Final Report

The First Touch on Landmark Detection

Machine Learning and Computational Statistics
Spring 2018

Team Members:

Xintian Han(xh1007)
Shuaiji Li(sl6486)
Xinyang Qiu(xq303)
Tianyu Wang(tw1682)

Advisor:

Dr. Vitaly Kuznetsov

Abstract

Using image pixels to identify landmark labels is an important and tricky multi-class problem. In this project, we present a systematical data science approach, from collecting data to predictive modeling. We compare the performance between logistic regression, Random Rorests, AlexNet, and our designed model on this task. Current state-of-the-art architecture such as VGG-19 and Inception-v3 are employed as well.

1 Introduction

Being able to automatically identify landmark labels directly from image pixels is a challenging task, but it could have great impact, for instance by helping people better understand and organize their photo collections. For this project, we aim to predict the key landmark (i.e. Yosemite, Forbidden City) on each validation image. The task is harder than the well-studied image classification or object recognition tasks in the sense that the data we use are uploaded and collected directly from users' Google Cloud, with variable quality. We will show how we process raw images and build effective predictive models from a data science perspective.

2 Data Preparation

The images we use are unstructured data that needed to be pre-processed carefully before fed into any predictive models.

2.1 Description

Thank Google for sharing the largest worldwide dataset to date on Kaggle to foster progress in landmark recognition research. We utilize this competition data to approach the task. The original dataset provided by the host contains 1,225,029 training images and 117,703 testing images, where each image has a landmark label anonymized as a categorical number. The train.csv and test.csv files follow the same format: each instance has a unique id, one url from the image address, and one label id. There is no missing data. There are 14,951 distinct labels and many least frequent labels have less than five images. The distribution of all categories/labels is shown below:

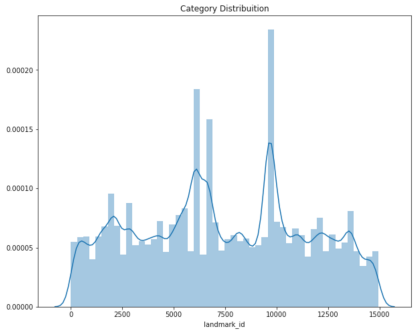

Figure 1: Distribution of landmark category

2.2 Scraping and Exploration

Figure 1 illustrates how labels are unevenly distributed. Due to the limitation of computational resources and time, it's hardly possible to process and establish a throughly trained model over all the original raw

images (roughly 340 gigabytes). Instead of pursuing a top ranking score on the leader board, we aim to approach the task systemically and channel more efforts into model refinements.

We pick 10 labels with ample amount of images and randomly draw 1500 training images and roughly 1500 validation images. Both training and validation images are randomly shuffled first. Since all the images are public and accessible via Google UserContent API, we directly fetch the images through url links and load them based on their shuffled ids.

Our dataset ends up with 29,357 total images, 15,000 of them are training images, and 14,357 are validation images. Some training samples are visualized below:

Figure 2: Sample images with clearly identifiable landmarks

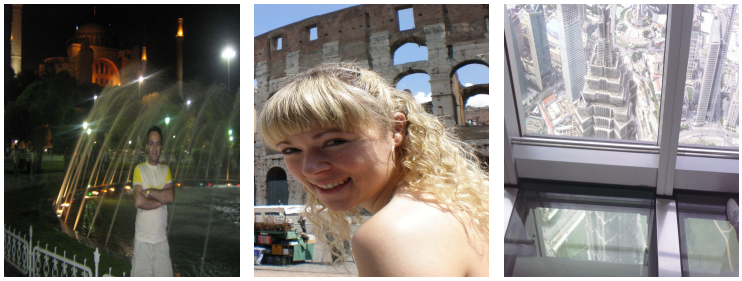

Figure 3: Sample images with obscure landmarks

Noticed that images in Figure 2 are well-constructed with clearly identifiable landmarks, whereas images in Figure 3 are containing either obscure objects or gloomy backgrounds. For instance, the second image in Figure 3 contains a human body which takes over more than 70% space of the image. In case like this, it's even infeasible to correctly identify the landmark as a human being.

