CONVOLUTIONAL NEURAL NETWORK BASED APPROACH FOR LANDMARK RECOGNITION

Aditya Surana, Harshit Mathur
Jaipur Engineering College and Research Centre, Jaipur, Rajasthan, India.
Email: to.adityasurana@gmail.com, harshitmathur10@gmail.com

ABSTRACT

In recent years, the world has witnessed a tremendous increase in digital cameras and mobile devices which has led to an even greater increase in the number of photographs clicked every year on average. Tourists, among others, contribute a considerable share to this effect with 1.323 billion international tourist arrivals worldwide in 2017 alone. With this insurgence in the amount of photographs clicked, arises the problem of recognition. A person might wonder what that one landmark was that he once visited. Landmark recognition technology helps people better understand and organize their photographs by predicting a landmark label directly from image pixels. A possible solution for landmark recognition has been presented by application of deep learning techniques. More specifically, convolutional neural networks have been used for detection of these images, with the help of open source pre-trained models. A person submitting a snap of a landmark can get relevant information regarding the landmark according to the match found from the millions of images present in the database.

KEYWORDS: *Artificial Intelligence, Computer Vision, Convolutional Neural Networks, Data Science, Deep Learning, Image Classification, Landmark Recognition, Transfer Learning.*

I. INTRODUCTION

The Google landmark recognition challenge involves building appropriate models that can classify the various landmark images provided to it in such a way that it will correctly match each unique input image with the actual landmark label.

Among the many challenges faced was the sheer size of the dataset; containing a massive amount of classes at 14,916. Furthermore, each individual class may or may not contain a significant amount of images for training purposes. The test set provided also contained many images that were not actually landmarks. The total number of images being well over 1.2 million asked for a lot of patience and time to download the images and have them sorted according to their unique classes. Many URLs being broken and some images being corrupt only added to the challenges faced. Next, the dataset had to be split into three separate folders namely train, val, and test, for training, validation, and testing respectively as holdout sets for testing the accuracy of the model.

Some previous solutions applied by others were based on significantly smaller number of classes, i.e., lesser number of unique landmark labels which only displayed good results for a narrow portion of the entire dataset.

Our approach aims to provide a robust solution based on 14,916 classes while being comparatively less complex in terms of architecture and implementation. We applied transfer learning by using various pre-trained models such as VGG16[5], InceptionResnetV2[8], Xception[6], MobileNet[12] and Resnet50[9]. Our approach with every pre-trained model involved freezing the convolution base and not including the top fully connected layers of the architecture. Next, we added our own fully connected layers over the convolution base to form the final architecture. Finally, the model was trained and the hyper-parameters were subsequently adjusted to achieve the desired accuracy.

II. RELATED WORK

One of the previous approaches [16] used the VGG16 [5] pre-trained model on 2000 landmark classes using sub-sampling method. For the architecture, they used the bottleneck layers of the VGG16 [5] and only trained the top three fully-connected layers. The images to be downloaded were first resized to a smaller size to make the download feasible, and split into holdout sets before downloading, i.e., each set contained only URLs and not images. Afterwards, the images were downloaded according to their sets and their classes by individually fetching the images from their URLs through a python script. Ultimately, the model was able to achieve an accuracy of 71%. Although the previous approach [16] managed to achieve a decent accuracy with minimal effort, the result was obtained on a significantly smaller number of classes than the complete dataset which makes it difficult to pinpoint the performance of the model over larger datasets which is usually observed in real life scenarios.
Our approach aimed at providing a solution for the entirely of the dataset instead of only a subset. This approach took into consideration how datasets usually are for real life scenarios. Instead of only freezing the bottleneck layers and training the top three fully-connected layers, we froze the convolutional base of the VGG16 [5] model and added another convolutional layer on top of the bottleneck layers for improved feature extraction before adding our fully-connected layers on top of it to form the complete architecture.
The same paradigm was followed for various other pre-trained models such as InceptionResnetV2 [8], Xception [6], MobileNet [12] and Resnet50[9]. We observed that although the accuracy would improve slightly for some of the other pre-trained models, the validation accuracy would fluctuate wildly and the inference scores would be less than desired, which led us to eventually consider VGG16 [5] for the task.

