Machine Learning Engineer Nanodegree

Capstone Proposal

Luiz Antônio Nonenmacher Júnior
March 2019

Proposal

Domain Background

The idea for this project came from the combination of some tendencies that I see in the machine learning field with some skills that I would like to develop as a machine learning engineer.

The first tendency is that deep learning requires a lot of data to be successful [1][2], which is generally not available for small and medium size companies. The second tendency is that most of the job of machine learning practitioners are data cleaning and preparation [3]. Because of this, my project will be based on transfer learning using a small dataset collected and prepared by myself, so I can train those data processing skills and I can simulate an environment of a small company.

The domain where I will apply this project is computer vision, which is one the fields where deep learning has a much better performance than traditional models. To exemplify this, we could look at the ImageNet performance in Figure 1:

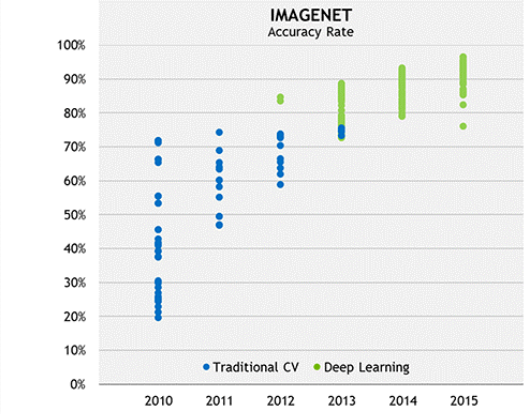

Figure 1: Performance on ImageNet. Extracted from [4]

In 2012 the ImageNet competition was won by a deep learning algorithm that performed better than handcrafted software written by computer visions experts. In 2015, another deep learning algorithm has performed better than humans on classifying those pictures [4].

There are several application of computer vision in different industries, like automotive, retail, financial services and healthcare [5], which make this field an important skill to master for machine learning practitioners.

Problem Statement

Given the background, I want to do a project that involved computer vision, transfer learning and the collecting and processing of data. To this end, I will use my two cats as subjects. My problem will be, given a photo of one of them, classify each cat it is. This problem *per se* is not useful for companies, but the same idea (given a photo classify into two or more categories) can be applied for different kind of problems.

Datasets and Inputs

The dataset will consist of 400 images taken by myself using a smartphone, half of them containing one cat and the other half the second cat (I took and selected the same number of pictures from each cat so the dataset would be balanced). Those pictures were taken in my home and the size and position of the cats will vary. Above ,there are two sample pictures from each of the cats (Gepetto and Kuki):

Figure 2: Sample picture 1 (Gepetto)

Figure 3: Sample picture 2 (Kuki)

The original pictures are in .JPEG and .JPG formats and all the pictures are colored. Most of the pictures has a width of 1536 and a height of 2048, but they are some pictures in a wide format (for example, 3088 x 2320 or 3724 x 2096). In the first step of the project all pictures will be processed to 331x331x3 (for the NASNetLarge model) and 299x299x3 (for the InceptionResNetV2 and Xception models).

I will use 60 images (15% of the dataset), 30 from each cat, as test set and the remaining 340 pictures as training/validation set. When training, I will use 15% of the remaining 340 pictures as validation set.

The original pictures are separated into two folders (Kuki and Gepetto) and are as a zipped file in a Google Drive folder (https://drive.google.com/drive/folders/1Vd1d-YemnPgA88VNZ9HR_CnsBNKFutAo?usp=sharing).

Solution Statement

To solve the problem, I will use transfer learning by using a model pre-trained on ImageNet (dataset of images that contains cats as one of the categories), then using my self-taken pictures to train the last layer of the model.

Benchmark Model

I will implement use three models as the basis for the transfer learning: NASNet[6], InceptionResNetV2[7] and Xception[8]. Those three models can be found in Keras applications [9] and I choose those specific three because they had the best top-1 and top-5 accuracy on ImageNet.