At this point, we are not sure if those poorly qualified images have a negative impact on our predictive models. So we manually mark and filter all the images with the similar conditions. The resulting sub-dataset contains 22,060 normal images where 11,224 are training and 10,836 are validation images. We apply each configuration on both datasets (completed and filtered) and further compare the evaluation result.

2.3 Data Transformation

Images in different sizes can hardly be used as inputs for a predictive model directly. We thus resize all images to 150×150 to balance the trade-off between amount of trainable pixels and limitation of computational resource. Instead of applying data cropping which selects a subset of the original image only, resizing makes sure that target landmark is always preserved throughout the transformation.

After resizing, two major transformations are applied to both datasets. First, as illustrated in Figure 3, images uploaded by Google Cloud users have variable quality. Some images are taken from a normal photo perspective, whereas others are fairly random. To preserve the consistency of landmark patterns' distribution between training and validation set, we randomly pick some training and validation images and flip them

horizontally. This transformation takes direction and angle of view into account. Second, we normalize images following the MinMax normalization schema. We examine some sample images and find that the raw pixel values come with different scale. To expedite the training speed and avoid potential gradient explosion (for neural network specifically), we scale and shift pixels values to a range of $[0, 1]$.

3 Evaluation

Our problem is a multi-class classification problem. We can use several multi-class evaluation metrics to measure the performance.

3.1 Accuracy

The easiest way is to use the accuracy. Suppose we have n samples. For each sample i, denote y_i to be the true label and $\hat{y}_i$ to be the predicted label. The accuracy is defined by:

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} 1(y_i = \hat{y}_i).$$

3.2 Global Average Precision

The second evaluation metric is the Global Average Precision (GAP) at $k = 1$, or micro Average Precision (microAP) invented by [Perronnin et al., 2009]. Here $k = 1$ means we only use the label with highest predicted confidence score. For each query image, we predict one landmark label and a corresponding confidence score. The confidence score can be the margin in SVM or the predicted probability in neural networks. Then we treat each prediction as an individual data point in a long list, sort them in descending order by confidence scores, and compute the Average Precision based on this list.

If we have n predictions (label/confidence pairs) sorted in descending order by their confidence scores, then the Global Average Precision is computed as:

$$\text{GAP} = \sum_{i=1}^{n} P(i) rel(i)$$

where $P(i)$ is the precision at rank i and $rel(i) = 1$ if the prediction is correct and 0 otherwise. GAP is a way to balance precision and recall.

3.3 Comparison of Accuracy and Global Average Precision

Accuracy only cares about the target with highest confidence level but sometimes we need to take ranking into account. For example, in this Landmark Detection task, if a person wants to know where the photo is taken from, we may give a list of possible places sorted by confidence levels in our algorithms. The GAP will be high if the prediction with high confidence level is correct for most time. That means not only the prediction we get is correct but also the confidence level we get is correct. With larger confidence level, we should obtain higher accuracy. This GAP helps in search service with ranking.

In short, if we only care about one prediction, we should use accuracy; if we need a list of prediction, GAP is what we need. In this task, we choose both criteria to evaluate our model since accuracy is a straightforward way to show the goodness of the model and GAP meets what we need in application.

4 Training Methodology and Experimental Results

We start from traditional machine learning algorithms, like logistic regression and Random Forests. And then we try different type of convolutional neural networks. We also tune the hyperparameters and choose the best combination of hyperparameters.

4.1 Baseline Algorithms

We start by proposing some baseline models that use relatively simple algorithms to solve this detection problem.

4.1.1 Logistic Regression

The simplest approach to attack this problem is to use the multi-class logistic regression. Here we use the version in [Nasrabadi, 2007]. The posterior probability are given by a softmax transformation of linear functions of the feature variables. Then we use maximum likelihood to determine weights in the linear functions. We try different learning rates and weight decays. The validation accuracy is 33% on the complete dataset, and 40% on the filtered dataset. GAP is also quite low in both cases, as shown in Table 1.

