**User Documentation for the Research Study Titled: A Deep Learning Approach to Fine-Grained Dog Breed Classification**

**Abstract**

Classifying dog breeds is a difficult task that requires a vast knowledge of breed characteristics. Knowing a dog's breed can be helpful in a variety of ways, including health issues, life span, and energy levels. The task of classifying dogs by breed based on images is prime for transfer learning, via a deep learning pipeline. The process of classifying breeds via deep learning is not novel, however, little data exists comparing what neural networks work best for this problem. Along with comparing models this study also aimed to classify mixed breed dogs (non-AKC breeds). This study compares four models with how well they train on the Standford dogs dataset using transfer learning: Xception, Densenet201, ResNet50, and VGG19. The results of training showed that DenseNet201 performed the best with the highest results in both accuracy and loss. Further training via fine-tuning the base model layers led to a model that had a 98.25% accuracy during testing. This study found that the architecture of DenseNet201 was ideal for feature extraction of dog breeds. A method for classification of mixed breed dogs was unsuccessful as there was not enough data to train the models to identify multiple breeds at once, more research is needed to develop this aspect of the study.

# **1. Implementation details**

GPU: NVIDIA GeForce RTX 3060

CPU: Intel i7-10875H @ 2.3 GHz

RAM: 16 GB

OS: Windows 10 Home 64 bit

Python 3.9.7

TensorFlow: 2.6.0

Keras: 2.6.0

Numpy: 1.19.5

Pandas: 1.3.2

matplotlib: 3.4.3

SKLearn: 1.0

TQDM: 4.62.3

# **2. Starting Code and Our Implementation**

We were having trouble with getting anything to train from scratch, therefore we found something that worked as-is and we modified it to fit our constraints. The starting code helped us get a better reference to how everything flowed as well as all the steps needed to train a model efficiently.

* 1. **Starting Code Link**

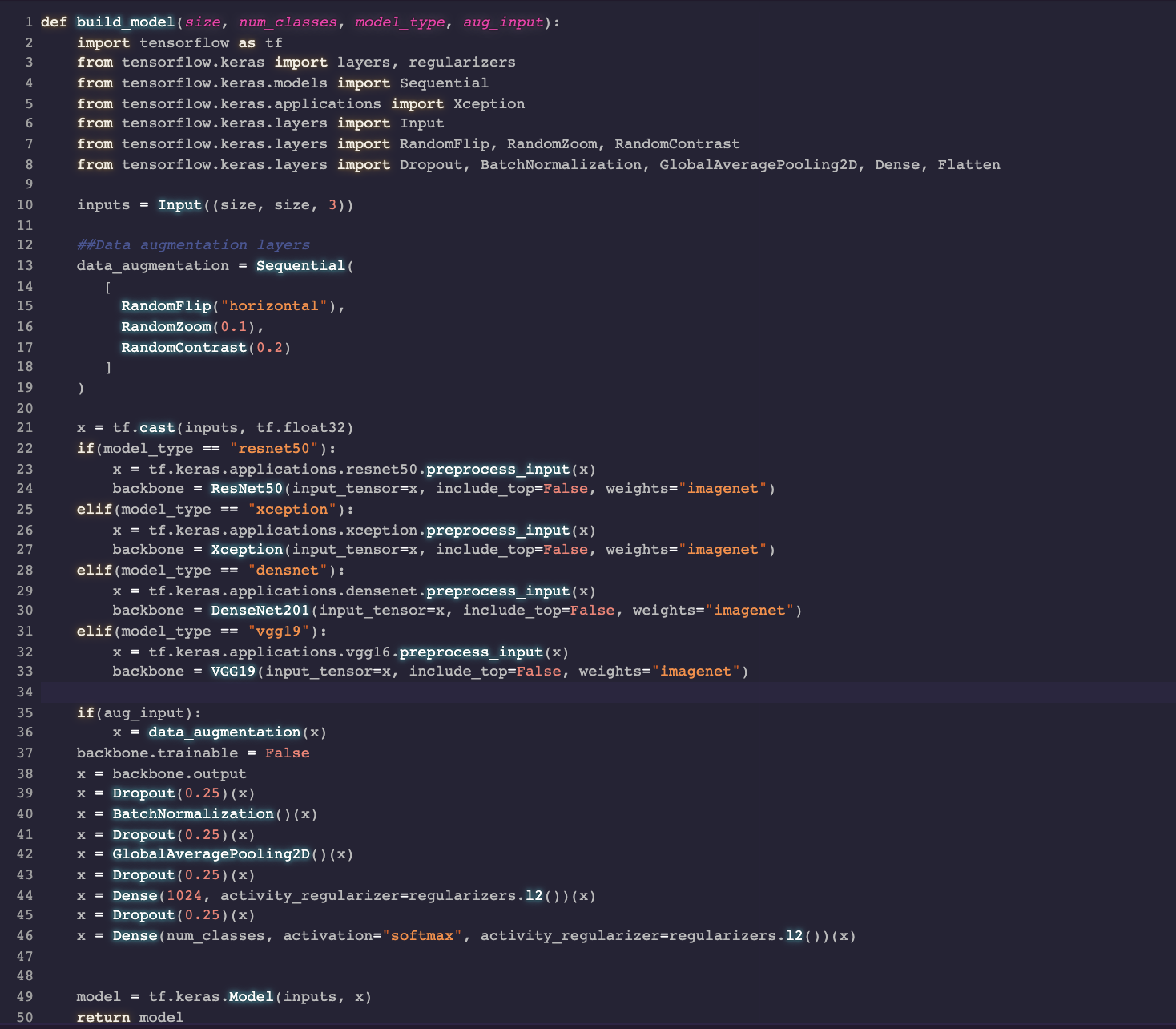
<https://github.com/nikhilroxtomar/Dog-Breed-Classifier-using-TF2.0>

