Image-based geographical localization using transfer learning and convolutional neural networks

Rishab Paruchuri

1. Introduction

Predicting the geographical location of an image taken using only the image's pixels is known as image-based geolocation. Performing this task without any prior information is very challenging due to the fact photos taken from all over the earth depict a large amount of variation, e.g., different times, objects, or camera positions. Consequently, this project aims to answer the question of if a computer can associate the visual clues of an image with the location of where it was taken. With the rise of GPS technologies, creating a model designed to solve this task may seem redundant. However, the task still has many contemporary applications—namely geopositioning, geotagging, and geocoding—all of which will only become more apparent in our current technological culture.

This project treats geolocation as a classification problem, using images from ten of the most famous cities on the planet to create a model that achieves a superhuman level of accuracy. By simplifying the problem and considering only ten distinct cites, the task becomes significantly easier but still remains challenging nonetheless. Images frequently contain relevant indicators such as landmarks, weather patterns, flora, road markers, and architectural characteristics, which when combined can allow one to identify an approximate location. The approach taken in this project is to use transfer learning to train a convolutional neural network, more specifically the

VGG16 model, using a database of approximately 40,000 images from Flickr (a public image hosting website). The aim of the project was to build a model that could classify the correct city from an image more than half the time. After training the model, it was able to achieve an accuracy of 61% on the testing dataset, confirming the usefulness of convolutional neural networks in this task. This paper is organized as follows:

2. Related Work

The task of image-based geolocation is notoriously difficult and researchers have used many different techniques in their efforts to solve the problem. Attempts to solve the geo-location problem in the past have mostly focused on feature retrieval. Im2GPS (Hays, J., & Efros, A. A. 2008), which compares an image to similar photographs from a collection of millions of Flickr photos, was the most noteworthy implementation of this technology. This approach, which uses similarity scores, may achieve satisfactory geo-location accuracy, categorizing around 25% of photographs with country-level accuracy.

S. Belongie and J. Hays (2015, June 1) undertook work leveraging aerial images for geolocalization in an attempt to overcome the problem of scant data in rural regions. A Siamese CNN is used in this study to match ground pictures to aerial shots with known locations. In specific cities, the model was able to localize more than 22% of queries.

Until recently, relatively little prior research has applied CNNs to this task. Hong and Peddada (2015) attempted this task with a subset of the CRCV dataset containing 62,058 high-quality Google Street View images. The photos depict the downtown and surrounding districts of Pittsburgh, PA; Orlando, FL; and, to a lesser extent, Manhattan. The researchers were able to achieve an astounding 97.5% accuracy performance in coarsely labeling cities in the CRCV dataset with a CNN trained from scratch. However, only three cities were used to train and test the model, significantly reducing the complexity of the task.

A recent Google paper (Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., & Xu, C., 2020) uses CNN for a similar geo-localization assignment. They created PlaNet, a network that categorizes pictures into variously sized areas throughout the world. They utilized a Flickr dataset that comprised geo-tagged photographs with a wide range of topic matter. As a result, they attain very high accuracy on photographs of landmarks or cultural subjects, which are more likely to be the topic of a Flickr shot. They achieve cutting-edge accuracy in global geo-localization, classifying images with city-level accuracy 25% of the time.

Hershey and Wulfe's study (2016) used a GoogleNet model that had been pre-trained on the MIT Places205 dataset to train a classification model using a dataset of 100,000 Google Street View photos. Photographs were made by filtering images from random places within 25 kilometers of 10 distinct city centers. As a result, their model didn't have to deal with many images containing little to no architectural features. On their testing dataset, their network, LittlePlaNet, achieved an accuracy of 75%. However, when tested on a Flickr dataset, LittlePlaNet only obtained a 17 percent accuracy.