The results of the models on ImageNet are not totally compared to my results, because they are trying to classify images into several classes, where I will just classify of type of image (cats) into two categories. Also, one of the fundamentals of my project is that I will be using self-taken pictures, so there is no benchmark model on the internet that used the same dataset (especially because the quality of the pictures and the small size of the dataset).

To serve as a benchmark, I will create a CNN that will be trained without using transfer learning, just the 400 pictures in my dataset. Because the amount of data is small, I will use a small model without ton of layers, to reduce overfitting.

The model will take 299x299x3 images as input and then pass it trough three pairs of Convolution/MaxPooling layers, with the convolution layers containing 16, 34 and 64

filters respectively. All convolution layers will use a filter size of 2, same padding and relu activation. The MaxPooling layers will use a pool size of 2. After those 6 layers, I will use a Gloval Average Pooling (GAP) layer to flatten the result of the last layer, past trough a 10-neuron fully connected layer and then, finally, trough a 2-neuron fully connected layer that will output the class. In both of the fully connected layers I will use the relu activation function.

Evaluation Metrics

The images trained on ImageNet has two main evaluation metrics: top-1 accuracy and top-5 accuracy. Top-1 accuracy is calculated by looking at the class that the model gives the biggest probability and to compare with the actual label. Top-5 accuracy is similar, but we look at the five classes with more probability and compare it with the real class.

In my problem, there is no top-5 accuracy because the model just predicts two probabilities, so I will use just the simple accuracy, by counting how many pictures did the model has correctly classified.

Project Design

This project will be divided in three macro steps: Data Preparation, Model Selection and Data Augmentation.

In the Data Preparation step, I will first collect all the pictures (today they are scattered into smartphones and computers), organize into a single folder and then create a label if it contains one cat or another. After it, I will process the images by scaling it to the desired size for each model (331x331 for the NASNetLarge and 299x299 for the InceptionResNetV2 and Xception) and then scaling the pixel values to a 0, 1 range.

The last step of data preparation is to obtain the bottleneck features [10]. Those features are obtained for each of the three models by passing the input images trough the network except for the last layer and then saved. Those features will be utilized to train the new fully connected layer that will classify the images into my two categories. This is done to speed the training process, because when training we just need to train one layer without having to pass the image trough all the convolutional layers before.

The model selection step consist in training the three models and comparing their performance on a test set, using accuracy as measure. The model with the best accuracy will be selected to use in the next step (Performance Improvements). Below are a description of each model used:

The NASNet Large (Neural Architecture Search Network) is a model developed by Google AI in 2018 [6] that was obtained by using the AutoML project, an approach that makes the model learn the best architecture for some task by using some building blocks (like convolution, average pooling and max pooling layers) [11][12]. This specific model architecture was built to solve the CIFAR-10 dataset and after was applied to the ImageNet dataset, obtaining a state-of-the-art performance at that time. The model implemented in Keras has a top-1 accuracy of 0.825 and a top-5 accuracy of 0.960 on the ImageNet dataset.

The InceptionResNetV2 [7] is a model developed by Google AI in 2016 that combines the earlier model version Inception V4 with Microsoft's ResNet models[13]. This combination is based on using the Inception blocks developed by the Inception models with the Residual connections used on the ResNet models. Below are a diagram of the model architecture (taken from [7]):

