Machine Learning Engineer Nanodegree

Capstone Project

Hoang Chung Hien
February 16th, 2018

I. Definition

Project Overview

While go through vacation photos, people always ask themself: What is the name of this temple I visited in Thailand? Who created this monument in France? Landmark recognition can help! This technology can predict landmark labels directly from image pixels, to help people better understand and organize their photo collections.

In this project, a model use for landmark recognition will be built based on ImageNet Classification With Deep Convolutional Neural Networks[1]

Problem Statement

The goal of this project is to build a classification model that will be used to recognize landmark from a photo. A few problems need to be solved to achieve the goal:

- The landmark dataset that will be used to train the model, this can be solved by using dataset provided by Google Landmark Recognition Challenge[2].
- Train a classifier model using the dataset above would take a lot of time and computational resource. This problem can be solved by using transfer learning[3].

Metrics

This is a categorical classifier model which can be evaluate effectively using categorical accuracy metric for this balanced dataset.

$$accuracy = \frac{number\ of\ correct\ classes\ predicted}{dataset\ size}$$

[1] http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[2] https://www.kaggle.com/c/landmark-recognition-challenge/data
[3] https://en.wikipedia.org/wiki/Transfer_learning

In this landmark recognition problem, it doesn't matter if the classifier have some false recognition. In other words, user won't be affected if the model give some wrong recognize. Instead, it will be fun experience. Thus, accuracy score will work good in this case.

II. Analysis

Data Exploration

The training set was constructed by clustering photos with respect to their geolocation and visual similarity using an algorithm similar to the one described in [4][4]. Matches between training images were established using local feature matching. Note that there may be multiple clusters per landmark, which typically correspond to different views or different parts of the landmark.

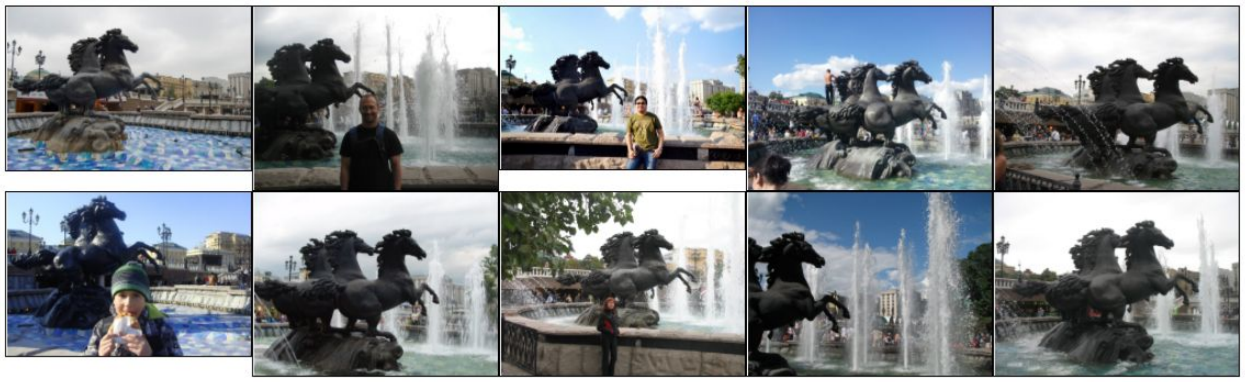


Fig. 1 Some images from the dataset.

The dataset provided by [5][5] which is obtained from a kaggle challenge[6]. The provided dataset was very large which have 1,225,029 images which belongs to 14,951 classes. Let say each image take 200 kilobytes, then download all the images would requires up to 244 gigabytes which will take a week, train for only one model will take a month (based on my current limiting computational resource). For that reason, I will use only images from about 1% (149 classes) of the total classes and split the training dataset into 75% for training, 25% for validation and testing. I believe 1% of the total classes would be enough to evaluate the solution model with a reasonable training time.

Exploratory Visualization

The chart below show the distribution of images among the classes in the original dataset.

