Recognizing City Image Locations with Machine Learning

Team: Gregory Campbell (519, Kyle's cohort), Zohar Singer (419, Kyle's Cohort), Spencer Solit (519, Kyle's cohort)
Project Mentor TA: Kyle Vedder
Link to Code: https://tinyurl.com/4rjuu9c

Abstract

Current image capture devices encode GPS location in their metadata, but there are countless images that have been already stored without this information. The purpose of this project is to geolocate an image through machine learning, relying only on the contents of the image; a fully functional implementation of this concept would eliminate the need for GPS metadata within photos, and allow broadly for forensic locating of an image. To achieve this within the scope of our project, we created a custom dataset of 10800 total images from 8 worldwide cities and applied ideas from classical convolutional neural network (CNN) architecture, transfer learning, and ensembling. When predicting the location of each image, basic CNNs were able to achieve just over 50% accuracy (approximately the same as the human benchmark), while our best model reached accuracies of up to 83%. Our work shows that by leveraging transfer learning and ensembling, CNNs can use a small dataset to acquire significantly greater accuracy than humans on this small scale version of the geolocation problem.

Introduction

Geolocation is useful in a variety of areas to tailor location-based notifications to a user, such as weather alerts, marketing, and navigation. Therefore, many services could benefit from the ability to infer this information from raw image data. Geoguesser (https://www.geoguessr.com/) has turned this task into a game for humans, but a fully functional implementation of this concept would eliminate the need for GPS metadata within photos, and allow broadly for forensic locating of an image. Due to time and computation restraints, we have conceptualized a more narrow form of the problem in which a learner must categorize Google Street View images based on the major city in which they were taken. We have limited the domain to 8 of the 10 largest cities in the world, choosing to exclude cities in China, as they lack sufficient data on Street View. Based on our trials, even this simplified geolocation task is difficult for humans, with an average success rate of 47% in correctly identifying a city.

Related Prior Work

ImageNet Large Scale Visual Recognition Challenge (ILSVRC): ImageNet [1] is a research effort competition that has been held annually since 2010, with the goal of categorizing the ImageNet Dataset. This database consists of 15 million labeled, high-resolution images from over 20,000 categories [2]. Winners of this competition include well known CNN's such as AlexNet [3], GoogleLeNet [4], and ResNet [5]. AlexNet was novel because it used multi-layer neural networks to jointly learn features and classifiers and introduced Rectified Linear Units (ReLUs) as activation functions. This architecture achieved "top-5" error values just over 15% (almost 10% lower than the previous best) using just an 8-layer CNN. GoogleLeNet then introduced the non-sequential format of the Inception Module in a 22-layer CNN. Finally, ResNet emphasized residuals in an extremely deep, 152 layer CNN to bring the top-5 error below 4%.

Prior work specific to geolocation: Using CNN's for image-based geolocation has been explored since 2015. Prior approaches include using k-nearest neighbors and having a massive database of tagged

images such that new images can be compared to the database [6]. When researching using purely CNN's, early research simplified the problem by restricting the classification tasks to a few cities or landmarks, or classifying a location within a single city [7]. Transfer learning from trained ImageNet models dramatically increased the success rate in these tasks [7]. More modern models now attempt to classify from images GPS coordinates without limiting the classification domain, but greater global accuracy in these tasks tends to come at the cost of lower local accuracy [6].

Formal Problem Setup (T, E, P)

Task (T): The learner is presented with three photos taken from the same location, each with a 120 degree offset, such that the combined photos cover a 360 degree view of the location. Given this input, the learner is tasked with classifying which of the following 8 cities the images were taken in: New York, Cairo, Kinki, Mumbai, Sao Paulo, Mexico City, Delhi, and Tokyo.

Experience (E): For a given location, the initial orientation is randomly generated, and then three images with 120 degree field of view at 120 degree heading increments were downloaded. Image resolution at download was set to 640x640, allowing for significant choice in the degree of downsampling desired at train time. 400 locations from each city were included in the training set (9600 images total), and 50 locations from each city were included in the validation set (1200 images total). We also recorded the date the pictures were taken, the header of each image, continent, and country in order to help explain the results of the model.

