Synopsis of Dog Breed Identification using Deep Learning

Bharadwaj Kacharla, Duo Zhou, Jenny Jiang, Rich Johnson

Deep Learning and Image Recognition, University of Chicago

Course Instructor: Ashish Pujari

* Author names in order of alphabetical order

Table of Contents

Abstract

Our Deep Learning project is based on a dog breed identification problem posted in a Kaggle competition. The objective of the competition is to determine the breed of a dog based on images of dogs. The competition provides a sample of dog images from the ImageNet dataset which is based on the Stanford Dogs dataset that contains 20,580 images from 120 breeds of dogs. The problem type is an example of fine-grained image categorization.

The dog breed identification problem is challenging because there are a limited number of images for each breed of dog. Our data preprocessing starts by filtering down the dataset by requiring at least 100 images for each breed of dog that results in a subset of the dataset of 20 breeds. We expect a deep neural network with more layers to outperform machine learning models and plain neural nets with fewer layers, however, we also recognize, as discussed in the paper titled “Deep Residual Learning for Image Recognition", that deeper models are not only harder to train due to the vanishing gradient problem, but their accuracy saturates and degrades with a large number of layers. Therefore, we approach the dog breed image identification problem by building alternative models including: 1) a baseline machine learning (ML) model, using both Logistic Regression and a Random Forest Classifier, 2) a basic convolutional neural network (CNN) with 7 hidden layers that includes batch normalization to put all the data on the same scale and dropout to prevent model overfitting, 3) a convolutional neural network that is 19 layers deep (VGG-19) including 16 convolutional layers, 3 fully connected layers, 5 max pooling layers, and 1 SoftMax layer, 4) a convolutional neural network that is 50 layers deep (ResNet-50) and pretrained on images of animals from the ImageNet database,5) ResNet-151V2 and 6) Xception from the Keras package.

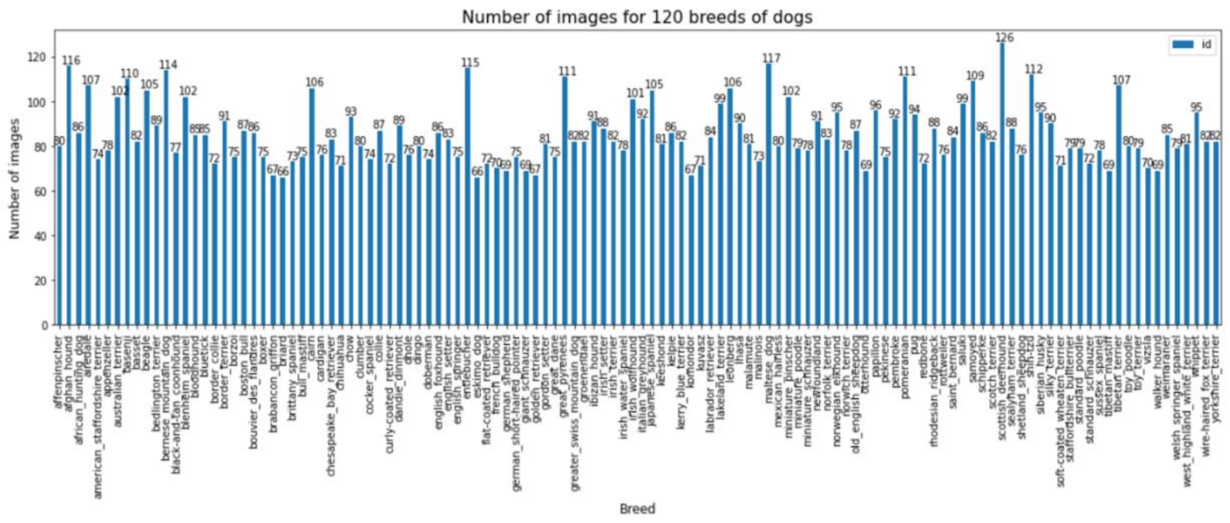
We found that the Xception model (224,224,3) performs the best with near 80% accuracy.

