

**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.

# **Introduction**

Anyone who has adopted a dog will know that identifying specific breeds in a mixed breed dog is either an expensive mystery to be solved or a guessing game. This study aims to make a tool that makes it easier for the public to identify the specific breed and sub-breeds within their dogs to help them have a better understanding of the medical/health issues their dog may face. Our tool to detect breeds in dogs uses a CNN to classify image data and determine via supervised learning what dog breeds are contained within a photo.

Many breeds of dogs have specific health issues that become very prominent as they reach old age. Being able to determine the sub-breeds of a mixed breed dog would help dog owners understand the possible health issues their dog may face in their old age. This would allow them to take preventative action and improve their dog’s quality of life.

Similarly, breed recognition could help with training by helping determine a dog’s interaction behavior and inherent instincts. These behaviors and instincts are sometimes heavily linked to a dog’s genetics.

Using image classification instead of DNA testing is not only a less expensive alternative to breed detection, but it could also have a less ecological impact due to not requiring DNA samples to be shipped to labs and processed.

The importance of collecting data on which neural networks solve this problem the most efficiently is to aid in further research, saving time for others with similar studies, and adding to the pool of data and results.

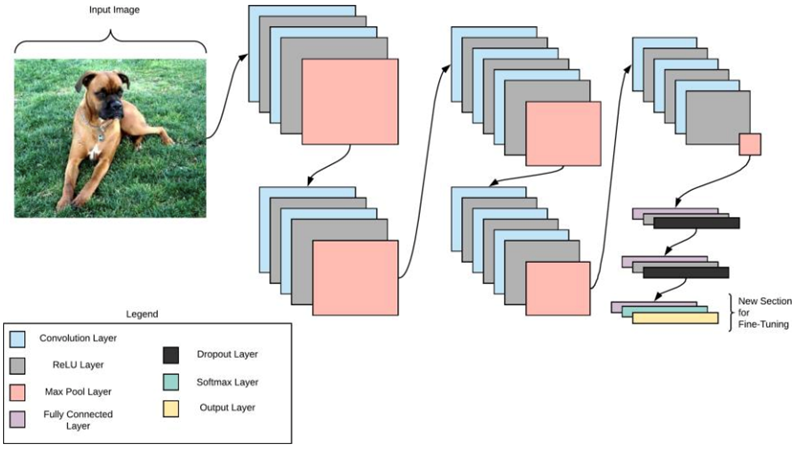
# **Related Work**

Several deep learning approaches to fine-grained breed classification have been done previously. A paper, published in December 2020 and written by Ding-Nan Zou, Song-Hai Zhang, Tai-Jiang Mu & Min Zhang, discussed the creation of a dataset to train several pre-existing models, Inception V3, WS-DAN, PMG & TBMSL-Net (4 classification deep neural networks) [1]. A paper written by Xavier Higa in April of 2019 discusses the use of two different CNN’s, (VGG-16 & Densenet-201) trained on the Stanford Dogs dataset, to determine a specific breed for a dog [2]. Kaitlyn Mulligan and Pablo Rivas published a paper in 2019 about Breed Identification using the Xception neural network (a CNN) [3]. Deep Learning: How to build a dog detector and breed classifier using CNN?! [6] written by Rahil Bagheri goes into some detail about how they built a deep learning model to do fine-grain dog breed classification using VGG19, InceptionV3, ResNet50, and Xception. Solving the mystery of my dog's breed with ML [7] written by Guilherme Marmerola describes how he used the feature extraction of the Xception model and logistic regression to get a very quick and accurate prediction of his dog’s breed. And, our final reference for our research at this point is a GitHub repository called Dog-Breed-CLassifier-using-TF2.0 [8]. This repository has been a helpful guide in how to set up and train a somewhat accurate deep learning model on our project’s dataset.