Performance Metrics (P): A model's city predictions were scored based on the overall accuracy of predictions, where a correct prediction is a binary 1 and an incorrect prediction is a binary 0. This performance is measured on a validation set that was held-back during training. Human accuracy is similarly measured as a comparison using the same binary metric and a random 50 image subsample of the validation set.

Methods

Our procedure to create and validate a CNN that can geolocate based on image data took place in three stages: creating a dataset, recording a human baseline, and developing our algorithms. This last stage took place through an iterative process that can be summarized through three different approaches. First, we began by experimenting with a handful of novel architectures. Second, we implemented a transfer-learner based on a traditional ResNet18 implementation. Lastly, we incorporated an ensembling method natural to our dataset and experimented with hyperparameters to fine-tune performance.

We have 400 locations for each city with minimal repetitions (this was checked by hand and was the reason we decided not to use images downloaded from cities in China as there are only a few distinct locations with street view in China that could easily be over fit for). We chose the cities as we thought the most populated cities would be the most urbanized and therefore easiest to differentiate, as well as to ensure that there was enough data in each location for the model to make an accurate prediction. We collected the data using Google's public static street view api where we could pick a longitude and latitude location and it would give us the closest google street view location within a predetermined radius. These latitudes and longitudes were picked uniformly at random from our approximation of the city. We then repeated the exact same process for 50 locations per city (1200 images) to form a validation set.

We implemented a human baseline to establish a lower bound on the level of accuracy needed to be useful in the real world. Each group member was tested, and we also picked three additional people external to the project so as to remove any advantage we would have gained by working on the project ourselves. We then gave them a short training period of 25 locations in order for them to get the

participant used to the mechanics of the task. We then gave them 50 more locations to test their accuracy. We decided on this number by balancing enough data to remove chance while respecting the time of our volunteers. The images for the human testing came from the training data. The application we used to do the human testing can be found here (https://human-tester.netlify.app/).

Additionally, our initial attempts at creating simple architectures serves as an informal baseline to judge how applicable more advanced techniques are to this problem setting. We tried to create a small CNN with only 5 convolution layers, a larger CNN with 8 convolutional layers, and a CNN with skip layers totaling 10 layers. We then trained each model with Adam as the optimizer until the validation accuracy began to consistently decrease. The accuracy of these models is summarized below in the results section.

Next, we implemented a transfer model using PyTorch's model library and copied weights from a ResNet18 architecture that was trained on the ImageNet dataset. We then resized the final linear layer to fit our 8 output options and froze all other parameters so that they did not update during gradient descent. Because the model structure needed to be fixed for transfer, the model needed to automatically downsample images to the correct size, and inputting all three images at once would lead to massive loss of information. We therefore decided to have the model take in one 120-degree slice at a time during training and viewed each slice as a separate input, which also had the benefit of tripling our data points. Then, the model evaluated its performance using cross entropy loss and was optimized using Adam. During validation, this model would only use one image to make it's prediction.

Lastly, we created a transfer model that also leveraged ensembling to make its predictions. During train time, the model would output a vector for each of the three images at a given location. These outputs were then averaged and the cross-entropy loss was calculated based on this average. During validation, an average was calculated using the same procedure and the city with the highest confidence was chosen. We also experimented more deeply with hyperparameters for this model, such as training epochs, learning rates, and batch sizes. The final model was trained for 30 epochs with an initial learning rate of 0.001 and 64 locations in a batch.

Experimental Results:

Questions: We aim to answer the following questions: (1) Can a CNN trained on a small (9600 image) sample of Google StreetView images correctly predict the city location of other images? (2) Can transfer learning be used to apply a pre-existing CNN to the same problem, and outperform the custom CNN? (3) Will this system make the same types of mistakes that humans make, i.e. mixing up cities that are in the same country? (4) Are there inherent biases in these models that cause it to guess one location more than others?

