Landmark Recognition

Sushanth Samala, Jingshuai Jiang
{ss12852, jj2903}@nyu.edu

Abstract: Landmark recognition is an instance recognition task and has its own set of challenges. The first and most critical of them is that it is impossible to provide a concrete concept of a landmark. Landmarks do not have a common structure and can be just about anything; buildings, monuments, cathedrals, parks, waterfalls, castles, etc, while some buildings are landmarks while others are not. There are many state of the art machine learning models for image recognition task but none for landmark recognition which is a harder problem to solve given that a good recognition system should be able to identify a landmark from different shooting angles from the inside and outside and it should also recognize if a building is a landmark or not. In this project, we used various deep learning models to fit the Google landmark recognition dataset v2. The main challenge lies in the fact that it's very complicated to give a precise definition of what is and what is not a landmark. Some buildings, statues, and natural objects are landmarks; others are not. Results show that our enhanced Inception model architecture out-performs and gives the best results out of all the existing models that have about the same number of trainable parameters. The enhanced Inception model has been deployed as a part of our landmark recognition solution on a Kubernetes cluster accessible for the next 30 days.

1 Introduction

Image retrieval and instance recognition are fundamental research topics that have been studied for decades. The task of image retrieval is to rank images in an index set w.r.t. their relevance to a query image. The task of instance recognition is to identify which specific instance of an object class (e.g. the instance "Mona Lisa" of the object class "painting") is shown in a query image.

As techniques for both tasks have evolved, approaches have become more robust and scalable and are starting to "solve" early datasets. Moreover, while increasingly large-scale classification datasets like ImageNet [1], COCO [2] and OpenImages [3] have established themselves as standard benchmarks, image retrieval is still commonly evaluated on very small datasets. For example, the original Oxford5k [4] and Paris6k [5] datasets that were released in 2007 and 2008, respectively, have only 55 query images of 11 instances each, but are still widely used today. Because both datasets only contain images from a single city, results may not generalize to larger-scale settings.

The motivation for this work is that most of the existing machine learning models have only been bench-marked on small landmark datasets, these datasets do not present real-world challenges. For instance, a landmark recognition system that is applied in a generic visual search app will be queried with a large fraction of non-landmark queries, like animals, plants, or products, which it is not expected to yield any results for. Yet, most instance recognition datasets have only "on-topic" queries and do not measure the false-positive rate on out-of-domain queries. Therefore, larger, more challenging datasets are necessary to fairly benchmark these techniques while providing enough challenges to motivate further research.

A possible reason that small-scale datasets have been the dominant benchmarks for a long time is that it is hard to collect instance-level labels at scale. Annotating millions of images with hundreds of thousands of fine-grained instance labels is not easy to achieve when using labeling services like Amazon Mechanical Turk, since taggers need expert knowledge of a very fine-grained domain[6].

We trained our models on the Google Landmarks Dataset v2 (GLDv2)[6], a new large-scale dataset for instance-level recognition and retrieval. GLDv2 includes over 5M images of over 200k human-made and natural landmarks that were contributed to Wikimedia Commons by local experts. Fig. 1 shows the geographical distribution of the landmarks present in the images. The dataset includes 4M labeled training images for the instance recognition task and 762k index images for the image retrieval task. The test set consists of 118k query images with ground truth labels for both tasks. While the Google Landmarks Dataset v2 focuses on the task of recognizing landmarks, approaches that solve the challenges it poses should readily transfer to other instance-level recognition tasks, like logo, product or artwork recognition.

In this paper we present a formulation in which we enhanced the famous Google Inception machine learning model architecture to perform better on landmark recognition and we also tuned the hyper-parameters of the existing famous deep learning models to better the performance on this new massive dataset. We compared the performance of various deep learning models on this task and our model's performance is the best among all others. Our primary contribution is the enhancement of the Inception model architecture and deployment of the model, its implementation has been deployed to production at scale and used for recognition on user photos.

