Dog Breed Identification with a Neural Network over Learned Representations from The Xception CNN Architecture

Kaitlyn Mulligan[1] **and Pablo Rivas**[2]
[1]Department of Mathematics, Marist College, Poughkeepsie, NY, United States
[2]Department of Computer Science, Marist College, Poughkeepsie, NY, United States

Abstract— *Machine learning is a growing field that has greatly increased with the continuing advancements in technology. This area provides many tools that can perform different tasks on large data sets. The focus of this paper is on classification tools. Classification tools are utilized in order to classify or predict the breeds of dogs based on an input image. Many methods are used in attempts to classify the images in the data set. The data set comes from a Kaggle competition in which the goal is to predict the breed of dog in the image. Participants tried many different methods, some of which helped inspire this research. The classification tools that are explored here are a Convolutional Neural Network and Xception with a Multilayer Perceptron. The paper explores the trial and error in all of the methods as well as the final model that was used to predict and classify the dog breeds. While the final model has a much better prediction rate than the original attempt, there is an acknowledgement of the errors made throughout the process. With this acknowledgement comes areas to improve and ideas to further explore this model as a classification tool on dog breeds.*

Keywords: Machine Learning, Classification, Xception, MLP, CNN, Deep Learning, Merge Models

1. Introduction

Machine learning is a widely growing field. Continued advancements in technology have greatly impacted the development of machine learning allowing the field to gain increased momentum. Machine learning allows data analyses to be performed by the computer in place of humans. Although some machine learning algorithms have been around for some time, we have not always had the abilities with it that we do now. With current technology and advancements, we have the "ability to automatically apply complex mathematical calculations to big data - over and over, faster and faster"[6]. Machine learning falls under a branch of artificial intelligence based on the idea that "systems can learn from data, identify patterns and make decisions with minimal human intervention"[6]. The process that the systems use to learn from the data is an iterative process. This is important because as the models continue to be exposed to more and more data, it is able to continuously learn from the new data.

The motivation for this research was to learn how to use a classification tool on images in order to classify the image. To do so, I utilized a data set which consisted of images of different breeds of dogs. The goal was to build a model which will learn the different breeds of dogs in order to predict the breed of dog represented in the input image. While in the process of building this model, I came across many different techniques, of which, I tried a few. First, I tried to use a Convolutional Neural Network to predict the dog breeds. After trying a few different settings with the CNN, I decided to then try Logistic Regression using Xception and a Multilayered Perceptron. Once the optimal model was found, it could then be analyzed to see how well it performs in classifying the different dog breeds. Utilizing these results, I then determined where improvements could be made or if different methods should be attempted to further improve the systems' ability to predict the dog breeds.

2. Background and Related Work

Throughout this process, I analyzed Convolutional Neural Networks and Xception. Convolutional Neural Networks are gaining popularity when it comes to classifying images, but I wanted to find a method that would better predict the breed of dog. Therefore, the method I decided to use for predicting was Xception and an MLP. Xception seems like its own method, but in simple terms it is a very large convolutional neural network. It is "trained on more than a million images from the ImageNet database" [9]. Xception is 71 layers deep and can classify a variety of different images. Xception is a newer method compared to Inception. It was named Xception because it is an "extreme version of Inception" [8]. With Xception, I used a Multilayer Perceptron. The Multilayer Perceptron is a deep, artificial neural network. In [1], they state that "the multilayer perceptron is the hello world of deep learning: a good place to start when you are learning about deep learning."

There are many examples on the internet of how people use different methods to classify images. In regards to dog breed identification, many people have attempted to make a model because it was a Kaggle competition. Many people participated in this competition, using a variety of different methods in an attempt to predict the dog breeds. After

viewing some of the participants' ideas on how to go about this problem, it seemed that using a Convolutional Neural Network was a fairly popular method. Participants also used Xception, Inception, and other methods to predict the breeds of dogs.

3. Methodology

The process of programming a system to predict the breed of dog in an image required a great deal of trial and error. I made several attempts before deciding to use Xception and an MLP to predict the breed of dog in an image. Initially, I planned to use a Convolutional Neural Network. During my time in a machine learning class, I learned about CNNs and their applications. I realized that it would be a good method to apply as I attempted to predict dog breeds based on images. During the process of learning about CNNs, my classmates and I applied them to classify images that we took from around the building. This data set included chalk, lights, chairs, and a few other items. The model created was able to correctly predict to a certain extent. From this lesson, I adapted the model to my data set to try to predict dog breeds. In using the CNN, I first tried it with two convolutional layers and five full connection layers. These full connection layers consisted of relu, sigmoid, and dropout. I then attempted another version of a CNN without any convolutional layers. This produced a few problems which will be touched upon in the following section.