We compare our custom-built CNNs to human trials and a transfer learning model that begins with a PyTorch implementation of ResNet18 [5] to address Questions (1) and (2). Our smallest CNN with 5 layers achieved 48% accuracy, our larger CNN with 8 layers achieved 39% accuracy, and our 10-layer model with layer skipping achieved 51% accuracy. These custom CNN models started over-fitting after about 15 epochs, with training accuracies of over 85%. Training accuracies were above 85% for all methods. The gains from layer skipping demonstrate the usefulness of this technique, yet even so the model quickly began to overfit the data. Given the sub-5% error rate ResNet18 achieved on the more difficult ImageNet challenge, this seems to imply our dataset is too small to learn more general features. It is possible corrections such as dropout may have improved this result further. However, our small-dataset interpretation is further supported by the jump to 67% achieved by our transfer learning model.

Name	Training Accuracy	Validation Accuracy
Zohar	56%	66%
Ishaan (Zohar's volunteer)	36%	40%
Spencer	32%	42%
Sam (Spencer's volunteer)	24%	30%
Greg	36%	40%
Julia (Greg's volunteer)	44%	66%

Table 1: Human testing accuracies

Method Name	Validation Accuracy
Human Benchmark	47%
Custom small CNN	48%
Custom large CNN	39%
Custom skip layer CNN	51%
Transfer Learner	67%
Ensemble Learner	83%

Table 2: Model validation accuracies

Adding ensembling to our model led to another 16% increase in accuracy, equal to the gain made from implementing transfer-learning rather than learning from scratch. Some portion of this rise comes from the fact that ensemble-model has access to triple the data during prediction when compared to the simpler transfer model. If the first validation image used an unlucky heading (i.e. it was pointing at a not very useful part of the 360 degree view), the transfer model would be unable to use features present at other headings to help identify the city. The ensemble model, on the other hand, had access to the entire 360 degree view and didn't lose too much information through downsampling just to fit the Resnet18 architecture.

An example of results from the ensemble-transfer-learning model applied to training data is shown below:

Figure 1: Ensemble model predictions compared to actual cities shown for a sampling of training data

We hypothesize that the accuracy increase from ensembling mostly comes from this increased information access, rather than the strategy of ensembling itself. First, ensembling typically works best when using many weak learners, rather than our implementation using a few more complex learners. Second, neural networks are prone to produce overconfident outputs from softmax layers [8] and there could easily be an input where there is some extremely indicative feature, such as english signs in New York, that might only be present in one 120-degree slice. However, if the model overconfidently mispredicted on the other two slices, the indicative feature is essentially ignored.

On the other hand, it is difficult to conclude exactly what led to such significant gains without further testing, and ensembling has been an essential strategy in many other tasks. It is quite possible that ensembling helped because there is some indicative feature that is over-valued, and forcing the model to look for consistency across the slices leads to greater gains. To expand on our earlier example, English signs might be very indicative of New York, but other cities might occasionally have signs in english as well. In this scenario, ensembling would help prevent the learner from getting too distracted by any single instance of a feature. More testing is required to confirm either case.

In comparing the performance of the ensemble model on individual cities (within the validation set), it was clear that it excelled at guessing some cities but came up short on others. Interestingly, the model never mislabelled Cairo. This city also had the highest accuracy for the human testers, as most noticed Cairo has a “distinctive color,” making it much easier to identify. While Dehli also reached very high accuracy, it was also guessed significantly more than Mumbai, meaning the model may have defaulted to Delhi when it was unsure between the two cities.