Exploratory Data Analysis

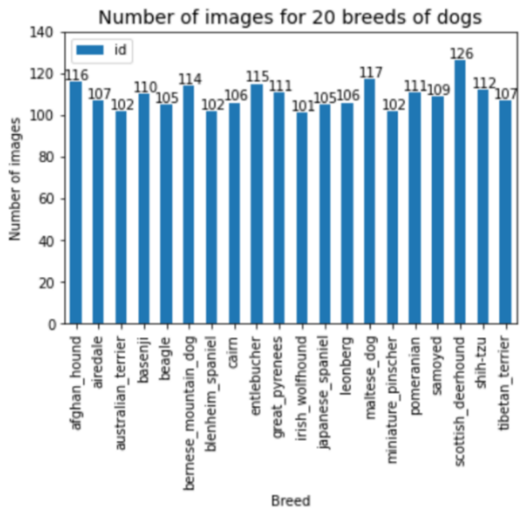
Dataset overview

- **Dataset size**

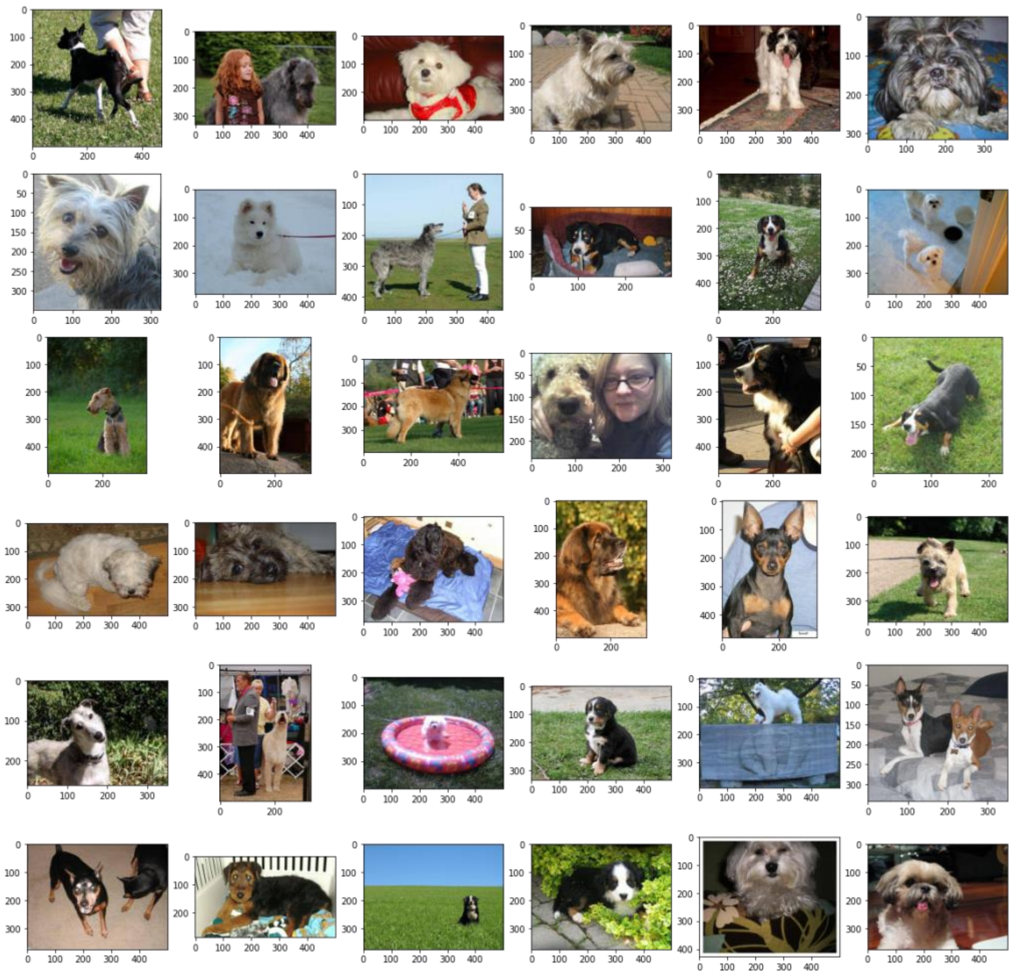
The whole dataset contains 20,580 images from 120 breeds of dogs. Although the number of images varies for different breeds, there are at least 60 images for each breed.



Given that processing 20,580 images is a computationally expensive and time-consuming task, we only choose the top 20 breeds with at least 100 images for each (a total of 2,184 images) for our project. Below are the 20 breeds we pick.



The following are samples of the images in the 20 selected categories.



- **Input Image size**

The images are 2D multichannel images with coordinates format (rows, columns, channels). The number of channels is three for all images. However, since the sizes of images are different, the number of rows and columns of images varies. This suggests that we need to resize the images before feeding them to our models.

Feature Engineering

- **For Traditional Machine Learning Models**

We identified the minimum dimensions for rows, columns, and channels to determine the resizing shape for all images. The resizing shape was (97, 97, 3). We resized all images to that shape and then converted these 3D NumPy arrays to 1D NumPy array to form tabular data. That meant an image with shape (97, 97, 3) was converted to one-row data with 28,227 columns (97*97*3). Since we had 2,814 images, eventually we got tabular data with 2,184 rows and 28,227 columns.

Then we loaded the label data and ensured that each row of the tabular data was matched with the correct label.