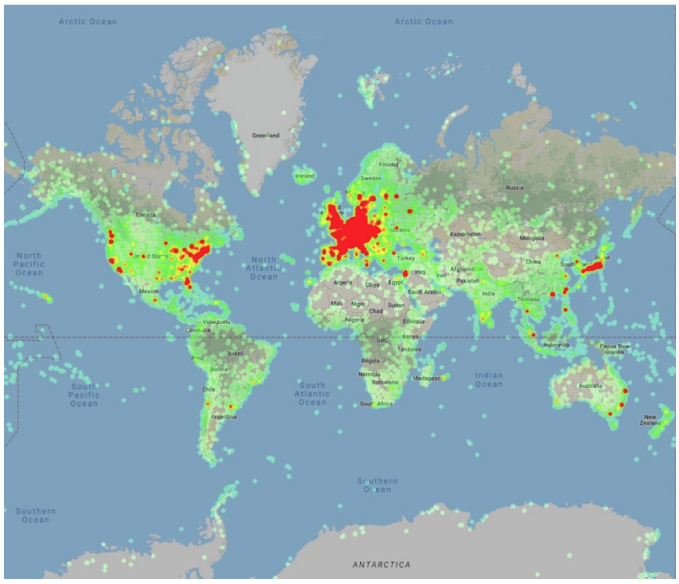

Figure 1. Heatmap of the places in the Google Landmarks Dataset v2[6].

2 Related Work

Image recognition problems range from basic categorization ("cat", "shoe", "building"), through fine-grained tasks involving distinction of species/models/styles ("Persian cat", "running shoes", "Roman Catholic church"), to instance-level recognition ("Oscar the cat", "Adidas Duramo 9", "NotreDame cathedral in Paris"). Our machine learning model focuses on tasks that are at the end of this continuum: identifying individual human-made and natural landmarks. In the following, we review image recognition and retrieval models, focusing mainly on those which are most related to our work.

Over the past two decades, there has been significant progress in the field of image retrieval, and the main approaches can be divided into two groups. The first one is classical retrieval approaches using local features, such as: methods based on local features descriptors organized in bag-of-words [7], spatial verification [8], Hamming embedding [9], query expansion [10]. These approaches dominated in image retrieval until the rise of deep convolutional neural networks (CNN) [11], which are used to produce global descriptors for an input image. According to [12], the pipeline of a typical local-features based method includes local features detection, descriptor extraction, quantization using a visual code-book and aggregation into embeddings.

Recognizing Landmarks[13] learn models for landmarks with a multi-class support vector machine, using classic vector-quantized interest point descriptors as features. They also incorporate the non-visual metadata available on modern photo-sharing sites, showing that textual tags and temporal constraints lead to significant improvements in classification rate. They also experiment with Single Image Classification Using Bag of Words Models to perform image classification adopting the bag-of-features model proposed by Csurka et al.[14], where each photo is represented by a feature vector recording occurrences of vector-quantized SIFT interest point descriptors [15]. As in that paper, they built a visual vocabulary by clustering SIFT descriptors from photos in the training set using the k-means algorithm.

Landmark Recognition via Deep Metric Learning[16] proposes a metric learning-based approach that successfully deals with existing challenges and efficiently handles a large number of landmarks. Deep Metric Learning method uses a deep neural network and requires a single pass inference that makes it fast to use in production. The basis of the method is the usage of embeddings of the deep convolutional neural network. This CNN is trained by curriculum learning technique with modified version of Center loss. To decide whether a landmark is present on a given image model computes distance between the image embedding vector and one or more centroids per class. For each landmark the centroids are calculated over clusters obtained from hierarchical clustering procedure.

Metric learning aims at finding appropriate similarity measure between image pairs that preserve desired distance structure. A good similarity can boost the performance of image search, especially when the number of classes is large or unknown [17]. Gordo et al. [6] use regional max-pooling and fine-tuning by Triplet loss [18]. Radenović et al. [12] employ Contrastive loss [19] and generalized mean-pooling, this approach shows good results, but brings additional memory and complexity cost [12].

DELF (DEep Local Features) by Noh et al. [20] demonstrates promising results. This method combines classical local features approach with deep learning. DELF uses features from one of the layers of the deep CNN and train an attention module on top of it. This allows us to extract local features from input images and then perform geometric verification with RANSAC [21]. As follows from the experiments described in [12], DELF and its extension [22] are the state-of-the-art approaches.

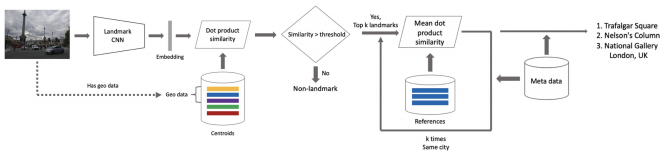

Figure 2. Overview of Deep Metric Learning System.[16]

3 Models

To create a state of the art landmark recognition model we need to have a good source of landmark images and we are using the recently released Google Landmark Recognition dataset v2 after looking at many different datasets which did not meet our requirements. The main problems with openly available image datasets like ImageNet are that the

number of images that contain landmarks is very low and by training our model on those won't give us good results. We used the following models to train on the Google Landmark Recognition dataset v2.

