Predicting US Cities from Street View Images

CSC 461 Final Project

Jacob Kopacz

Professor Marco Alvarez

December 18, 2024

Introduction

This project, Predicting US Cities from Street View Images, aims to develop machine learning models capable of classifying US cities based on street view images. Cities across the United States exhibit unique visual characteristics shaped by their architecture, landscapes, climate, and local design. These features create a distinct visual identity for each city. By using these differences, the model will aim to accurately predict the city depicted in a given street view image.

The project will compare the performance of different convolutional neural networks (CNNs), to determine the most effective approach for this task. By analyzing the results, we aim to identify differences in how these models interpret visual data and assess their ability to generalize across different street view environments. The ultimate goal is to create a model that will achieve high accuracy for unseen data, demonstrating its practical usability.

To optimize the performance of the aforementioned models, we will experiment with a slew of parameters. For the custom CNNs, this will include adjusting the number of filters, kernel sizes, pooling strategies, dropout rates, and batch normalization. Additionally, we will adjust the learning rates and weight decay during training. These experiments aim to find the best configurations for the custom CNNsand compare their performance with the pretrained ResNet18. By performing these experiments, we aim to identify the most effective configurations for street view image classification.

Problem Definition

The problem of this project is to classify images from Google Street View into the US city it belongs to using convolutional neural networks (CNNs). To achieve this, the project will use three CNN architectures: a simple custom model, a more complex custom model, and a pretrained ResNet18 leveraging transfer learning. Each model must learn the differences in street view images, such as lighting, perspectives, and urban features, which complicates accurate classification. To optimize performance, we will experiment with different parameters such as learning rates, batch sizes, dropout rates, weight decay, pooling strategies, and batch normalization. The goal is to identify the architecture and parameter configurations that achieve the highest accuracy and generalization, providing insights into the relative strengths of custom versus pretrained models for this task.

Data

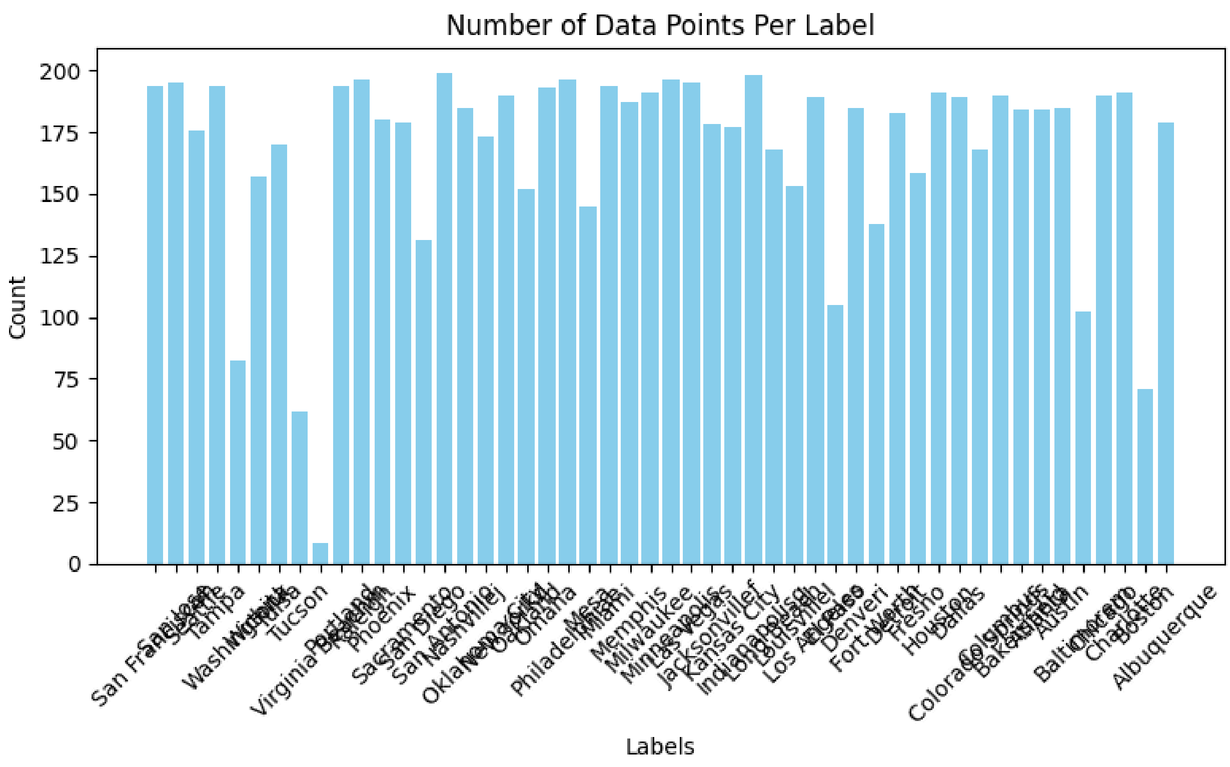
The dataset for this project consists of 8,400 street view images sourced from Google Street View, representing the 50 most populous cities in the United States. Each image is labeled with the corresponding city name and includes the geographical coordinates of the location where it was captured. The images were originally standardized to a resolution of 500x500 pixels.

For the custom CNN models and ResNet18, the images were further processed to fit the input size requirements and to reduce training time/resources. All images were normalized to [-1, 1], and a random horizontal flip was applied. For the custom models, the images were resized to 64x64 pixels to reduce computational load while preserving key visual details for classification.