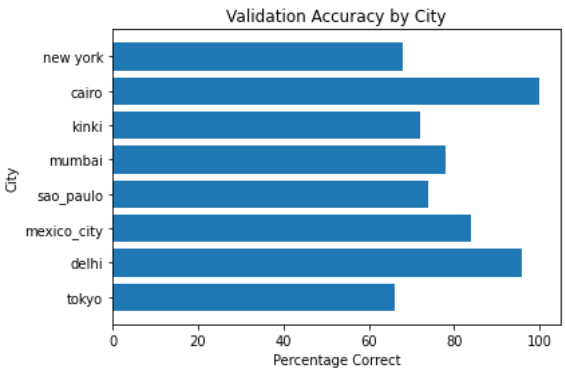


Figure 2: Ensemble model validation accuracy

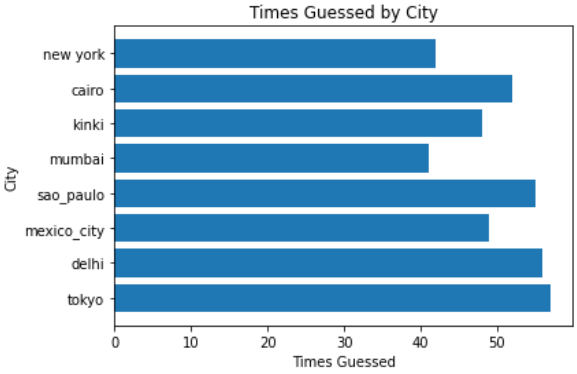


Figure 3: Ensemble model prediction distribution

Additionally, the model often seemed to make errors on cities within the same country, just as our human testers did. Tokyo was the most common incorrect guess for the Kinki and vice versa (see S1 and S5). The same can also be seen for Mumbai and Delhi (S2 and S3).

Conclusions and Future Work

To achieve optimal performance on our geolocation task, it was important to take advantage of both transfer learning and ensembling methods. Taking this approach, our learner significantly outperformed humans in categorizing city images. However, the ensemble model still struggled in many of the same areas humans struggled in, which might highlight some inherent ambiguity in the task.

Subsequent work can improve the scope of our geolocator, such as expanding the dataset to include more cities, or allow more fine-grain classification of locations within each city. Additionally, it may be worthwhile to experiment more thoroughly with ensemble architectures, varying how many learners work together and how their input is weighted. Neural networks are prone to produce overconfident outputs from softmax layers, and many techniques have been proposed to handle this, such as Monte Carlo Dropout [8]. Currently, because our model simply averages together predictions instead of using more recent techniques, one overconfident result may skew the final result.

Ethical Considerations and Broader Impacts:

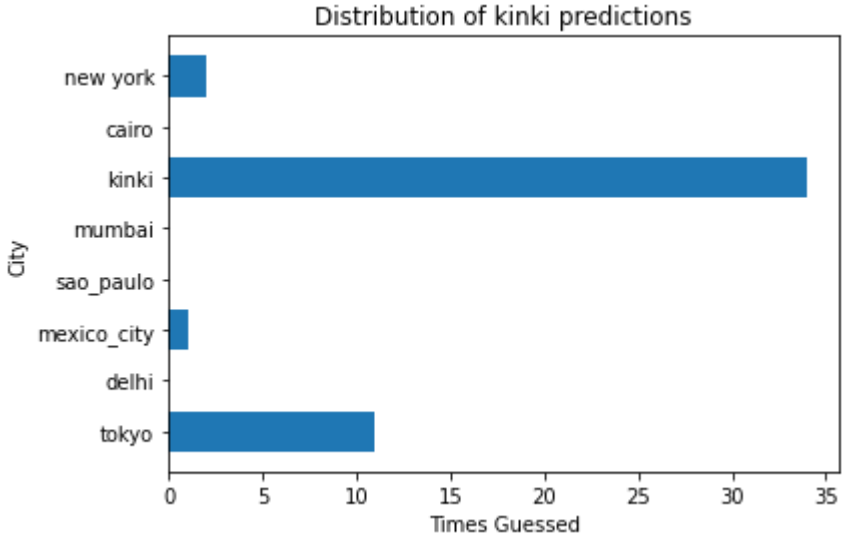
While geolocating images restricted to only 8 cities has few ethical ramifications, a more comprehensive geolocater will restrict one’s ability to share images without giving away location data. This raises significant privacy concerns, from invasive marketing to empowering predators on dating apps. Additionally, Google’s publicly available image data biases towards more attractive, affluent areas. To cover cities in Haiti, for example, one would need to rely on images from media coverage, which may lead to unreasonable and biased decisions. On the other hand, geolocation from images can be a powerful and constructive tool when used responsibly. We hope that with human guidance and careful attention to bias, our work can be an instrument for public good.

