Edwin He

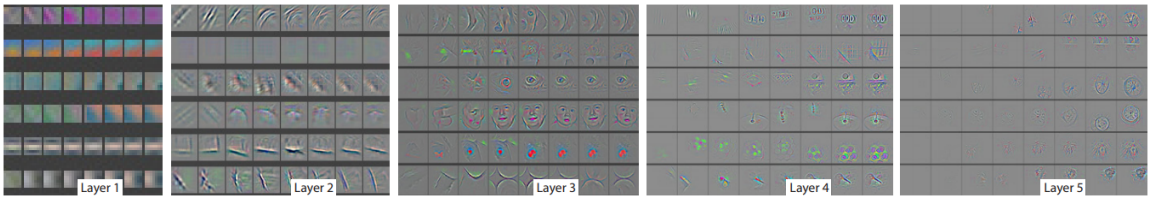
31 Oct 2017 • on machine learning

ML Nano Degree - Capstone Project Proposal

1. Introduction

Deep learning with Deep Neural Network has gained huge traction in the past few years owning to the blooming of data and GPU accelerated computation. The former makes it possible to collect massive amount of training data to tune millions or even billions of parameters; the later makes running deep learning computation approachable for everyone have access to a consumer PC with GPU installed.

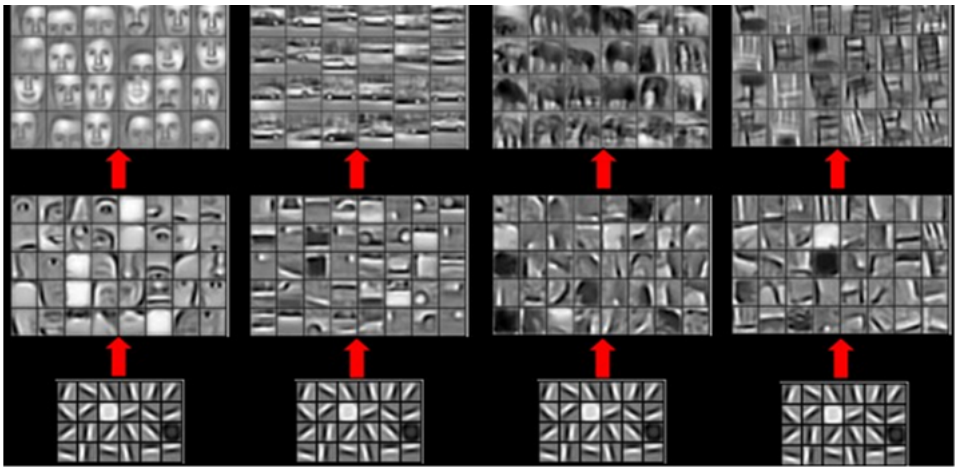
One of the most exciting fields of deep learning application is computer vision which gives machine the ability to see. One of the most influential innovations in the field of computer vision is Convolution Neurual Network(CNN), first published by LeCun et al [1], became prominence as Alex Krizhevsky used it to win 2012 ImageNet competition. The key idea of CNN is weight sharing unlike Multilayer Perceptrons (MLP) where nodes are fully connected, each input has its own weight. This allows CNN to extract patterns from images regardless of their position. First layer extracts simple patters such as edges. Following layers extract more abstract patterns based on the previous layers [2].



2. Task

Though empowered by GPU, training an image classifier with CNN still requires a long computation time. CNN itself needs to go deeper to achieve more complex tasks which requires massive amount of training data. Neither is often feasible for individuals.

As aforementioned, the first few layers extract relatively simple patterns which are roughly the same for different image classification tasks. For instance, basic patterns like edges and curves are the basic elements of every image [3] (Lee et al 2009).