3. Dataset

I utilized the Flickr API to build and download the dataset used in this project. When compared to many other geolocation models, this strategy is rather widespread. Google Street View photos are also a frequently utilized alternative for this task, but due to the costs associated with using their API, I decided against using them. The Flickr image database, with over 5 billion photographs and many of them including useful metadata like tags, geolocation, and Exif data, was an ideal instrument for constructing the geolocation dataset. The data set was created using 41,000 photos in total, with approximately 4000 photographs from each of the 10 cities. Because of the easy availability of high-quality photographs, the following 10 cities were chosen as the different categories for the image geolocation problem: Paris, Tokyo, London, Bangkok, New York, Amsterdam, Mumbai, Barcelona, Venice, and Los Angeles.

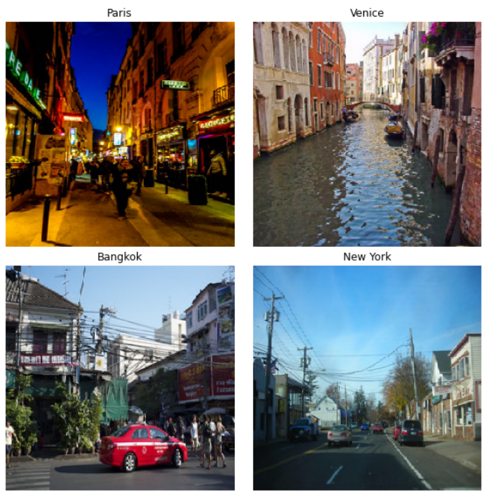

Figure 1. Randomized Street Images from Four of the Ten Cities

To ensure that images were not only of monuments, landmarks, and other distinctive attractions, the Flickr images were taken by including the key search word "streets" when searching for the city photos. The images ranged in size and quality and hence were standardized by downloading the Small 320 size: “url_n: Small 320 (320 × 229)”. Higher resolution images could have possibly led to better results as they may have included finer features such as textual elements and other intricate objects. At training and testing time, all the image pixel values were scaled from 0 to 1, and the images were rescaled to 224 x 224 pixels, which is the input size specified by the VGG16 model used.

4. Background

4.1 Convolutional Neural Networks

A convolutional neural network is a type of artificial neural network that is often used in deep learning to assess and solve problems involving visual images. CNNs may frequently achieve high accuracy when dealing with spatial information, making them crucial when considering picture classification challenges. The CNN used as this project’s model (VGG16) has three components: convolutional layers, pooling layers, and fully connected layers (Albawi, S., Mohammed, T. A., & Al-Zawi, S. 2017).

4.1.1 Convolutional Layer

The convolutional layer entail a set of kernels that will be learned throughout the training phase. Convolutional layers are useful for extracting features from images because they cope with spatial redundancy through weight sharing. Redundancy is minimized as the input is routed across the network, and the characteristics become more exclusive and informative. This is mostly due to cascading convolutions that occur repeatedly. This is accomplished by running a kernel across the input and generating a feature map. As redundancy is eliminated, a compressed feature representation of the image's content is produced.

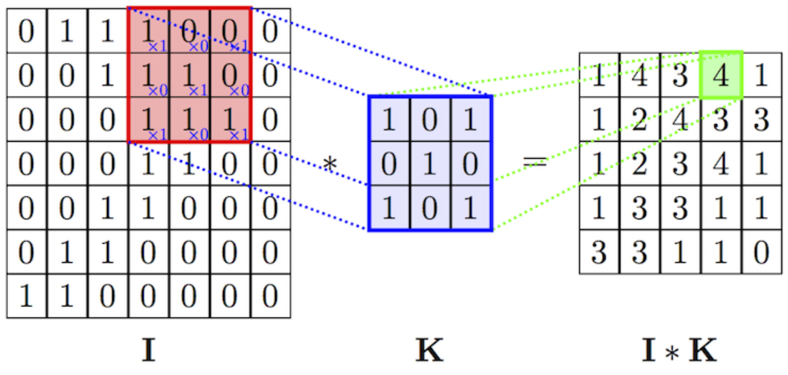

Figure 2. An example of convolution operation in 2D (Mohamed, 2017)

The convolutional layer works by sliding a defined two-dimensional matrix across the input and performing a convolutional arithmetic operation. Figure 2 shows an example of the convolution operation (Albawi, S., Mohammed, T. A., & Al-Zawi, S. 2017).

