Deep Learning for Image-based City Classification

Pranav Devabhaktuni
Georgia Institute of Technology
Atlanta, Georgia
pdevabhaktuni6@gatech.edu

Dhruv Adha
Georgia Institute of Technology
Atlanta, Georgia
dadha3@gatech.edu

Akshay Raj
Georgia Institute of Technology
Atlanta, Georgia
araj72@gatech.edu

Arjun Birthi
Georgia Institute of Technology
Atlanta, Georgia
abirthi6@gatech.edu

Abstract

Cities around the world possess distinct architectural styles, street layouts, and landmarks that reflect their unique histories and cultural identities. These visual identifiers enable the classification of cities based on photographic inputs. In this study, we address the problem of predicting the city where a street-level image was captured, using deep learning methods. Our primary dataset, GSV-Cities, contains approximately 530,000 labeled images that span 23 prominent cities around the world. We explore Convolutional Neural Networks (CNNs) and additional models, such as ResNet18, DenseNet, and OpenAI's CLIP to benchmark performance. Our approach includes rigorous data preprocessing, augmentation techniques, and a standardized evaluation using metrics such as precision, F1 score, precision, and recall. We found that the DenseNet model achieved the highest overall performance, significantly surpassing random baselines. Potential applications of this research include smart city guides, location-based content filtering, and urban planning. This work highlights the efficacy of CNN-based approaches for geolocation tasks and lays the foundation for future advancements in visual place recognition and urban analysis.

1. Introduction

Thousands of cities exist around the world, each characterized by their own unique architectures, climates, street layouts, landmarks, and several other distinct features that reflect their storied histories. These characteristics serve as essential visual identifiers, allowing us to distinguish one city from another. In this paper, we aim to identify the city a picture was taken in using these key features. Understanding public architectural styles is important in urban planning, analyzing cultural differences, sustainability efforts, and location-based services. To achieve this, we plan to apply deep learning and convolutional neural networks (CNNs) for city classification. CNNs are well-known for their effectiveness in image recognition and classification, making them an ideal solution for the problem.

Since there are thousands of cities around the world, we decided to narrow down our focus to a subset of twenty-three prominent cities. The selection of these cities was primarily based upon easy availability of data. Also, cities within a country can look very differently while a single city is more likely to have distinguishing characteristics that a model can identify. Our dataset contains approximately 530,000 unique images and contains information about the latitude, longitude, month, and year each image was taken. Our model will take in a street-view level image as an input and output the predicted city where the image was taken. Based on past research, we have found that CNNs have had significant success in similar domains, such as predicting countries, GDPs, and population density, so we hope to replicate this success with accurate city prediction. Some possible challenges we may encounter during this project include class imbalances, different image resolutions, and seasonal variability. We plan to evaluate our model on several metrics, including accuracy, F1 score, precision, and recall. Potential applications of this technology include smart city guides, automatic content filtering based on image location, environmental monitoring, and disaster response and recovery.

2. Related Work

One of the first breakthroughs in geolocation was IM2GPS in 2008. This paper compared individual images to millions of others and used similarity scores to predict the location. This group had a 25% accuracy in pre-

Figure 1. Example of image from Chicago. We plan to identify which city images like this one are from.