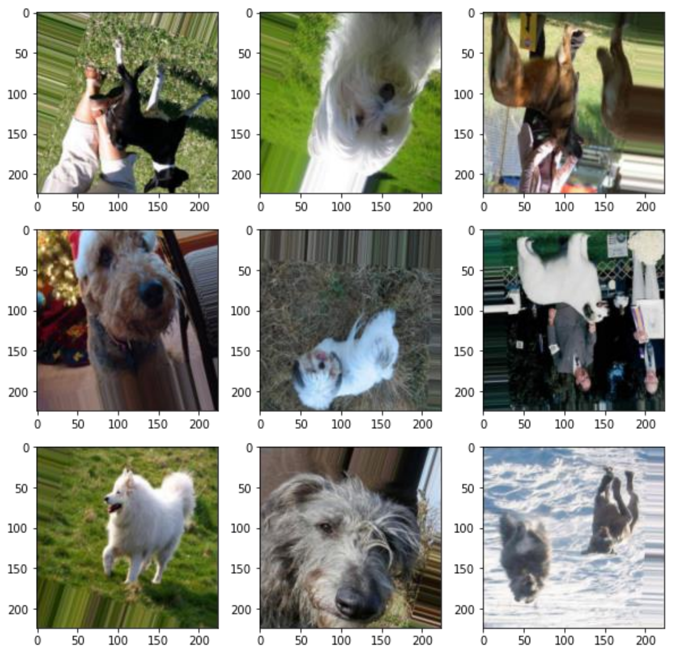
- **For Deep Learning Models**

Apart from the resizing mentioned above, we also resized the shape of images to (224, 224, 3). That was because we planned to implement transfer learning in our project. Many transfer learning models, such as Resnet and Xception, were trained using images with a shape of (224, 224, 3).

Since neural networks usually have an output neuron for each class, we encoded our labels (target variable) using the one vs. all approach. For instance, if an image was labeled as class A, then that class was denoted by 1 and the rest of these classes became 0.

Because regular Convoluted Neural Networks are not rotation invariant, we generated rotated images based on the training data. All images were transformed by randomly rotating the training set where the rotation may have been shifted on both dimensions and then flipped around on both the horizontal and vertical axes. Rotation is a transfer learning technique, and it is believed that rotated image data may improve the performance of a deep neural network.

The following images are examples of training sample data that was transformed by randomly rotating the image samples. The images may have been randomly shifted or flipped.

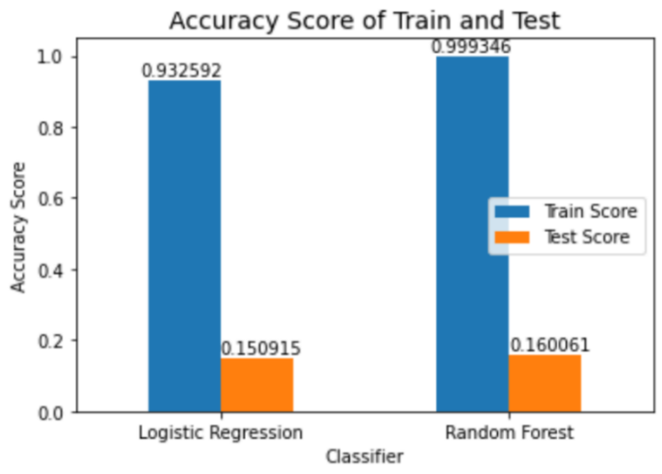


Model Building

Competing Algorithms

Traditional Machine Learning Models

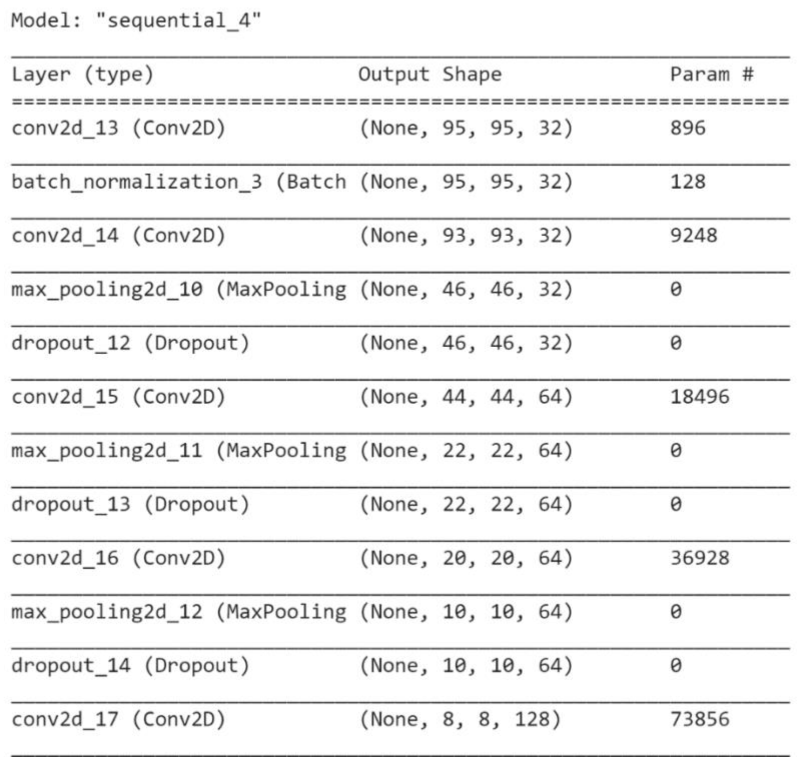
The tabular data with 2,184 rows and 28,277 columns was fed to two traditional machine learning models: Logistic Regression and Random Forest Classifier. Although the accuracy scores for training data reached 93% or above for these two models, the accuracy scores for test data only ranged from 15% to 16%, which indicated very low prediction accuracy. This strong sign of overfitting (low bias and high variance) which may have been caused by having more features than observations.



