Project overview

For my capstone project, I chose to train a convolutional neural network (CNN) to predict dog breeds from user images.

Problem statement

It can be difficult for humans to guess the breed of a dog, but CNNs have proven quite successful for image classification tasks, with state-of-the-art (SOTA) models capable of predicting a class from among 1000 potential classes to a high degree of accuracy! Since these models have such excellent predictive power, I expect they should easily adapt to predicting 133 dog breeds, far better than a human could.

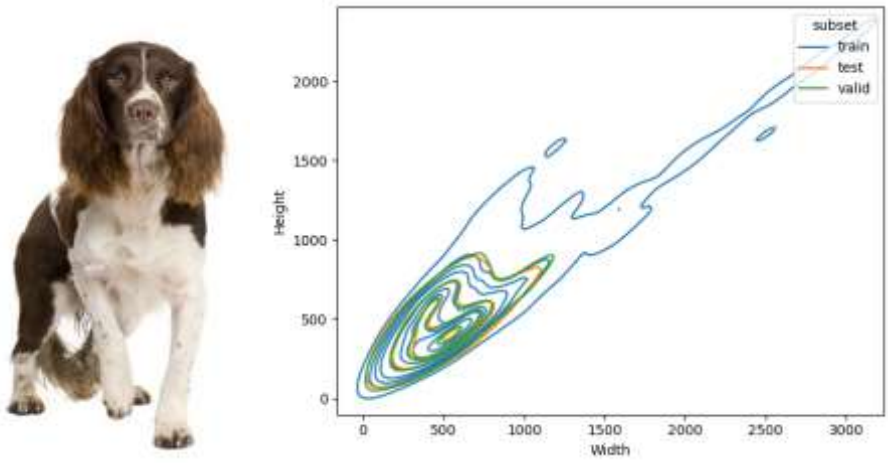
Metrics

A CNN will be trained on a training dataset and a validation dataset will be assessed in parallel to ensure the model is improving and can generalize well to unseen data rather than just learning the training set. To determine the success of this approach, prediction accuracy on a test dataset will decide which model performed best on this task.

Data exploration and visualization

This dataset contains 133 dog breeds from Affenpinscher to Yorkshire Terrier and is already split into training, validation, and testing subsets containing 6680, 835, and 836 RGB images, respectively. It is similar to the Stanford Dogs dataset which contains over 20,000 images of 120 dog breeds.

Each dog breed is present approximately 50 times (min 30, max 80) in the training subset, 6.3 times (min 4, max 9) in the validation subset, and 6.3 times (min 3, max 10) in the testing subset. The width and height of the images from each subset of the data are comparable, thought the training data has a larger range of image dimensions to work with (figure below).



Example dog image (English Springer Spaniel, Left); KDEplot of X and Y dimensions per subset (Right)

Methodology

The images vary by width and height so reshaping will be necessary for the CNN to accept these data as input. Normalization of input data to a certain range, either (0, 1) or (-1, 1), helps when training the CNN so each image channel was divided by the maximum pixel value of 255.0 to get floats in the range (0, 1).

A naïve implementation of a CNN was tested for simplicity. The basic components of these networks are convolutional blocks which contain filters to detect features in the input data. In early layers of the network, these filters function as edge detectors, while filters learned in later stages of the network respond to more complex features such as ear shape. After passing the data through a number of convolutional layers, the feature maps are compressed into a 1D tensor by global average pooling. These data are then passed through a fully-connected layer and a final layer with a softmax activation to predict the class identity.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 222, 222, 8)	224
batch_normalization_1 (Batch	(None, 222, 222, 8)	32
activation_50 (Activation)	(None, 222, 222, 8)	0
conv2d_2 (Conv2D)	(None, 220, 220, 16)	1168
batch_normalization_2 (Batch	(None, 220, 220, 16)	64
activation_51 (Activation)	(None, 220, 220, 16)	0
conv2d_3 (Conv2D)	(None, 218, 218, 32)	4640
batch_normalization_3 (Batch	(None, 218, 218, 32)	128
activation_52 (Activation)	(None, 218, 218, 32)	0
global_average_pooling2d_1 (	(None, 32)	0
dense_1 (Dense)	(None, 256)	8448
dense_2 (Dense)	(None, 133)	34181