dicting the correct country [4]. Until 2015, with the rise of CNNs, there was not much progress in this space. A group from Google developed PlaNet for the task of geolocation, a CNN based on the Inception architecture with batch normalization. Their model achieved a 25% city-level accuracy and performed better than manual prediction [8]. From here, methods to improve upon the geolocation task diverged. Using a special version of the VLAD descriptor called NetVLAD, deep features were condensed into one feature before being plugged into a CNN [2]. On the other hand, R-MAC extracted regions of interest from CNN feature maps to form representations. The next couple of years brought many more advances and techniques, but almost all high-performing models were based on CNNs [7]. CNNs were also found to perform well at predicting features other than location, such as population, elevation, and household income, reaching accuracies of 75% for population density and 73% for GDP [6]. More recent studies use representation learning, including the paper that published our dataset. The "GSV-Cities" paper serves as the background paper for our dataset, which provides around 14 years of images and around 40 cities. Existing models that focus on visual place recognition (VPR) are found to have better performance when trained on the GSV-Cities dataset. Specifically, the paper highlights key loss functions on the performance of VPR and a unique convolutional aggregation layer that outperforms existing SOTA methods such as GeM and NetVLAD. This aggregation layer could be useful for localizing cities more accurately, and the GSV-Cities dataset serves as an initial dataset that we plan to use [1]. "PIGEON: Predicting Image Geolocations" from Stanford University is the most recent work in the field (May 2024) we have found. This paper provides a unique approach to image geolocation that improves the accuracy of predictions using vision transformers and multi-task learning. PIGEON employs OpenAI's CLIP model in contrast to popular DL models which utilizes auxilary data such geography, demographics and climate to tune the model. The methods provided in this research paper may be useful when we incorporate auxiliary data such as urban city layout or transportation options to predict gentrification or walkability [3].

3. Technical Approach

3.1. Data Collection and Processing

For this problem, we collected data from two primary data sources. The first source of data was the GSV-Cities dataset, which contains a diverse set of street-level images from cities around the world. In total, the dataset contains approximately 530,000 unprocessed images from 23 different cities across the globe. The distribution of cities the images come from is seen in Figure 3. The second source of data was Google's Street View Static API and Places API, which allowed us to generate street-level images in arbitrary locations. We used the Google Street View API and Places API to generate additional images of the cities found within the GSV-Cities dataset. Once we collected the necessary data, we performed several steps of data preprocessing, including data visualization and image transforms (displayed in Figure 4), to properly format our data to receive the highest accuracy possible with our models. Additionally, we split our model into 80% training data, 10% validation data, and 10% testing data.

3.2. Models

Convolutional neural networks (CNNs) have been extensively studied and are well known for their success in image classification tasks. CNNs have the ability to reduce the high dimensionality of images but retain the relationships between parts of the image using parameter sharing. Furthermore, based upon prior research, several similar geoclassification tasks made thorough use of CNNs with significant success [1] [5]. As such, we decided to use a CNN as our primary model since we believed it would be extremely effective in identifying the unique features between cities to classify them. Our basic CNN model consists of three convolutional layers with Batch Normalization, ReLU activation, and max-pooling. The connected layers map the 32768 dimensional space to a flattened output of 512 hidden units which is then mapped to the number of classes. We tested with more convolutional layers and a dropout layer but our accuracies were not as high as the simplified approach. We observed that a low learning rate of $1e{-}4$ produced the highest accuracy. We used the Adam optimizer as it is known to converge quickly which ensures stability and speed in our model.

In addition to the usage of CNNs, another model we implemented is the OpenAI CLIP (Contrastive Language–Image Pre-training) model covered in the PIGEON paper [3]. This model leverages multimodal learning, such as associating images with geographic and demographic data, which can fine-tune model accuracy. CLIP is intended

Figure 2. Examples of images from Barcelona in the training set.

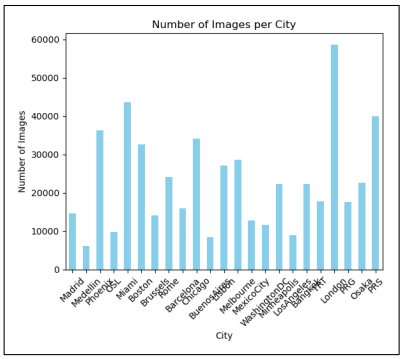

Figure 3. Images per City in the GSV-Cities Dataset.

to use as a zero-shot model, similar to GPT-3 in that it often requires no extra training. CLIP was developed to function well on a variety of datasets so images were associated with a few samples of a set of 32,768 randomly sampled text snippets. This ability to generalize comes at a cost of requiring 63 million parameters. However, unlike other deep learning models such as ImageNet, the CLIP model does not need to be fine-tuned to be applied on a new dataset - we just need to pass in label names pertaining to our task. The CLIP model contains two processes: the text encoder takes labels and converts them to vectors and the image encoder takes in images and also outputs vectors containing semantic data. These encoders are pretrained so that the vectors produced by text-image pairs are similar. Now on a new task, cosine similarity between the text and image vector is applied to select the label that is most appropriate to the image.

