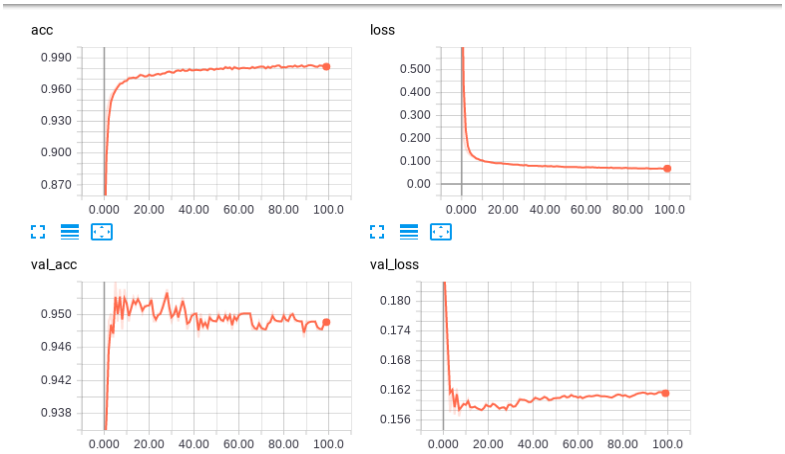


Hybrid model

- This ensemble is of features from the pretrained models. Current log_loss of this model is

around 0.165

- **model**- In this model **original images** were resized to (400,400) and flipped images were added to this data and features were extracted from inceptionv3(9k,2048), inception resnet v2(9k,2048), xception(9k,1536) and concatenated to form the input(9k,5632) features to fully connected neural network (2048,120).



Accuracy and loss of training and validation data of original+flipped images

Refinement

In this we have tried different sizes of images to load and see the best resulting image size.

Image size	accuracy
224*224*3	~83%
300*300*3	~89%

400*400*3	~95%
450*450*3	~92%

After the image size , I have tweaked the fully connected layers .i.e for the 2048 layer a dropout of 0.5 was added and it took the val_accuracy to 95.65 with a logloss of 0.165 . The optimizer values were adjusted manually to get to this result.

The initial Final model:

1) Load the images with dimensions 400,400,3.

2)Extract features from Xception+Inceptionv3+inception_resnetV2

3) Concatenate the features

4) For this features, we build a deep neural network

5) Configuration of DNN is:

Layer1: 2048 neurons ,activation='elu'

Layer 2 : 120 neurons , activation ='softmax' the optimizer used is SGD

The Refined Final model:

Mostly same as the previous one , but with a drop out added to the dense network

1) Configuration of DNN is:

Layer1: 2048 neurons ,activation='elu' with dropout=0.5

Layer 2 : 120 neurons , activation ='softmax'