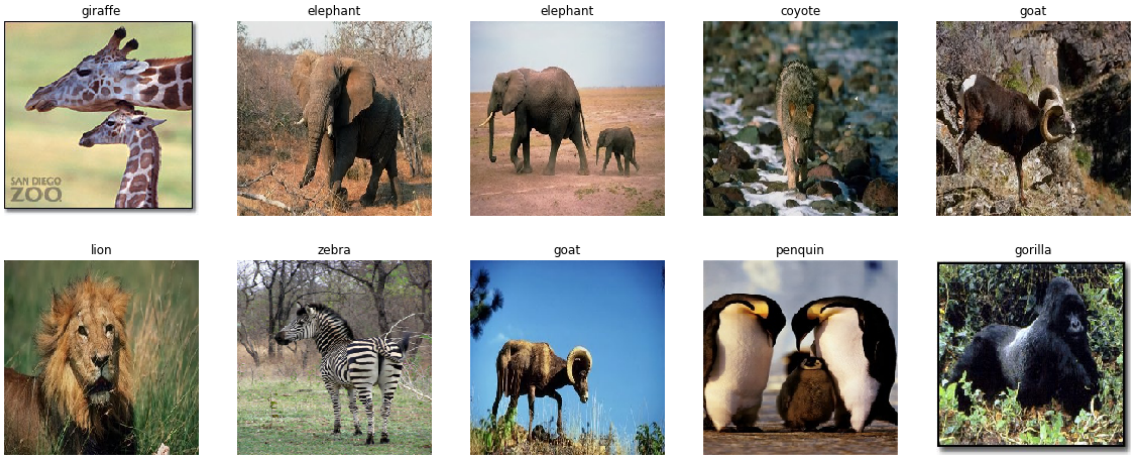
Naturall we would want to find a way to reuse the less abstract and less task orianted layers pretrained by massive data sets. This is when transfer learning come to the rescue. Transfer learning allows us to reuse the less abstract layers of a pretrained model for a similar tasks and achieve a different goal by training the more abstract layers with way less data and effort due to the massive reduction of trainable parameters.

In this study, I look to use transfer learning to train an animal image classifier with 19 classes and very few training data.

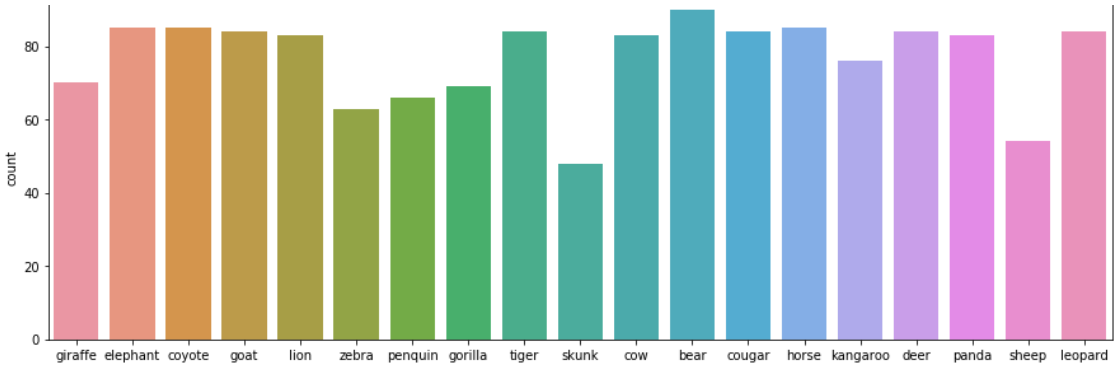
3. Project Plan

3.1 Dataset

The KTH-ANIMALS dataset[4] contains only 1740 images in total and are of different shapes from ~200 to ~400 pixels width and height. I first load them as the same 224 by 224 pixels images. Plot them to validate if class labels have been aligned correctly.



As can been seen in the training data, the images are taken from different angles in the wild. This gives us even more troubles. For example, a more complex model is required to learn a goat looking from the front as well as from the side when the goat horn seems to have different shapes from different angle.



I then cut the data into training, validation and test set containing 1460, 180 and 100 images respectively. Leaving us merely 78 images on average per animal class which is far from sufficient to train a CNN model from stratch to classify such a diverse dataset.