[4] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher T.-S. Chua, and H. Neven, "Tour the World: Building a Web-Scale Landmark Recognition Engine," Proc. CVPR 09

[5] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han, "Large-Scale Image Retrieval with Attentive Deep Local Features", Proc. ICCV'17

[6] https://www.kaggle.com/c/landmark-recognition-challenge/data

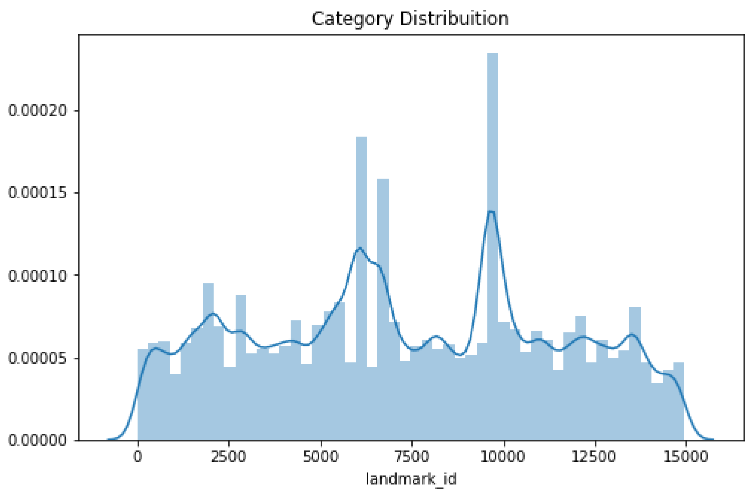
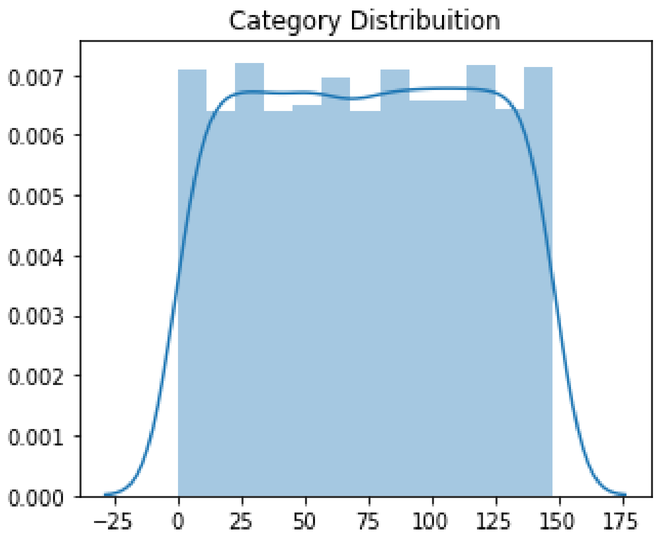


Fig. 2 A plot showing how images distributed among the classes of the original dataset. The chart clearly indicate an imbalance of the distribution. There are few classes which contains exceptionally high number of images.

Fig. 3 A plot showing the distribution of the reduced version of the original dataset. The dataset now contains only 1% (149 of 14,951 classes) of the original classes. The x-axis is the class number from 0 to 149. The chart clearly indicate the reduced dataset is balanced.



Algorithms and Techniques

The classifier will be built using Convolutional Neural Network (CNN) technique which use in almost image processing tasks, including classification. Specifically I will use a VGG16 pre-trained model[7] implemented by keras team[8]. This is the Keras[9] model of 16-layer network used by VGG team in the ILSVRC-2014 competition[10]. This pre-trained model would do a

[7] Very Deep Convolutional Networks for Large-Scale Image Recognition K. Simonyan, A. Zisserman arXiv:1409.1556
[8] https://github.com/keras-team/keras/blob/master/keras/applications/vgg16.py
[9] https://keras.io/
[10] http://www.image-net.org/challenges/LSVRC/2014/

hardest task of extracting features from an image and act as input for my fully connected deep neural network to detect which landmark that image belongs to.

The following parameters will be tuned to optimize the classifier:

- Number of epochs
- Learning rate
- Optimizer
- Number of fully connected layers as well as nodes of each layer

Benchmark

To create benchmark for the solution, I will build a simple CNN model and train with the same dataset that I will use to train my solution model. To understand how best my solution perform, I compare the accuracy score in both training and validation of the solution model with the simple one.

