LANDMARK RECOGNITION
Large-scale Classification on Noisy Imbalanced Dataset

Renke Cai
CS231N, Stanford University
Renke.cai@stanford.edu

Chenjiao Wang
CS230, Stanford University
Chenjiao.wang@stanford.edu

Abstract

Landmark recognition is a large-scale image classification problem on imbalanced datasets. This paper discusses a transfer learning approach to training the classifier using CNN architectures such as VGG16 and Xception that are pre-trained on ImageNet, as well as a few practical aspects including data augmentation, model fine-tuning as well as their impact on the performance of the classifier. Some data/model visualization and qualitative analysis are also conducted in order to better explain the model prediction performance.

1. Introduction

The landmark recognition problem comes from a Kaggle Challenge launched by Google Inc. in April, 2019. The goal is to build models that recognize the correct landmark (if any) in a large-scale dataset of images.

This problem of extreme classification is prevalent in the data science community today with the advancement of deep learning. The dataset for this challenge is provided by Google and the Dataset is with 5 million images depicting human-made and natural landmarks spanning 200 thousand classes. Dealing with large-scale dataset but also a large number of classes with very few images in many classes is particularly challenging. And in terms of computational resources, increasing image resolutions may significantly increases the training time therefore there will have to be a tradeoff.

The input data are raw landmark pictures. We experimented with VGG16 and Xception initialized with weights trained on ImageNet with our customized top layers attached in order to predict a label for each image, followed by which are various experiments and analysis of the results. We also designed a voting mechanism at the stage of inference by doing cropping and prediction for each of the test image and came up with a heuristic confidence score associated with each predicted class label.

2. Related Work

Our work is concerned mainly the deep learning architecture, as it is advancing a number of pattern recognition and machine learning areas and deep convolutional neural networks (CNNs) could always have a noticeable performance in the computer vision problems. As our task is associated with large scale data set process, the deep CNN is a good choice to tackle the task.

We choose to use VGG16[1] by Karen el at's as our starting point for this deep learning project as it is very powerful model and useful as image classifier and as the basis for new models that use image inputs. It increasing depth using an architecture with very small (3*3) convolution filters by pushing to depth to 16 weight layers.

Xception[11] also have been used as our base model as it is a relatively new model by Francois and it outperformed Inceptions V3 on a larger image data classification dataset by replacing the standard inception modules with depth-wise separable convolutions. And the weights serialization are smallest compared to the other classic models VGGs and ResNet50.

Transfer learning allows us to train deep networks using significantly less data/time than we would need if we had to train from scratch. With transfer learning, we are in effect transferring the "knowledge" that a model has learned from a previous dataset, to our current one. The idea is we can leverage whatever network parameters that model has learned through its extensive training, without having to do that training ourselves. As Transfer learning has been consistently proven to boost model accuracy and reduce require training time, we will use transfer learning through out the project.

3. Dataset and Features

The dataset was constructed by clustering images based on geo-locations. The original training dataset (over 500 GB) contains 4,132,914 images of 203,094 famous as well as not so famous landmarks all over the world. The numbers of images varies high from classes to classes. Maximum images in a class are 10247 while a number of other classes contain only 1 picture. Figure 1 plots the number of samples for the most frequent and least frequent labels. This imbalanced dataset will require a more careful data augmentation technique.

In the time of interest, we filtered out 6,401 classes which consist of 1,243,915 images in total for our training. However, even on such a subset of universe, the training set

is still quite noisy. Figure 2 are the images that share the label with the most observations. This does not contain any landmark information.

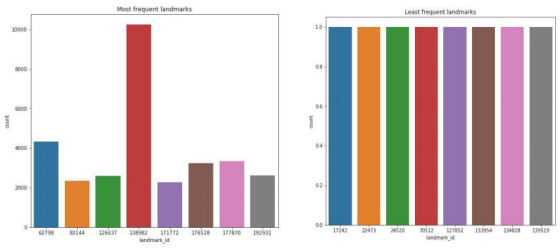

Figure 1 Sample counts of most/least frequent classes

Such images of plants do exist everywhere in many different labels and may confuse the training to some extent. And this top class label takes a significant portion of all the training samples.