Prior Work / References:

1. Image-net.org
2. https://www.kaggle.com/getting-started/149448
3. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.
4. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
5. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
6. Müller-Budack E., Pustu-Iren K., Ewerth R. (2018) Geolocation Estimation of Photos Using a Hierarchical Model and Scene Classification. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11216. Springer, Cham. https://openaccess.thecvf.com/content_ECCV_2018/papers/Eric_Muller-Budack_Geolocation_Estimation_of_ECCV_2018_paper.pdf
7. Peddada, A.V., & Hong, J. (2015). Geo-Location Estimation with Convolutional Neural Networks. http://cs231n.stanford.edu/reports/2015/pdfs/CS231N_Final_Report_amanivp_jamesh93.pdf
8. Y. Gal, R. Islam, and Z. Ghahramani, “Deep Bayesian Active Learning with Image Data,” *International Conference on Machine Learning (ICML)*, 2017. https://arxiv.org/abs/1703.02910

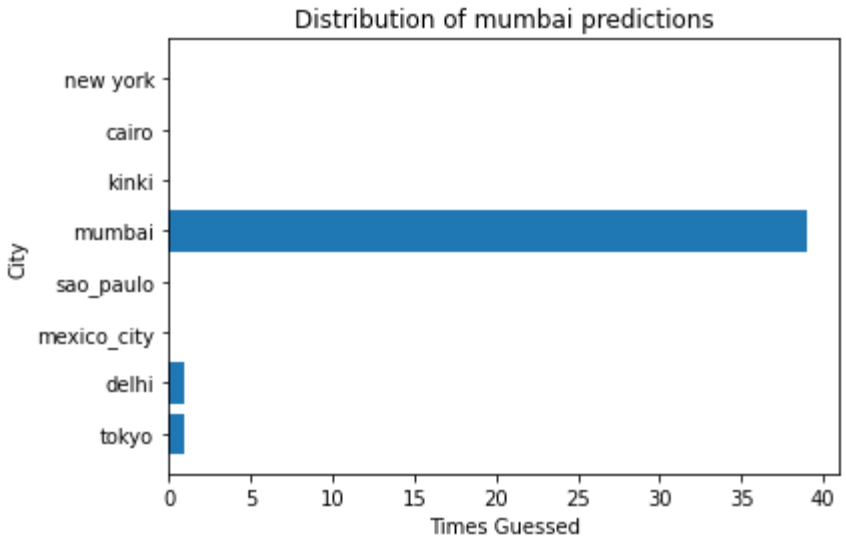
Supplementary material:

Distributions of ensemble learner guesses for each city (S1-S8). The titular city designates the actual location of the images, the y-axis cities represent the city that was guessed.

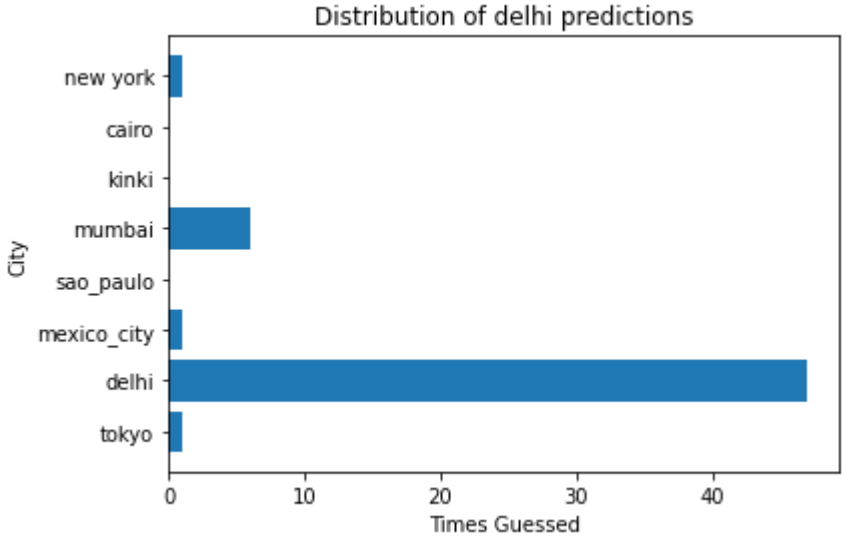


S1.

