Deep Learning for Image-based City Classification

Pranav Devabhaktuni, Dhruv Adha, Akshay Raj, Arjun Birthi

Georgia Institute of Technology

INTRODUCTION

The Problem

Thousands of cities worldwide have unique architectural styles, climates, and street layouts, which serve as visual identifiers. Our goal is to classify the city where a picture was taken using these distinguishing features.

Why It Matters

Understanding public architectural styles has practical applications in:

- Urban planning
- Analyzing cultural differences
- Sustainability efforts
- Location-based services

Our Approach

We leverage deep learning and Convolutional Neural Networks (CNNs), which excel in image recognition tasks, to classify images by city. Our dataset contains ~530,000 street-view images from 23 prominent cities, selected for data availability and diversity. Each image includes metadata like latitude, longitude, and time taken.

Challenges

- Class imbalances
- Varied image resolutions
- Seasonal variability

Applications

Potential real-world applications include:

- Smart city guides
- Environmental monitoring
- Disaster response and recovery

Figure 1. Examples of images from Barcelona in the training set.

RELATED WORK

Early Breakthroughs in Geolocation

- **IM2GPS (2008)**: Compared individual images to millions of others using similarity scores; achieved **25% country-level accuracy** [4].
- **PlaNet (2015)**: Google's CNN-based model (Inception architecture with batch normalization) improved performance, achieving **25% city-level accuracy** [8].

Advances in Feature Extraction

- **NetVLAD**: Condensed deep features into a single feature vector for CNN input [2].
- **R-MAC**: Extracted regions of interest from CNN feature maps for representation [7].

Broader Applications of CNNs

- Accuracies: **75% for population density**, **73% for GDP** [6].
- Demonstrated versatility beyond geolocation tasks.

Our Dataset and Current SOTA

- **GSV-Cities Dataset**: Provides 14 years of street-view images from ~40 cities; includes metadata like latitude, longitude, and time [1]. Introduced a convolution aggregation layer outperforming methods like GeM and NetVLAD.
- **PIGEON (2024)**: Latest work leveraging OpenAI's CLIP model with vision transformers and multi-task learning to improve accuracy by incorporating auxiliary data (e.g., urban layout, demographics, climate) [3].

TECHNICAL APPROACH

Data Collection and Processing

- **GSV-Cities Dataset**
  - ~530,000 unprocessed street-level images from **23 cities worldwide**.
  - Provides diversity in architecture, geography, and urban layouts.
  - See distribution of cities in Figure X.
- **Google Street View API and Places API**
  - Generated additional street-level images for the cities in the GSV-Cities dataset.
  - **Preprocessing Steps**
- Data visualization to identify trends and outliers.
- Image transformations: resizing, color jittering, and random flips (Figure Y).
- Split the dataset into:
  - **80% training**
  - **10% validation**
  - **10% testing**

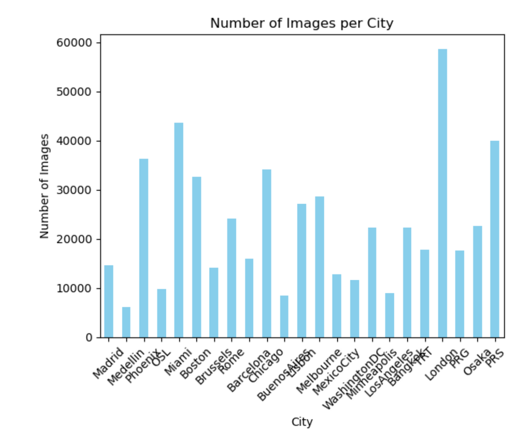


Figure 2. Images per City in the GSV-Cities Dataset.

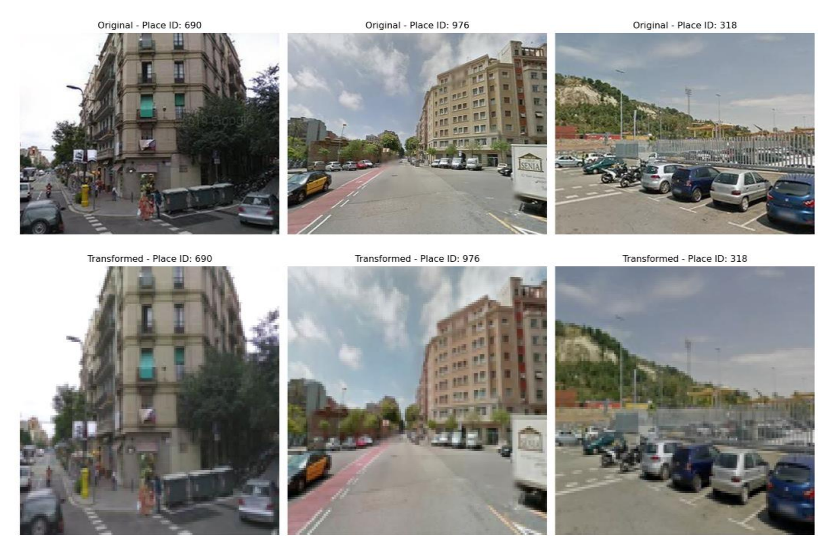


Figure 3. Images underwent transformation as seen above.