4.1.2 Pooling Layer

CNNs employ pooling layers to consolidate the features learned by the convolutional layer feature map. Compressing and generalizing the features in the feature map reduces overfitting during model training. This eliminates the requirement to train the model on precisely positioned elements, making it more robust and resilient. Pooling layers are relatively straightforward since they commonly employ maximum or average input values to downsample the data (Albawi, S., Mohammed, T. A., & Al-Zawi, S. 2017).

4.1.3 Fully Connected Layers

Fully connected layers, or Dense layers, are the last layers in the convolutional neural network. The fully connected layers work similarly to how neurons would work in traditional neural networks. Each node in a fully connected layer is connected to every node in the previous and next layer. The values in the layer are computed by doing a weighted sum of all the values of the previous layer. After this, the output is passed through the nonlinear activation function. In most cases the ReLu activation function is used as defined below:

$$f(x) = max(0, x)$$

In contrast to the other side product produced by the convolutional kernel, this transform eliminates the negative component of the input, resulting in a clearer contrast of important features. In order to predict a multinomial probability distribution, the last fully connected layer of the network uses the softmax activation function. This layer also contains the same number of nodes as the number of classes the model can predict (Wang, H., & Raj, B. 2017).

4.2 Transfer Learning

Transfer learning is the term used in machine learning to describe using a previously trained model on a different problem. In transfer learning, a computer leverages the information gained from prior training to improve prediction about a new task. Neural networks are often used in computer vision to identify edges, shapes, and other task-specific properties. Transfer learning uses the early layers of the neural network, whereas only the subsequent layers are retrained. It employs the labeled data from the task on which it was trained. The biggest benefits of transfer learning include shorter training times, higher neural network performance (in most cases), and the lack of a significant amount of data (Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., & Xu, C. 2020).

5. Methods

5.1 Image Data Generator

The images were fed to the model using an Image Data Generator. This meant that the training images were being fed to the model in batches of 15 rather than all at once. This was used to handle the large dataset without going over the Google Colab usage limits. The Image Data Generator also allowed for preprocessing of the images. All the images were rescaled to a size of 224 x 224 pixels so they could be fed into the network. In addition to this, the image pixel values were normalized from 0 to 1 as inputs with large integer values can slow down the training process because the neural networks use small weight values to process the integer values.

5.2 VGG16

The task of image-based geolocation is notoriously difficult and therefore training a network from scratch would require an extensive amount of time and computational power. Therefore, transfer learning was used to reduce both training time and computation. The VGG16 model trained on the ImageNet dataset was used for the project.

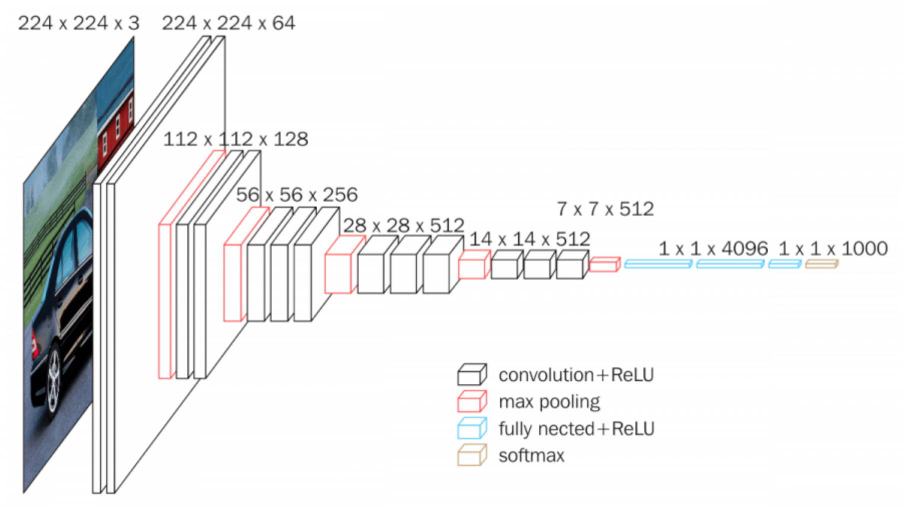

Figure 3. VGG16 Model Architecture (VGG16—Convolutional Network for Classification and Detection. (n.d.).)