3.2 Benchmark Model

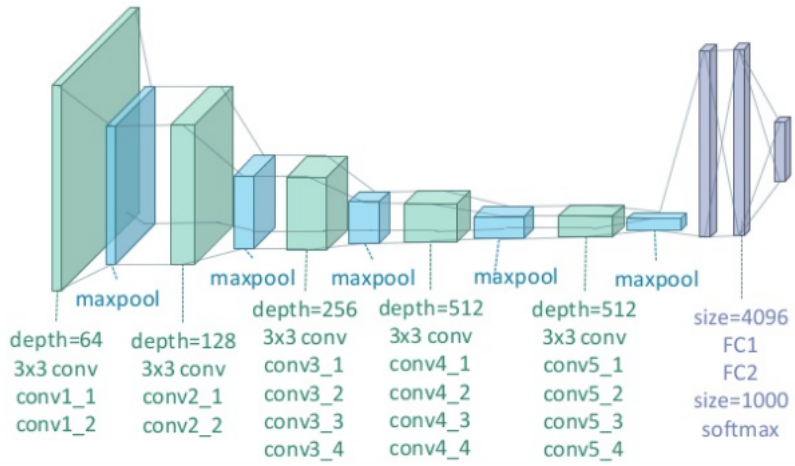
The benchmark model - a CNN model with at least 5 ConvNet layers similar to VGG19 except that every block has only one CNN layer(subjected to minor changes).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           (None, 224, 224, 3)       0
_________________________________________________________________
Conv_1 (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
MaxPool_2 (MaxPooling2D)     (None, 112, 112, 64)      0
_________________________________________________________________
Conv_3 (Conv2D)              (None, 112, 112, 128)     73856
_________________________________________________________________
MaxPool_4 (MaxPooling2D)     (None, 56, 56, 128)       0
_________________________________________________________________
Conv_5 (Conv2D)              (None, 56, 56, 256)       295168
_________________________________________________________________
MaxPool_5 (MaxPooling2D)     (None, 28, 28, 256)       0
_________________________________________________________________
Conv_7 (Conv2D)              (None, 28, 28, 512)       1180160
_________________________________________________________________
MaxPool_7 (MaxPooling2D)     (None, 14, 14, 512)       0
_________________________________________________________________
Dropout_8 (Dropout)          (None, 14, 14, 512)       0
_________________________________________________________________
Conv_9 (Conv2D)              (None, 14, 14, 512)       2359808
_________________________________________________________________
MaxPool_10 (MaxPooling2D)    (None, 7, 7, 512)         0
_________________________________________________________________
Dropout_11 (Dropout)         (None, 7, 7, 512)         0
_________________________________________________________________
GAP_12 (GlobalAveragePooling (None, 512)               0
_________________________________________________________________
Dropout_13 (Dropout)         (None, 512)               0
_________________________________________________________________
Output (Dense)               (None, 19)                9747
=================================================================
Total params: 3,920,531
Trainable params: 3,920,531
Non-trainable params: 0
```

A simpler model tend to easily overfit while a deeper model cannot learn both due to the limitation of the insufficient number of training data. I run the benchmark model and visualize the loss and accuracy during the entire training phase. Measure the classification accuracy on test set. (some graphs here)

3.3 Project Design

The transfer learning model - a VGG19 with weights pretrained on ImageNet dataset with top layers(two final Fully Connected layers) replaced with a top model.



Feed the training, validation and test data through VGG19 to obtain the bottleneck features. Then use the bottleneck features as the input to train the weight of the top model. Measure the classification accuracy.

Then freeze the first 4 CNN blocks of VGG19 to fine tune the last(5th) block of VGG19 as well as our top model. Visualize the loss and accuracy during the training phase and compare the classification accuracy with our benchmark model.

The transfer learning model should show a significant increase in accuracy.

At the end I will also explore utilizing the last CNN output weights for nearly effort free object localization [6][7].

Reference

[1] http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf
[2] https://arxiv.org/pdf/1311.2901.pdf
[3] https://www.cs.princeton.edu/~rajeshr/papers/icml09-ConvolutionalDeepBeliefNetworks.pdf
[4] https://www.csc.kth.se/~heydarma/Datasets.html [5] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html [6]
http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf [7]
https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/