For this research, as mentioned, I utilized Xception and a Multilayered Perceptron. Before going into the experimentation phase of implementing my model, I will establish some basics of these methods.

Xception utilizes depth-wise separable convolutions. In total, there are 36 convolutional stages within this methodology. The Xception model performs the one by one convolution first, and then moves into the channel wise spatial convolution. This is depicted by Fig. 1.

Xception does not have an intermediate activation. Due to this, it has the highest accuracy compared to other methods, such as Inception [8]. Lastly, Xception performs better due to the better use of the model parameters [8].

Briefly described earlier, the Multilayer Perceptron is a deep, artificial neural network. Multilayer Perceptrons are composed of three key components. The first is the input layer. This is the layer that receives the signal. Next, I have an arbitrary number of hidden layers. Lastly, I have the output layer. This is the layer that makes the decision or prediction regarding the input. A simple model of a Multilayer Perceptron is depicted in Fig. 2.

With the MLP, I implemented classification. The classification provides the results that can be utilized to determine how well the system is performing. Metrics such as confusion matrices, LogLoss, and Balanced Accuracy Score are all provided as output to understand the performance of the system.

When implementing Xception and the MLP, the data is split into testing and training data. The model then learns on the training data. The goal for the system is to obtain a high accuracy rate and minimize the error by adjusting parameters. With this comes some trial and error which will be explored in the following section.

4. Experiments

4.1 Data

The dog breed data set, which was the focus of this research, came from a Kaggle competition, Dog Breed Identification[1], in which the task was to determine the breed of a dog based on an image. The data set included a training and testing set of images for different breeds of dogs. Each image has a file name that corresponds to a unique ID which can be found in the file `labels.csv`. This file contains the ID which is the file name, and the breed of dog in the image. A preview of this file is shown in Fig. 3. The data set is comprised of 120 unique dog breeds. For my model, I only utilized the training data set provided and from there, split it into training and testing sets. The original training data set included 10,222 images of dogs.

4.2 Convolutional Neural Network

To reiterate, my initial idea was to use a CNN to predict dog breeds. After making the necessary design choices, I was able to begin experimenting with it. I started off with a small number of epochs and a small batch size. In doing so, I did not obtain sufficient results. From this initial run, I realized I had to keep adjusting the values of the parameters and try larger and smaller values for the number of epochs and the batch size. In addition to this, I also tried different combinations of the two. After trying several combinations, results obtained were still not ideal. I continued to change things around and moved on to trying different full connection layers. I did not initially have dropout layers in my CNN so I tried to add some. Once changing my full connection layers, I continued to change around the number of epochs and the batch size to see if I could improve my results. After trying to predict some dog breeds, I realized that the model was greatly overfitting the data. Overall, throughout these attempts with my CNN model, the highest accuracy rate I obtained was 10%. After consulting the professor, I decided to change some more aspects of the CNN. I changed more parameters and got rid of the convolutional layers. After several attempts to make this work, I continually obtained errors. From there, I decided to try a different method instead of the CNN.