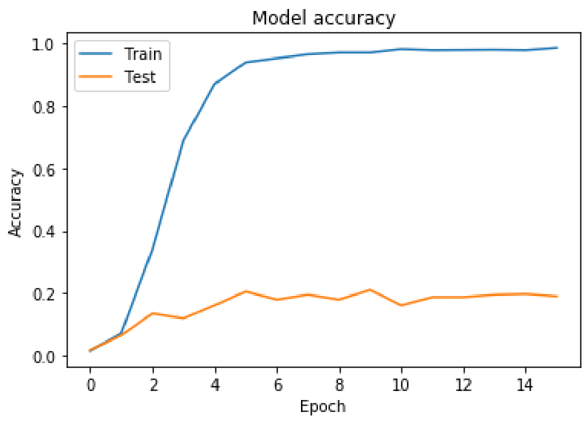


Fig. 4 Without data augmentation

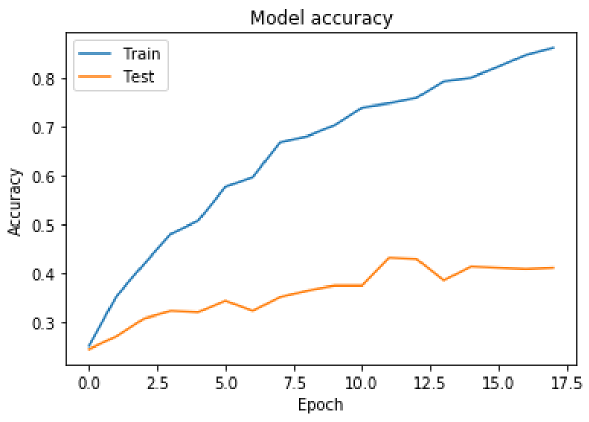


Fig. 5 With data augmentation

The charts above show the performance of the Simple CNN model. The first one (Fig. 4) without apply data augmentation yield the accuracy score of 0.1879. The last one (Fig. 5) apply data augmentation yield the accuracy score of 0.4359. Generally speaking, the first one can recognize 18.79% of the test dataset, the last one can recognize 43.59% of the test dataset. Pretty good recognition by a simple classifier.

III. Methodology

Data Preprocessing

1. The images dataset have been divided into 75% for training, 15% for validation and 10% for testing.
2. The size of the images need to be scaled to a fixed size of 224x224 and the pixels value need to be transformed to a range of [0.0 to 1.0] before fetch into a learning model.

Implementation

1. Load a VGG16 model and use it to extract the bottleneck features from dataset images. Then save the extracted bottleneck features that will be used for training step.
2. Define a fully connected model with input size equals to the number of features that load from the bottleneck features file saved from the previous step. For the next layer have a size of 512 nodes and using relu activation. The last layer have a size equals to total classes of the dataset (149 nodes).
3. Compile the model with the rmsprop optimizer. If the score doesn't work well after training, I will change other optimizer and train again.
4. Fit training and validation bottleneck features to the model and use checkpointer to save best weight during training process. If the validation score doesn't increase after 5 epochs, then terminate the training process.

Refinement

The benchmark model reported earlier could recognize **43.59%** of the test dataset. The initial solution does a little better job than the benchmark model, it could recognize **65.1%** of the test dataset. **Figure 6** clearly show how good the initial solution perform during few epochs.

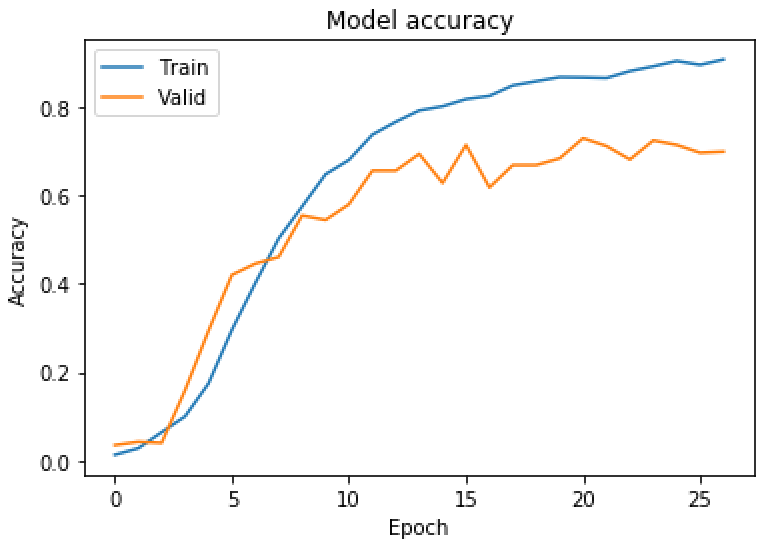


Fig. 6 Initial solution

After fine tuning, the final solution model yield better result by using the momentum optimizer (SGD[11]) with learning rate of 0.0001 and momentum of 0.9. The final model can recognize **83.22%** of the test dataset. **Figure 7** clearly show how good the final solution model perform.