Figure 2 Samples from the most frequent landmark

Due to the above nature of our dataset, though the essence of this problem is image classification, it differs from classical classification problem (like ImageNet Challenge) in several perspectives:

- Landmarks do not only have transition and rotation invariance like objects, but also could be totally different in each picture – indoor scene, outdoor landscape, or even just a piece of statue in a person's selfie, etc.;
- There is a lot of noise in the dataset – some pictures are labeled by a landmark but it actually cannot represent anything, and this data may significantly affect the training (see data preprocessing section for details);
- The large number of classes results in many similar-looking landmarks which actually belong to different classes. This adds to the complexity of the problem as human beings may not tell the difference as well.
- As the data is being highly imbalanced, various image augmentation techniques need to be used to minimize difference in the number of image per class.

All of these special characteristics indicate that a more delicate design of data augmentation as well as training and fine tuning process needs to be done.

4. Methods

We trained two classifiers based on the well-known CNN architectures – VGG16 and Xception, as described in the related work, along with our customized top-layer models.

4.1. Baseline: VGG16

VGG16 architecture is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth, which is an improvement over AlexNet that uses kernel of sizes 11 and 5 in its first two convolutional layers. Reducing volume size is then handled by max pooling.

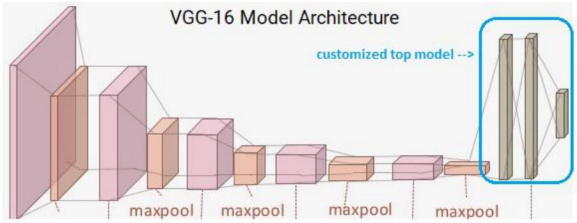

Figure 3 VGG16 Model Architecture
Picture Source: towardsdatascience.com

We used a relatively small top-layer model including two fully-connected layers with dropout and regularization added as well. To predict the final class label we used a softmax layer that has dimension equal to the number of our selected classes.

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$$

The loss function we used in our first experiment is the categorical cross entropy loss

$$L = -\sum_{i=1}^{n} y_i \log(S(f(x_i)))$$

4.2. Xception

Xception stands for extreme Inception. An Inception module computes multiple different transformations over the same input map in parallel, concatenating their results into a single output, using 1x1 convolutions to perform dimensionality reduction. Xception takes this one step further. Instead of partitioning input data into several compressed chunks, it maps the spatial correlations for each output channel separately, and then performs a 1x1 depth-wise convolution to capture cross-channel correlation.

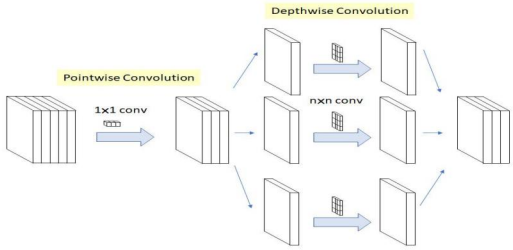

Figure 4 Modified Inception Module in Xception
Picture Source: towardsdatascience.com

Our top model consists of 3 layers: generalized average

pool layer replacing full-connected layer, a dropout layer and an output layer (Softmax).

The generalized average pool layer is utilized to reduce the spatial dimension of a three dimensional tensor while extract the features from the base Xception model. It also helps to flatten the data and minimize over-fitting by reducing the total number of parameters in the model. To further avoid over-fitting, we add another dropout layers scale down the neurons. Adding another dense layer was tried but did not improve the performance while led to quicker over-fitting instead, therefore not used in the final architecture. This means our model only uses the minimum number of parameters, since we do not any parameterized layers besides the output layer.

We still use the most common categorical cross entropy loss for the optimization. In addition, we calculate the Global Average Precision (GAP) as an auxiliary evaluation metric. For N predictions, the Global Average Precision is computed as:

$$GAP = \frac{1}{M}\sum_{i=1}^{N} P(i)rel(i)$$

where:

- M is the total number of queries with at least one landmark from the training set visible in it (note that some queries may not depict landmarks)
- $P(i)$ is the precision at rank i;
- $rel(i)$ denotes the relevance of prediction i: it's 1 if the i-th prediction is correct, and 0 otherwise.