Additionally, we used PyTorch to implement a basic DenseNet model to function as a supplementary model for our benchmark of ResNet. We achieved high validation accuracies with the DenseNet model, so we may pursued parameter tuning with this model as well.

3.3. Metrics

As a very rudimentary baseline, we compared our model against random choice, which would give a $\frac{1}{23}$ (4.3478%) chance of accurately guessing an image's location. As a more thorough baseline, we compared our model against a pretrained ResNet model based upon the ImageNet database. Since our dataset is relatively balanced, we chose to use accuracy as our primary evaluation metric, but we also utilized F1 score, precision, and recall as secondary metrics. we also used a confusion matrix to visualize our mismatches and identify if there are any particular trends within our models, and plotted the training and validation curves for loss and accuracy.

4. Data

4.1. GSV-Cities Dataset

Our main source of data in this project was the GSV-Cities dataset, a large-scale resource designed for training deep learning models in Visual Place Recognition tasks, which fits our project's goal. This dataset, published as part of the Neurocomputing 2022 study "GSV-Cities: Toward Appropriate Supervised Visual Place Recognition," consists of approximately 530,000 images representing over 62,000 distinct physical locations spread across 23 cities worldwide. Each location is depicted by at least four images (up to 20), ensuring that there is decently significant visual variability. There is a minimum distance of 100 meters between any two locations to avoid redundancy.

The dataset focuses on street-level imagery, making it particularly suitable for city classification tasks like ours. We selected this dataset because it provides a well-structured benchmark for analyzing architectural, geographic, and cultural differences that distinguish cities. By narrowing the scope to 23 cities, the dataset is able to balance between diversity and manageability, allowing us to focus on finding and extracting meaningful features without overwhelming the model or complicating the results with

Figure 4. Images underwent transformations as seen above.

significant overlap.

Some notable properties of the dataset include how it has important metadata, which provides contextual information such as latitude, longitude, and timestamps (month and year). These attributes enhance the dataset's utility for geographic and temporal analyses - although our work primarily focuses on the visual aspects, temporal analysis could be area for future work. Geographically, having more information helps us to better understand how the dataset is diverse. We also note that the dataset is inherently limited by its geographic scope and does not represent all cities globally. Its images predominantly capture urban environments, which may restrict generalization to other settings, including rural or suburban areas. For us, this is a positive. We want it to be able to classify between only these 23 world cities with a high accuracy.

The dataset has been preprocessed to standardize image resolution and augment training data, using techniques such as resizing, color jittering, and random flips. This allows us to ensure better generalization of our models while addressing challenges like class imbalances and seasonal variability. One important thing to note is that, due to the vast number of visual similarities between urban locations, there is a lot of significant overlap between features in the images. In the future, a more diverse selection of cities might help to mitigate this potential bias towards specific features.

4.2. Street View API/Places API Images

To supplement the GSV-Cities Dataset, we decided to use Google Maps Street View API. This is because the Street View API can function as an evaluation for how realistic the GSV-Cities dataset mirrors the "real world" dataset. Initially, the Places API was used to generate 20 images from each of the 23 cities. After creating a unique API key and call the API for 20 random locations (latitude and longitude) in a city with radius and point of interest, we observed many of the images were not useful. Specifically, some images, such as in Figure 5, were indoors and are not representative of what we aim to accomplish. Following this, we moved to Google Maps Street View, which again required a unique API key to call the Street View API. We called this API for 150 images per city based on location (latitude/longitude) and source/point of interest. Specifically, we requested for outdoor locations by checking the metadata of the image. The resulting image data gave us more promising results, as seen in Figure 6. Through this process, we obtained 150 images from each city in the GSV-cities dataset (total of 3000+ images). Due to the quota on the number of free API calls, we were unable to gather a substantial number of images compared to the GSV-cities dataset, so we decided to use the street view images we collected to further evaluate our ResNet and DenseNet models.