3.1 VGG

The first model we used is the VGG network from Visual Geometry Group[23]. It got a remarkable result of 93.2% top-5 accuracy, and 76.3% top-1 accuracy on the ImageNet test set. With a deeper network, gradient vanishing from multi-layers back-propagation becomes a bigger problem. This problem limits researchers to keep adding more layers, otherwise, the network will be really hard to converge.

During training, the input to VGG ConvNets is a fixed-size 224 x 224 RGB image. The image is passed through a stack of convolutional layers, where use filters with a very small receptive field: 3 x 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise 1 x 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 x 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolutional layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 x 2 pixel window, with stride 2.

A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification ReLU[24] non-linearity. We note that none of the networks (except for one) contain Local Response Normalisation (LRN) normalisation[24], such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of [24].

3.2 Residual Learning

Residual Learning Networks(ResNet)[25] explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning un-referenced functions. These residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8x deeper than VGG nets [26] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set.

For VGG network, the biggest hurdle of getting even deeper is the gradient vanishing issue, i.e, derivatives become smaller and smaller when back-propagate through deeper layers, and eventually reaches a point that modern computer architecture can't really represent meaningfully. GoogLeNet tried to attack this by using auxiliary supervision and asymmetric inception module, but it only alleviates the problem to a small extent. If we want to use 50 or even 100 layers, will there be a better way for the gradient to flow through the network? The answer from ResNet is to use a residual module.

ResNet added an identity shortcut to the output, so that each residual module can't at least predict whatever the input is, without getting lost in the wild. Even more important thing is, instead of hoping each layer fit directly to the desired feature mapping, the residual module tries to learn the difference between output and input, which makes the task much easier because the information gain needed is less. Imagine that you are learning mathematics, for each new problem, you are given a solution of a similar problem, so all you need to do is to extend this solution and try to make it work. This is much easier than thinking of a brand new solution for every problem you run into. Or as Newton said, we can stand on the shoulders of giants, and the identity input is that giant for the residual module[28].

3.3 EfficientNet

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. Inception model carefully balances network depth, width, and resolution that leads to better performance. Based on this observation, this new scaling method uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient and can be used in most the production systems today.

A drastic change in network structure usually only offers a little accuracy improvement. Even worse, when the same network applied to different datasets and tasks, previously claimed tricks don't seem to work, which led to critiques that whether those improvements are just over-fitting on the ImageNet dataset. On the other side, there's one trick that never fails our expectation: using higher resolution input, adding more channels for convolution layers, and adding more layers. Although very brutal force, it seems like there's a principled way to scale the network on demand. The EfficientNet baseline model architecture is shown in Table 1.

EfficientNet[28] uses the optimal building block to make sure a good foundation to start with. On the other hand, it defined three parameters alpha, beta, and rho to control the depth, width, and resolution of the network correspondingly. By doing so, even without a large GPU pool to search for an optimal structure, we can still rely on these principled parameters to tune the network based on their different requirements. In the end, EfficientNet gave 8 different variants with different width, depth, and resolution ratios and

got good performance for both small and big models.

EfficientNet changed the deep learning landscape by a lot, as the depth/width and other hyper-parameters are taken care of by the model we did not have much improvement to make on EfficientNet so we experiment by changing the architecture of Inception model.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 1. EfficientNet-B0 baseline network.

3.4 Inception

In the Inception module, multiple branches of different transformations are aggregated together to achieve a topology sparsity. The Inception v3 network architecture is progressively built, step-by-step.

Firstly, factorized convolutions are used to help reduce the computational efficiency as they reduce the number of parameters involved in a network. It also keeps a check on the network efficiency. Smaller convolutions replace bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5 x 5 filter has 25 parameters; two 3 x 3 filters replacing a 5 x 5 convolution has only 18 (3x3 + 3x3) parameters instead. In Figure 5, in the middle we see a 3x3 convolution, and below a fully-connected layer. Since both 3x3 convolutions can share weights among themselves, the number of computations can be reduced.