5. Experiments and Results

The 1,243,915 training images dataset is split into 70% for training, 20% for hold-out validation and 10% for testing. The split is performed per class label, ensuring the same distribution across the three sets.

5.1. Data augmentation

Various standard image augmentation techniques such as flip, crop, scale, rotation, translation are used as real-time data augmentation process when the training the images on-the-fly.

Beyond that, we customized an image cropping function that crop the original image into resolution of 150 * 150. Note that the original image we prepared are of width 256 while height is set so that original scale is preserved. This function takes in three parameters:

- Cropping probability: the probability of whether to perform image cropping or not;
- Image resizing scale: resizing the original image into this size before cropping
- Way of random cropping: cropping from the center or corners or edges.

The Python Code of the data cropping and training data generator (that basically implements the Keras ImageDataGenerator API) as well as a few other helper functions are inspired by and adapted from Jan Daldrop 's Github: *https://github.com/jandaldrop/landmark-recognition-challenge/*

Figure 5 illustration of image cropping

This cropping method is useful from two perspectives: first it helps avoid overfitting especially for classes where the number of sample images is small; secondly, each cropping will carry additional information from epoch to epoch since the original image has a higher resolution than target (150*150).

5.2. Transfer learning and VGG16 training

As a baseline method we trained the VGG16 classifier based on weights pre-trained on ImageNet.

First we examined that pre-trained VGG16 does predict meaning patterns using inputs from our landmark dataset. The top three predictions for the sample image (Figure 6) are: *beacon, stupa and barn*. They are reasonable predictions, indicating that the general features have been learned by the neural networks.

Figure 6 Landmark used to generate features from ImageNet

Therefore we decided to continue with the transfer learning approach. We then froze all the convolutional part of the model (from input layer all the way to before fully-connected layers) and trained for 30 epochs.

Table 1 Hyper parameters of VGG16 training

Base Model Name	VGG16
Top-layer Model	FC (dim=256, Relu) – FC (dim=256, Relu) – output(softmax)
Frozen Layers	All convolutional layers

Learning Rate	0.0001
Optimizer	Adam
Loss function	Categorical Cross Entropy

The result is not promising. As a second stage, the top convolutional layers was then set to trainable as well, but that did not bring any improvement.

Table 2 Results of baseline (VGG16) training

	Accuracy	GAP
Training set	0.045	0.0053
Validation set	0.053	0.0067

5.3. Xception training and fine-tuning

Our Xception architecture was trained by three stages with changing hyper parameters. Please note that since we did not have resources to grid search the hyper parameters, they were set in the following way because they either gave impressive results on toy dataset, or were empirical/intuitive enough (hopefully) for direct use.

5.3.1 Top model + a few convolutional layers training

First we froze the bottom 80 layers (Note: This number follows the model implementation of Keras, and is the detailed layer info instead of the general number of conv layers in most architecture flow charts) and train using the following hyper parameters for 20 epochs:

Table 3 Hyper Parameters of Xception training -- stage 1

Base Model Name	Xception
Top-layer Model	Generalized mean pool - dropout – output(softmax)
Frozen Layers	Bottom 80 conv layers
Learning Rate	0.0001
Optimizer	Adam
Loss function	Categorical Cross Entropy
Cropping probability	0.1 to 0.5 linear increasing by epoch
Resizing scale	Resizing to 180*180 before cropping
Way of cropping	Random center position

The following plot (Figure 7) displays the training and validation set accuracy for each epoch during training. We can see a significantly improvement when coming to the second stage. Since the validation accuracy became saturated after a few epochs in this stage, we stopped our training accordingly.

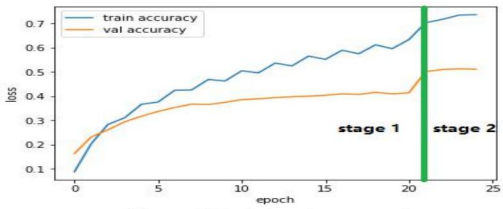

Figure 7 Training Accuracy Logs

5.3.2 More convolutional layers fine tuning with lower learning rate

Another 4 epochs were trained with more layers allowed for training and lower learning rates as well as higher image cropping probability.