[1] `www.kaggle.com/c/dog-breed-identification`

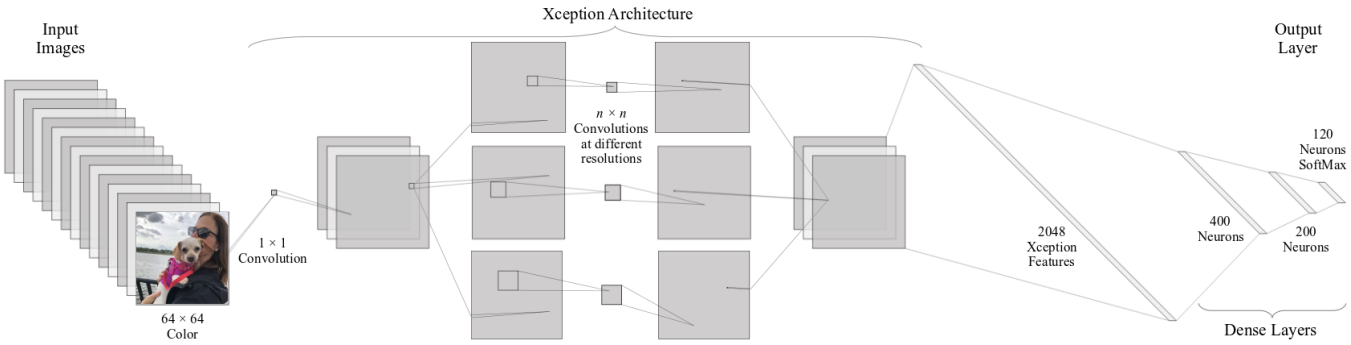

Fig. 1: Proposed model architecture that uses the Xception model [10]. First the model receives input color images of size 64×64 that go through the Xception model using 1×1 convolutions (point-wise convolutions) followed by $n \times n$ convolutions at different scales (depth-wise convolutions) leading to a feature vector of size 2048. This feature vector is connected to a series of dense layers using the traditional perceptron model ending in 120 SoftMax neural units, one for each class. Note that the Xception network is not re-trained, but it is independent of the dense network, which is trained separately.

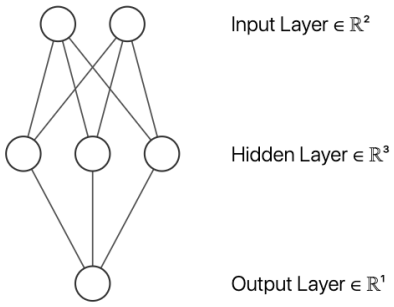

Fig. 2: A simple multi-layer perceptron model that maps an input $\mathbf{x} \in \mathbb{R}^2$ to an output $y \in \mathbb{R}$. It achieves such mapping through a hidden layer of three neurons, each with its own activation function. This simple model is capable of approximating non-trivial non-linear function mappings.

	A	B
1	id	breed
2	000bec180eb18c7604dcecc8fe0dba07	boston_bull
3	001513dfcb2ffafc82cccf4d8bbaba97	dingo
4	001cdf01b096e06d78e9e5112d419397	pekinese
5	00214f311d5d2247d5dfe4fe24b2303d	bluetick
6	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever
7	002211c81b498ef88e1b40b9abf84e1d	bedlington_terrier
8	00290d3e1fdd27226ba27a8ce248ce85	bedlington_terrier

Fig. 3: Sample of image IDs and corresponding breed labels. More information on the structure of the data set is publicly available.

4.3 Xception and MLP

When implementing this new method to predict dog breeds, the first step I needed to do was to load Xception and split the data into training and testing sets. With the data split, I then needed to run the training and testing sets through Xception. In my model, I needed to learn which set of neurons along with what value of eta, η, would provide me with the best model to predict dog breeds. The parameter η represents the initial learning rate of the model. This value "controls the step-size in updating the weights" [7].

These values are not known without running the model to see what the cross validation score is and comparing it amongst others. Therefore, I created a for loop within a for loop. I utilized eight different sets of neurons and four different values of η. At the end of each for loop run through, the cross validation score would be compared against the current, best cross validation score. If it was better, these new values would be stored, but if it was not better it would move on to the next set of neurons or the next value of η. Once all of the sets of neurons and values of η were scanned through, I would have a final, best model to use for predictions.

Before running the model with all the sets of neurons and value of η, I wanted to confirm the model worked in the manner in which I hoped. Therefore, I decided to run it with only two sets of neurons and two values of η initially. This was useful because it allowed me to correct all of the errors it produced before running the complete model. Once the errors were corrected, I moved on to running the model with all the sets of neurons and values of η to determine which out of all of them created the best model to predict dog breeds.

During the process of running the full model, I learned that it took a long time to complete running. In order to simplify the process, I broke it down to run one set of neurons at a time with all of the values of η. While running the model with one set of neurons at a time, I recorded all of my observations to compare all of the results in the end. When comparing the results in the end, I examined the cross validation scores and not the balanced accuracy