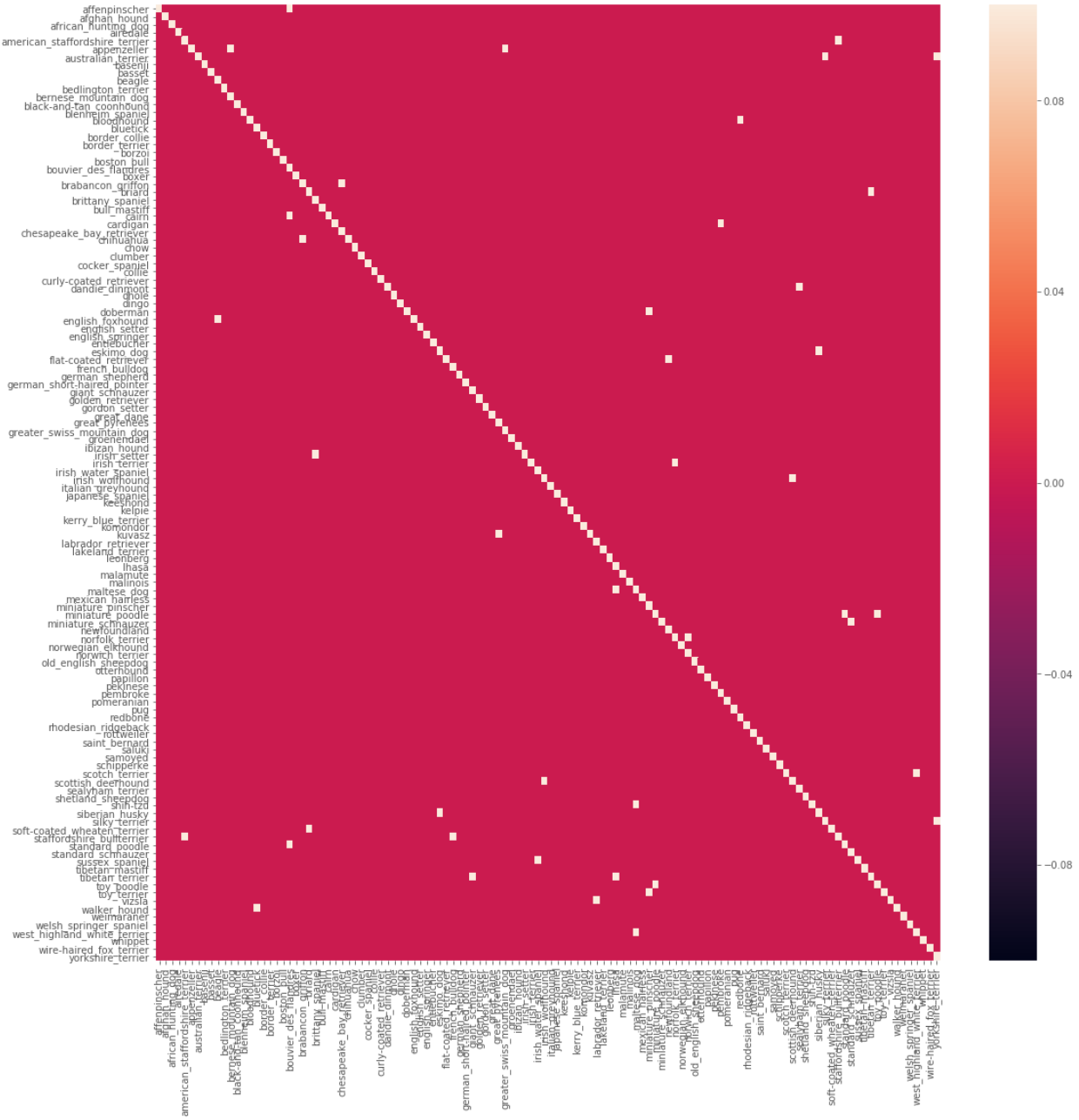
the optimizer used is SGD with lr=0.085, decay=0.03, momentum=0.92