Table 4 Hyper Parameters for Xception training -- stage 2

Base Model Name	Xception
Top-layer Model	Generalized mean pool - dropout – output(softmax)
Frozen Layers	Bottom 20 conv layers
Learning Rate	0.00001
Optimizer	Adam
Loss function	Categorical Cross Entropy
Cropping probability	0.5 to 0.8 linear increasing by epoch
Resizing scale	Resizing to 200*200 before cropping
Way of cropping	Random center position

5.3.3 Prediction through voting

Since we do not have resource to build up ensemble learning by having different model trained, we designed a voting mechanism using image cropping at test time.

We cropped the images 10 times and use majority win among the predicted labels as our final prediction. This yields better performance on hold-out set.The confidence score is then based on the weighted average probability of the majority label (i.e. the final prediction).

5.3.4 Final results (Xception model with best validation set accuracy)

The following table 5 summarizes the quantitative performance of the best Xception model trained. Please note that the GAP for training and validation set are not listed here to avoid confusion, since we calculated it batch by batch (each batch size is 64), which are not comparable with that calculated on test set.

Table 5 model performance

	Accuracy	GAP
Training set	0.7496	--
Validation set	0.5111	--
Test set	0.5159	0.4709
Test set with voting mechanism	0.5389	0.5964

5.4. Qualitative analysis

The model prediction accuracy is roughly centered around 50% with a shape of Normal distribution. It does not have significant correlation to the number of samples in the class.

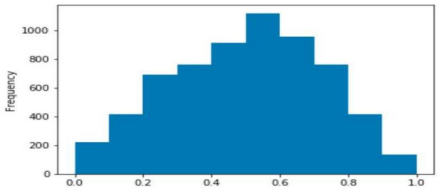

Figure 8 All classes' prediction accuracy histogram

This generally concludes that our trained classifier is doing a superb job (at least for this randomly-selected label).

5.4.1 Sample Class with 0 accuracy

This is fairly reasonable as the pictures do not even depict a landmark (noisy data)

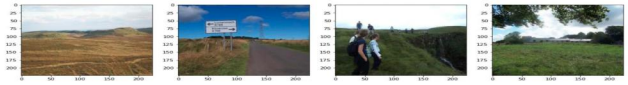

Figure 9 Sample images with 0 classification accuracy

5.4.2 Sample Class with 50% accuracy

There is an important observation from the correctly classified images -- the model is able to recognize indoor scene and outdoor landmark shape as the same class, which is a different problem nature compared with traditional image classification problem.

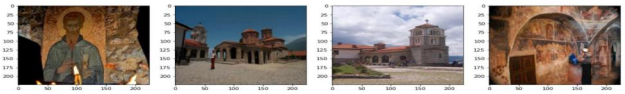

Figure 10 Correctly-classified sample images with 50% classification accuracy for this label

The misclassified images again do not depict any landmarks.

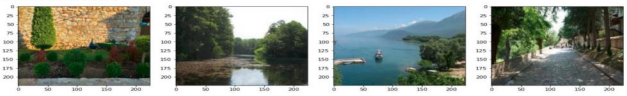

Figure 11 Misclassified sample images with 50% classification accuracy for this label

This generally concludes that our trained classifier is doing a superb job (at least for this randomly-selected label).

5.5. Saliency Map

As saliency refers to unique features (pixels, resolution etc.) of the image in the context of visual processing, these unique features depict the visually alluring locations in an image[13]. We want to compute the gradient of output category with respect to input image to represent the saliency at every location in the visual field by a scalar quantity and to guide the selection of attended locations based on the spatial distribution of saliency.

$$\frac{\partial output}{\partial input}$$

This tell us how the output value changes with respect to a small change in inputs. For a better visualization, we employed the method of guided saliency. In guided saliency, the backpropagation step is modified to only propagate positive gradients for positive activations[15]. The Saliency map generated from our trained model indicates that our classifier is looking at the shape of the construction (especially the unique parts) as well as the surrounding environments (slightly).