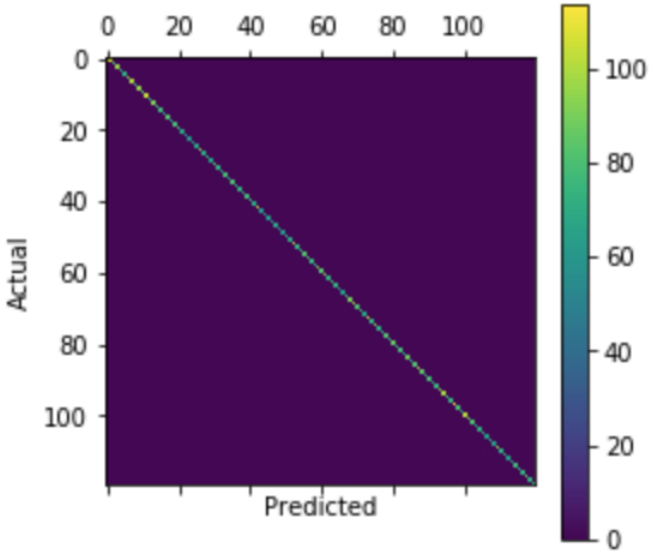

Fig. 4: Confusion matrix on the training set. The main diagonal is evidence of very good performance during training.

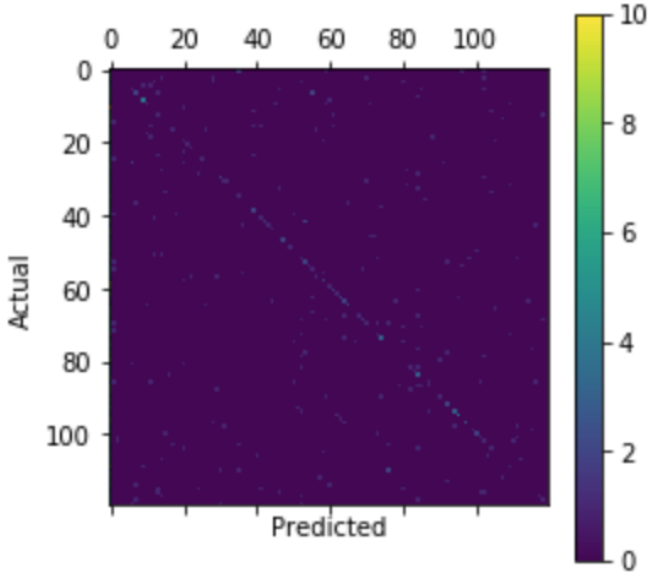

Fig. 5: Confusion matrix on the test set. This indicates a clear diagonal pattern; however, notice that there are other classes that are also incorrectly predicted, leading to a balanced accuracy of 54.8%. While this may seen like it is above the random noise level, we need to recall that this is not the traditional accuracy, but the balanced accuracy rate.

score. I looked for the highest cross validation score because utilizing cross validation takes into consideration and helps control overfitting. I did not look for the highest balanced accuracy score because this value was not being run through cross validation, therefore, this value should not be trusted as a truly accurate representation of the prediction accuracy. Looking for the highest cross validation score out of all of the results I obtained, I found that the set of neurons (400, 200) and η being 0.001 would produce the best model for the classification of dog breeds. Once these values were saved, I no longer needed to run the entire model. Instead, I was able to just run the MLP Classification with these hyperparameters to obtain the performance metrics which will be analyzed in the following section.

5. Discussion and Analysis

In order to analyze the performance of the model I created, I needed to run performance metrics. The performance metrics I obtained included confusion matrices, LogLoss, and Balanced Accuracy. The confusion matrices show how well the model was trained and how well the model predicted the test data set. It compared the true dog breed to the predicted dog breed regarding specific images. For my optimal model I obtained the following confusion matrices for the training and testing data sets.

As you can see, in Fig. 4, the model is training well as there is a clear diagonal line which represents the true breed matching the predicted breed. Then analyzing the testing confusion matrix, in Fig. 5, I could see that the diagonal line, which represents the accurate predictions, is not as prevalent. Instead, I could see that there are scattered dots throughout the rest of the confusion matrix, which represent incorrect predictions.

The testing LogLoss and Balanced Accuracy were also reported. These measures are not a good representation of the true rates of LogLoss and Balanced Accuracy because they are not being passed through cross validation, but they provide a good idea of how the system is performing. The optimal model produced a LogLoss of 9.5954 and a Balanced Accuracy of 0.5480, or 54.80%. In future uses of my system, I plan on editing the LogLoss and Balanced Accuracy so that they are running through cross validation in order to obtain a better representation of these values. Another future improvement of this model is to increase the number of splits utilized. These results were obtained using only three splits. An increase to ten splits will allow the model to train more which will potentially increase the accuracy rate of predictions.