Total params: 48,885
Trainable params: 48,773
Non-trainable params: 112

Naïve architecture

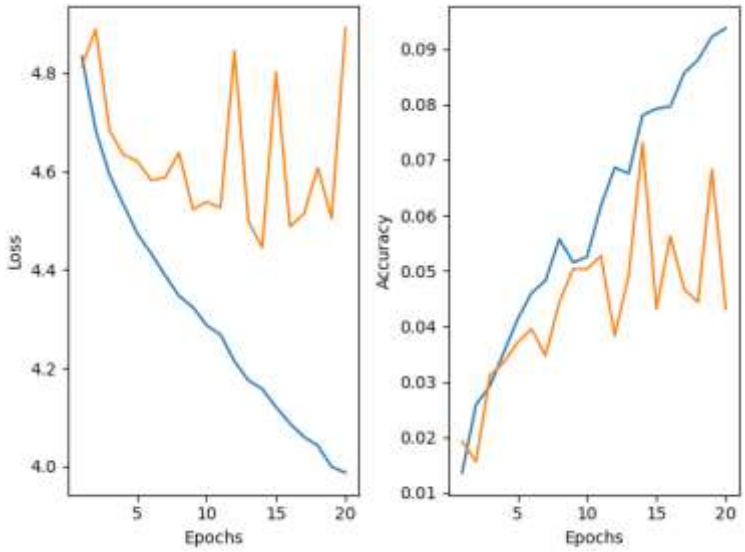
Many deep learning libraries provide SOTA architectures with pretrained weights to enable so-called “transfer learning” and allow the CNN to be adapted to a different classification task. It is believed that a pretrained SOTA model may only need a few iterations of training and so would be quicker to train than a naïve implementation.

Refinement

Batch normalization was also applied before activations to transform batches to have a mean of 0 and a standard deviation of 1. This practice helps to prevent exploding gradients, resulting in smoother loss curves and faster convergence.

Model evaluation and validation

The models were compiled with the Adam optimizer and categorical cross-entropy was used as a loss function for this multi-class classification problem. Models learned on the training data and were assessed on the validation data after each epoch to prevent overfitting. The naïve model had a test accuracy of 6.1% on the test data after 14 epochs but would clearly need some structural tweaks to achieve better results. The curves in the figure below show a slight improvement in validation loss and accuracy after a few epochs but the small model rapidly begins to overfit on the training data.

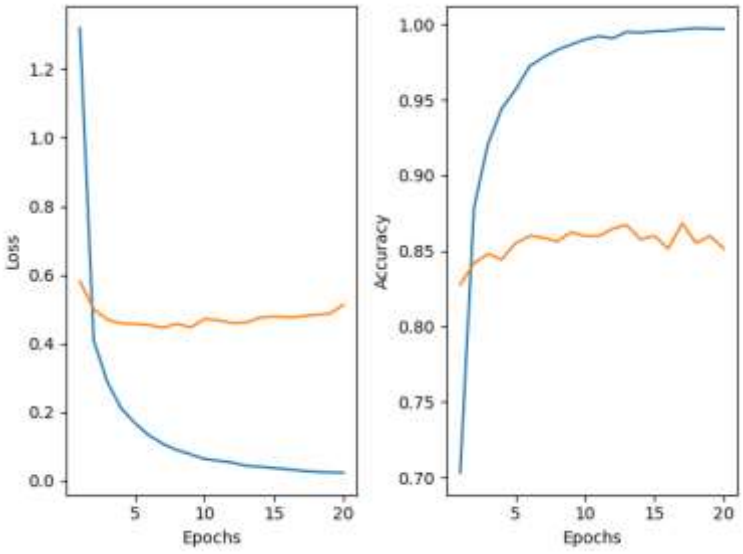


Training curves for the naïve model

Justification

There are endless possible improvements one could make to a naïve model, but that may not be necessary since transfer learning using SOTA image classification models may already have the predictive power needed and may only require slight adjustments to the weights to achieve sufficient accuracy on a new dataset. Popular models for transfer learning included VGG16, InceptionV3, ResNet50, and Xception architectures, which have millions of parameters and are known for exceptional performance in classification

challenges. Of the choices provided, the Xception architecture proved most attractive as it uses a concept called “depthwise separable convolutions” which provide a more computationally efficient way of applying a convolutional block in the network, thus reducing prediction times. By dropping the fully-connected layers at the end of the network and replacing them with layers which would allow prediction of the 133 dog breeds, we could achieve 86% validation accuracy and 85% test accuracy. The validation curves below show minute improvements in during training but it seems there is also a tendency to overfit to the training data.



Training curves for the adapted Xception model

Conclusion (Reflection, Improvement)

CNNs possess incredible predictive power to discriminate between classes especially when SOTA architectures are adapted to new classification tasks. These models could be enhanced by using (i) data augmentation techniques to diversify the images the model might have to classify, (ii) acquiring additional data, (iii) a callback to reduce the learning rate when validation loss stagnates, or (iv) a combination of all three.