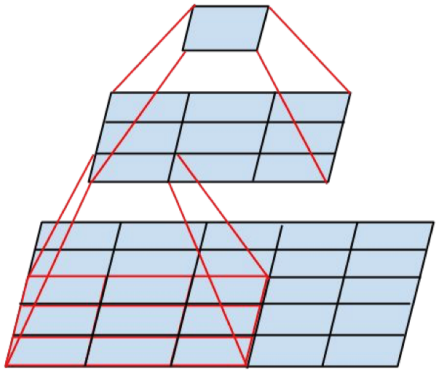

Figure 3. Mini-network replacing the 5 x 5 convolutions.

Inception module[29] uses asymmetric convolutions where a 3 x 3 convolution is replaced by a 1 x 3 convolution followed by a 3 x 1 convolution. If a 3 x 3 convolution is replaced by a 2 x 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed. Auxiliary classifiers are an important step in the Inception model architecture, an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.

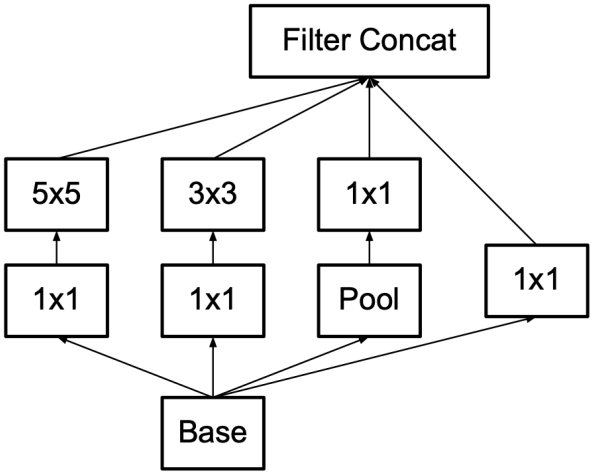

Figure 4. Original Inception module.

Finally, to reduce the grid size we use pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed in the original Inception paper.

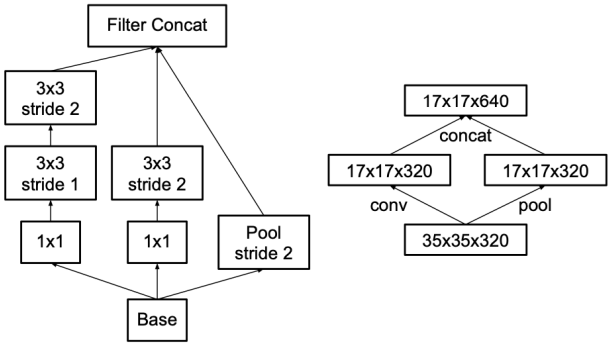

Figure 5. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

All of the above discussed techniques are combined to make the most optimal Inception model, we experimented by using many different architectures and figured out the best performing two architectures on our huge landmark dataset, the models are described in Figure 6, Figure 7,

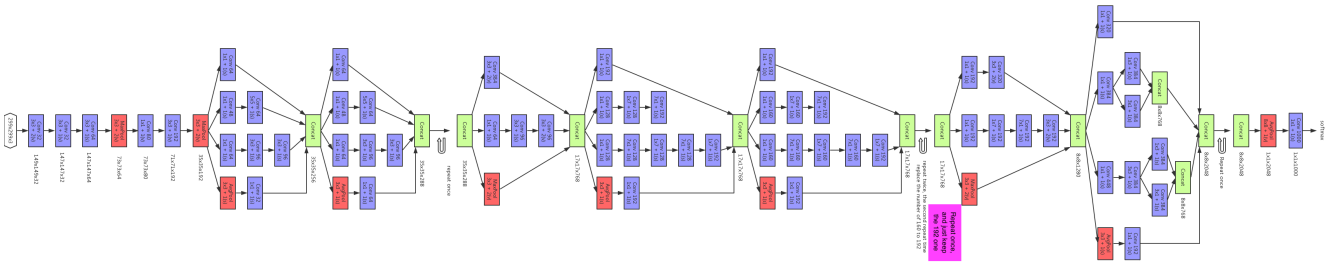

Figure 6. Our enhanced Inception model architecture(InceptionV200).

experimental details about our models are discussed in the next section. We changed the Inception module which is the basic unit for the Inception architecture and it is shown in Figure 7.