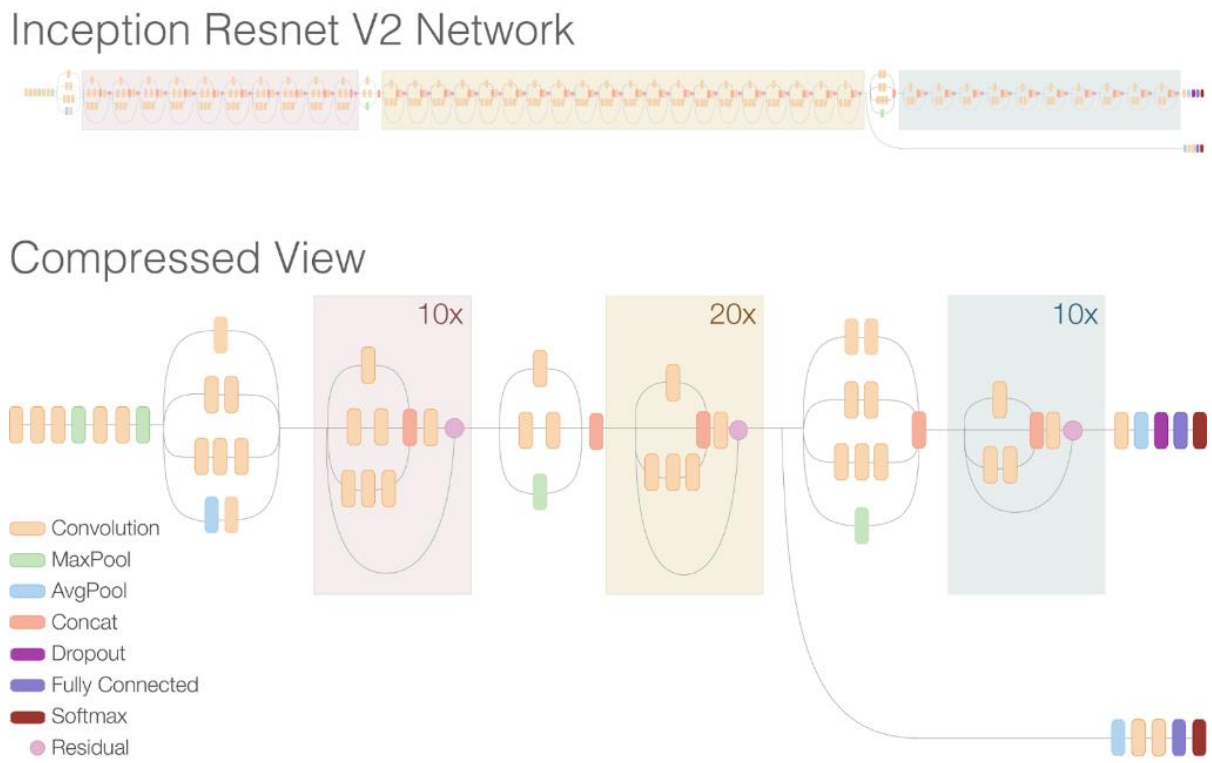

Figure 4: InceptionResNet V2

When the model had launched it had a state-of-the-art performance on the ILSVC 2012 image classification benchmark (that used the ImageNet database) [14]. On the models implemented on Keras it has a top-1 accuracy of 0.803 and a top-5 accuracy of 0.953, the second-best values on the models implemented.

The Xception model (Extreme Inception)[8] is a model developed by François Chollet, a AI Researcher at Google and one of the core developers of Keras [15], in 2017. The basic idea of the model is to create an architecture based entirely on depthwise separable convolution layers, which can be understand as a stronger version of the Inception hypothesis used on the Inception family of models (hence the name Extreme Inception). Figure 5 contains the model architecture, as found on [8]. The implementation on Keras has a top-1 accuracy of 0.790 and a top-5 accuracy of 0.945.