VGG16 is a convolutional neural network trained on a portion of the ImageNet dataset, which contains over 14 million photos classified into 22,000 categories. VGG16 obtained 92.7 percent classification accuracy in the 2014 ImageNet Classification Challenge. It is more important to this project that it has been trained on millions of photos. The model is widely regarded as one of the best computer vision models available to date, and it is thus employed in transfer learning across a wide range of contexts. Its pre-trained architecture is capable of detecting generic visual elements in this project's Geolocation dataset (Simonyan, K., & Zisserman, A. 2015).

5.3 Project Model

As shown in Figure 2, the architecture of the model contains thirteen Convolutional layers, five Max Pooling layers, and three Dense layers of which sixteen of these layers are trainable. After loading the VGG16 model the Dense layers were taken out and replaced. The Convolutional layers and Max Pooling layers were frozen and thus their weights could not be changed during training. A full connected layer of 1024 neurons was added to the model. Then a Dropout layer was added to help prevent overfitting. The Dropout layer will randomly leave out a specified amount of neurons from the fully connected layer, allowing the model to become more robust. After this, a final Dense layer was added with ten neurons, the number of classes, to act as the output layer. The softmax function was used to display the probability distributions of all the classes and provide the final output. Figure 3 shows a visual representation of the model's architecture, and Figure 4 shows a shortened representation of the model's layers.

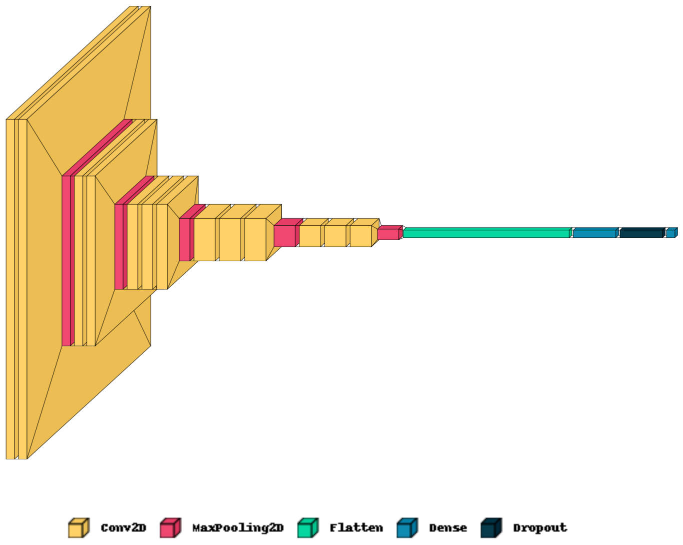

Figure 4. Model architecture created using VGG16

A callback function was created to stop the training after the model retained a minimum validation loss accuracy for seven epochs in a row. The model reached a peak accuracy without overfitting at 11 epochs. The use of transfer learning was adequate for the task as the model was able to reach a superhuman accuracy in a short time period.

6. Results

Table 1 gives the classification accuracy for the model on the Flickr dataset as well as the classification accuracy for each of the ten cities. To determine the model's accuracy, 500 random images from the dataset were chosen from each of the ten cities and then passed through the model. The accuracy was then calculated by dividing the total number of correct predictions by the total number of predictions. The model performed with above human-level accuracy and achieved an accuracy of 61.0%.

City	Accuracy
Amsterdam	.612
Bangkok	.564
Barcelona	.604
London	.345
Los Angeles	.727
Mumbai	.489
New York	.857
Paris	.416
Tokyo	.658
Venice	.823
Overall	**.610**

Table 6. Model accuracies for each of the ten cities and overall accuracy

As seen from Table 1, the categorization accuracy of certain cities varies greatly. These disparities might be explained by a variety of factors, including architectural similarities between cities or noisy data.