4 Experiments and Results

The main goal of these experiments is to figure out which set of hyper-parameters give the best results for the baseline architectures of VGG, ResNet, EfficientNet, Inception architectures. We ran the machine learning models on the Google Landmark Dataset v2, a new large-scale dataset for instance-level recognition and retrieval. GLDv2 includes over 5M images of over 200k human-made and natural landmarks that we recontributed to Wikimedia Commons by local experts. The dataset includes 4M labeled training images for the instance recognition task and 762k index images for the image retrieval task. The test set consists of 118k query images with ground truth labels for both tasks, but since the Google landmark recognition Kaggle competition is past submission deadline there was no way for us to get our submission scored so we used the dataset to split it into train and test set and used the test set for our metrics.

Implementation: Our implementation setup consists of 4 Million images allocated for all the models training and 1 Million images allocated to test the models performance. We experimented with different sets of hyper-parameters on each of the model, we are displaying the final and most optimal set of hyper-parameters for each model.

VGG: We trained VGG16 model on the training set with epochs = 50, batch size = 32, learning rate = 1e-3, optimizer = Adam. The model performed as expected and got an accuracy of 70.12% Top-1 and 88.21% Top-5.

ResNet50: We trained ResNet50 model on the training set with epochs = 50, batch size = 32, learning rate = 1e-3, optimizer = Adam. The model performed as expected and got an accuracy of 74.42% Top-1 and 87.24% Top-5.

EfficientNet: We trained EfficientNet-B0 architecture on the training set with epochs = 50, batch size = 64, learning rate = 1e-3, optimizer = Adam, weight decay = 1e-4. The model performed as expected and got an accuracy of 80.54% Top-1 and 90.27% Top-5. The model was very resilient to changes in the hyper parameters compared to the other models, a small change in any of the hyper parameters other than the learning rate did not have a significant reduction in performance of the model. The model has about 11M trainable parameters.

Inception: We trained baseline InceptionV3 model on the training set with epochs = 10, batch size = 64, learning rate = 1e-3, optimizer = Adam. The model performed as expected and got an accuracy of 84.28% Top-1 and 95.39% Top-5. Our enhanced InceptionV100 model shown in Figure 7 trained for epochs = 10, batch size = 64, learning rate = 1e-3, optimizer = Adam got a Top-1 accuracy is 81.84 and Top-5 is 92.14. Our enhanced InceptionV200 model shown in Figure 6 trained for epochs = 10, batch size = 64, learning rate = 1e-3, optimizer = Adam out-performed the InceptionV3 model and InceptionV100 and after training for 50 epochs the Top-1 accuracy is 90.59 and Top-5 is 96.20.

*InceptionV100, InceptionV200 are our enhanced models

Model	Epochs	Top-1(%)	Top-5(%)
VGG16	50	70.12	88.21
ResNet50	50	74.42	87.24
EfficientNet-B0	50	80.54	90.27
InceptionV3	10	84.28	95.39
InceptionV100	10	81.84	92.14
InceptionV200	10	86.72	95.12
InceptionV200	50	90.59	96.20

Table 2. Experimentation Results.

Model	Accuracy(%)
Baseline RANSAC	48.5
Pretrained DELG	49.07
EfficientNet	54.32
Unknown-Best-Kaggle-Result	68.38

Table 3. Kaggle Competition Peer Results for Comparison.

The scores shown in Table 3 are from the Google Landmark Recognition Kaggle competition leaderboard, the best performing model's score is 68.38% which secured the number 1 spot on the competition leaderboard, but the team did not mention which model they used. The other models got a score around 50% as you can see from the table. Our models significantly out-performed the Kaggle competition peers. However, the point to be noted here is that the accuracy score for peers are based on the test data provided in Kaggle,

but we could not submit our model to get the test accuracy from Kaggle as the competition has ended, we got our scores by splitting the train set into train and validation set. We are very confident that our models will out-perform all the other models on the test set. We deployed our model and is accessible at "http://169.57.85.66:30010" for the next 30 days.

5 Discussion

Following the famous AlexNet's[30] design, the VGG network has two major updates: 1) VGG not only used a wider network like AlexNet but also deeper. VGG-19 has 19 convolution layers, compared with 5 from AlexNet. 2) VGG also demonstrated that a few small 3x3 convolution filters can replace a single 7x7 or even 11x11 filters from AlexNet, achieve better performance while reducing the computation cost. Because of this elegant design, VGG also became the back-bone network of many pioneering networks in other computer vision tasks, such as FCN for semantic segmentation, and Faster R-CNN for object detection.