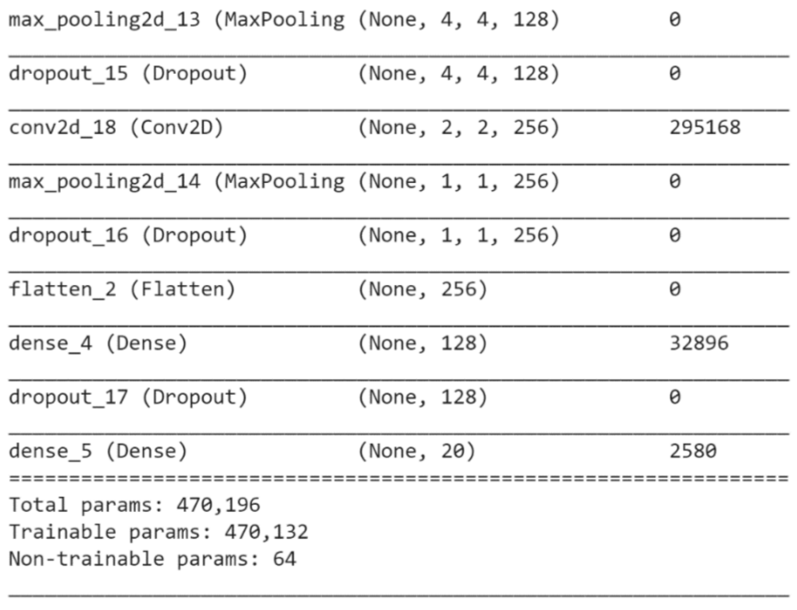
We picked the accuracy score of 16% as our baseline model performance and seek to improve our model performance beyond that point.

Convolutional Neural Networks (CNNs)

We built a Convoluted Neural Network with seven hidden layers and applied batch normalization and dropout in this neural network. Then we fed images of two different sizes to the model. One input size was 97*97*3 and the other input size was 224*224*3. Below is a summary of the model with input size 97*97*3. Total parameters (470, 196) needed to be trained in this process.