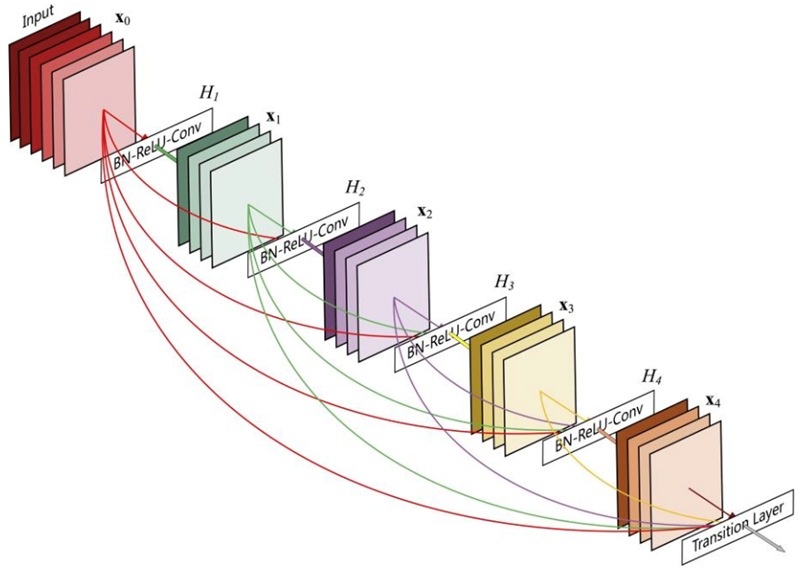
# **Proposed Method**

## **VGG-16 & Densenet-201**

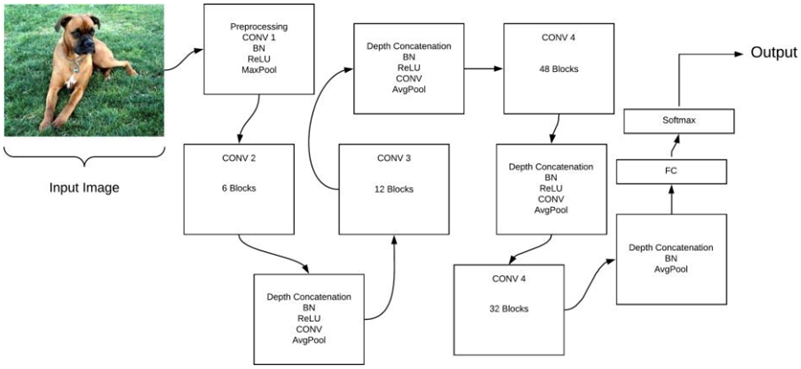
“Dog Breed Classification Using Convolutional Neural Networks: Interpreted Through a Lockean Perspective”, the paper written by Xavier Higa is most like our project topic [2]. The paper discusses using two CNN’s, (VGG-16 & Densenet-201) trained on the Stanford Dogs dataset, to determine a specific breed for a dog [2]. They trained the CNN’s using supervised learning, on labeled image data from ImageNet, then fine-tuned each Neural Network using a test set of the Stanford Dogs Dataset [2]. The two CNNs used were VGG-16 (Figs. 6 & 7) and DenseNet-201(Fig. 8) [2].