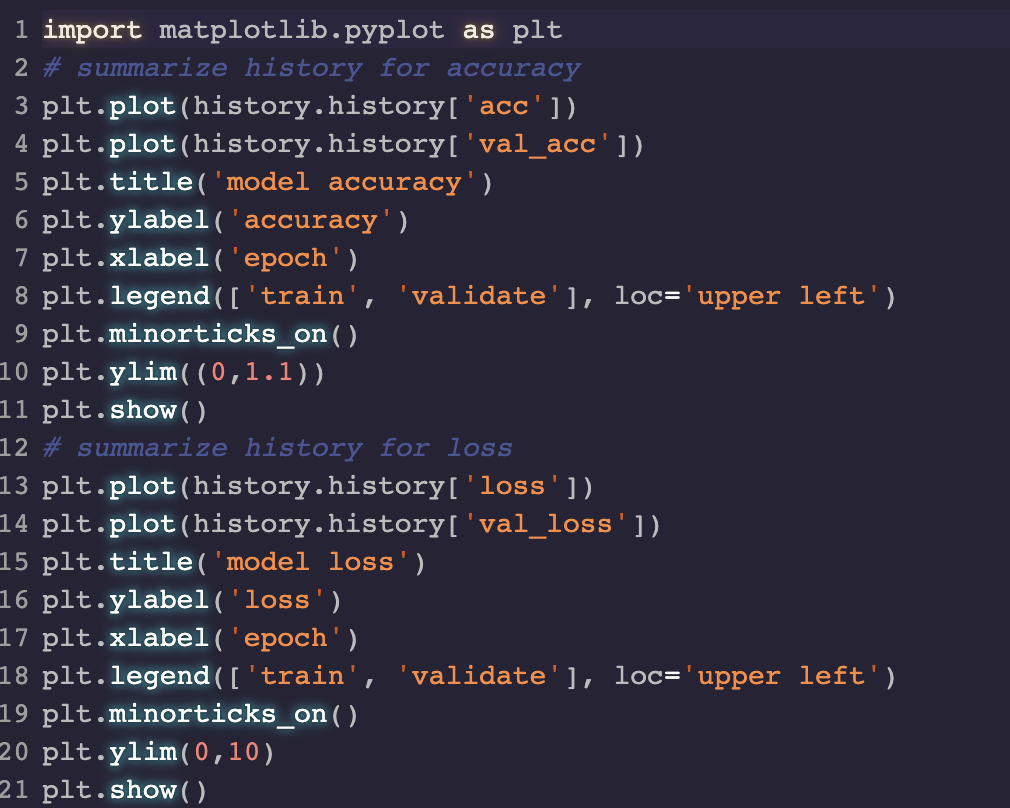
* 1. **Difference Between the Starting Code and Our Implemented Code**

