Machine Learning Engineering Nanodegree

Proposal

For the capstone project for the Udacity Machine Learning Nanodegree, I have selected the classification of dog images using Convolutional Neural Networks (CNN).

Domain Background

Image classification is a common Machine Learning task, for this project we will be using different ML techniques and will compare the results obtained from them. I will use different techniques to build an image classifier that will determine the breed of the dog.

Dog breed classification is a well-tested machine learning. For example, the following paper describes building CNN to classify the breed, in order to help lost dogs be returned to their owners.

https://arxiv.org/pdf/2007.11986.pdf

Problem Statement

The purpose of this project is to evaluate different machine learning techniques, and compare them . In order to accomplish this I will use pre trained models, apply Transfer learning techniques and finally create a Convolution Neural Network from scratch.

I will use VGG-16 model pre trained against the ImageNet dataset to build a dog classifier. VGG-16 has been trained with the image-net dataset which consists of over 14 millions images and 1,000 different labeled feature classes which includes 120 dog breeds.

Also I will use pretrained weights of the VGG-16 model and apply Transfer Learning techniques to refine the results,   also I will create a CNN from scratch and train it with the Dog dataset.

Dataset

For this project I will be using the Standford Dog dataset, This dataset consists of 120 different dog breeds with around 150 images per breed for a total of 20,580 images.

This dataset is balanced for all the classes :

- 12,000 images for training
- 8,580 images for testing

Using a total of 20,580 images.

This is a popular dataset for dog breed classification models used in research. And it is also available on Kaggle Playground Prediction Competition https://www.kaggle.com/c/dog-breed-identification

The original data source is found on http://vision.stanford.edu/aditya86/ImageNetDogs/

While doing research for the project I realized that the “Stanford Dogs” dataset is available from the tensorflow-datasets package with training/testing split, and I decided to use this instead of manually downloading the data and transforming the data into a format suitable for use with Tensorflow.

Few Sample images from the dataset

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.*

Evaluation Metrics

This is a simple classification problem, I decided to use accuracy as the evaluation metric, since we have 120 different classes, and other metrics such as Recall and F1 score are harder to interpret with this large number of classes. While Recall and F1 score can provide us with metric for each class they provide less information about the overall model performance, instead they provide information about he prediction for each class using a one-vs-rest approach.

Also the Stanford dog dataset is balanced, the training dataset includes 100 images for each of the classes.

Project Design

For this project I decided to use Tensorflow and the Keras API, mainly because I have previous experience using them and they are suitable for this project.

The Keras API includes APIs to quickly take advantage of pretrained image classification models, as well as implementing Transfer Learning.

In order to accomplish the goals outlined for this project we will build a data pipeline that can perform the preprocessing required for the images. Such has image resizing, and scaling.

The VGG16 model includes a pre processing module that converts the image from RGB to BGR and zero-centers the color channel which is the same as the values used to train against the ImageNet dataset. Also I used a basic data augmentation on the images random horizontal flip, and random rotation of the images when training.

I was able to train the model on my PC with a GTX1650S GPU, with a batch size of 4 (unable to use larger batch sizes due to the low memory on the GPU), even with GPU usage running the entire notebook would take over 18 hours to run (with most of the time spent training the model from scratch).

Network Models

For this project I used the following Network Models :

VGG16

VGG16 is CNN model that was introduced in 2014 and was trained against ImageNet dataset, training took several weeks using GPU computers.

The main disadvantage of this model is that is slow to train due to the large number of parameters and the model is huge on the network, and also that the mode takes a lot of disk space (528 MB) but inference with this model is reasonably fast and disk space was not a major deciding factor for me.

This model was selected because I wanted to use a model that had been trained against the ImageNet dataset, since the ImageNet dataset includes 1,000 classes and includes 120 dogs breeds.

Transfer Learning with VGG16

With Transfer learning we can take advantage of the work others have done and use the pretrained weights (knowledge) in the model and just change the last layer. Transfer Learning reduces the training time required (since we only have to train a few layers of the model) and usually improves the performance of the model. Also Transfer Learning can usually get better performance while using less data for training. In this case we are training with 12,000 images instead of the 14 Million images that were used to train VGG16.

