The Dog Breed Identifier

Machine Learning For Your Dog

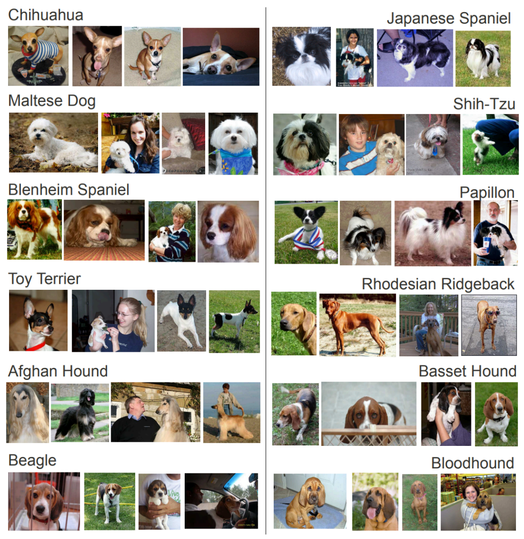
By: Trey Gower & Steven Oh

SECTION 1: SETUP

We live in a world where dogs are the best friend known to man. You love em, I love em, we all recognise the value of a good dog. Part of taking pride in ownership of one's dog is the identity of the dog. That is, what breed your dog is. Sure, DNA testing is a great way to give sound results for your dogs genetic makeup, not everyone has time for that process. For this reason, Steven Oh and I, Trey Gower, introduce to the world, The Dog Breed Identifier.

SECTION 2: DATA AND TECH USED

The dataset that will be used is from Stanford and it contains 20,580 images of 120 different dog breeds. The Dataset can be found here. Since this dataset is so large we decided to split it up into a 70% training, 15% testing, and 15% validation. From this, we are training three models (1) VGGNet, (2) AlexNet, and (3) ResNet, in which we will deploy the best result to a website.



For loading this data, Kaggle has an api making it easy to take any dataset and prepare it for use with python. After pip installing kaggle this command can be used to download the data: kaggle datasets download -d jessicali9530/stanford-dogs-dataset (for convenience, we have included all images from the dataset in our repo). From here, we needed to do minimal preprocessing, which means splitting the data set and rescaling the images for training, which only requires python and basic functions from the os library. This was done with the first few lines of code here on jupyter notebook.

SECTION 3a: TRAINING THE MODELS

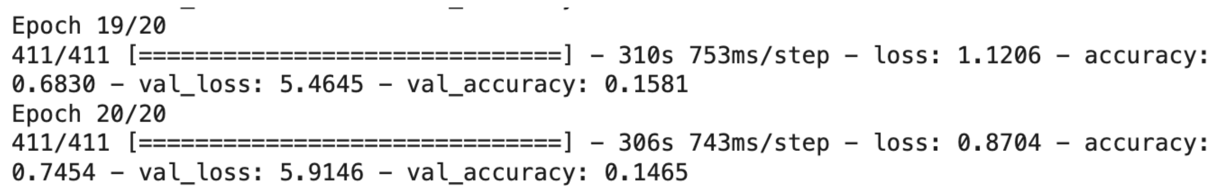
1. **VGGNet:** This architecture consists of a series of convolutional layers followed by max-pooling layers. The main feature is the use of small (3x3) convolutional filters with a stride of 1, which helps in capturing intricate features in the images. The fully connected layers at the end progressively reduce the number of channels until reaching the final output layer of 120 channels, which predicts the class probabilities. It's important to note that the original VGGNet has 16 or 19 weight layers (ours has 16).
2. **ResNet:** ResNet is similar to VGGNet but addresses the vanishing gradient problem by introducing skip connections or shortcut connections that allow gradients to flow more easily during training. These connections skip one or more layers, mitigating the degradation problem and allowing for the training of very deep networks (hundreds of layers). This architecture has residual blocks containing convolutional layers and skip connections.
3. **AlexNet:** AlexNet is a seminal deep learning model that popularized convolutional neural networks (CNNs). It consists of one initial convolutional layer followed by max-pooling, another convolutional layer and max-pooling, and three final concolutional layers and one max pooling layer. It then has three fully connected layers. This model excels in preventing overfitting by employing dropout layers and data augmentation techniques which also improves generalization of the model.

SECTION 3b: BUILDING THE WEBSITE

To build and deploy the website, React, Node.js, and GitHub Pages were used. The npm packages that were downloaded to be used in the scripts were react packages and tensorflowjs packages. As the goal of this website is to create a fully front-ended application that would perform proper dog breed classification, no back-end server was used. Lastly, the model that was trained was converted using the tensorflowjs library in the jupyter notebook [tfjs.converters.save_keras_model(model, ‘./modeldirectoryname’)].

SECTION 4a: RESULTS OF THE MODELS

Unfortunately, all of our trained models tended to want to memorize the data instead of learn it. This can be seen in one very obvious way. All of the accuracies of our models were achieving relatively high percentages, but in every case the validation accuracies were terribly low. Look at this output below with 75% accuracy and 15% validation as an example:



```
Epoch 19/20
411/411 [==============================] - 310s 753ms/step - loss: 1.1206 - accuracy:
0.6830 - val_loss: 5.4645 - val_accuracy: 0.1581
Epoch 20/20
411/411 [==============================] - 306s 743ms/step - loss: 0.8704 - accuracy:
0.7454 - val_loss: 5.9146 - val_accuracy: 0.1465
```

Above is the output of the AlexNet model, which already employs methods to prevent overfitting and still we are getting terrible results. Through various techniques there were attempts to fix this persisting overfitting issue. These included batch normilization, experimenting with learning rates, reducing number of filters, and even early stopping to monitor the validation loss. With all of these methods employed still the error persisted.

The running theory for this issue is that the dataset is so large that the simpler versions of the models we used were not complex enough to capture the data effectively. To check this theory we used larger pre-trained ResNet50 model which can also be found in our jupyter notebook. From this and from comparison of the model summaries it was clear that this could have very well been the defining issue.

SECTION 4b: THE FINAL PRODUCT

The project's objective was to deploy the trained model to a website (using GitHub pages) and allow users to input images of dogs to classify. While it was unsuccessful in using the model trained through this project, the concept was proven successful with the use of a different model that already exists. The model that was used to test the concept of this project is in the tensorflowjs library and is named MobileNet.

While unsuccessful with the model that was trained in this project, there are lines of code that have been commented out that have lots of potential to work. The issue that occurred was the improper conversion of the image to a tensor and then reshaping it. It is believed that it was successful in converting the image to a tensor but failed when trying to reshape the image.

Ultimately, the website that uses the MobileNet model was quite successful as it was able to predict the dog breed of any image, providing the top 3 possible breeds. Future implementations include fixing and figuring out the reshape error in order to use any personally trained Keras models.

SECTION 5: REFERENCES

Dataset:
https://www.kaggle.com/datasets/jessicali9530/stanford-dogs-dataset/data

CNN Models Article:
https://towardsdatascience.com/various-types-of-convolutional-neural-network-8b00c9a08a1b