The starting code didn’t contain any comments or documentation, we corrected this by adding comments for nearly everything. We also changed the build\_model function to be able to build one of four models on the fly without having to change more than one variable/line of code. We also added data augmentation layers, however, these were not needed to get the best model trained adequately. The top layers were also changed/added to. We added multiple dropout layers set to 0.25 as well as a batch normalization layer and added activity regularizers in each dense layer. The original code was trained on MobileNetV2, whereas we trained on Xception, DenseNet201, ResNet50, and VGG19. We added graphical output of our training data. We changed the testing method to collect a percentage of accuracy on the test. We also changed the testing method to easily change the number of images being tested. Along with this, the text on the images was hard to read so a white box with overlaid black text was superimposed on the image using cv2. We also, of course, changed up the learning rates, what metrics to evaluate, and implemented early stopping.

* 1. **Our own contributed codes**

In trainDogBreedClassifier.ipynb lines 1 through 50 of the section Helper Functions



In trainDogBreedClassifier.ipynb lines 1 through 21 of the section titled “Plot Data” 

In the file testDogBreedClassifier.ipynb lines 31 through 55 were modified in order to better represent the data and get more quantitative results.



In CreateBreedLabelCSV.py lines 1 through 15 are a script in order to preprocess the data, creating a CSV with a breed name for each image name, without its file extension.

# 

# **3. Datasets**

* 1. **Dataset**

The dataset was collected from the following website.