With the performance metrics obtained in mind, attempts to predict dog breeds were made. One way that predictions can be made are by utilizing the test data, which the data was split into at the beginning of training. Using the test data, I passed in an image that is from the data set and saw what the system predicted it as. To determine this, I analyzed the predictive probability and looked for the maximum value in the array. The index of the maximum value represents the index of the dog breed predicted. Since this input image is an image from the original data set, I was able to obtain the true value of the dog breed in the image. Comparing

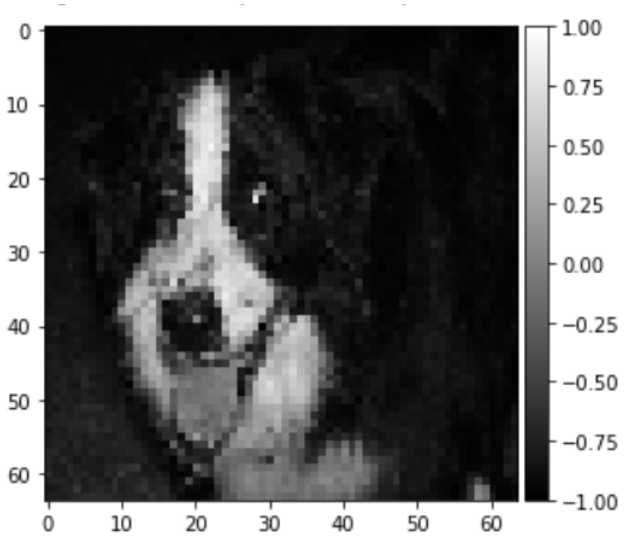

Fig. 6: Sample input image of a dog. Note that the images are scaled to the range $[-1, 1]$ to ease the tensor computations and minimize the problems associated with gradient descent calculations.

```
[[3.14358755e-16 2.65322107e-22 7.73992984e-20 6.66627257e-24
  3.22729149e-15 5.08625661e-10 3.13380935e-23 3.65731049e-22
  1.64392625e-14 1.00899261e-16 3.76889158e-25 1.03811010e-02
  4.30512700e-22 2.20296388e-21 2.79050585e-28 2.08868149e-21
  9.95585958e-01 1.47748973e-22 1.61882701e-18 6.05653480e-09
  3.18968908e-31 4.39479747e-18 5.53907842e-29 6.35374693e-15
  2.24829173e-23 1.37677784e-25 6.25947033e-18 8.10185542e-20
  1.68843810e-37 7.61821215e-18 4.87353519e-18 1.81913166e-14
  1.28806238e-13 1.34708955e-14 7.92577240e-24 2.76187679e-12
  6.68035224e-26 2.42961211e-18 3.92340440e-19 7.11721599e-25
  1.60198159e-16 4.59433835e-07 2.56661459e-11 2.85874060e-20
  5.41318276e-25 6.48693727e-17 5.12969451e-17 4.37609679e-27
  7.84579659e-24 2.09136901e-21 8.33357488e-19 6.46702656e-23
  1.35503526e-22 3.81100351e-12 3.67057571e-19 4.44696079e-35
  6.93158937e-32 3.09976705e-21 1.19516163e-20 2.35176318e-15
  1.32733572e-19 1.04379395e-13 2.41403153e-06 3.57842032e-16
  6.37876253e-24 2.33943470e-23 8.90206912e-20 1.34706611e-23
  2.25708399e-23 3.85890528e-22 9.71657028e-15 1.70901056e-07
  1.50597359e-15 3.89274308e-17 1.12939053e-27 1.35286416e-19
  1.83479694e-16 2.16146104e-19 1.88815121e-15 6.82979164e-20
  2.08394820e-14 1.58576430e-27 1.62305832e-11 4.45084164e-16
  4.50906314e-09 3.57366664e-14 8.98627278e-13 3.53371617e-17
  3.41331829e-17 8.42882706e-27 2.07775107e-09 4.70181957e-15
  6.95429109e-10 1.91673129e-21 2.33166991e-13 2.69818252e-17
  3.94208859e-23 9.07050092e-15 5.05154685e-23 4.26953710e-10
  2.31669953e-11 9.72932774e-14 6.71191132e-11 8.61322664e-21
  7.01712893e-19 3.27100483e-20 1.06532512e-12 2.95452919e-14
  2.43074267e-07 6.00642281e-11 1.99273981e-17 1.55174462e-20
  2.33998703e-24 7.21385527e-19 4.44798211e-25 4.02349990e-14
  8.91277102e-14 8.05844589e-19 2.10029087e-23 1.35802024e-14]]
```