Figure 5. Places API Barcelona Image 3

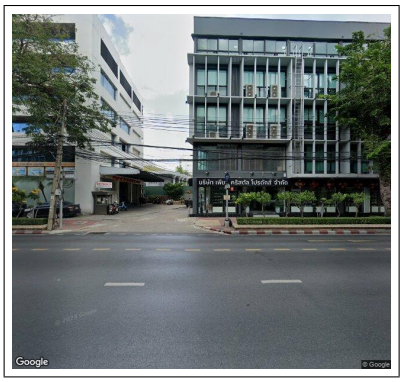

Figure 6. Street View API Bangkok Image 9

5. Results

5.1. Data Preprocessing

Utilizing the Kaggle API, we downloaded the GSV-Cities dataset. The GSV-Cities dataset contains csv files for each unique class and the actual images themselves. The csv files provide general information regarding the time and location that each image was taken at, which we used to determine if there were any class imbalances within our data. After, we used the PyTorch library to load and manage the large amount of images being processed. Since the dataset contained approximately 530k images, we decided to limit our model to train on 1000 images per class to reduce training time. We split the remaining data into 80% training data, 10% validation data, and 10% testing data. Additionally, we shuffled the data to ensure the model did not recognize any inherent trends within the data. This helped to reduce bias within batches and also improved the generalization of our model. Additionally, we applied several image transformations, including image resizing, color jittering, and random horizontal & vertical flips to help reduce model overfitting. Another benefit of the image resizing was a significant decrease in training time.

5.2. Results

Model	Accuracy	Precision	Recall	F1 Score
CLIP	0.5048	0.56	0.50	0.50
ResNet18	0.7165	0.7422	0.7165	0.7168
DenseNet	0.8109	0.8155	0.8109	0.8105
CNN	0.2917	0.2938	0.2917	0.2798

Table 1. Model Performance of All Models

We began with the CLIP zero-shot model as a baseline, passing in 2300 total image samples. For our class names, we entered the cities in the format ”City, Country”. Adding the country in as extra textual data would allow the model to perform better in case it identifies the country but an incorrect or missing city. Considering that the CLIP model was not trained on our street view image dataset, the model performed fairly well with 52.3% accuracy. We do not expect the CLIP model to perform much better since it performs best at identifying images within an image, not more abstract factors such as where the picture was taken. We then examined the 32,578 text snippets that the CLIP model is trained on. While countries such as United States and Spain were in the data, individual cities were not. So, we predicted that the model is likely able to detect the country that the image is taken in but not the city. To test this, we created two new lists of labels: a list of just the city names and a list of just the country names to compare the results against the original model that uses labels which include both the city and country names.

Labels	Accuracy	Precision	Recall	F1 Score
City, Country	0.5048	0.56	0.50	0.50
City	0.5161	0.56	0.52	0.52
Country	0.3474	0.31	0.35	0.30

Table 2. CLIP Model Performance on Different Label Formats

We were surprised to see that using both the city and country did perform significantly better than the models using only the city or only the country. The model may have been trained on similar data so that, for example, city names such as ”Barcelona” may be associated with labels that are in the training data such as ”beach”. The low accuracy for country labels is misleading because we did not try to see whether the model is able to predict the correct country; rather, we just omitted city names so both ”Barcelona, Spain” and ”Madrid, Spain” were mapped to ”Spain”, confusing the model. Our goal was to investigate whether the model is actually able to use the city names to classify images despite not existing in the data. When we only used unique countries as labels, the CLIP model achieved a 0.4478 accuracy, 0.62 precision, 0.45 recall, and 0.45 F1 Score. This clearly shows that the CLIP model is able to utilize the city names to classify images.