III. METHODOLOGY

A. Obtaining dataset

The dataset was in the form of a CSV file which contained three columns namely id, URL, and landmark_id. The id column contained id for individual images, URL column contained the address for the individual images, and landmark_id contained the class label for that image. The total number of images was over 1.2 million. The images were downloaded through a python script that downloaded the images according to their class labels and sorted them accordingly by creating separate folders for each class label.

Fig. 1. Sample landmark image

B. Pre-processing

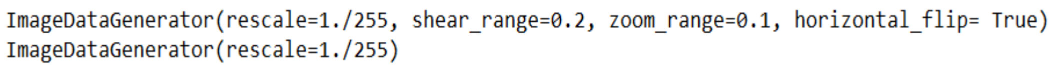
The downloaded images had to be split into three separate sets as holdout sets for the purpose of training, validation, and testing respectively. The split_folders package in python was used to accomplish this with a ratio of 80:10:10 for the three sets.

Moreover, several images could not be downloaded due to copyright reasons since the dataset was prepared by crowdsurfing method. Thus, many labels were missing in the final dataset. Therefore, empty folders had to be created for these labels.

C. Image Augmentation

To account for variations in our data to better mimic real life image queries we applied several image augmentation techniques such as shear, zoom, and horizontal flip.

Fig. 2. Sample landmark image



```
ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.1, horizontal_flip= True)
ImageDataGenerator(rescale=1./255)
```

Fig. 3. This figure shows various data augmentation techniques applied

D. Model architecture

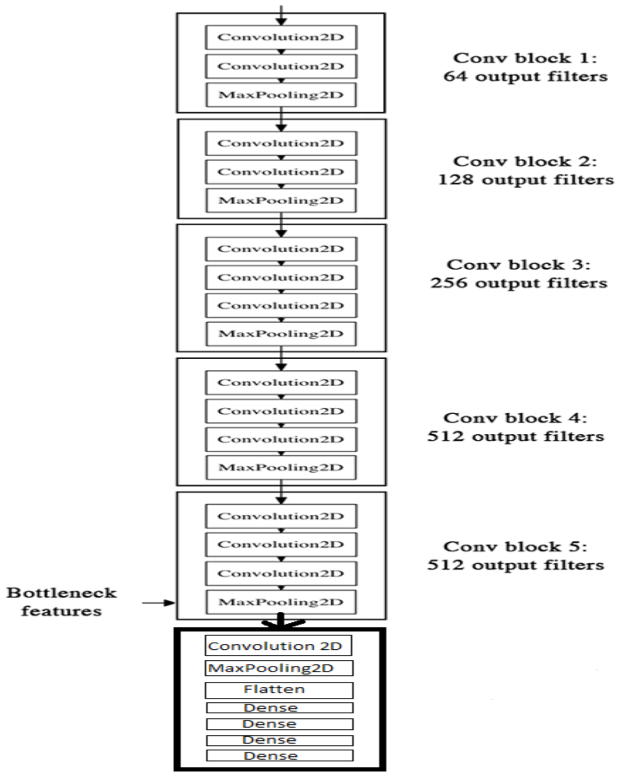


Fig. 4. Model architecture

The VGG16[5] pre-trained model was used for applying transfer learning in our approach. The first 16 layers of the model were frozen and the top layers were not included. Afterwards, four fully-connected layers were added on top of the convolution layers to form the final model architecture.
During the initial runs of our model, we observed low training accuracy and slight underfitting which indicated that our model needed better feature extraction. As a result, we added
another convolution layer of our own in between the VGG16[5] layers and the four fully connected layers followed by a MaxPooling layer and a Flatten layer, which was then fed into the fully-connected layers.
The fully-connected layers consisted of three layers of decreasing units of 128, 64, and 32 with activation ReLU leading to the final output layer consisting of 14916 units with activation as Softmax.