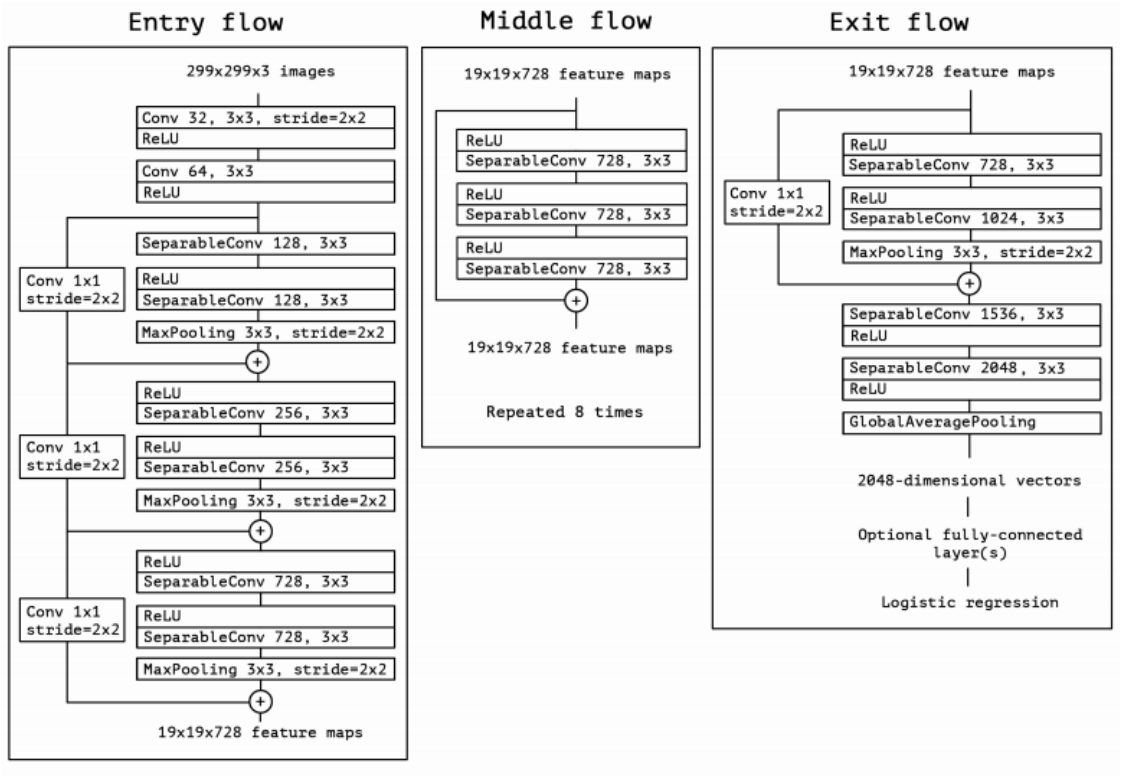

Figure 5: Xception network

The next macro step, Data Augmentation, consists in taking the best model on the last macro step use Image Augmentation in the training of the model, which is done by adding some minor changes (like rotation and translation) to the images in our dataset [16]. This will be implemented by using the ImageDataGenerator available in Keras[17]. Data Augmentation seem very important for this problem because of the small size of the training set, so the model can learn trough those new augmented pictures and reduce overfitting.

References

[1] - https://towardsdatascience.com/7-practical-deep-learning-tips-97a9f514100e

[2] - https://hackernoon.com/%EF%B8%8F-big-challenge-in-deep-learning-training-data-31a88b97b282

[3] - https://www.theverge.com/2017/11/1/16589246/machine-learning-data-science-dirty-data-kaggle-survey-2017

[4] - https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/

[5] - https://indatalabs.com/blog/data-science/applications-computer-vision-across-industries

[6] - https://arxiv.org/abs/1707.07012

[7] - https://arxiv.org/abs/1602.07261

[8] - https://arxiv.org/abs/1610.02357

[9] - https://keras.io/applications/

[10] - https://ai.stackexchange.com/questions/3089/what-are-bottleneck-features

[11] - https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html

[12] - https://towardsdatascience.com/everything-you-need-to-know-about-automl-and-neural-architecture-search-8db1863682bf

[13] - https://ai.googleblog.com/2016/08/improving-inception-and-image.html

[14] - http://image-net.org/challenges/LSVRC/2012/

[15] - https://hackernoon.com/interview-with-the-creator-of-keras-ai-researcher-fran%C3%A7ois-chollet-823cf1099b7c

[16] - https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

[17] - https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html