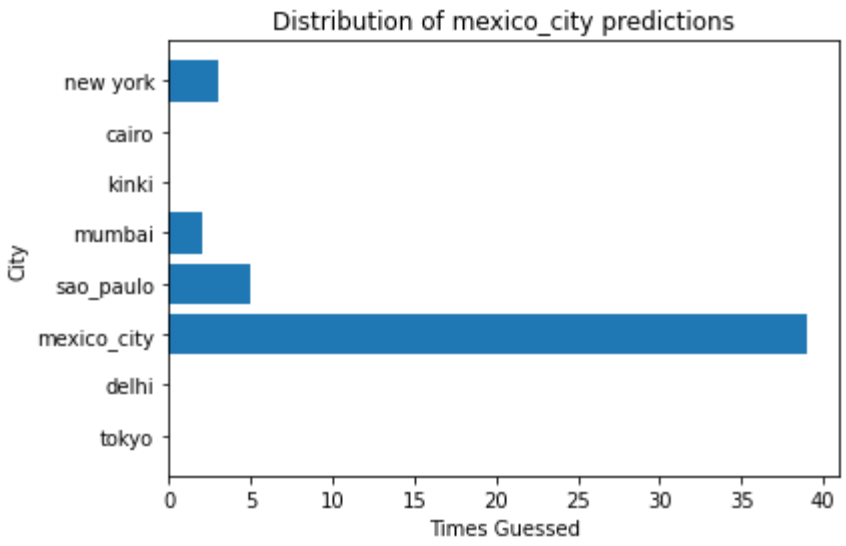
S2.


Distribution of mumbai predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
5
10
15
20
25
30
35
40
Times Guessed

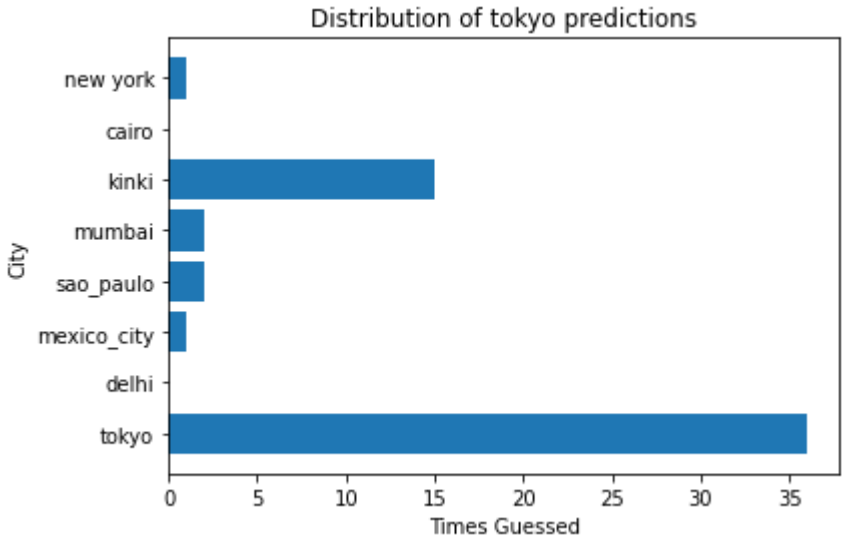
S3.


Distribution of delhi predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
10
20
30
40
Times Guessed

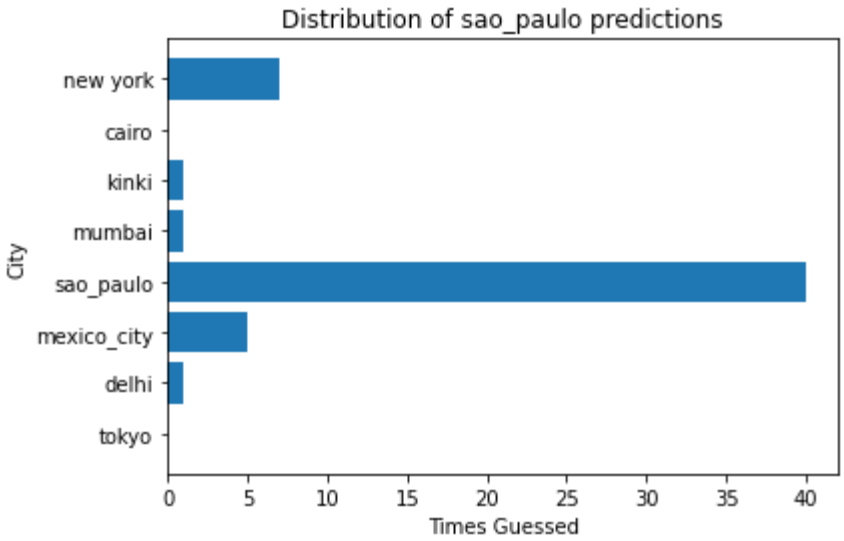
S4.