****

*Figure 1: VGG-16 Structure [2]*

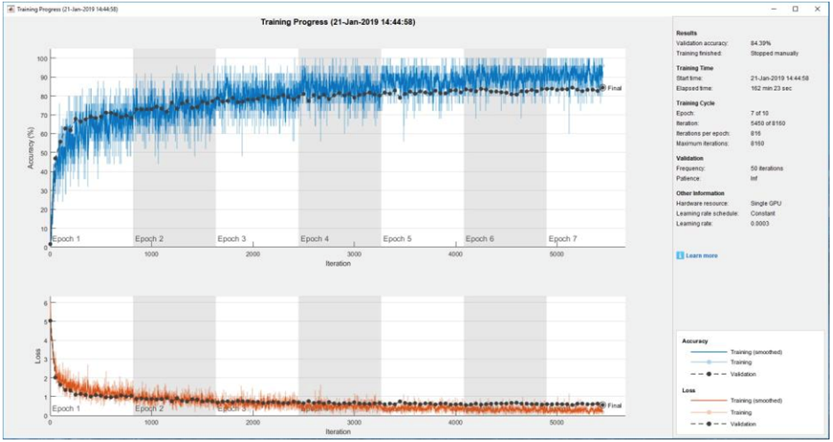


*Figure 2: Schematic representation of a dense block with five layers [2]*

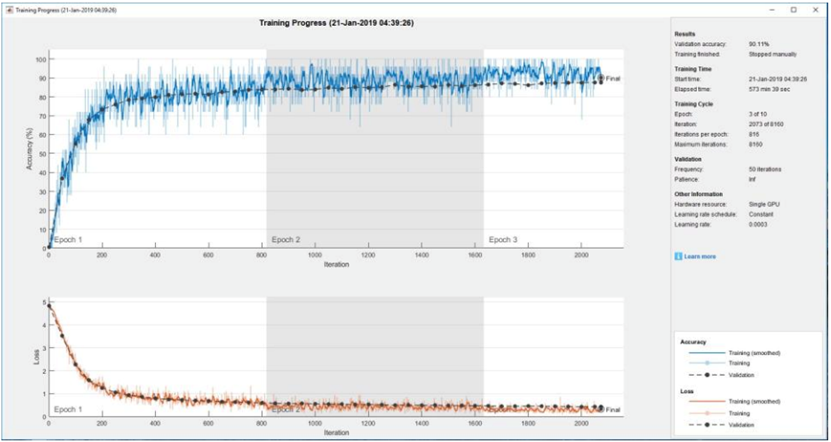


*Figure 3: Schematic representation of DenseNet-201 [2]*

Using MATLAB, they gathered results of training and testing on each Network (Figs. 10 & 11) using a standard method for training both Networks [2].



***Figure 4: VGG-16 Training Progress [2]***



***Figure* 5*: DenseNet-201* Training Progress [2]**

## **Xception Feature Extraction & Logistic Regression Fitting**

Solving the mystery of my dog’s breed with ML [6] is a blog post by a data scientist about the process of using machine learning to determine the breed of his dog whom he got from a local shelter. In this post, the author, Marmerola, describes the steps he takes in some detail. The author sets up his dataset using some custom-built utilities to gather the images’ file paths and break them into a metadata block. Then the Xception neural network is used to extract the features from that metadata block into a .csv file. The primary component analysis module of the science kit learn dictionary is used to reduce the size of the feature set to speed up the pipeline. Then, finally, logistic regression was fitted and evaluated on the feature set.

This whole process runs very quickly on Google Colab Pro (first run: <40 minutes, sequential runs: ~3 minutes) and has a very good accuracy: >80%.

On running a test of this implementation, after almost an hour of environment setup, the model gave this output when given an input image from Google of a Great Pyrenees:

