Dog Breed Classification Presentation

A short synopsis of the final lab

Jean-Sebastien Roger[1]

[1]Florida Atlantic University

Biostatistics, Fall 2020

Abstract

In this experiment, we were tasked with tuning a model to classify our own curated dataset. Classifying images of dogs with the correct breed was the specific goal which drove the creation and modeling of this dataset. Our classifications consisted of 4 different dog breeds, each with over 400 images to use for training purposes. Below are the 4 dog breeds incorporated into this dataset:

- Husky
- Labrador
- Poodle
- Pug

Our experiment was successful in the sense that, we were able to create a dataset of dog images, split the images into test and train, load the images, properly transform and normalize our images into tensors, re-tune a ResNet-18 model using our training set, and correctly classify the majority of our validation set. The next slide has numerous examples of the images used in training.

These images below have been cropped, transformed, and normalized

Examples

Procedure

Using python code, we:

1. Download images from Google search and curate data set
2. Split data into train and valid
3. Read in our dataset of images
4. Crop each image, transform it into a tensor, and normalize the resulting tensor
5. Load in a pre-trained model: ResNet-18
6. Create and use a function to train the model on dog breed training set
7. Incorporate the trained model into another function for visualization on random samples
8. Prove the model's flexibility by reading a personal image and predicting the breed of the dog in the picture
9. Empirically support the accuracy of the model using confusion matrix and evaluation metrics

Model Training

The model training step was the most intensive section of this lab. In the training function we required both the training and validation datasets, that way with each iteration, our training function aimed to optimize the model and minimize the loss while validating. Within our training function we had three key tuning tools:

- Criterion - This allowed us to calculated the loss in each iteration
- Optimizer - This optimized our model allowing us to apply stochastic gradient descent
- Scheduler - This was what controlled the progression of the descent based on our learning rate and optimizer

In the end, we were left with a model tuning function which we could call in a single line:

Final model train function call

$$model = train\_model(model, num\_epochs = 15) \tag{1}$$

Visualization

Our final visualization stitched everything together, allowing us to present the image, the prediction, and the label all in one visual

Example prediction results

Results

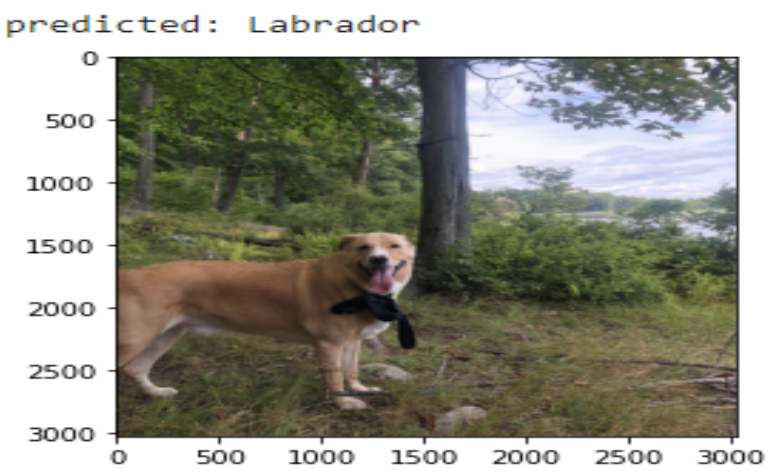
In order to summarize our results, we can look at a confusion matrix of our validation set and various evaluation metrics. From the confusion matrix shown below, we can see that this model is very accurate. We show here some of the main evaluation metrics, with others included in the lab report.

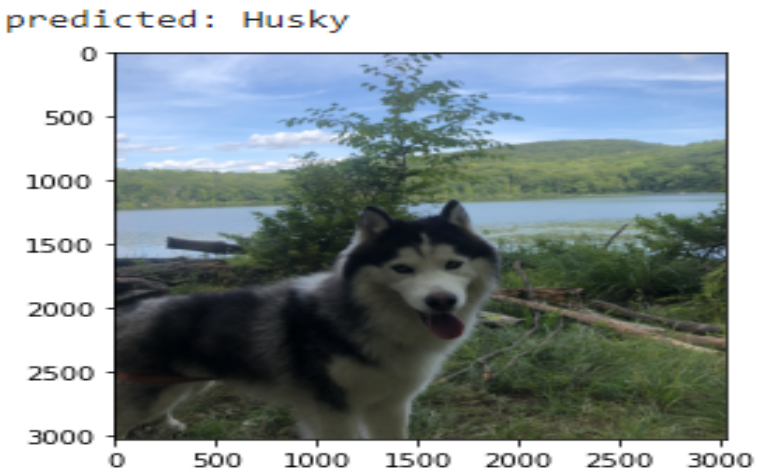
- F1 Score: 0.99788
- Sensitivity: [1, 0.99174, 1, 1]
- Specificity: [1, 1, 1, 0.99719]

```
Confusion Matrix :
array([[118,   0,   0,   0],
       [  0, 120,   0,   1],
       [  0,   0, 117,   0],
       [  0,   0,   0, 115]])
```

Conclusion

From this experiment, we were able to train a pre-built model and validate its accuracy for our own image classification task, we were able to achieve a validation accuracy of around 99%. With the level of detail required to distinguish certain characteristics within images like dog pictures, and associate these characteristics with given labels, the expectation was for a model that did not perform very well.





Our final task within this lab was to ensure that our model could now be used on images outside of the typical datasets used to train and validate. After loading the images of my two dogs from my personal drive account, we see yet again, the model produced two correct classifications.