Distribution of mexico_city predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
5
10
15
20
25
30
35
40
Times Guessed

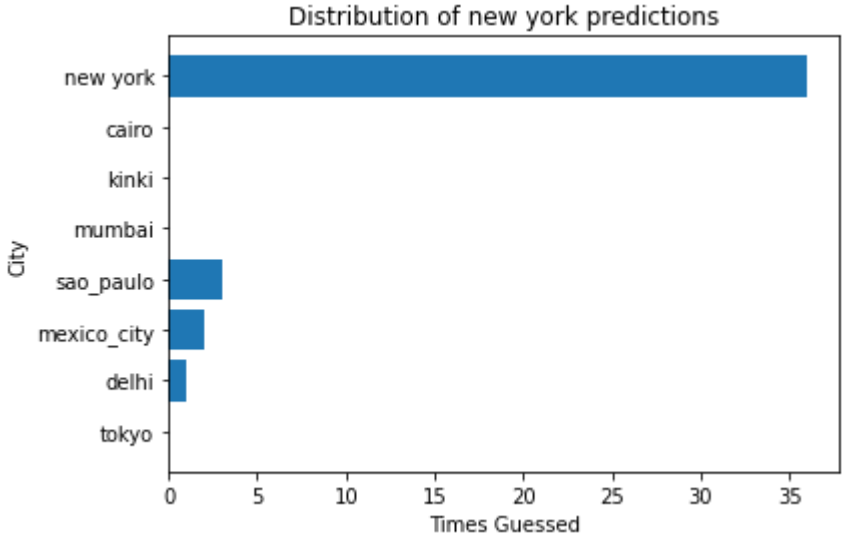
S5.


Distribution of tokyo predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
5
10
15
20
25
30
35
Times Guessed

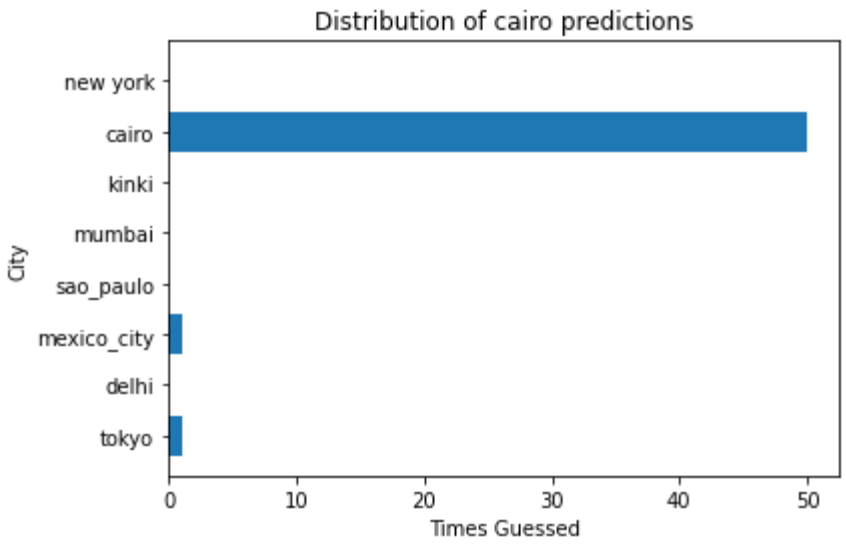
S6.


Distribution of sao_paulo predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
5
10
15
20
25
30
35
40
Times Guessed

S7.


Distribution of new york predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
5
10
15
20
25
30
35
Times Guessed

S8.


Distribution of cairo predictions
new york
cairo
kinki
mumbai
sao_paulo
mexico_city
delhi
tokyo
City
0
10
20
30
40
50
Times Guessed