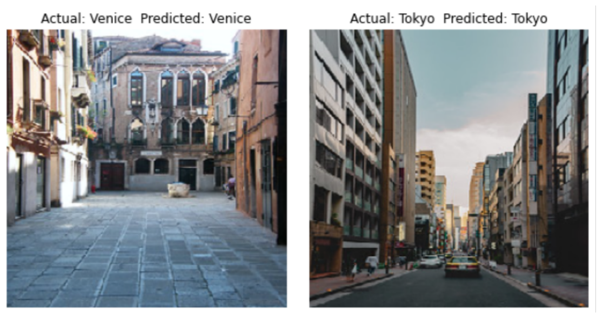
Actual: Venice Predicted: Venice
Actual: Tokyo Predicted: Tokyo

Figure 7. Images Correctly Classified by the Model

Figure 2 shows two images that the model correctly classified. Although it may not immediately recognizable to humans, the model was able to distinguish the correct cities by looking at the distinctive architecture. The model is generally good at predicting cities when the image contains certain characteristics of a particular city such as high-rise buildings or gravel roads.

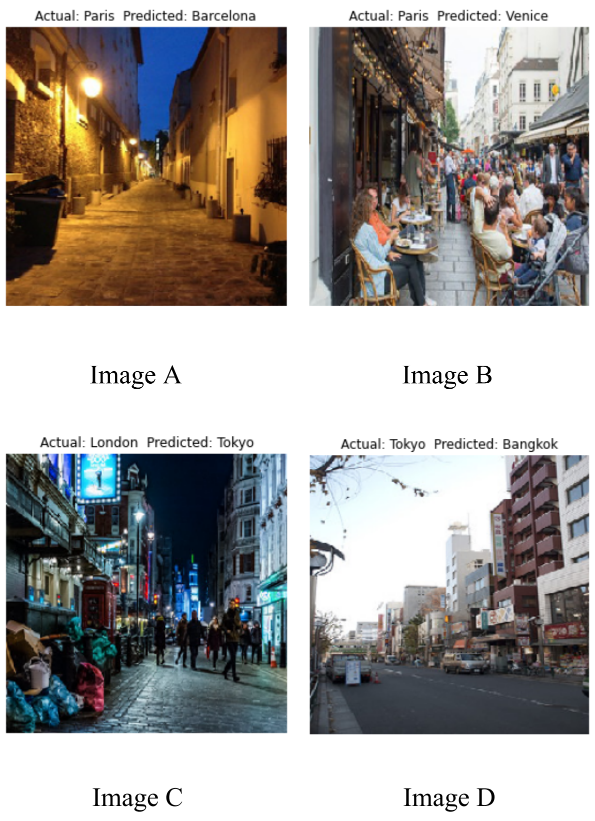
Actual: Paris Predicted: Barcelona
Actual: Paris Predicted: Venice
Image A
Image B
Actual: London Predicted: Tokyo
Actual: Tokyo Predicted: Bangkok
Image C
Image D

Figure 8. Images Incorrectly Classified by the Model

Figure 3 depicts a set of photos that the model misclassified. When these images are examined, it is clear that the algorithm generated predictions based on the architectural styles of the cities. Image A, for example, depicts a standard European alley that may easily be predicted wrongly. Image B is another image of a Paris café that was mistakenly identified as a Venice cafe due to similarities in architectural design. Image C depicts a view of London with an abundance of night lights reminiscent of Tokyo at night. Finally, the model mistakenly identified Image D as Tokyo rather than Bangkok. Perhaps if the model was larger or the image's input size was larger, the model would have been able to recognize the scriptures on the building as Japanese rather than Thai and correctly classify the image.

A confusion matrix was built in order to obtain insight into where the model was lacking. The confusion matrix was built using the same 5000 photos used to calculate the model's accuracy.