4.1.2 Random Forests

Inspired by [Bosch et al., 2007], we think tree algorithm such as Random Forests may also be a fair baseline algorithm for the task. Random Forests is a special type of ensemble learning algorithm that utilizes bagged decision tress which fit on bootstrapped samples of the training data. Unlike the general bagging algorithm, Random Forests modifies the tree-growing procedure to control the correlation between trees by restricting choice of splitting variable to a randomly chosen subset of features at each non-leaf node. Compared with logistic regression, Random Forests should better capture non-linearity in the data. In our case, since all features are image pixels which are stationary and locally correlated, we should expect Random Forests works better than the linear-type of algorithm.

One thing to note in training phase is that Random Forests may easily overfit the data if we do not put any constraint on hyper-parameters. In specific, if we do not limit the number of trees or the minimum number of samples required to split an internal node, the model will recursively optimize the objective function until all the training instances are correctly labeled. To reduce overfitting, we grid-search the splitting criterion, the number of tress, and the minimum samples to split with regularized conditions. We obtain a best configuration with 51% validation accuracy and 0.4 validation GAP on both datasets, presented in Table 1. As we expected, Random Forest outperforms Logistic regression in this task.

4.2 Advanced Models

Now given our baseline models, we propose the following architectures and frameworks to expand the predictive power of our models.

4.2.1 AlexNet

Convolution is a breakthrough in image recognition. A Convolutional Neural Network (CNN) consists one or more convolution layers (often with a sub-sampling step), followed by one or more fully connected layers. The architecture of CNN is designed to capture the 2-D spatial correlation of an feature map. For every kernel, the weight matrix is shared across whole input map. The sub-sampling layer (max/avg pooling) creates translation invariance over the pooling region.

One of the first world-famous CNN that achieves unprecedented low error rate on ImageNet, AlexNet, proposed by [Krizhevsky et al., 2012] in Figure 4, takes the virtue of both convolution layer and sub-sampling layer. For our task, we decide to adopt and modify AlexNet first. Our modified AlexNet contains seven weighted layers where the first six are convolution layers and the last one is fully connected. Raw input image passed through the convolution layers is embedded as a vector. The last layer of our net than takes this feature representation and output the likelihood of the input image being in each landmark class through a softmax function. Noticed that maximizing log-likelihood is equivalent to minimizing cross-entropy in the training setting.

Compared with traditional machine learning algorithms such as linear regression and Random Forests, AlexNet can learn pixel-based features, local correlated features, and overall characteristics holistically. Thus

reflected in the result, AlexNet exceeds our best validation accuracy and validation GAP of baseline models by 10 percent, illustrated in Table 1.

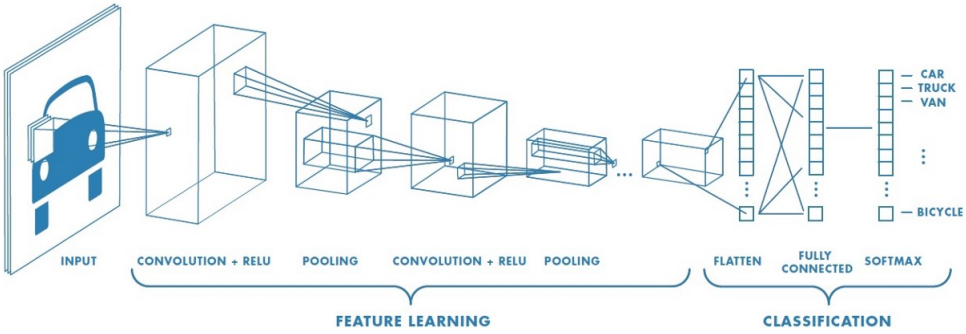

Figure 4: AlexNet Architecture, [Thomas, 2018]

4.2.2 OurNet

Although we have obtained considerable progress on AlexNet over the baseline algorithms, two potential problems raise from this neural network structure. First, a simple seven-layer network may not be "strong" enough to extract all the useful representations from input images. Second, pixels with various scales tend to lower the training speed of the network.

To deal with these issues, we come up with our own neural net, OurNet, with detailed structure presented in Figure 5. OurNet appends six convolution layers on top of AlexNet and three more fully connected layers at the end to expand range of different transformations. In order to balance the increasingly complex architecture with training time, we add batch normalization layer before each activation function. Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. Not only the scaled and shifted input maps reduce forward passing time to adjust activations, gradients backpropagated through backward pass also take a lower risk of being vanished/exploded.