With a deeper network, gradient vanishing from multi-layers back-propagation becomes a bigger problem. To deal with it, VGG also discussed the importance of pre-training and weight initialization. This problem we faced while fine-tuning VGG and to keep adding more layers, is that the network got really hard to converge. The performance of VGG16 was as expected and got a Top-1 accuracy of 70.12% and a Top-5 accuracy of 88.21% after 50 epochs.

As we discussed earlier for VGG network, the biggest hurdle of getting even deeper is the gradient vanishing issue, i.e, derivatives become smaller and smaller when back-propagate through deeper layers, and eventually reaches a point that modern computer architecture can't really represent meaningfully. If we want to use 50 or even 100 layers, will there be a better way for the gradient to flow through the network; The answer from ResNet is to use a residual module.

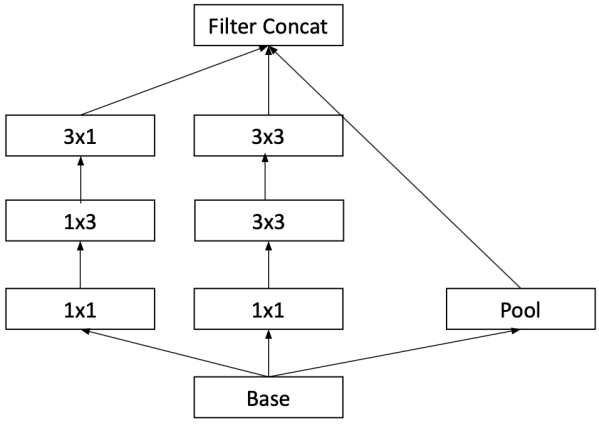

Figure 7. Our Inception module architecture (InceptionV100).

ResNet added an identity shortcut to the output, so that each residual module can't at least predict whatever the input is, without getting lost in the wild. Even more important thing is, instead of hoping each layer fit directly to the desired feature mapping, the residual module tries to learn the difference between output and input, which makes the task much easier because the information gain needed is less[27]. Imagine that you are learning mathematics, for each new problem, you are given a solution of a similar problem, so all you need to do is to extend this solution and try to make it work. This is much easier than thinking of a brand new solution for every problem you run into.

In addition to identity mapping, ResNet also borrowed the bottleneck and Batch Normalization from Inception networks. Eventually, it managed to build a network with 152 convolution layers and achieved 74.42% Top-1 accuracy and 87.24% Top-5 accuracy on Google Landmark Dataset v2. Thanks to ResNet's simple and beautiful design, it's still widely used in many production visual recognition systems nowadays.

A drastic change in network structure usually only offers a little accuracy improvement. Even worse, when the same network applied to different datasets and tasks, previously claimed tricks don't seem to work, which led to critiques that whether those improvements are just over-fitting on the ImageNet dataset. On the other side, there's one trick that never fails our expectation: using higher resolution input, adding more channels for convolution layers, and adding more layers. Although very brutal force, it seems like there's a principled way to scale the network on demand.

Researchers realized that even with the help from a computer, a change in architecture doesn't yield that much benefit. So they start to fall back to the scaling the network. EfficientNet is just built on top of this assumption. On one hand, it uses the optimal building block from mNASNet[31] to make sure a good foundation to start with. On the other hand, it defined three parameters alpha, beta, and rho to control the depth, width, and resolution of the network correspondingly. By doing so, even without a large GPU pool to search for an optimal structure, engineers can still rely on these principled parameters to tune the network based on their different requirements. In the end, EfficientNet gave 8 different variants with different width, depth, and resolution ratios and got good performance for both small and big models. EfficientNet-B0 got an accuracy of 80.54% Top-1 and 90.27% Top-5 accuracy. We did not have much room to improve on the EfficientNet architecture as it is already highly tuned.

One of the main contributions of Inception model is to push the limit of the network depth with a 22 layers structure. This demonstrated again that going deeper and wider is indeed the right direction to improve accuracy. Unlike VGG16, Inception tried to address the computation and gradient diminishing issues head-on, instead of proposing a workaround with better pre-trained schema and weights

initialization.

First, it explored the idea of asymmetric network design by using a module called Inception. Ideally, the authors of the Inception model pursued sparse convolution or dense layers to improve feature efficiency, but modern hardware design wasn't tailored to this case. So they believed that a sparsity at the network topology level could also help the fusion of features while leveraging existing hardware capabilities.