Models

1. Convolutional Neural Networks (CNNs)

- **Why CNNs?**
  - Excellent at image classification tasks.
  - Reduce high-dimensional data while retaining spatial relationships through parameter sharing.
- **Our Implementation**:
  - **Architecture**: 3 convolutional layers with Batch Normalization, ReLU activation, and max-pooling.
  - **Output Mapping**: From a 32,768-dimensional space to 512 hidden units, then to the number of classes.
  - **Optimization**: Used Adam optimizer and a learning rate of **1e-4** for stability and speed.
  - **Observations**: Simpler models outperformed deeper architectures in this task.

2. OpenAI CLIP

- **Why CLIP?**
  - Leverages multimodal learning (e.g., images + text) for geolocation tasks.
  - Generalizes across datasets without additional fine-tuning.
- **Key Features**:
  - Pretrained on **32,768 text snippets** and uses cosine similarity to match image and text vectors.
  - Contains a text encoder and an image encoder that map inputs to semantic vectors.
  - Requires **63M parameters** to operate efficiently.

3. DenseNet

- **Why DenseNet?**
  - Known for efficient feature reuse and gradient flow.
- **Our Implementation**:
  - Used PyTorch to implement a basic DenseNet model.
  - Achieved high validation accuracy, making it a strong benchmark for comparison.
  - Further parameter tuning planned for improved performance.

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

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

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

Model	Accuracy
Final ResNet	0.278
Final DenseNet	0.294

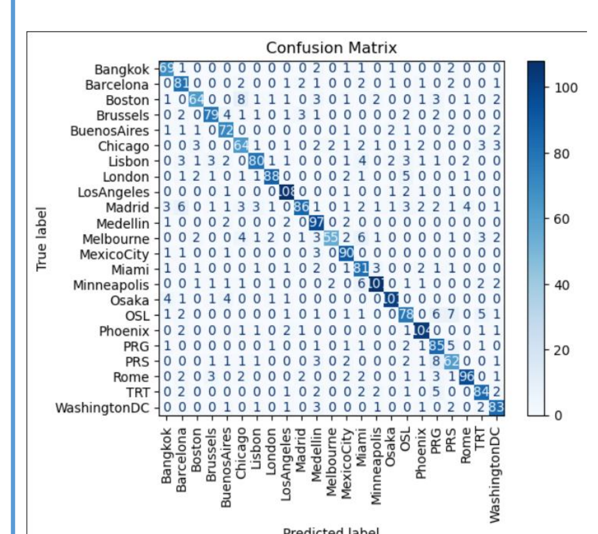
Table 4. Model Performance on Street View API Images

We began with the CLIP zero-shot model as a baseline, achieving 52.3% accuracy despite its lack of training on our dataset. Testing different label formats revealed that using both city and country names significantly outperformed using city or country names alone. However, attempts to fine-tune CLIP provided minimal improvement.

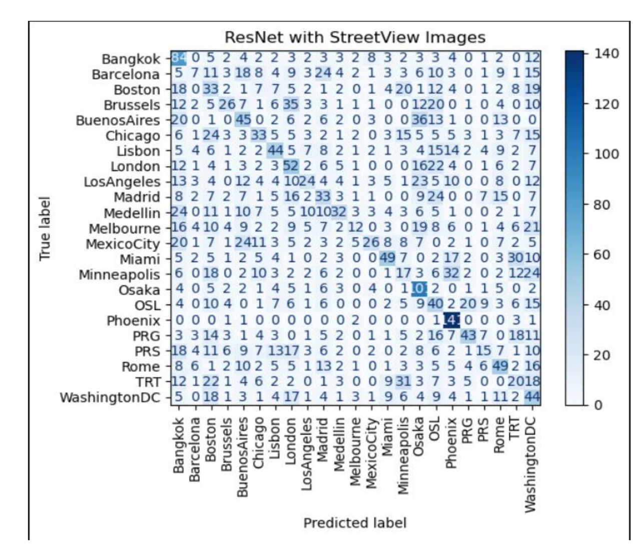
Our CNN model demonstrated poor performance compared to ResNet18 (72% accuracy) and DenseNet (81% accuracy). DenseNet's architecture, with efficient feature reuse and regularization, proved the most effective, outperforming ResNet despite hyperparameter optimization and advanced architectural modifications for ResNet.

Both models faced overfitting issues, with training accuracies over 95% and validation below 80%, even after introducing early stopping. When tested on Google Street View API images, DenseNet showed slightly better generalization (accuracy: 29.4%) than ResNet (accuracy: 27.8%). Interestingly, Phoenix-specific data consistently performed well due to distinctive geographic features like deserts and cacti.