Fig. 7: Sample array of probabilities for each of the 120 classes. Notice that in the first column and fifth row, the maximum value is shown with probability 99.58%, and thus, the predicted class would be number 17.

the value that the system outputted against the true value, I could determine if the system predicted that specific image correctly or not. With the index of the predicted breed, I could also obtain the name of the breed that the system predicted. This allowed me to compare the input image of the dog with what the system predicted it to be. For example, suppose we are trying to predict the breed of the dog in Fig. 6. The system will return the array of probabilities represented in Fig. 7. The model also outputs the index at the maximum value of the probabilities. In this case, the value it outputted was 16. I also programmed it to output the true value of the dog breed represented in the image. In this case, this output was also a 16, indicating that the breed was predicted correctly, with a probability of 0.99559.

Another way to predict images is by inputting your own images into the system. For example, I was able to input an image of my dog and see what the system predicted it to be. The system would again output the probabilities for the image being each breed in the data set. As in the last case, taking the maximum value of this gave me the index of which breed the system is predicting for this image. Using the index, I could determine which breed this is. Since it was a self-inputted image, and I know the breed of my dog, I was able to determine if it was predicting correctly or not. Unfortunately, with the image I inputted, it did not predict my dog correctly. The system predicted an index of 88 for one photo which represents an Entlebucher and an index of 82 for a second photo which represents a Malamute. My dog is a Golden Doodle, which, unfortunately, is not very similar to either of these breeds.

6. Conclusion

The motivation of this model was to learn how to use a machine learning classification tool in order to classify images, namely, dog breeds. In future research, I would like to improve the prediction accuracy rate. One way to approach this goal would be to run it with more sets of neurons and values of η. There is a potential this could increase the prediction rate, but this is unknown until further research is performed. Additionally, to further improve the model, the MLP Classification, LogLoss score, and Balanced Accuracy score should be run through cross validation. This would provide a better measure of the actual prediction rate because it takes into consideration overfitting and deals with that inflating or deflating the prediction rate. Lastly, to improve my current model, I would like to train my model with ten k-folds of cross validation instead of three. My current results represent the results obtained using three folds. In the future, I would be interested in seeing how changing the number of folds to 10 would change the prediction rate.

In addition to improving my current model, I am also interested in testing other methods' abilities to predict the dog breeds. I have already tested a Convolutional Neural Network and Xception throughout this process. I would be curious as to how other methods work and how the results would or would not improve. It would be interesting to show how the different methods compare against each other as well. Additionally, as I was performing my preliminary research about this data set and what participants in this

competition did, it seemed a lot of them limited the number of dog breeds they included in their models. It would be interesting to test different combinations of a certain number of breeds to see how the accuracy could increase or decrease due to this. This can be greatly impacted depending on if one dog breed is more or less prevalent in the entire data set as compared to the other dog breeds.

References

[1] "A Beginner's Guide to Multilayer Perceptrons (MLP)," Skymind. [Online]. Available: https://skymind.ai/wiki/multilayer-perceptron. [Accessed: 14-May-2019].

[2] A. Escontrela, "Convolutional Neural Networks from the ground up," *Towards Data Science*, 16-Jun-2018. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1. [Accessed: 24-Mar-2019].

[3] "Dog Breed Identification," *Kaggle*. [Online]. Available: https://www.kaggle.com/c/dog-breed-identification. [Accessed: 13-Feb-2019].

[4] H. Bendemra, "Build Your First Deep Learning Classifier using TensorFlow: Dog Breed Example," *Towards Data Science*, 26-Apr-2018. [Online]. Available: https://towardsdatascience.com/build-your-first-deep-learning-classifier-using-tensorflow-dog-breed-example-964ed0689430. [Accessed: 24-Mar-2019].

[5] J. Brownlee, "When to Use MLP, CNN, and RNN Neural Networks," Machine Learning Mastery, 23-Jul-2018. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/. [Accessed: 08-May-2019].

[6] "Machine Learning: What it is and why it matters," SAS. [Online]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html. [Accessed: 14-May-2019].

[7] "sklearn.neural_network.MLPClassifier," scikit learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier. [Accessed: 08-May-2019].

[8] S.-H. Tsangm "Review: Xception - With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)," Towards Data Science, 25-Sep-2018. [Online]. Available: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568. [Accessed: 14-May-2019].

[9] "Xception," MathWorks. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/xception.html. [Accessed: 14-May-2019].

[10] Chollet, Francois. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.