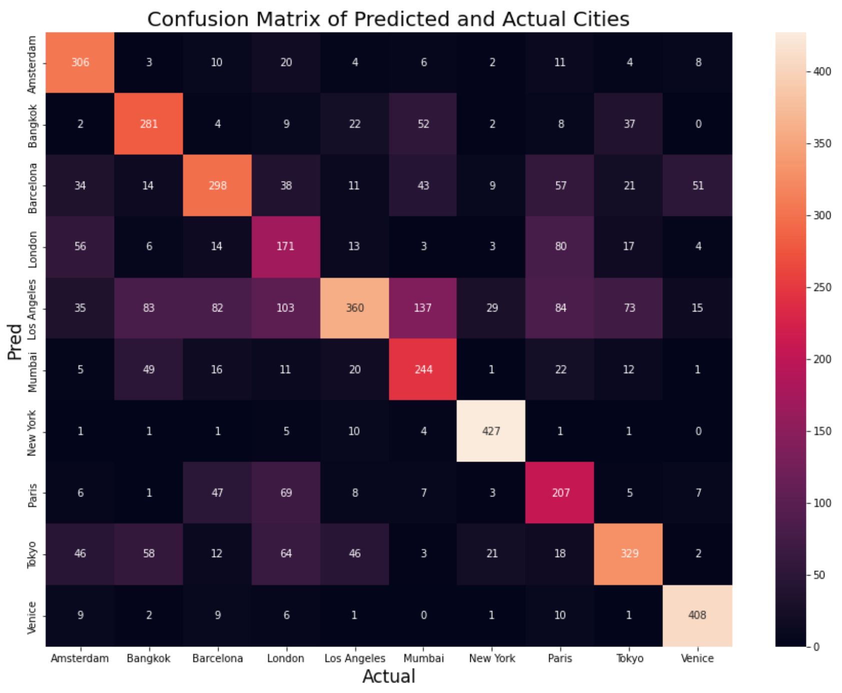

Figure 9. Confusion Matrix Visualization for the Geolocation Model

When the confusion matrix is examined, it is clear that the model mistakenly predicted the city as Los Angeles many times. After reviewing the inaccurately predicted photos, it became evident why the model was predicting Los Angeles for so many of them.

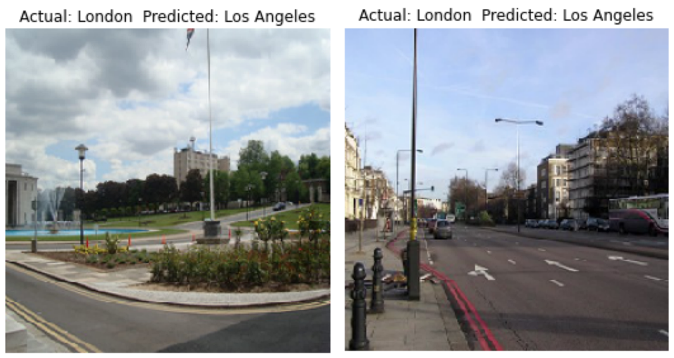
Actual: London Predicted: Los Angeles
Actual: London Predicted: Los Angeles

Figure 10. Incorrectly Classified Images of Los Angeles

The above two images show incorrect classification as Los Angeles. It was found out that whenever there was a lack of distinctive architectural features and instead generic images of roads, the model would predict the location as Los Angeles. This is a significant problem when it comes to image-based geolocation as it is difficult for a model to extract features from a simple picture. The reason for this was that the Los Angeles pictures in the database were also simple and contained many images of highways and empty streets.

7. Future Work

The findings demonstrate that utilizing CNNs can achieve extremely good performance in predicting geo-location from solely image data. However, there is still potential for development. Some possible extensions to this project to achieve even better results include model optimization, generalizing to more test data, and adding additional classes (Peddada, A. V., & Hong, J. 2015). Additional Dense layers might be added to the model to improve its complexity and pick up on some of the more nuanced features of the images. This would undoubtedly need more time and GPU computing power than is now available.