IV. Results

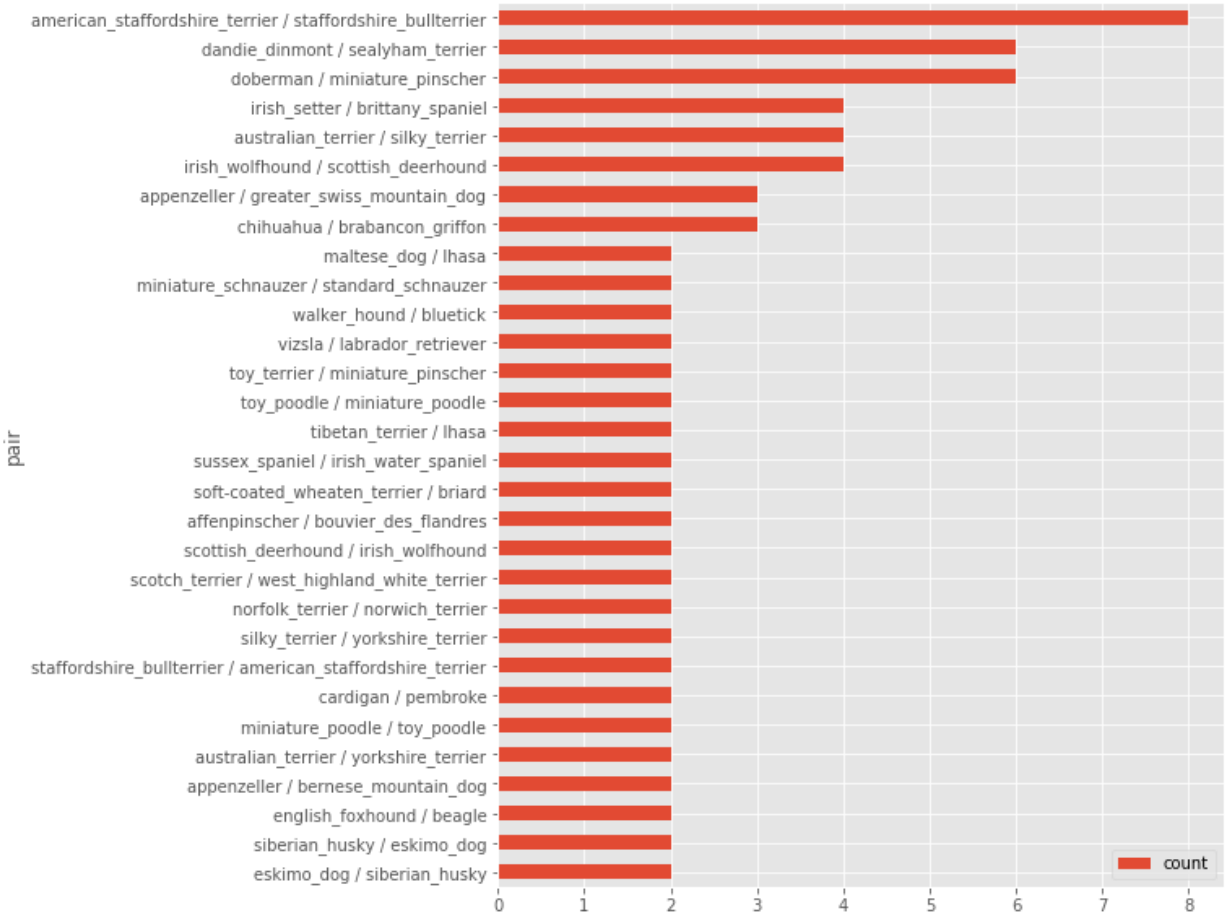
Model Evaluation and Validation

The final refined hybrid model gave me an accuracy of 95.65 which is a good validation accuracy and a logloss of around 0.16.This tells me that my model is performing well.

In the error analysis, I print a confusion matrix which tells me that most of the times it is being confused between two or three similar classes, other wise its working pretty well.



To see what breed pairs are mostly misclassified I print that bar graph to visualize that .

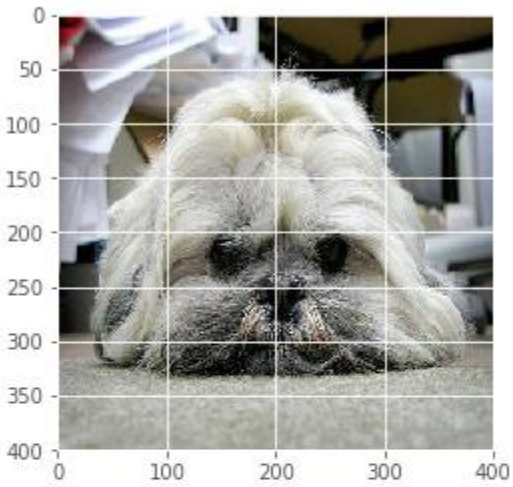


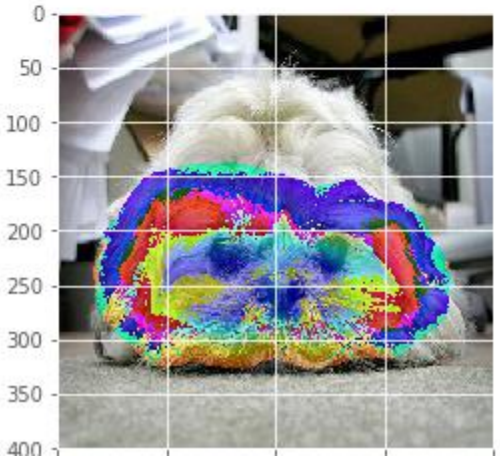
We can see that mostly the terrier pair or the similar type of dogs are misclassified.Even I couldn't tell the difference between the terriers just by looking at the pictures.So the model has done quite a good job in classifying the 120 breeds.

To be sure of that my model is seeing the correct features to predict the class , I have implemented a cam on the inception resnet v2 architecture as I couldn't do so on the hybrid model since it had last convolution layers of different sizes.So as a generalization I implemented cam on the inception resnet v2 which is dominant one in the hybrid model.

This cam shows the heat map or which part of the image was looked to give the prediction.

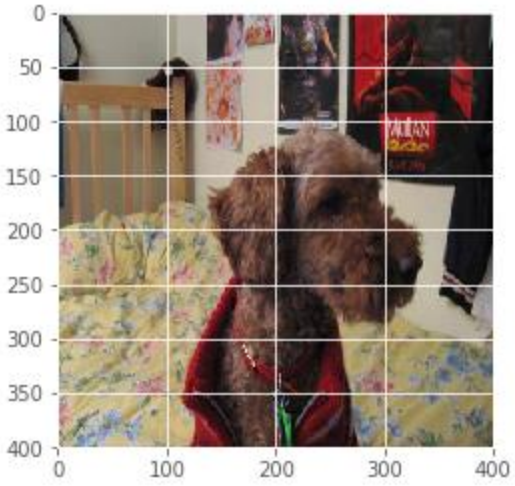
```
ground_truth :  lhasa
prediction :  lhasa
(11, 11, 1536)
(11, 1536)
```