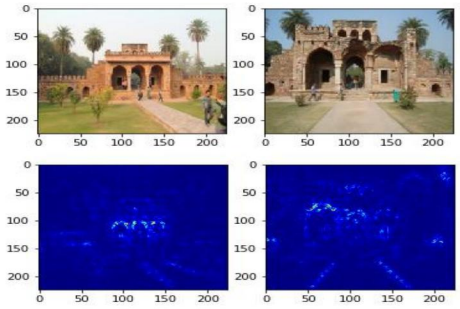

Figure 12 Saliency Map for two sample landmark images

6. Conclusion and future work

The quantitative and qualitative analysis of our trained model indicates a satisfactory performance on the 6k+ classes of landmarks. Data augmentation especially image cropping played a significant role in improving the accuracy. Also cropping at test time and use the majority win method gives better prediction accuracy, though it is heuristic.

However, once the problem is expanded to the original dataset with 200K+labels, it could be way more difficult to train a classifier using a pure deep learning architecture. Some hybrid method includes unsupervised learning such as KNN could be powerful. For example, we could try find the k nearest neighbors for a given test image based on the features from one bottleneck layer of some CNN architectures.

Some interesting method was not attempted yet due to the time constraint of this project. For example, a local feature descriptor for large-scale image retrieval called DeLF could be useful for this particular challenge. It extracts local features from images and matches them. We used it for matching local features of test images to images known to be landmarks. DeLF architecture is such that it selects the features with the highest score and then the query image is passed through and its features are matched with those of the database images after geometric verification. The matching of features is done through Ransac (Random Sample Consensus) and the number of inliers is used to make a decision. In order to tackle the no-landmark images in the Kaggle private test dataset, we can tune the threshold of the number of such inliers.

7. Contributions & Acknowledgements

Both members of the team were making significant contribution into this project including research, data downloading, cloud environment setup and report write-up.

Chenjiao Wang collected and researched related information, pre-processed the datasets, analyzed and visualized the data, built a pilot run pipeline, and conducted experiments by training baseline model with different hyperparameters, and also contributed to the improvement of the architecture, the model training supervision and the resource planning.

Renke Cai designed the main project pipeline, developed the major code base and training architecture for the Xception model using GPU and multi-threading, experimented different top-layer models, data augmentation mechanism, and prediction rules. He also conducted both the quantitative and qualitative analysis including the saliency map visualization and sample results analysis.

References

[1] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[2] C. McNabb, Et al. https://towardsdatascience.com/google-landmark-recognition-using-transfer-learning-dde35cc760e1

[3] Pedro Marcelino. https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[4] Felix Yu. https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

[5] Yunpeng Li, D. J. Crandall and D. P. Huttenlocher, "Landmark classification in large-scale image collections," 2009 IEEE 12th International Conference on Computer Vision, Kyoto, 2009, pp. 1957-1964.

[6] Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T. Chua, and H. Neven. Proceedings of International Conference on Computer Vision and Pattern Recognition, Miami, Florida, U.S.A, (June, 2009)

[7] O. M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, in: Proc. British Machine Vision Conference, Vol. 1, Swansea, UK, 2015, pp. 1–12.

[8] C. Lu and X. Tang. Surpassing human-level face verification performance on lfw with gaussianface. AAAI, 2015.

[9] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In Proc. ICCV, pages 1559–1566, 2011.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015

[11] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. arXivpreprint arXiv:1610.02357, 2016.

[12] Y. Zhu, X. Deng, and S. Newsam. 2018. Fine-Grained Land Use Classification at the City Scale Using Ground-Level Images. ArXiv e-prints (Feb. 2018). arXiv:cs.CV/1802.02668

[13] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. IEEE Patt. Anal. Mach. Intell., 20(11):1254–1259, November 1998.

[14] HC Shin, Et al. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, IEEE 2016.

[15] Springenberg, Et al. STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET, ICLR 2015.