[11] https://en.wikipedia.org/wiki/Stochastic_gradient_descent

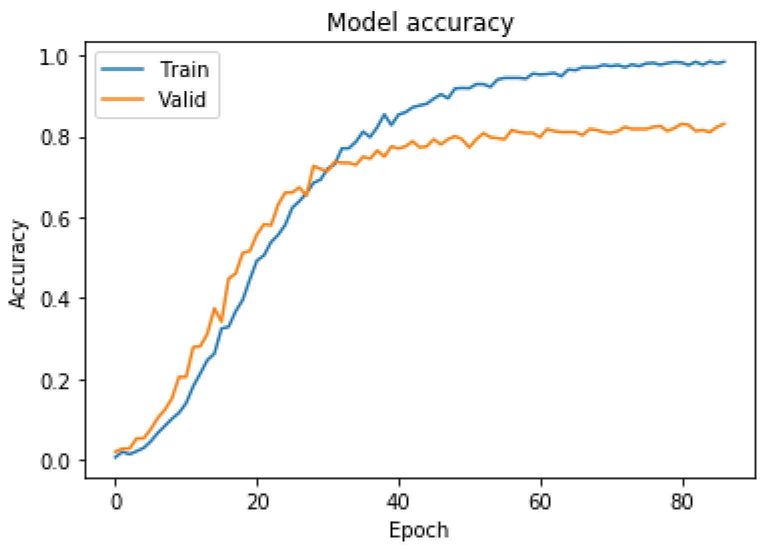


Fig. 7 Fine tuned model performance

IV. Results

Model Evaluation and Validation

The final architecture and hyperparameters were chosen because they performed best with the unseen data, in this case is the test dataset that I've splitted in the data preprocess step.

To test the robustness of the solution, I generate a different set of train dataset with the same number of images using data augmentation. Even though the necessary features could be crop, but the final result didn't change so significantly. It can recognize **76.51%** of test dataset, not far from the final result of **83.22%**.

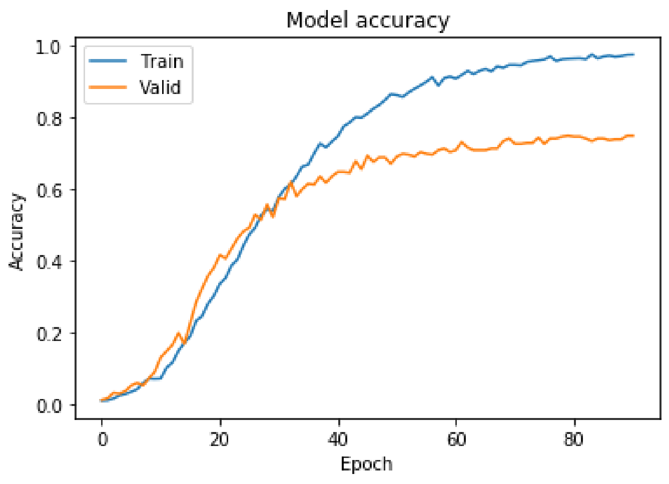


Fig. 8 Final model train by augmented dataset

The result of the final solution are robust and trustable because it can recognize over **75%** of the unseen images even though the train dataset are small, just a few images in each class.

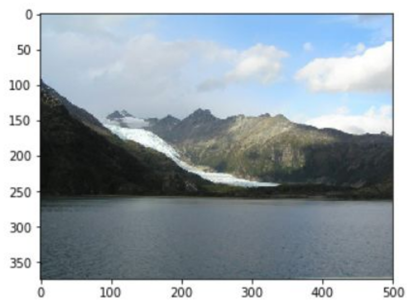
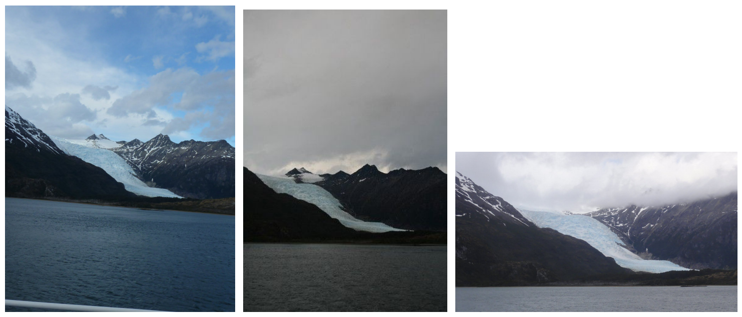
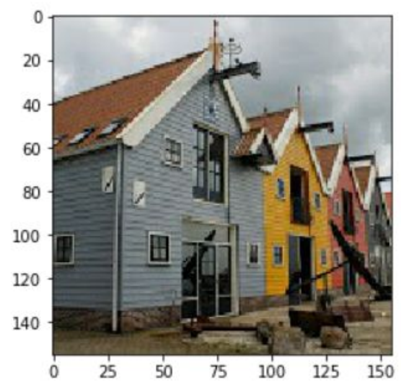
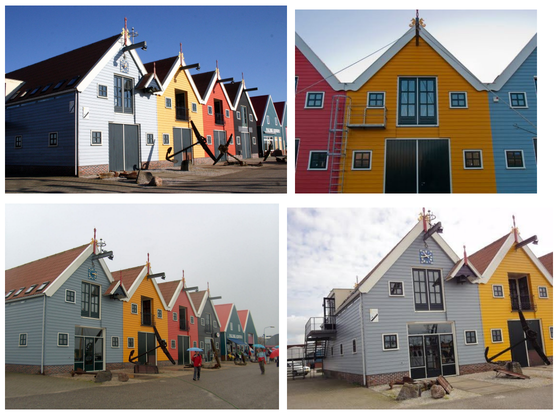
Justification