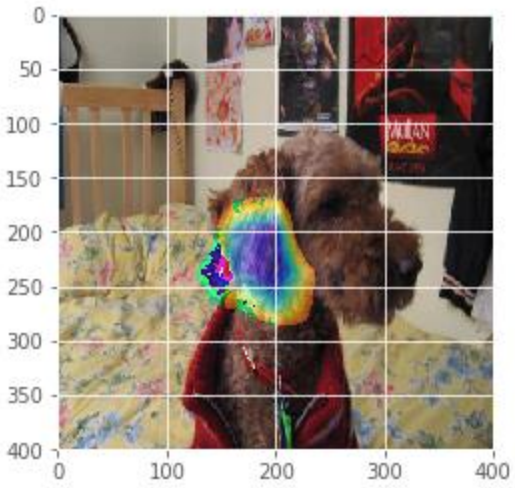




For the wrongly classified images it may look at the similar features or some non relevant part of the image.

```
ground_truth :  miniature_poodle
prediction :  toy_poodle
(11, 11, 1536)
(11, 1536)
```





From these visualization we can infer that most of the times our model is concentrating on the parts of the dogs, the interesting challenge would be to make our model look at what we we want it to look at.

Justification

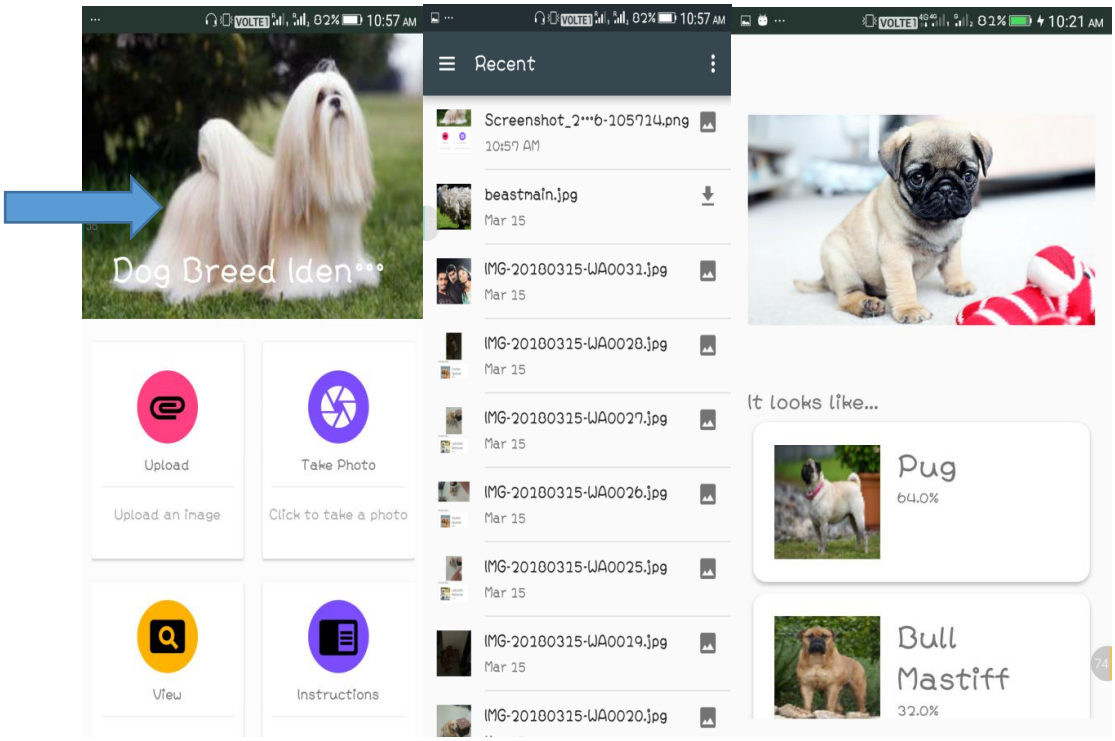
The VGG model has given a test accuracy of 78%, and the inception an accuracy of 92, but this model gives an accuracy of 95.65 with a logloss of 0.16.

This shows that our model which was a combination of different model could extract more important features which could boost up the validation accuracy and also decrease the log loss.