We then attempted to fine-tune our CLIP model to improve performance on detecting the City, Country labels but were unsuccessful. The loss only marginally improved through our 10 epochs even when we altered the learning rate.

We also experimented with a preliminary CNN model and later a ResNet model and DenseNet model. The preliminary CNN model had quite poor accuracy from our midterm compared to our ResNet having 72% accuracy

and DenseNet having 60% accuracy. Thus, we decided to move forward with the latter two models for continued exploration.

Model	Accuracy
Midterm ResNet	0.7165
ResNet with Improved Classifier & Freeze Layers	0.1403
ResNet with global average pooling, batch normalization, sequential layers	0.3106
Midterm DenseNet	0.6096
DenseNet with classifier and trained on all images	0.8109

Table 3. Model Performance Progression of ResNet & DenseNet

We were unable to improve our ResNet accuracy as seen in the above model performance table. Following our initial ResNet creation, we first experimented with freezing layers to limit the number of trainable parameters, adding batch normalization to stabilize learning, and introducing feature extraction steps in the forward method to refine the output. However, these changes resulted in worse validation accuracy, which was barely above an average guessing rate. We then transitioned to a more sophisticated ResNet architecture, incorporating global average pooling followed by sequential layers comprising dropout for regularization, linear layers for increased representation capacity, ReLU for non-linearity, and batch normalization for feature scaling. Despite these additions, this model only achieved an accuracy of 0.3106, which was lower than expected. After further analysis, we concluded that the initial ResNet model, with default weights and parameters, performed best with our dataset, likely due to better alignment with its pre-trained features.

To address the limitations of ResNet, we turned to a DenseNet architecture, leveraging its densely connected layers to enable efficient feature reuse and gradient flow. Specifically, we modified a DenseNet121 model by replacing its classifier with a single linear layer to match our input size of 1024 and output size of 23 classes. This modification resulted in a notable accuracy improvement, with the model achieving 0.65 accuracy during validation. Encouraged by these results, we decided to train this model on the entirety of the dataset, which further boosted the accuracy to 0.8109, marking our highest accuracy yet.

We used a learning rate of 1E-4, weight decay of 1E-5, and a batch size of 32, optimized with the Adam optimizer for all models. We experimented with variations of our hyperparameters (learning rate = 1E-3, weight decay = 1E-4), but we concluded after multiple runs that the learning rates aformentioned were the best. For each model, we ran for 10 epochs as a balance for training the model with accuracy and efficiency of time. Additionally, we found that the validation accuracy improvement plateaued around epoch 7 or 8 for each model, so it would not be useful to train for more epochs. The loss function we used was cross-entropy as it measured the error between predictions and ground truths. DenseNet generalized better due to its architectural efficiency, dropout, and batch normalization, achieving faster convergence and superior accuracy, making it the more effective choice compared to ResNet for this task. Both models were implemented in PyTorch and evaluated using metrics like accuracy, precision, and F1 score. For both the ResNet and DenseNet models had training accuracies greater than 95% with validation accuracies of 80% or lower, which is a sign of overfitting. We added an early stopping mechanism to help mitigate overfitting, but this did not achieve significant results.

Model	Accuracy
Final ResNet	0.278
Final DenseNet	0.294

Table 4. Model Performance on Street View API Images

Following the training and evaluation of our models, we tested their performance on Street View API images, which presented unique challenges. While our DenseNet model showed promising results in generalized GSV-Cities datasets, it failed to adapt effectively to diverse street view scenarios, yielding limited generalization capabilities. Surprisingly, Phoenix-specific data retained high accuracy, as seen in the confusion matrix in Figures 10 and 11. This suggests that the model benefited from region-specific features that aligned better with its learned representations, as Phoenix is unique with its deserts and cacti. Despite achieving slightly higher accuracy compared to ResNet (0.294 vs. 0.278, as shown in the table), the performance of both models on these images remained significantly lower than expected. These results highlight the need for further refinement, such as incorporating data augmentation or region-specific fine-tuning, to better capture the complexities of real-world street view imagery. Another future implementation would be to aggregate the GSV-Cities dataset with street view API images or extensively use street view API images to ensure the model has seen modern day street images (as some images in GSV-Cities date to over 15 years ago).