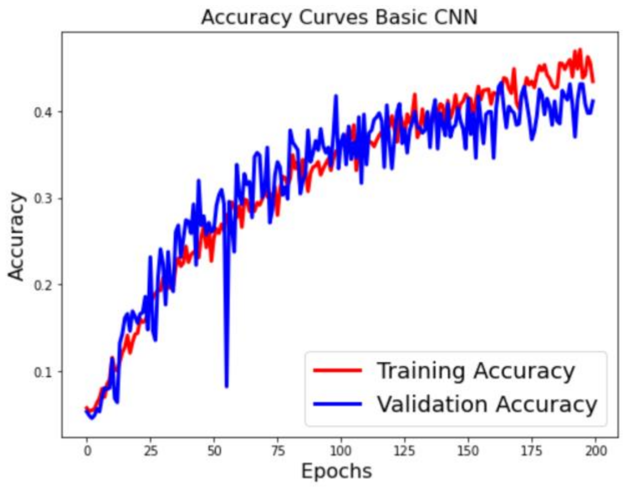
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 95, 95, 32)        896
_________________________________________________________________
batch_normalization_3 (Batch (None, 95, 95, 32)        128
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 93, 93, 32)        9248
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 46, 46, 32)        0
_________________________________________________________________
dropout_12 (Dropout)         (None, 46, 46, 32)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 22, 22, 64)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 20, 20, 64)        36928
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
```



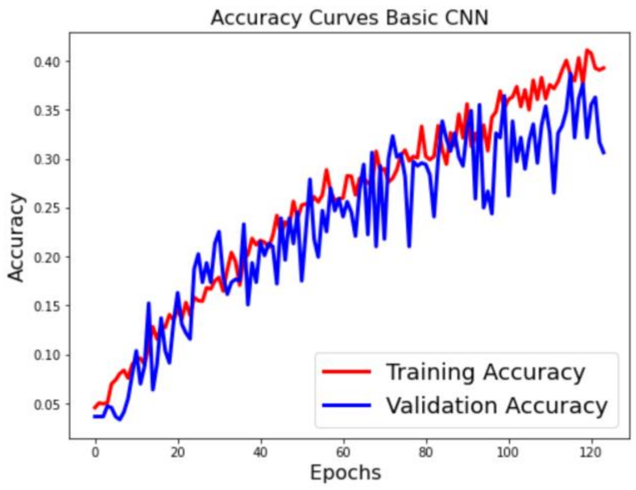
```
max_pooling2d_13 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_15 (Dropout)         (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 2, 2, 256)         295168
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 1, 1, 256)         0
_________________________________________________________________
dropout_16 (Dropout)         (None, 1, 1, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_17 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 20)                2580
=================================================================
Total params: 470,196
Trainable params: 470,132
Non-trainable params: 64
_________________________________________________________________
```

For the input size 97*97*3:

We got test accuracy scores fluctuating between 40% and 45%. The accuracy curve below showed that as the epochs increased, the Convoluted Neural Network extracted more information from the images and was more likely to make correct predictions. Thus, the accuracy scores increased drastically. When the epochs reached 150, the accuracy scores of the train data continued the upward trend, however, the accuracy scores of test data oscillated at a level lower than that of the training set.



For input size 224*224*3:



We got test accuracy scores that oscillated from **30% to 38%.**

In addition to a basic CNN model, we conducted transfer learning by employing the following pretrained models. Two different image resolution sizes, 97X97 and 224X224 were chosen to conduct the model training. 97 is the smallest dimension of image sizes in all of the training samples and 224 is the recommended input image size dimension for many of the transfer learning packages.

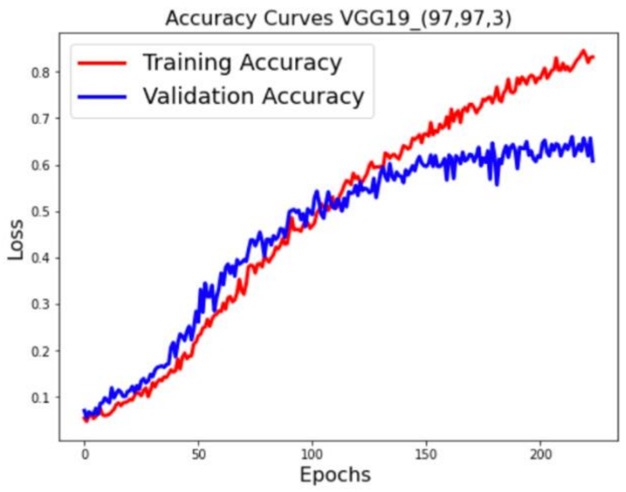
VGG-19

Apart from the baseline models we have also used transfer learning to train our data for better predictions. One of the models is VGG-19. As we already know VGG-19 is a convolutional neural network that is 19 layers deep. A pre-trained version of the model is available that has been trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as a keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

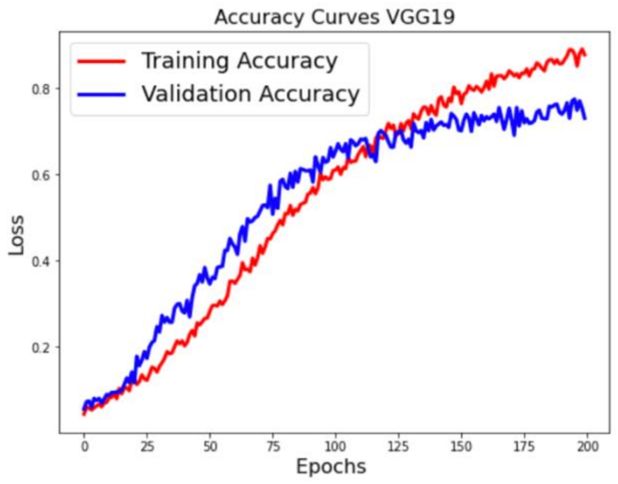