V. Conclusion

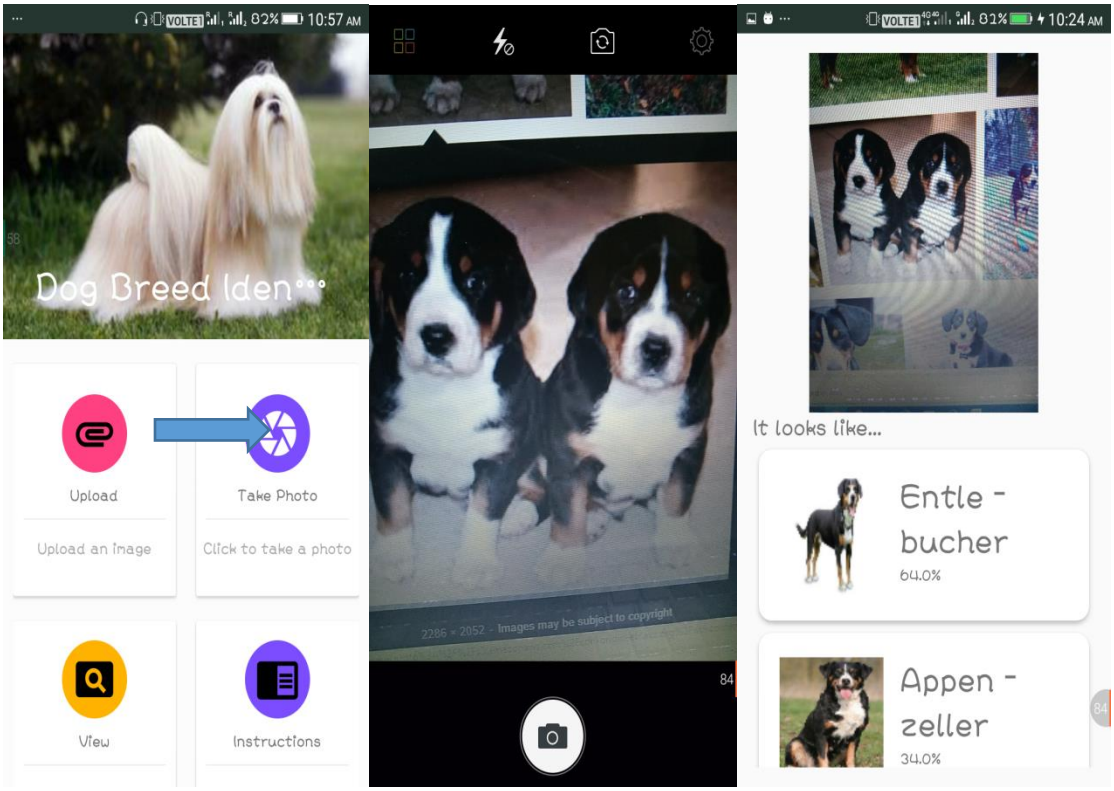
Free-Form Visualization

Here are some of the screenshots of the android app deployed using our model as the backend.



We can see that it predicts the image as pug with 64% onfidence and the next best prediction is bull mastiff.

The important thing is , it even recognizes the pup image of the dog even though it wasn't present in the dataset.



The images captured were noisy and of pup , but still the model could predict the corres=crt class with a 64% confidence.

Reflection

In this project , I have built a model which utilizes the three pre-trained model's bottle neck features and there upon built a fully connected network on these concatenated features .

As I had less data , I went with pretrained models ,as the scratch cnns weren't performing well, I aslo added the flipped images to the daaset to increase the samples.

After building it , I verified whether my models predictions could be trusted using the confusion matrix , bar graph of most misclassified pair and also the cams to gain an intuition regarding how my model is performing.

Here it was interesting as I could what my model looks at to give a prediction and if it ave a wrong prediction , I could gain an intuition to why it is giving that.

The difficulty I identified is to even classify the similar looking dogs at any given posture or colour.More approaches are to be explored to make our model robust to them.

The final model was deployed on android platform using a server and link to application .When mobile data is not present a small model is there as the backend in the app which is the mobile net trained on our data.

The final solution is quite similar to what I had in my mind but I feel that it can be made more better.

Improvement

Here I am using the mobile net a backend for the app when there is no mobile data, but instead I can use specific techniques to reduce the size of the final model I trained to reduce the size to make it small and to be used as the backend for the real time application instead of it neing on a server.

I can use the face land mark detection to crop out the dog face and then train my model on the dog faces

I think with more information like the its habits or some added text information my model's accuracy can be surpassed using the cnn and nlp techniques.