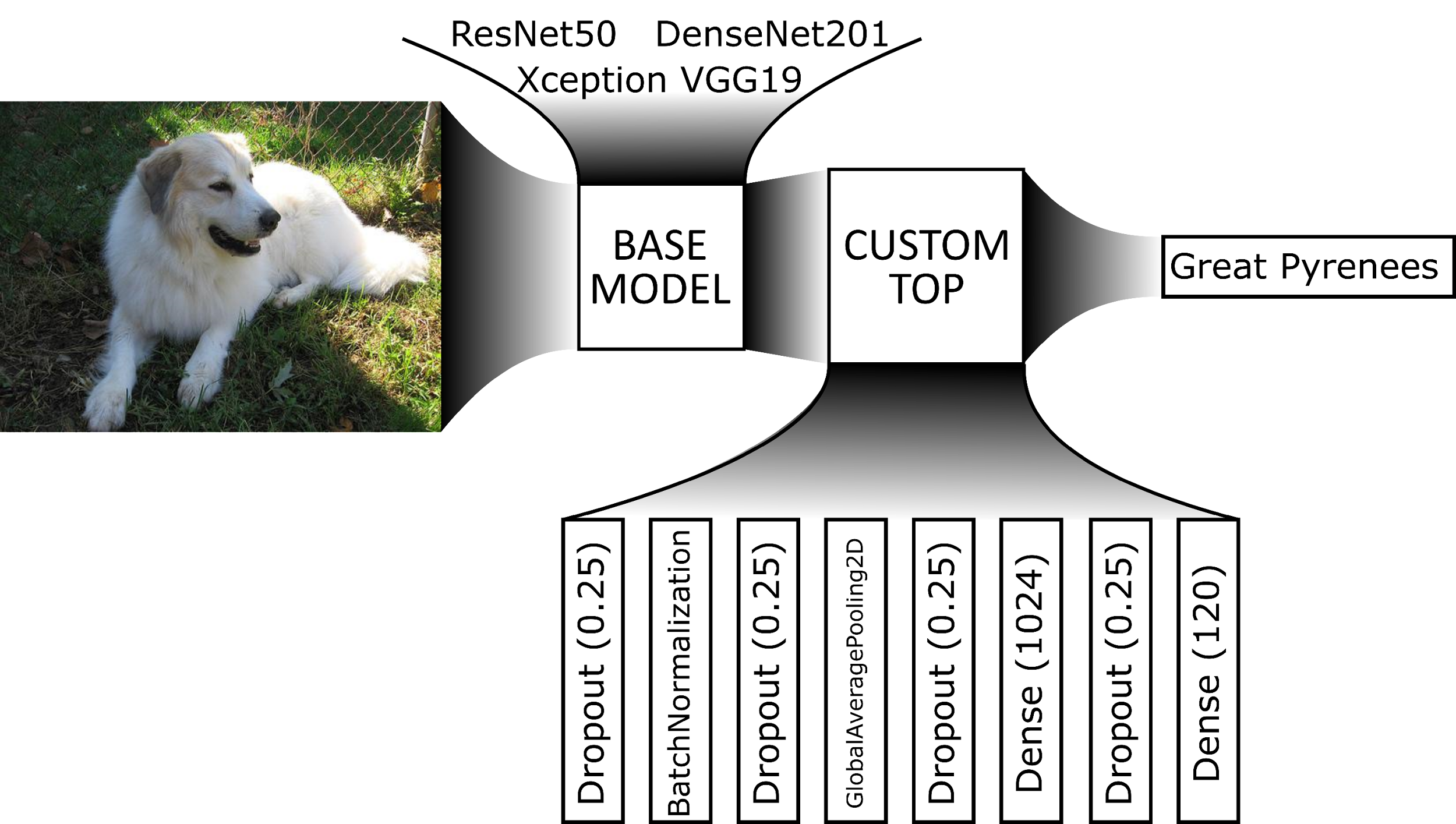
| Great\_Pyrenees | 0.968686 |
| --- | --- |
| kuvasz | 0.008912 |
| golden\_retriever | 0.000995 |
| Maltese\_dog | 0.000529 |
| Australian\_terrier | 0.000502 |
| Saint\_Bernard | 0.000455 |
| Walker\_hound | 0.000401 |
| Tibetan\_terrier | 0.000392 |
| vizsla | 0.000388 |
| Lakeland\_terrier | 0.000377 |

***Table 1: Confidence levels of fine-grained breed prediction***

Overall, the blog’s method has a good accuracy level and the output style this project is aiming for.

## **Our proposed method**

Our research compared the performance of ResNet50, Xception, DenseNet201, and VGG-19 models in the fine-grained identification of different dog breeds. The complete model uses transfer learning with its base being made up of either the ResNet50, Xception, DenseNet201, or VGG-19 models. The model has a set of layers built on top of it. In order, these are: dropout, batch normalization, dropout, global average pooling 2D, dropout, dense, dropout, and dense. These top layers provide a lot of flexibility and trainability for the complete model. This capacity for training is very important because the base model’s weights are set using the default image net pre-trained weights and then frozen throughout most of the training as will be discussed in 4.3.



***Figure 6: Deep Learning Pipeline***

The goal of this research is to determine the best base model for accuracy in the fine-grained identification of dog breeds. To do this the completed models, using each of the base models were trained on the dataset. The results of that training were evaluated for each model. The model with the best results was given another round of training. Then that model was tested against the test data to get the final accuracy.