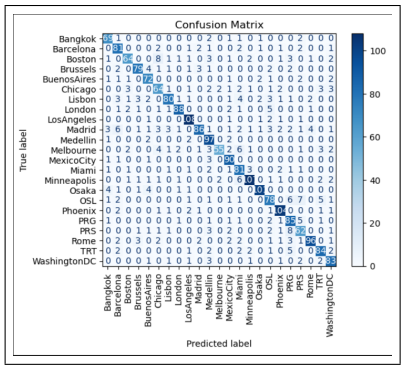

Figure 7. Confusion Matrix for ResNet18 model.

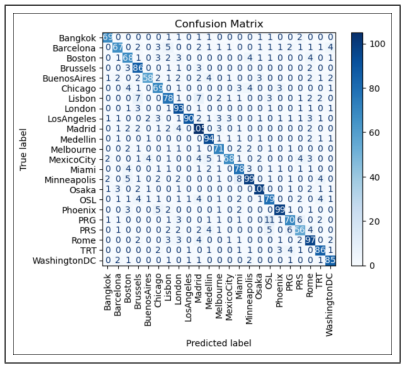

Figure 8. Confusion Matrix for DenseNet model.

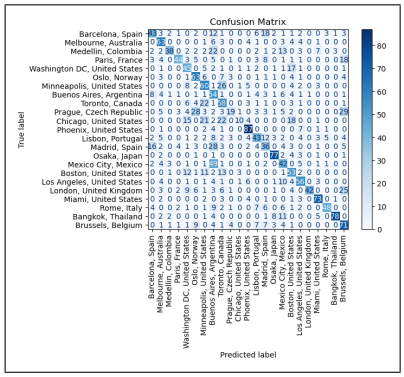

Figure 9. Confusion Matrix for CLIP model.

6. Conclusion

In this study, we explored the problem of city classification using street-level images, leveraging several deep learning models, including ResNet18, DenseNet, CNNs, and OpenAI's CLIP model. Our results highlight the challenges of building generalizable geolocation models and the importance of architectural complexity and pretraining in achieving strong performance. While our models performed well when we tested on the GSV-Cities dataset, particularly with DenseNet leading the results, they struggled when generalized to a testing set composed of images from the Google Street View API. This limitation underscores the implicit dataset bias present in the GSV-Cities dataset, where images may lack the variability necessary for robust generalization. However, an exception was found with Phoenix, a city that consistently performed better due to its distinctive desert features, suggesting that unique geographic characteristics can play a significant role in model accuracy.

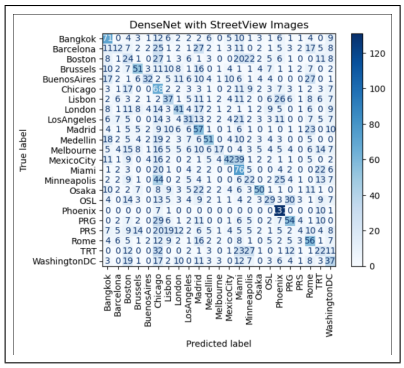

Figure 10. Confusion Matrix for DenseNet model on Street View images.

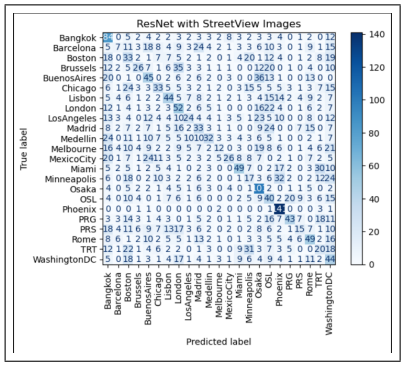

Figure 11. Confusion Matrix for ResNet model on Street View images.