E. Hyper-parameters

The batch size for training was set to 500 as a lower batch size led to a poor accuracy. The batch size for validation was set to 200. Since the activation function used in the last layer was softmax, therefore the loss function used was categorical cross-entropy to assist multi-class classification.

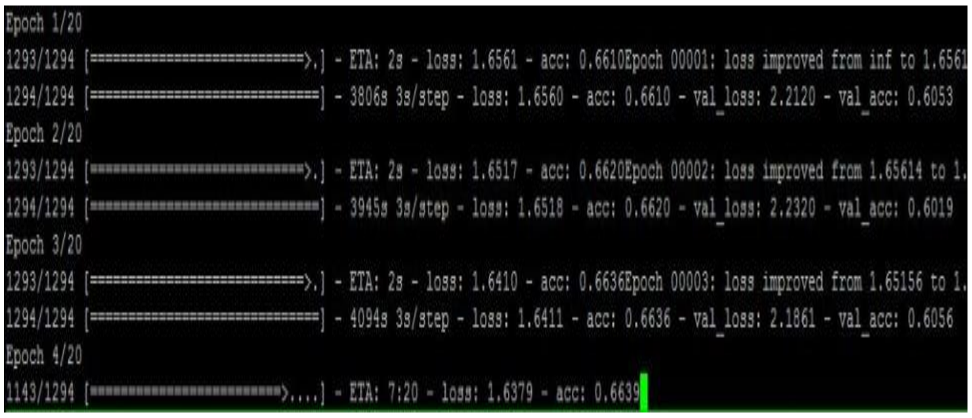
Table- I:. Hyperparameter values

Hyperparameter	**Value**
Batch size (Train)	500
Batch size (Validation)	200
Optimizer	Adam
Learning Rate	0.001
Epochs	60
Loss function	Categorical Cross-Entropy

IV. Experimental Results

A. Training

The model was run for two sessions, the first training session was run for 40 epochs at the end of which the model achieved an accuracy of 63%. We noticed that the loss for still decreasing up till the last epoch thus the accuracy could be improved in further epochs.
Thus, the second session was run for 20 epochs to improve the accuracy of the model. The model was able to achieve an accuracy of 67.4% on a total of 14916 classes during the second training run of the model.



```
Epoch 1/20
1293/1294 [==========================>.] - ETA: 2s - loss: 1.6561 - acc: 0.6610Epoch 00001: loss improved from inf to 1.6561
1294/1294 [============================] - 3806s 3s/step - loss: 1.6560 - acc: 0.6610 - val_loss: 2.2120 - val_acc: 0.6053
Epoch 2/20
1293/1294 [==========================>.] - ETA: 2s - loss: 1.6517 - acc: 0.6620Epoch 00002: loss improved from 1.65614 to 1.
1294/1294 [============================] - 3945s 3s/step - loss: 1.6518 - acc: 0.6620 - val_loss: 2.2320 - val_acc: 0.6019
Epoch 3/20
1293/1294 [==========================>.] - ETA: 2s - loss: 1.6410 - acc: 0.6636Epoch 00003: loss improved from 1.65156 to 1.
1294/1294 [============================] - 4094s 3s/step - loss: 1.6411 - acc: 0.6636 - val_loss: 2.1861 - val_acc: 0.6056
Epoch 4/20
1143/1294 [=======================>....] - ETA: 7:20 - loss: 1.6379 - acc: 0.6639
```

Fig. 5. Training

B. Results

Table- II: Pre-trained model comparison

PRE-TRAINED MODEL	**TRAINING ACCURACY (2000 CLASSES)**
VGG16[5]	92.2%
MobileNet[12]	92%
ResNet50[9]	87%

InceptionResnetV2[8]	90%
Xception[6]	89%

Both VGG16[5] and MobileNet[12] gave quite promising results but the inference scores of VGG16[5] turned out be better than that of MobileNET[12]. Therefore, VGG16[5] was chosen for the final model architecture to be run on 14916 classes.