# **Experiments**

## **Datasets**

Our team used the Stanford Dogs Dataset for our training and testing. This dataset contains 20,580 images of dogs across 120 different breeds. It is also readily available through Stanford University [4]. The dataset was split into 90% for training and 10% for validation and testing. Every image in the dataset was preprocessed according to the requirements of the model being used with the model’s built-in preprocessing method. A simple program was created to tabulate the file name and correct dog breed for every image into a single .csv file. This .csv file assisted in loading the dataset into and training and testing on the different models.

## **Evaluation metrics**

The evaluation metrics for the results of the training and testing in this research were training and testing validation accuracy and validation loss. Validation accuracy provides an estimate of how accurate a model would be on images not within its training dataset. Low validation loss values ensure that a model will provide consistent predictions.

## **Implementation details**

The models used in this research used the OS, CV2, NumPy, Pandas, Glob, TensorFlow, Kera’s, and Matplotlib Libraries. The training and testing of the models were completed on a system with 16GB of ram and an NVIDIA GeForce RTX 3060 Laptop GPU using Google Colab as a workspace.

The four completed transfer learning models, based on ResNet50, Xception, DenseNet201, and VGG-19, were each trained on the complete training split of the dataset. The weights of the base for each model were frozen and untrainable for the first two iterations of training. Each model was initially trained across a maximum of 20 epochs, in batches of 64 images, with a learning rate of 0.1. The training used an early stopping system that would finish the training early if after five epochs the validation loss of the model being trained did not improve.

*See 4.4.1.1 for results from round 1 of training.*

Each model was then trained once more using the same specifications, except that the learning rate was reduced to 0.01.

*See 4.4.1.2 for results from round 2 of training.*

The model with the best results from this training, the model with the DenseNet201 base, was then selected for fine-tuned training. For this round of training, the model weights of the transfer learning model’s base model were unfrozen and allowed to be adjusted through training. The training used the same specification as the previous round: max 20 epochs, 64 images per batch, and learning rate of 0.01.

*See 4.4.1.3 for results from round 3 of training.*

After completing this final stage of training the model was tested to confirm its accuracy. As was mentioned in 4.1, testing was done on the validation split of the dataset. Three tests were run on different-sized sections of the testing data split. One on fifty images, one on one thousand images, and the final test on the complete 2,058 image data split.

*See 4.4.2 for results from testing.*

## **Results**

**4.4.1. Training**

The graphs of the epoch-by-epoch results from each round of training provide some insights into how that round of training affected the model.

**4.4.1.1.** **Training Round 1**

Line chart

Description automatically generated

***Figure 7: ResNet50 Training Round 1: Accuracy***

Graphical user interface

Description automatically generated with medium confidence

***Figure 8: ResNet50 Training Round 1: Loss***

Text

Description automatically generated with medium confidence

***Figure 9: Xception Training Round 1: Accuracy***

Graphical user interface

Description automatically generated with medium confidence

***Figure 10: Xception Training Round 1: Loss***

A picture containing graphical user interface

Description automatically generated

***Figure 11: DenseNet201 Training Round 1: Accuracy***

Graphical user interface

Description automatically generated with medium confidence

***Figure 12: DenseNet201 Training Round 1: Loss***

A picture containing graphical user interface

Description automatically generated

***Figure 13: VGG19 Training Round 1: Accuracy***

Graphical user interface

Description automatically generated with medium confidence

***Figure 14: VGG19 Training Round 1: Loss***

| Training Round 1 Results | | | | |
| --- | --- | --- | --- | --- |
| Base Model | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy |
| ResNet50 | 0.8832 | 0.9279 | 1.3031 | 0.7162 |
| Xception | 4.1828 | 0.1370 | 4.5909 | 0.0811 |
| DenseNet201 | 0.7611 | 0.9192 | 0.9486 | 0.8095 |
| VGG19 | 1.5191 | 0.6987 | 1.5720 | 0.6531 |

***Table 2: Round 1 Training Results***

As can be seen from figures 7 – 14 the training accuracy of the model surpasses the validation accuracy within the first five epochs. Across the rest of the epochs of training the validation, accuracy makes small gains and losses but stays in the same general area. The validation loss follows a very similar pattern. Almost entirely leveling off after five epochs. These training results show that the custom layers on top of the model had plenty of trainability to make all the classifications. With the large dataset, we were able to avoid most overfitting problems.