Besides, we add a dropout layer before the last fully connected layer to prevent potential overfitting. In this layer, we set the output of each hidden neuron to zero with probability 0.5. The neurons which are "dropped out" in this way do not contribute to the forward pass or participate in backpropagation. Every time an input is presented, the neural network samples a different configuration. The network thus becomes less sensitive to the specific weights of neurons.

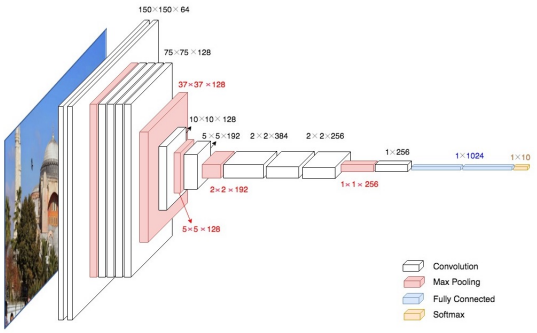

Figure 5: OurNet Architecture

We train OurNet throughly on both datasets. The validation accuracies are 73.9% and 76.4% respectively, demonstrating a great improvement over AlexNet, as presented in Table 1. Noticed that the discrepancy of

validation GAP on two datasets drops to 0.029, implying that input images with variable quality now have less effect on our model formulations.

4.2.3 VGG19

OurNet and AlexNet share a similar architecture in terms of feature extraction. We conduct a further study in this architecture to see whether improvement upon this part leads to a better result.

As shown in Figure 6, VGG19, proposed by [Simonyan and Zisserman, 2014], contains nineteen layers. The first sixteen convolution layers extract useful feature representations sequentially with batch normalization applied. The last three fully connected layers flatten the feature map and match the hidden representations with corresponding class labels.

Compared with AlexNet which adopts a "shallow" network with large kernel-sized filters solely, VGG19 takes one step further by using multiple small size filters and a deeper net. Although they share similar performance on global feature extraction, multiple stacked smaller size kernels can learn more detailed and complex features at a lower computation cost. For each input map, local feature correlations(primarily from adjacent pixels) missed by large kernels will be retained by small kernels. The training result meets with our expectation. VGG-19 achieves around 90% validation accuracy and 0.9 GAP on both datasets, shown in Table 1.

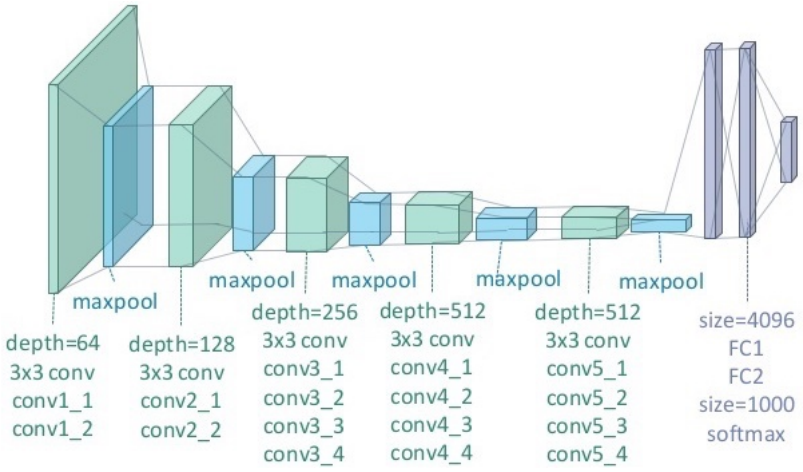

Figure 6: VGG19 Architecture, [Li, 2018]

4.2.4 Inception-v3

Previous endeavors have shown that deeper convolutional neural network with various kernel sizes enhances the predicting capability of our models. In addition to increasing the depth of our neural network, we believe a "wider" structure where multiple transformations are implemented over a same input map should provide more information regarding the adjacent pixels.