Dataset link: <http://vision.stanford.edu/aditya86/ImageNetDogs/>

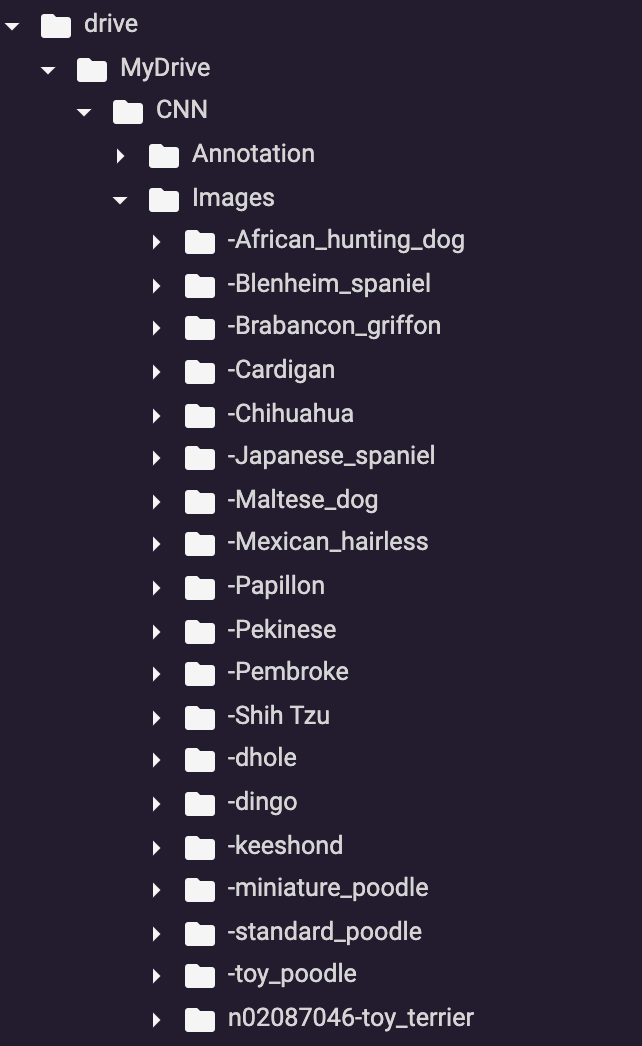
* 1. **Preprocessing**

createBreedLabelCSV.py is the file that was created to collect the image names and map them to what breed was shown. It outputs a .csv file with the file names and their respective labels.

Each base model requires a slightly different input, luckily this is made easy to change base models on the fly using the tf.keras.apptf.keras.applications.{model\_name}.preprocess\_input()method.

* 1. **Dataset path**

Jon’s Dataset path:



Sean’s Local Host Dataset path:

C:/Users/seanm/OneDrive - MNSCU/Fall 2021/CS 465/CS465\_final\_project/datasets/Images/\*\*/\*

The CSV with the labels should be stored in the same directory as the folder containing all the dataset images.

# **4. Execution**

Assuming the use of the provided dataset, the provided labels\_full.csv file, and the google Colab processing resources.

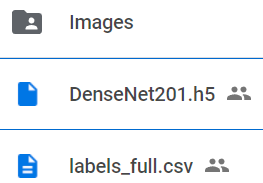
**testDogBreedClassifier.ipynb**

The .h5 files used for each step of training and testing are provided in the code source, and can be used to test the accuracy of the different models at different stages.

1. Set the userDirectory variable in the second block of code to the source directory for this project.



1. Load a copy of the desired base model .h5 file into the datasets directory



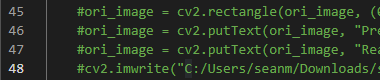
1. Update the second argument of the os.path.join function on line 31 of the Test Model block to use the filename of the desired base model .h5 file



1. Update the denominator variable on line 33 of the Test Model block to represent the number of images to test the model on



1. If the images being tested should be saved with their prediction results, uncomment lines 45 - 48 of the Test Model block and update the first argument of the cv2.imwrite function on line 48 to the desired directory for the images to be saved to



1. If the images being tested should be displayed in the Google Colab results, uncomment lines 50 and 51 of the Test Model block

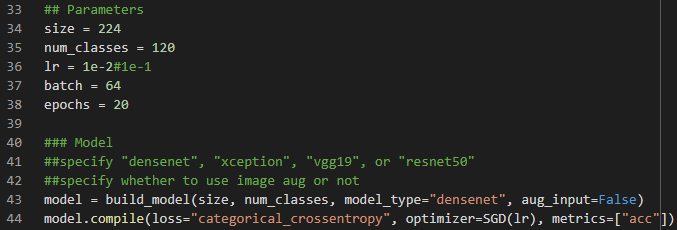


**trainDogBreedClassifier.ipynb**

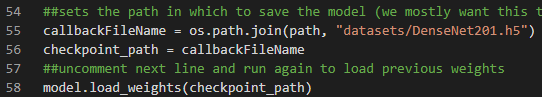
1. Set the userDirectory variable in the second block of code to the source directory for this project.



1. Update the parameters for the model training on lines 36, 37, 38, and 43 of the Build & Train Model block



1. Line 36 is the learning rate of the model
2. Line 37 is the batch size
3. Line 38 is the number of epochs to run
4. The model\_type on line 43 determines the base model using the values on line 41
5. the agu\_input on line 43 determines whether the model using image augmentation or not
6. Update the callbackFileName variable on line 55 to determine where the callback file of the training will be saved



1. Comment out line 58 if the model should not be loaded from a .h5 file a the callbackFileName location

# **5. Final submission link**

Final submission link: https://drive.google.com/drive/folders/1A-VRgZLSYFO6Incj5cmotTn8eG7TahtJ?usp=sharing

# **6. Miscellaneous**

None