**4.4.1.2. Training Round 2**

Graphical user interface

Description automatically generated with medium confidence

***Figure 15: ResNet50 Training Round 2: Accuracy***

Graphical user interface, text

Description automatically generated with medium confidence

***Figure 16: ResNet50 Training Round 2: Loss***

Graphical user interface, text

Description automatically generated with medium confidence

***Figure 17: Xception Training Round 2: Accuracy***

A picture containing graphical user interface

Description automatically generated

***Figure 18: Xception Training Round 2: Loss***

Graphical user interface

Description automatically generated

***Figure 19: DenseNet201 Training Round 2: Accuracy***

Graphical user interface

Description automatically generated with low confidence

***Figure 20: DenseNet210 Training Round 2: Loss***

A picture containing graphical user interface

Description automatically generated

***Figure 21: VGG19 Training Round 2: Accuracy***

Text

Description automatically generated with medium confidence

***Figure 22: VGG19 Training Round 2: Loss***

| Training Round 2 Results | | | | |
| --- | --- | --- | --- | --- |
| Base Model | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy |
| ResNet50 | 0.7631 | 0.9530 | 1.2516 | 0.7250 |
| Xception | 4.1310 | 0.1431 | 4.5773 | 0.0807 |
| DenseNet201 | 0.6764 | 0.9405 | 0.9161 | 0.8071 |
| VGG19 | 1.4268 | 0.7244 | 1.5399 | 0.6531 |

***Table 3: Round 2 training results***

Figures 15 – 22 show that the second round of training with the reduced learning rate did not impactfully improve the validation accuracy or validation loss of any of the models. This round of training was not necessary and could be skipped in a replication of this research. The green values in table three represent values that improved with this round of training. The red values in table three represent values that declined from the previous round of training. The black values in table three represent values that have not improved or declined since the previous round of training.

**4.4.1.3. Training Round 3**

A picture containing text

Description automatically generated

***Figure 23: DenseNet201 Training Round 3: Accuracy***

A picture containing text

Description automatically generated

***Figure 24: DenseNet201 Training Round 3: Loss***

| Training Round 3 Results | | | | |
| --- | --- | --- | --- | --- |
| Base Model | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy |
| DenseNet201 | 0.3579 | 0.9991 | 0.5847 | 0.9091 |

***Table 4: Round 3 (fine-tuning) training results***

Figures 23 & 24 show that the third round of training, the fine-tuned training, gave a ten percent improvement to the training and validation accuracy. As in table three, the green values in table four represent values that improved with this round of training. Being able to make small updates to all the weights in the base model greatly improved the validation accuracy and loss.

**4.4.2. Testing**

The image outputs from and calculated accuracy of the model testing show the quality of the completed model.

**4.4.2.1. Image Outputs**

These images come from the output of the first test of fifty images on the final model.

A dog in the grass

Description automatically generated with medium confidence

***Image 1: Correctly predicted Basset***

A picture containing text, dog, brown, indoor

Description automatically generated

***Image 2: Correctly predicted Walker Hound***

A dog sitting in a car

Description automatically generated with medium confidence

***Image 3: Correctly predicted Pembroke***

A dog with its mouth open

Description automatically generated with low confidence

***Image 4: Correctly predicted Brittany Spaniel***

Images 1 - 4 are all very similarly colored dogs from different breeds. The completed model correctly identified each of them despite their similarities.

A dog sitting in the grass

Description automatically generated with low confidence

***Image 5: Incorrectly predicted Border Collie***

Image 5 was incorrectly predicted by the final model. However, the model did predict the correct family of dogs.