Train Model from scratch

The custom model trained from scratch uses smaller Xception model, The Xception model is very good image classification model by google. (Xception – eXtreme Inception)

The main disadvantage of training a model fully from scratch are:

- Large Number of data required
  Training a CNN requires a large number of images
- Training time Model

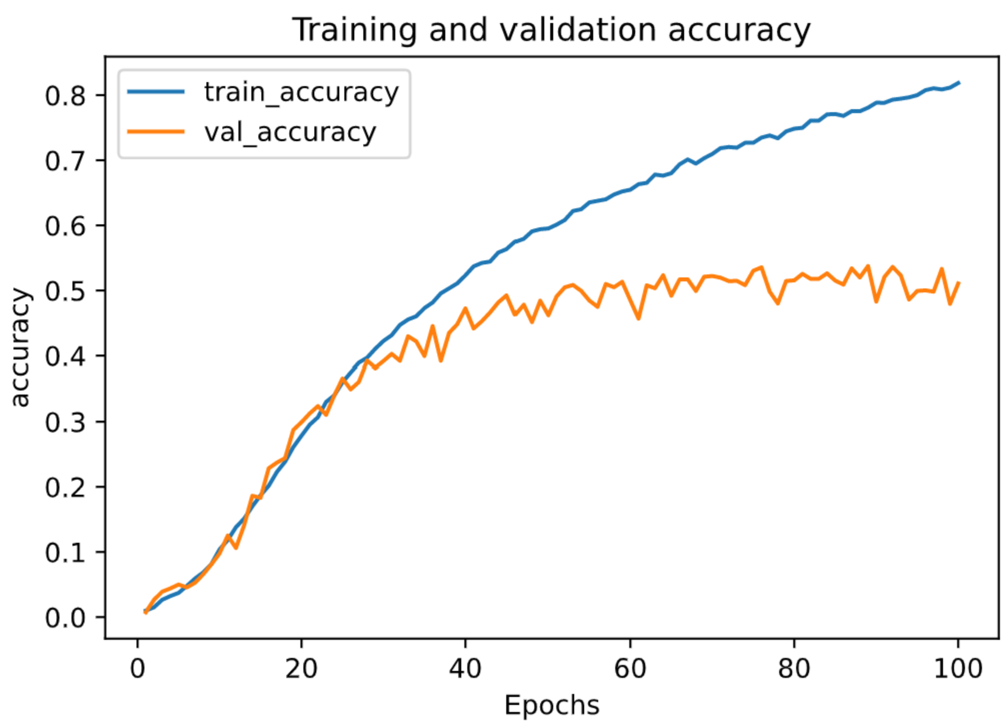
For Additional information on the models use see the resources section.

Conclusion¶

Using the pretrained model was really easy to use but unfortunately only obtained an accuracy of 39%, Transfer learning was able to get an accuracy of 64% and training was quick. Training the custom model took the longest (about 12 hours for 100 epochs on my setup) and the model only reached an accuracy of 51% but it looks like there is additional room to improvement.

When doing Transfer learning of the model I only used feature extraction (only trained the last layer of the network) the model also benefit from fine-tuning the model, while in traditional Feature Extraction only the last layer of the model is trained, with fine-tuning the last few layers of the model are trained.

Custom model trained from scratch training accuracy plot shows that training accuracy is still improving, better performance may be obtained by training for additional epochs.



Additional analysis of what classes the models were able to identify regular and which classes the models struggled with would be interesting.

Overall it was a very interesting exercise, and I had the opportunity to explore a lot of different image classification options.

Resources

- VGG16 model
- ImageNet
- Keras ImageNet trained models
- Using Public Datasets with TensorFlow Datasets
- Keras Pretrained image classification Models
- Checkpointing Deep Learning Models
- Transfer Learning and fine-tuning with Keras
- Image Classification from Scratch
- Xception Network