For this specific problem, the first 4 layers of the neural network were frozen. We have trained the models using two image sizes, starting with (97,97,3) and the other being (224,224,3)

VGG-19 with image size of (97,97,3)

The validation accuracy fluctuates between 50% - 60%. Based on the accuracy curve we can observe that as the number of epochs has increased the validation accuracy has increased, however, the model has converged around the range 50% - 60%.



VGG-19 with image size of (224,224,3)



Since the original VGG-19 was trained on the image sizes of 224*224, we have opted to train our dataset using the same image size. Compared to the model with 97*97*3 input size, the 224*224*3 model has performed significantly better. The validation accuracy of the new model has significantly improved when compared to the previous one, the value converges between

65% - 70%. Based on the above graph we can conclude that the model has converged post the 150-epoch run.

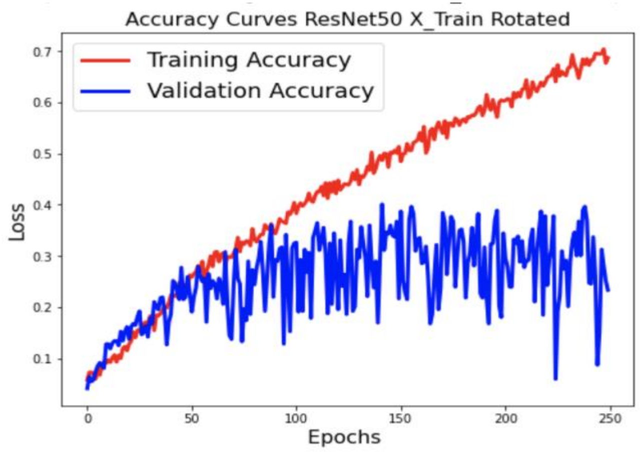
ResNet50

We have also used ResNet 50, as we already know ResNet 50 is a convolutional neural network that is 50 layers deep. We can load a pre-trained version of the network that has been trained on more than a million images from the ImageNet database.

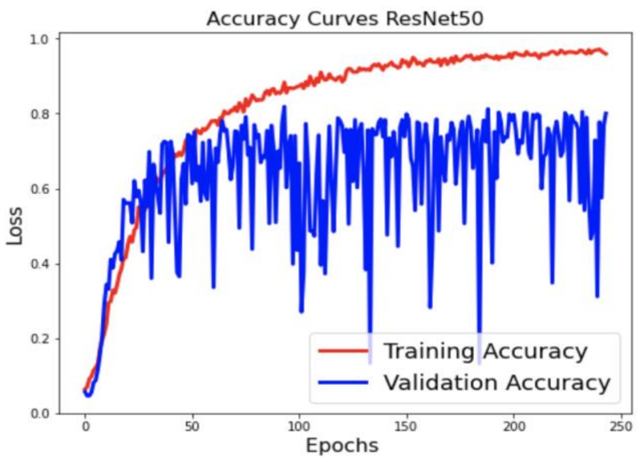
For the purpose of this project, we have frozen the first 40 layers and built a couple of custom neural networks. We have also trained the data on two different images sizes. Firstly, with (97,97,3) as the input size and then (224,224,3) as the input size.

ResNet50 with image size of (97,97,3)

The validation accuracy fluctuates between 25% - 35%, except for a couple of outliers, based on the accuracy curve we can observe that the as the number of epochs has increased the training loss has decreased, however, the validation loss hasn't progressed in the same manner as the train loss. This may be due to the fact that the ResNet50 has been trained on image size of (224,224,3).



ResNet50 with image size of (224,224,3)

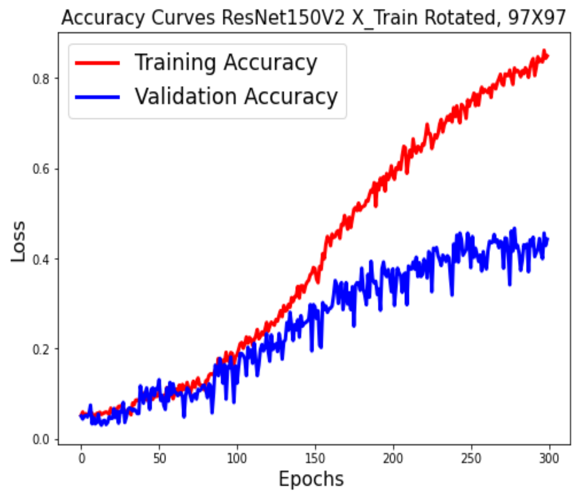


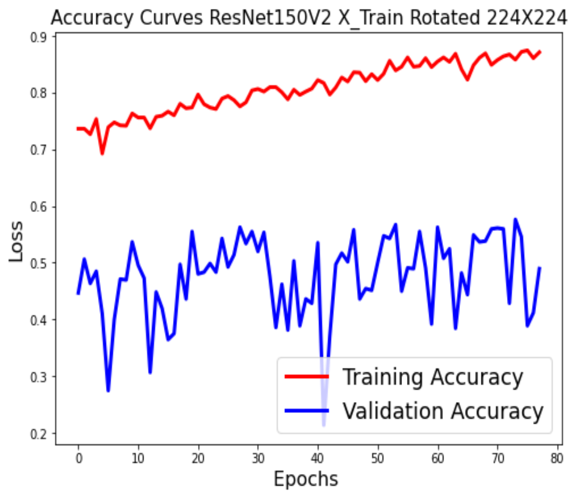
Since the original ResNet50 was trained on the image sizes of 224*224, we have opted to train our dataset using the same image size. Compared to the model with 97*97*3 input size, the 224*224*3 model has performed significantly better. The validation accuracy of the new model has significantly improved when compared to the previous one, the value fluctuates between 30% - 80%. Based on the above graph we can conclude that the model has converged post the 150-epoch run.

We can also observe that the range of train accuracy is fluctuating between 30% - 80%, which is a very large window for train accuracy. Based on this we can infer that the ResNet50 is a highly unstable model when compared to other transfer learning models.

ResNet151V2

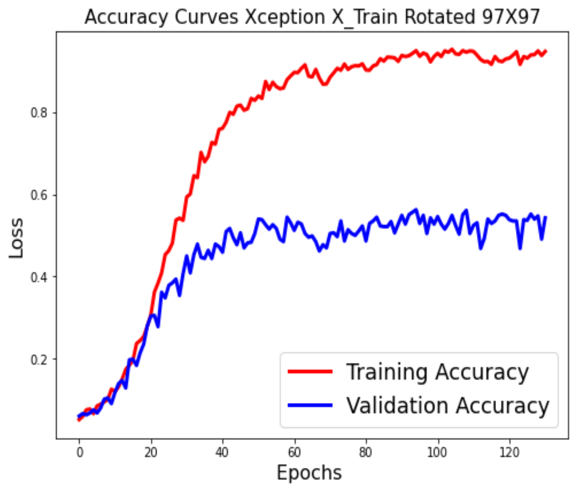