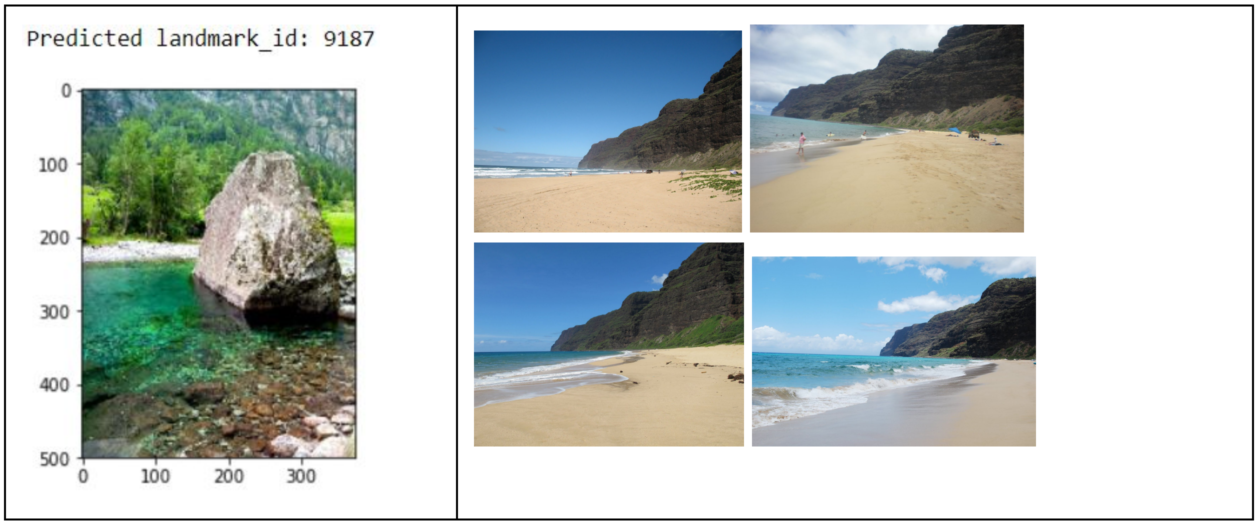
Figure 5 clearly show the performance of the benchmark model. Accuracy score of the benchmark model only **0.4359** compare to final model which yield **0.8322**, the performance which is significant increased.

The final model can solve the problem better than the benchmark model because it can recognize more images than the benchmark model.

V. Conclusion

Free-Form Visualization

Recognized	Images with the same landmark_id
`Predicted landmark_id: 205` 	
`Predicted landmark_id: 382` 	



```
Predicted landmark_id: 9187
```

Fig. 9

Look like the classifier does a pretty great job, it can recognize almost images in the test dataset. Take a look at the last image in the **Figure 9**, I can see the same patterns like water, mountain, trees, etc that caused the classifier to give wrong recognition.

Reflection

The entire process used for this project can be summarized in few step below:

1. Find an initial problem and a public dataset.
2. Analyse the dataset to know whether to reduce the dataset or not.
3. Download and preprocess the images from the reduced version of the dataset.
4. Create a benchmark model and train using the downloaded images
5. Extract bottleneck features from images using VGG16 model
6. Train the final model using the extracted features

Even though the reduced version of the dataset is small, but I found it harder to perform step 5 and 6, because it take a lot of time and hard to know the performance until I tried a lot of parameters tuning.

The dataset used for this project are the most interested to me. The images use to train to recognize most classes were small, just few images. But the final model can still recognize most of them. The transfer learning so powerful in this case. I believe transfer learning can be used to recognize almost everything in the future.

Improvement

The result of more than 80% might not satisfy every users. To increase more accuracy, I believe that it's necessary to collect a little more images for each class. There are many others models I'll try to compare with my final model, they are VGG19, ResNet, etc.

After archive the accuracy of more than 90%, I will have confident to train it with the full dataset to recognize 14,951 of classes.