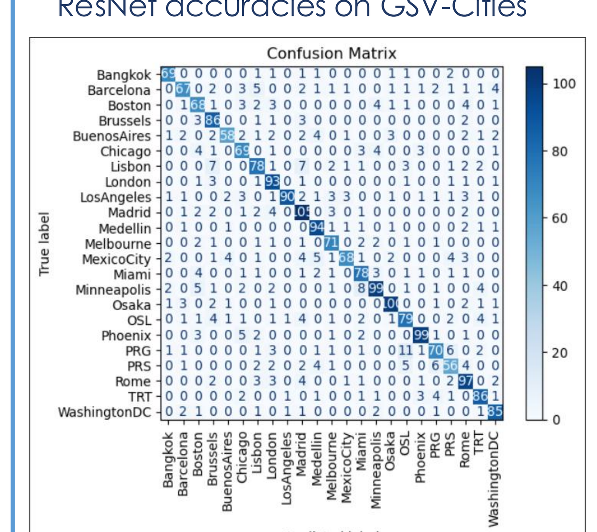
Future work involves integrating Street View API images with GSV-Cities for more varied training data and exploring region-specific fine-tuning to enhance generalization.



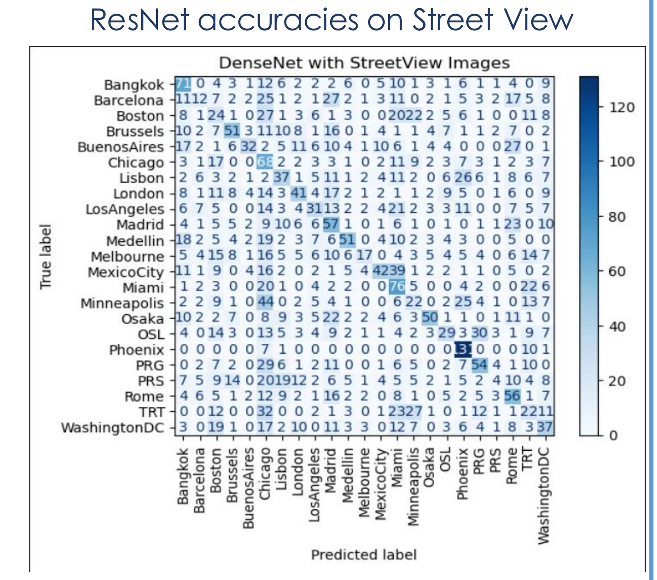
ResNet accuracies on GSV-Cities



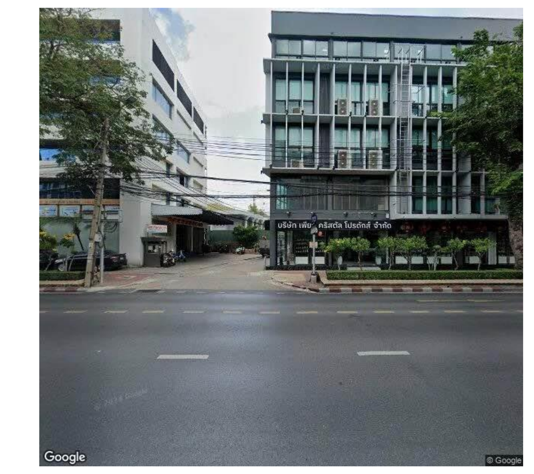
ResNet accuracies on Street View



DenseNet accuracies on GSV-Cities



DenseNet accuracies on Street View



CONCLUSIONS

Key Findings

- Successfully classified cities using street-level images with models like ResNet18, DenseNet, CNNs, and CLIP.
- DenseNet outperformed other models with **81% accuracy**, attributed to its efficient feature reuse and gradient flow.
- Phoenix-specific data consistently showed higher accuracy, suggesting that distinctive geographic features enhance classification.

Challenges

- Limited generalization when testing on Google Street View API images due to dataset biases in GSV-Cities.
- Simpler models, such as CNNs, struggled with the complexity required for high accuracy.

Lessons Learned

- **Dataset Diversity**: City classification depends heavily on the quality and diversity of the dataset.
- **Pretrained Models**: CLIP demonstrated potential for geolocation tasks with limited city-specific data but required more contextual fine-tuning.

Future Directions

- Incorporate additional data sources (e.g., Google Street View API) to improve dataset variability and generalization.
- Explore hybrid models that combine multimodal approaches (e.g., CLIP) with CNN feature extraction.
- Investigate new applications, such as indoor-city detection, country-level classification, and geographic similarity modeling.

REFERENCES

[1] Ali-Bey, A., Chaib-Draa, B., & Giguere, P. (2022). GSV-Cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 513, 194–203. https://doi.org/10.1016/j.neucom.022.09.127

[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., & J´egou, 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

CONTACTS

Pranav Devabhaktuni – pdevabhaktuni6@gatech.edu

Arjun Birthi – abirthi6@gatech.edu

Dhruv Adha – dadha3@gatech.edu

Akshay Raj – araj72j@gatech.edu