Second, it attacks the high computation cost problem by borrowing an idea from Network in Network[32] paper. Basically, a 1x1 convolution filter is introduced to reduce dimensions of features before going through heavy computing operation like a 5x5 convolution kernel. This structure is called "Bottleneck" later and widely used in many following networks. Similar to Network in Network[32], it also used an average pooling layer to replace the final fully connected layer to further reduce cost.

Third, to help gradients to flow to deeper layers, Inception model also used supervision on some intermediate layer outputs or auxiliary output. We modified the Inception model architecture to fine-tune it for the Google Landmark Recognition Dataset v2, we changed the Inception module architecture to get tackle the over-fitting issue. Our InceptionV200 model achieved 84.28% Top-1 accuracy and 95.39% Top-5 accuracy after running for just 10 epochs, this already outperforms all the other models that ran for 50 epochs, after 50 epochs our InceptionV200 architecture achieved 90.59% Top-1 accuracy and 96.20% Top-5 accuracy.

6 Conclusion and Future Work

In this paper, we demonstrated how various deep learning architectures that were mainly designed to perform generic image recognition can be fine-tuned to solve the landmark recognition task and also showed improvement over the prior work done on the InceptionV3 architecture by incorporating an enhanced Inception module unit. Our enhanced Inception model out-performed all the other models deep learning models presented in this paper by a substantial margin even when trained for just 10 epochs compared to 50 epochs trained model results. The above results provide an interesting demonstration of the capacities of deep learning methods on the landmark recognition task. The advantage of these methods is that they can easily be tuned for any image related tasks by making a few changes based on the task at hand and the kind of data we have available. Further research could make use of more recent deep learning model architectures and fine tune them to fit the Google Landmark Recognition Dataset v2. Finally, most critical follow-up work is to make our model more robust by using images from many different angles of the landmarks, we do have a few images of landmarks from different angles, but there is no dataset available at the moment that offers a large set of images from different angles of landmarks.

7 References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[2] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dolla r, and C. Zitnick. Microsoft COCO: common objects in context. In Proc. ECCV, 2014.

[3] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. arXiv:1811.00982, 2018.

[4] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In Proc. CVPR, 2007.

[5] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In Proc. CVPR, 2008.

[6] Tobias Weyand, Andre Arauj, Jack Sim, Bingyi Caoo.Google Landmarks Dataset v2:A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. rXiv:2004.01804v2, Nov 2020.

[7] Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In null. IEEE, 1470.

[8] James Philbin, Ondrej Chum,Michael Isard,Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 1–8.

[9] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2008. Hamming embedding and weak geometric consistency for large scale image search. In European conference on computer vision. Springer, 304–317.

[10] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. 2007. Total recall: Automatic query expansion with a generative feature model for object retrieval. In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE, 1–8.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey EHinton.2012.Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 1097–1105.

[12] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. 2018. Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. arXiv preprint arXiv:1803.11285 (2018).

[13] David J. Crandall, Yunpeng Li, Stefan Lee, and Daniel P. Huttenlocher. Recognizing Landmarks in Large-Scale Social Image Collections

[14] Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.:Visual Categorization with Bags of Key points. In: ECCV Workshop on Statistical Learning in Computer Vision (2004)

[15] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
[16] Andrei Boiarov, Eduard Tyantov. Large Scale Landmark Recognition via Deep Metric Learning. arXiv:1908.10192v3 2019.
[17] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In Advances in Neural Information Processing Systems. 730–738.
[18] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition. 815–823.
[19] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 2. IEEE, 1735–1742.
[20] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. 2017. Largescale image retrieval with attentive deep local features. In Proceedings of the IEEE International Conference on Computer Vision. 3456–3465.
[21] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition. 815–823.
[22] Marvin Teichmann, Andre Araujo, Menglong Zhu, and Jack Sim. 2018. Detect-to-Retrieve: Efficient Regional Aggregation for Image Search. arXiv preprint arXiv:1812.01584 (2018).
[23] Karen Simonyan Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 2019
[24] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.
[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385 2015
[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
[27] Ethan Yanjia Li. Understand Image Classification in the Deep Learning Era. Towards Data science 2020
[28] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946v5 2020
[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567v3 2015
[30] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM 2017.
[31] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. arXiv:1707.07012 2018
[32] Min Lin, Qiang Chen, Shuicheng Yan. Network In Network. arXiv:1312.4400 2014.