In this model, 50 pretrained layers of Resnet151V2 were frozen as input. Flatten and output layers with a density of 1024 were added to finalize the model. Regardless of the change in the image sizes, RsNet151V2 transfer learning model's test accuracy converges around 40%-50%. There is no significant accuracy improvement with the change of input resolution.

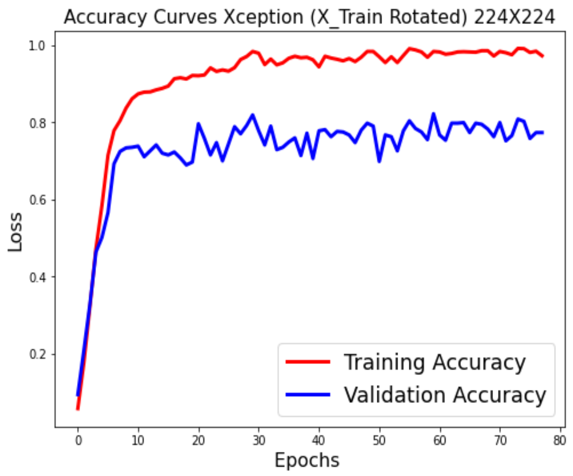




Xception

Finally, a transfer learning model using Xception was trained. Similar to the Resnet151V2 model, 50 pretrained layers of Xception were unfreezed as input. Flatten and output layers with a density of 1024 were added to finalize the model. The test accuracy converges around 40%-50% when using 97X97 input size, however, the test accuracy converges at 70%-80% when 224X224 input size is applied. Input size of 224X224 improves the test accuracy of this model by 25%-30% as compared to 97X97 input size.





Model Performance Evaluation

The below table summarizes the model performance across various neural networks.

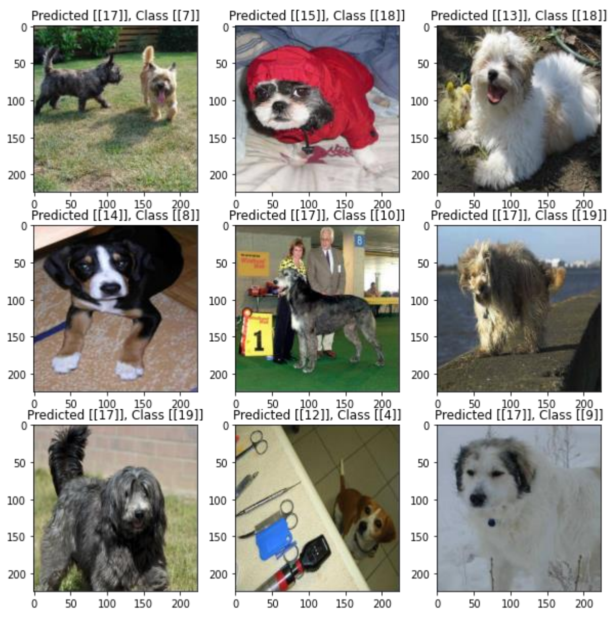
Model	Validation Accuracy	
	(224,224,3)	(97,97,3)
Logistic Regression	N/A	15.09%
Random Forest	N/A	16.01%
CNN	30% - 38%	40% - 50%
VGG-19	65% - 70%	50% - 60%
ResNet50	30% - 80%	20% - 30%
ResNet151V2	40% - 50%	40% - 50%
Xception	70% - 80%	40% - 50%

Best Model

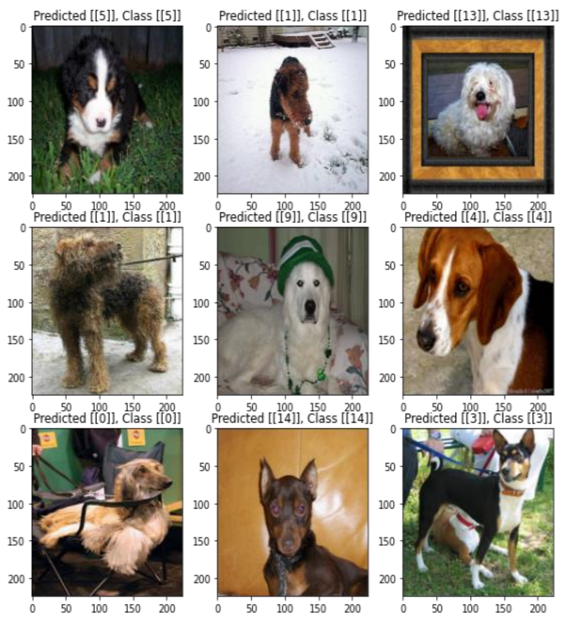
With close to 80% of test data accuracy, Xception with input size of dimensions of (224,224,3). It is important to note that the minimum image size of our training data is (103, 97, 3). The resize function applied in this project will reduce the information preserved in the images with dimensional size less than 224. Image processing functions that can interpolate information should be employed to resize training and test images to preserve the entire original image information.

The following are samples of correctly and incorrectly identified test images.

Incorrect Classification Samples


Predicted [[17]], Class [[7]]
Predicted [[15]], Class [[18]]
Predicted [[13]], Class [[18]]
Predicted [[14]], Class [[8]]
Predicted [[17]], Class [[10]]
Predicted [[17]], Class [[19]]
Predicted [[17]], Class [[19]]
Predicted [[12]], Class [[4]]
Predicted [[17]], Class [[9]]

Correct Classification Samples


Predicted [[5]], Class [[5]]
Predicted [[1]], Class [[1]]
Predicted [[13]], Class [[13]]
Predicted [[1]], Class [[1]]
Predicted [[9]], Class [[9]]
Predicted [[4]], Class [[4]]
Predicted [[0]], Class [[0]]
Predicted [[14]], Class [[14]]
Predicted [[3]], Class [[3]]

We identified following characteristics in the test images that were incorrectly classified:

- The focused subjects, dogs, are overwhelmed by other background subjects, i.e., human, plants and furniture.
- The focused subjects, dogs, are located on the outer edges of the image
- Dogs are oriented in different directions
- Dogs have a lot of accessories being put on them, i.e., clothes