A picture containing text, dog, brown, mammal

Description automatically generated

***Image 6: Incorrectly predicted Norfolk Terrier***

Image 6 was also incorrectly predicted by the final model. However, the model did choose a breed prediction that was in the same family and that has a very similar appearance to the real breed.

A dog sitting on a rock

Description automatically generated with medium confidence

***Image 7: Norwich Terrier from the dataset***

**4.4.2.2. Calculated Accuracy**

The testing accuracy of the completed model was calculated by dividing the count of the number of images it predicted correctly by the number of images it was given to test on. Table 5 contains the results of the tests.

| Testing Results | | | |
| --- | --- | --- | --- |
| Base Model | Round 1  (50 images) | Round 2  (1000 images) | Round 3  (2058 images) |
| DenseNet201 | 96.0% | 98.0% | 98.2507% |

***Table 5: Testing results***

This testing accuracy of 98% is well above the goal of 90% accuracy.

# **Conclusion**

During this research, the original source code was expanded upon in a few separate ways. First and most notably, the top of the model was expanded upon to provide more trainability. The source project this research is based on only had a top of four layers: one global average pooling 2D, one dropout with a value of 0.2, and two dense layers with 1024 and 120 neurons respectively. The improved top used in this research added a batch normalization layer and three 0.25 dropout layers. These additional layers provide many more weights and biases that could be trained to improve accuracy. The second improvement in this research is that the model was trained across the entire dataset. The source project was only being trained on 1/20th of the entire dataset. Using the entire training set drastically increased the required training time. To overcome this the model had to be run on a local system instead of the google cloud resources. To make the transfer between training resources requires many hours of research and setup. However, it provided great rewards in training and testing speed. Finally, and most important to this research, the model was trained on several different base models. The original source project used MobileNetV2. A deep learning model focused on performance to allow for use on mobile devices. This research replaced this base model with the RestNet50, Xception, DenseNet201, and VGG19 models.

This research could be expanded in a few different ways. A very important improvement would be testing the model on a dataset separate from the one it was trained on. Testing a model on the dataset it was validated with during training does not provide truly reliable results because the model has used those images before. A very simple improvement of this research is to retrain the model with the Xception base model. In previous tests, the Xception model gave much better results than were received by the final model. The image preprocessing for Xception probably is not set up correctly and a few hours of work and research could probably fix it. A final update that could be made would complete the original plan for this model. The output of the model could be expanded to show the top five to ten values of the model’s breed prediction for any given dog image.

# **References**

1. Zou, DN., Zhang, SH., Mu, TJ. et al. A new dataset of dog breed images and a benchmark for finegrained classification. Comp. Visual Media 6, 477–487 (2020). <https://doi.org/10.1007/s41095-020-0184-6>
2. Higa, Xavier. Dog Breed Classification Using Convolutional Neural Networks: Interpreted Through a Lockean Perspective. Diss. Lake Forest College, 2019. https://core.ac.uk/download/pdf/214315107.pdf
3. Mulligan, Kaitlyn, and Pablo Rivas. "Dog breed identification with a neural network over learned representations from the xception cnn architecture." 21st International conference on artificial intelligence (ICAI 2019). 2019. https://www.reev.us/pdfs/mulligan2019dog.pdf
4. Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao and Li Fei-Fei. **Novel dataset for Fine-Grained Image Categorization.** *First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
5. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).
6. Bagheri, Rahil. *Deep Learning: How to build a dog detector and breed classifier using CNN?!* 8 October 2019. Blog Post. 24 October 2021. <https://towardsdatascience.com/deep-learning-build-a-dog-detector-and-breed-classifier-using-cnn-f6ea2e5d954a>.
7. Marmerola, Guilherme Duarte. *Solving the mystery of my dog's breed with ML*. 14 June 2020. Blog Post. 24 October 2021. <https://gdmarmerola.github.io/discovering-breed-with-ml/>.
8. nikhilroxtomar. *Dog-Breed-CLassifier-uisng-TF2.0*. 16 January 2020. <https://github.com/nikhilroxtomar/Dog-Breed-Classifier-using-TF2.0>.