Table- III: VGG16 final model

Train accuracy (14916 classes)	67.4%
Validation accuracy (14916 classes)	60.76%
Pre-trained model giving most promising result	VGG16[5]
Time for final model training	57 hours

Table- IV: Comparison with previous approach [16]

Parameter	**Previous approach**[16]	**Our approach**
Number of classes	2000	14916
Train accuracy	71%	67.4%
Validation accuracy	78.6%	60.76%

V. CONCLUSION

A. Summary

The problem was solved using deep learning techniques, particularly CNN coupled with transfer learning to achieve desirable accuracy. The best pre-trained for this problem turned out be VGG16[5] which was used by freezing the convolution base and training the fully connected layers. The hyper-parameters were fine-tuned after the initial training run to better optimize the model and improve model performance.

B. Future Work

We aim to apply Deep Local Features (DeLF) to our trained model to make the model learn to distinguish between non-landmark and landmark images as the test set obtained largely consisted of non-landmark images. DeLF further lets us improve model performance by not assigning labels to non-landmark images and only perform predictions if an image contains a landmark.
Further, the model can be run for even more epochs as the loss was observed to still be decreasing even after 60 epochs.

ACKNOWLEDGEMENT

We would like to thank Dr. Sanjay Gaur, a researcher, and professor at Jaipur Engineering College and Research Centre, for guiding us throughout the research.

REFERENCE

[1]. Catherine McNabb, Anuraag Mohile, Avani Sharma, Evan David, Anisha Garg, “Google Landmark Recognition using Transfer Learning”, towards Data Science, 2018, available online at: https://towardsdatascience.com/google-landmark-recognition-using-transfer-learning-dde35cc760e1
[2]. Abhinaya Ananthakrishnan, Mark Babbe, Lining Jiang and Kimmy Zhuo, “A Data Geek’s guide to recognize landmarks”, Medium, 2018 available online at: https://medium.com/@abhinaya08/google-landmark-recognition-274aab3c71ae
[3]. Jason Brownlee, “How to Configure Learning Rate Hyperparameter when Training Deep Learning Neural Networks”, Deep Learning Performance, 2019 available online at : https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/
[4]. Keras Documentation available online at : https://keras.io/
[5]. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", Proc. Int. Conf. Learn. Representations, 2015.

[6]. F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.
[7]. C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovic., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.
[8]. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.
[9]. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
[10]. He K., Zhang X., Ren S., Sun J. (2016) Identity Mappings in Deep Residual Networks. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_38
[11]. Adrian Rosebrock, "ImageNet: VGGNet, ResNet, Inception, and Xception with Keras", available at: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
[12]. Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861.
[13]. Ferhat Culfaz, "Transfer Learning using MobileNet and Keras", Towards Data Science, available at: https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299
[14]. Raúl Gómez Bruballa, "Understanding Categorical Cross-Entropy Loss", available online at: https://gombru.github.io/2018/05/23/cross_entropy_loss/
[15]. Fast resized image download (Python 3), available at: https://www.kaggle.com/lyakaap/fast-resized-image-download-python-3
[16]. Anisha Garg, "Google-Landmark-Recognition", available at: https://github.com/anishagg/Google-Landmark-Recognition/tree/master/Scripts

AUTHORS PROFILE

Aditya Surana currently pursuing bachelor's in technology in computer science from Jaipur Engineering College and Research Centre, Jaipur. His research areas include Machine Learning, Data Science and Deep Learning.

Harshit Mathur currently pursuing bachelor's in technology in computer science from Jaipur Engineering College and Research Centre, Jaipur. His research areas include Machine Learning, Data Science and Deep Learning.