Training the model with images from different sources may also increase its accuracy and overall robustness. An example of this would be using city street images from other sources, such as Google Street View Images, Facebook, and others, to determine how correctly the model can localize these new photos. Lastly, by adding additional classes and using better filters to decrease noise, the ten-class Flickr dataset can be enhanced. The complexity of the model will also have to increase with the addition of more classes. In any event, the use of CNNs in geolocalization is a vastly untapped area that deserves additional exploration.

8. Conclusion

In this project, it has been shown that convolutional neural networks using transfer learning can achieve superlative performance at image-based geolocation on a street image dataset. Achieving an accuracy of 61% despite the numerous challenges posed by the task is substantial and thus emphasized the impressive capabilities of CNNs. However, some of this success can be attributed to the formulation of the problem. I focused solely on ten highly distinctive cities, making the problem easier than it may be otherwise. Expanding the model to predict a location from anywhere in the world would likely yield lower accuracies. The project also underlined some of the fundamental issues with the localization task. Over-predicting images from a class that contains “generic”, simple street images was a problem and is a possible area of improvement. This was a problem that all the related works faced as well, and developing a solution to this issue would undoubtedly boost the accuracy of geolocation models. Overall, this project emphasized the effectiveness of transfer learning in image-based geolocation and can most definitely be applied to practical scenarios such as geopositioning, geotagging, and geocoding.

9. References

1. *2: An example of convolution operation in 2D 2. | Download Scientific Diagram*. (n.d.). Retrieved July 9, 2022, from https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524
2. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186
3. Hays, J., & Efros, A. A. (2008). IM2GPS: Estimating geographic information from a single image. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. https://doi.org/10.1109/CVPR.2008.4587784
4. Hershey, D., & Wulfe, B. (2016). *Recognizing Cities from Street View Images*. https://www.semanticscholar.org/paper/Recognizing-Cities-from-Street-View-Images-Hershey-Wulfe/e067333952771611273 99e4d6015c39f658cbe8e
5. Lee, S., Zhang, H., & Crandall, D. J. (2015). Predicting Geo-informative Attributes in Large-Scale Image Collections Using Convolutional Neural Networks. *2015 IEEE Winter Conference on Applications of Computer Vision*, 550–557. https://doi.org/10.1109/WACV.2015.79
6. Lin, T.-Y., Cui, Y., Belongie, S., & Hays, J. (2015, June 1). *Learning Deep Representations for Ground-to-Aerial Geolocalization*. https://doi.org/10.1109/CVPR.2015.7299135
7. Peddada, A. V., & Hong, J. (2015). *Geo-Location Estimation with Convolutional Neural Networks*. https://www.semanticscholar.org/paper/Geo-Location-Estimation-with-Convolutional-Neural-Peddada-Hong/d02abc695627717a200215cb4caa9220e6a2fac7
8. *Places CNN*. (n.d.). Retrieved June 24, 2022, from http://places.csail.mit.edu/downloadCNN.html

9. *Release of Places365-CNNs*. (2022). [Python]. MIT CSAIL Computer Vision. https://github.com/CSAILVision/places365 (Original work published 2016)

10. Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

11. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions* (arXiv:1409.4842). arXiv. https://doi.org/10.48550/arXiv.1409.4842

12. *VGG16—Convolutional Network for Classification and Detection*. (n.d.). Retrieved July 9, 2022, from https://neurohive.io/en/popular-networks/vgg16/

13. Wang, H., & Raj, B. (2017). *On the Origin of Deep Learning* (arXiv:1702.07800; Version 4). arXiv. https://doi.org/10.48550/arXiv.1702.07800

14. Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., & Xu, C. (2020). Pay Attention to Features, Transfer Learn Faster CNNs. *ICLR*.

15. Weyand, T., Kostrikov, I., & Philbin, J. (2016). *PlaNet—Photo Geolocation with Convolutional Neural Networks* (Vol. 9912, pp. 37–55). https://doi.org/10.1007/978-3-319-46484-8_3

16. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems*, *27*. https://papers.nips.cc/paper/2014/hash/3fe94a002317b5f9259f82690aeea4cd-Abstract.htmlaeea4cd-Abstract.html