- Dogs have similar color as the background

Model Management

Model management is a new field that acknowledges the challenges faced in implementing machine learning models in practice. Currently, there are no easy answers as there is a lack of best practices. Model management conceptual challenges include 1) model definition 2) model validation 3) model retraining and 4) adversarial settings. Data management challenges also exist. For example, ML pipelines lack a declarative abstraction, meaning they include different abstractions such as data integration, feature transformation, and model training. Moreover, understanding metadata and the linkage between models becomes important. Engineering challenges also exist. Typically, multiple code bases are used and users have differing skill levels. Backward compatibility also needs to be considered. Model management covers a broad area with a lack of clear answers.

We certainly have more work to do in this area. We saved our best model as "best_model.h5". Regarding input data, new images with the shape of (224,224,3), can be fed to the model to obtain a classification result. And the model can be retrained using more images in the future. Model assumptions, data requirements and future model validation all need to be defined.

Conclusion

For the data set in this project, we selected a dog breed identification problem using ImageNet data. Our feature engineering included selecting dog breeds with the most images, resizing images and randomly rotating images, which is a transfer learning technique believed to increase deep learning model accuracy. We ran several competing models, and on (97,97,3) data, the CNN, VGG-19, ResNet151V2, and Xception performed similarly with accuracy in the range of 40% to 50% and performed much better than conventional ML methods. Of the models resized to (224,224,3), the Xception model saw the most improvement and performed the best with accuracy in the range of 70% to 80%. We found that choosing an appropriate transfer learning package can significantly improve test accuracy. Images that were the most challenging to identify included images with background subjects, images placed at the edges, images not facing the camera, images with accessories or clothing, and images with a similar color to the background. Future work may include removing background subjects or changing background color to help improve model accuracy. Target subjects near the outer edge of an image is a challenge for proper identification

References

https://www.kaggle.com/c/dog-breed-identification/data
https://www.kaggle.com/nafisur/dog-breed-identification-keras-cnn-basic

http://vision.stanford.edu/aditya86/ImageNetDogs/

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

https://link.springer.com/chapter/10.1007/978-3-319-11758-4_32

http://sites.computer.org/debull/A18dec/p5.pdf

Resources

www.colab.research.google.com