Through this work, we learned that city classification is highly dependent on both the quality and diversity of the dataset, as well as the architecture of the model employed. Pretrained models like CLIP, though designed for zero-shot learning, demonstrated potential for geolocation tasks when city-specific data was limited. However, simpler models, such as our CNN, struggled to capture the complexity needed for high accuracy, emphasizing the need for more sophisticated approaches. The DenseNet model proved to be among the best options for geolocation classification, which may be due to its densely connected architecture that promotes efficient feature reuse and gradient flow. This allows the ResNet model to learn more concrete representations from our dataset. The ResNet model strug-

gled to generalize despite modifications like global average pooling, dropout, and batch normalization compared to DenseNet. Its architectural efficiency, combined with regularization techniques such as dropout and batch normalization, likely contributed to faster convergence and better generalization.

In the future, we propose incorporating additional data sources into the training set as well (in our case, images from the Google Street View API) to generate a more diverse and representative dataset that can improve model generalization to real-world applications. One potential area of research includes the exploration of hybrid models that integrate the contextual strength of multimodal approaches like CLIP with the feature extraction capabilities of CNNs, which could further enhance performance. Further areas of research include indoor-city detection, country-detection and geographic-similarity models. Potential real-world applications of this research include urban planning, environmental monitoring, disaster response, and location-based services. Addressing dataset biases and expanding the scope of city classification models could unlock new possibilities for applying this technology in diverse, real-world settings.

Team Contributions	
Name	Contribution
Arjun B	Worked on data visualization, ResNet and DenseNet result visualization, Street View API and contributed to paper and posterboard.
Dhruv A	Developed the CLIP model and associated visualizations, tested model on different labels and attempted fine-tuning, and worked on paper and explanations.
Akshay R	Developed CNN Model, worked on DenseNet & ResNet models, worked on Places API & Street View API. Contributed to paper on respective topics.
Pranav D	Worked on dataset loading, image preprocessing, DenseNet and ResNet, and Street View API. Worked on poster and final paper.

7. References

References

[1] Ali-Bey, A., Chaib-Draa, B., & Giguère, P. (2022). GSV-Cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 513, 194–203. https://doi.org/10.1016/j.neucom.2022.09.127

[2] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016, June). NetVLAD: CNN architecture for weakly supervised place recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. http://dx.doi.org/10.1109/cvpr.2016.572

[3] Haas, Lukas and Skreta, Michal and Alberti, Silas and Finn, Chelsea. (2024). PIGEON: Predicting image geolocations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 12893–12902.

[4] Hays, J., & Efros, A. A. (2008). IM2GPS: Estimating geographic information from a single image. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 155, 1–8. http://dx.doi.org/10.1109/cvpr.2008.4587784

[5] Hershey, D., & Wulfe, B. (2016). Recognizing cities from street view images. Stanford University, CS231n: Deep Learning for Computer Vision. https://cs231n.stanford.edu/reports/2016/pdfs/422_Report.pdf

[6] Lee, S., Zhang, H., & Crandall, D. J. (2015). Predicting geo-informative attributes in large-scale image collections using convolutional neural networks. In *2015 IEEE Winter Conference on Applications of Computer Vision*, 550–557. http://dx.doi.org/10.1109/wacv.2015.79

[7] Tolias, G., Sicre, R., & Jégou, H. (2015, November 18). Particular object retrieval with integral max-pooling of CNN activations. arXiv. https://arxiv.org/abs/1511.05879

[8] Weyand, T., Kostrikov, I., & Philbin, J. (2016). PlaNet - Photo geolocation with convolutional neural networks. In *Lecture Notes in Computer Science* (pp. 37–55). Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-46484-8_3