Remember that in a traditional convolution net, each layer extracts information from the previous layer in order to transform the input data into a more useful representation. However, layers with different kernel size (i.e. 5×5 and 3×3) extracts different kind of information. At any given layer, it is often not easy to tell what transformation provides the most useful information.

One of the state-of-the-art architectures, Inception-v3, presented by [Szegedy et al., 2015], resolves this issue by computing multiple different transformations over the same input map in parallel, and then concatenating their results into a single output. As shown in Figure 7, the key innovation is to apply several 1×1 pointwise convolution/max-pooling layers first to performing dimensionality reduction. A 1×1 convolution only looks at one value at a time, but across multiple channels. It can extract spatial information and

compress it down to a lower dimension. Each Inception module hence stacks different layer transformations in parallel without much increase of computational costs.

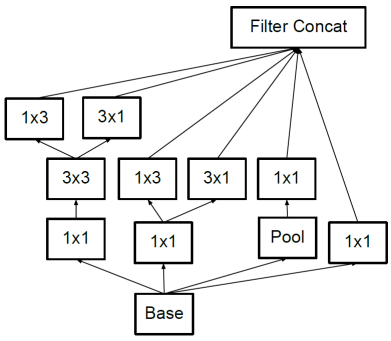

Figure 7: A sample Inception Module, [Szegedy et al., 2015]

The original Inception-v3 architecture pre-trained on ImageNet consists of 313 layers. We replace the last layer with a 10-neurons fully connected layer to transfer the learning objectives into our specific task. We fix the first 263 layers and fine-tune the last 50 layers. For each batch normalization layer, we do aggressive exponential smoothing of batch norm parameter, with momentum set to 0.8, to faster adjust to our new dataset. The resulting model achieves highest measurements between every evaluation metrics on both datasets, shown in Table 1. One crucial thing to note is that the gap of validation accuracy between the complete dataset and the filtered dataset shrinks to 1.4%. It implies that our advanced model effectively alleviates the potentially negative impact of poor-quality images on landmark prediction.

Table 1: Experiment Result

	Model	Evaluation Metrics					
		Completed Dataset			Filtered Dataset		
		Train Acc.	Val Acc.	Val GAP	Train Acc.	Val Acc.	Val GAP
Baseline	LR	0.851	0.330	0.179	0.927	0.402	0.239
	RF	0.999	0.506	0.395	0.999	0.508	0.397
Advanced	AlexNet	0.999	0.616	0.545	1.000	0.660	0.590
	OurNet	1.000	0.739	0.699	1.000	0.764	0.728
	VGG19	1.000	0.895	0.887	1.000	0.911	0.904
	Inception-v3	**1.000**	**0.953**	**0.907**	**1.000**	**0.967**	**0.931**

5 Conclusions and Future Work

In this project, we tackle the landmark detection problem from a machine learning perspective. We start by showing how to collect raw data, and what data transformation tricks are applicable for our case. Several baseline algorithms are employed to give some primary measures of this task. We then continuously refine our models until the best configuration is obtained, with a 95% accuracy on the validation set. Within the

process of enhancing our predictive modeling frameworks, we observe that complex architectures such as VGG19 and Inception-v3, are capable of capturing deep internal correlations despite the presence of some poor-quality training images.

As for potential future work, we can utilize more computer vision methods to identify interfering object (i.e.human body) and replace it with predefined filler. We can try Regions with Convolutional Neural Network (R-CNN) architecture with alignment, proposed by [Karpathy and Li, 2014], to predict landmark given the location of object. If that works well, then there is no need to ever manually filter data based on image quality.

References

A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. 1:14–21, 01 2007.

A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL `http://arxiv.org/abs/1412.2306`.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

L. Li. Classifying caltech 101 categories of images, 2018. URL `https://lihan.me/2018/01/vgg19-caltech101-classification/`.

N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

F. Perronnin, Y. Liu, and J.-M. Renders. A family of contextual measures of similarity between distributions with application to image retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2358–2365. IEEE, 2009.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

R. Thomas. Convolutional networks for everyone-rohan thomas- medium, 2018. URL `https://medium.com/@rohanthomas.me/convolutional-networks-for-everyone-1d0699de1a9d`.