Machine Learning Engineer Nanodegree

Capstone Proposal

Luis Gabriel Rezzonico

March 27, 2017

Transfer learning: comparison of VGG16, Resnet and Inception networks applied to image recognition

Domain Background

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a task that has already been learned.

We can see the concept of transfer learning in the real world, when a person tries to learn a new task, that person is going to use previously acquired concept to accelerate the process of learning. Learning to play the guitar is going to be an relatively easier task for a person who has an extensive experience playing music.

Most of the time, obtaining training data is an expensive task, this is one of the reasons to use Transfer Learning of a big previously acquired dataset [1] to a new task. Transfer Learning becomes increasingly important if we take into consideration that the "state-of-the-art" models are trained for months using high performance computers. [2] [3]

Image recognition is the task of determining whether or not a image contains a specific object or not. There has been a boom of startups using this technology to solve real life problems. [4] Currently the best algorithms to solve the task are Convolutional Neural Networks [1].

This capstone aims to provide a comparison of the performance of several pre-trained models for the task of image recognition. Can transfer learning be applied to image an recognition task? The findings of this capstone would be invaluable for people who either do not have the monetary resources or the time to train a model from scratch.

Problem Statement

The objective of this capstone is to compare the accuracy of the transfer of learned knowledge of state-of-the-art Convolutional Neural Networks. The pretrained CNN to be used are VGG16, Reasnet50 and Inception V3. All this models has been trained with the ImageNet dataset [1] . These networks are going to be applied to solve the Kaggle's Dogs vs. Cats competition. A small CNN trained directly on the Cats and Dogs dataset is going to be used as a becnhmark.

Datasets and Inputs

The dataset to be used contains 25,000 images of dogs and cats and can be obtained in the Kaggle's Competition site: https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data, for this capstone only a subset of the data is going to be used.

This capstone also use the "weights" (knowledge) of the pretrained models on the ImageNet dataset, these weights files are listed below:

- VGG: https://goo.gl/go2y7h
- Resnet50: https://goo.gl/MuftHu
- Inverption V3: https://goo.gl/R0Ig5B

Solution Statement

A small convolutional neural network consisting of two convolutional layers and a fully connected network is going to be trained using the Cats and Dogs dataset. These results are going to be used as a benchmark.

The pre-acquired knowledge from the ImageNet dataset is going to be used to configure the VGG, Resnet and Inception Networks. These knowledge is going to be transfer to the Cats and Dogs dataset by fine-tuning the top layers of these networks. The output networks are going to be used on the Cats and Dogs dataset to make predictions

Benchmark Model

The benchmark is going to be the results of the small CNN trained directly with the Cats and Dogs dataset. The loss of this network is going to be compared with the loss of the pre-trained models applied to the Cats and Dogs dataset.

As a secondary benchmark, the Public Leaderboard of the Kaggle's competition can be used.

Evaluation Metrics

The evaluation metric is going to be the one used in the competition itself: log loss. This can be calculated with the following formula:

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(p_i) + (1 - y_i) log(1 - p_i)]$$

where

n is the number of images in the test set

pi is the predicted probability of the image being a dog

yi is 1 if the image is a dog, 0 if cat

log() is the natural (base e) logarithm

Project Design

The capstone is going to be built using the open source neural network library Keras, using tensorflow as a backend. [5] [6]

The data is going to be read using Keras library function. Pixels are going to be normilize by dividing each one by 255. Folder structure is going to be rearrange for easy manipulation. All the images are going to be resized to a common size. The images are going to be splitted in train, validation and test. No further manipulation is going to be made to the data unless strictly necessary.

A small Convolutional neural network is going to be trained on the data and the logarithmic loss is going to be calculated for the test set. A small snippet of code is given as a reference:

```
x = Convolution2D(16, (3, 3),
                  activation='relu',
                  data_format="channels_last",
                  kernel_initializer="he_uniform")(x)

x = MaxPooling2D(pool_size=(2, 2),
                 strides=(2, 2),
                 data_format="channels_last")(x)

x = Convolution2D(32, (3, 3),
                  activation='relu',
                  data_format="channels_last",
                  kernel_initializer="he_uniform")(x)

x = MaxPooling2D(pool_size=(2, 2),
                 strides=(2, 2),
                 data_format="channels_last")(x)
```

On top of this CNN fully connected layers are going to be used, the following code can be used as reference:

```
x = Flatten()(x)
x = Dense(96, activation='relu',kernel_initializer="he_uniform")(x)
```

```
x = Dense(24, activation='relu', kernel_initializer="he_uniform")(x)
predictions = Dense(2, activation='softmax')(x)
```

For each of the pre-trained models we are going to compute the features for each image only once to accelerate future processing. The following code is only a snippet of the code to be used:

```
from keras.applications.vgg16 import VGG16
# create the base pre-trained model using ImageNet weights
base_model = VGG16(weights='imagenet', include_top=False)

concolutional_image_feature = base_model.predict(image)
```

On top of each of the pre-trained models a small fully connected model is going to be used:

```
x = Flatten()(x)
x = Dense(256, activation='relu'))(x)
x = Dense(2, activation='softmax'))(x)
```

This model is going to be trained with the features previously calculated.

References

1. Olga Russakovsky, *Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015. https://arxiv.org/abs/1409.0575

2. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. 2015. https://arxiv.org/abs/1512.00567
3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015. https://arxiv.org/abs/1512.03385
4. https://angel.co/image-recognition
5. https://keras.io/
6. https://github.com/tensorflow/tensorflow