For ResNet18, the images were resized to 224x224 pixels to match with the input dimensions required byResNet18, ensuring compatibility.

All data was split into training, validation, and test subsets in 60%, 20%, and 20% proportions, respectively. This split was chosen due to the dataset’s small size, with each of the 50 cities represented by approximately 170 images, ensuring a balanced distribution across all subsets.

Example Google Street View Images:
Left: Wichita, KS
Right: Albuquerque, NM



Distribution of Data

Methods

Custom CNN Models

The custom models used in this project include a simple CNN, named SuperTinyCNN, and a more complex architecture, named LayeredCNN, designed to test how different levels of model complexity affect performance. SuperTinyCNN consists of a single convolutional layer followed by a dropout layer and a fully connected layer. In contrast, LayeredCNN was designed with three convolutional layers, followed by dropout layers, optional pooling layers, and a final fully connected layer. Both models use ReLu and linear transformation and have the following parameters:

- **Number of Filters**: Determines the number of feature maps produced by each convolutional layer. Where more filters allow the model to learn more patterns (e.g., edges, textures, shapes) but increase computational complexity.
- **Kernel Size:** Specifies the size of the filter used in the convolution operation (e.g., 3x3 or 5x5). Smaller kernels focus on fine-grained details, while larger kernels capture broader patterns.
- **Pooling**: Boolean, either applied on the first and last layer, or not at all. Reduces the spatial dimensions of feature maps. Reducing computational load, but may cause loss of information.
- **Dropout Rate:** Specifies the probability of randomly setting some neurons to zero during training, helps with preventing overfitting by making the model not rely on specific neurons. Higher dropout rates provide stronger regularization.
- **Batch Normalization:** Normalizes the input of each layer enabling faster convergence and improving generalization.

ResNet18 Model

ResNet18 is made up of 18 layers. These layers include convolutional layers, batch normalization layers, ReLU activation layers, pooling layers, and a single fully connected layer which uses linear transformation. The model is fairly static, not allowing for custom configurations.

Hyperparameter Grid Search

To tune the hyperparameters of all three CNN models a two-step grid search strategy was used. This method used combinations of the parameters learning rates, batch sizes, dropout rates, weight decay, batch normalization, and pooling. The grid search function was used to evaluate these parameter combinations for each model, returning the best training configuration for the model based on the validation and test results.

First, a coarse grid search was performed with 3 epochs for each parameter combination. This approach allowed for a quick evaluation of the parameters, identifying general trends and ruling out poor configurations. After finding the best configuration of the course grid search, a fine grid search was performed using 12 epochs. Here, the best dropout rate, batch normalization, pooling, and batch size was used. While we took the best three of learning rate and weighted decay. The final results of the fine grid search were the final results of our models.

Below are the ranges of the hyperparameters we used. Unfortunately, due to limited resources, we had to reduce the parameter ranges of our coarse grid search.

Original Hyperparameter Tuning

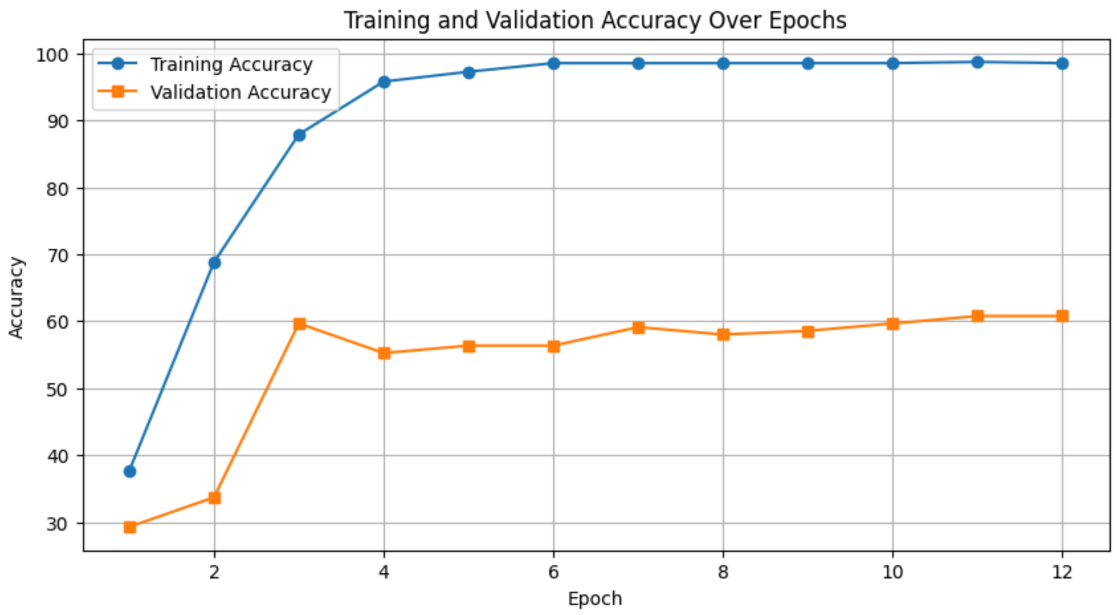
Learning Rate	1e-1	1e-2	1e-3	1e-4
Weight Decay	1e-3	1e-4	1e-5	0
Batch Size	8	16	32	64
Dropout Rate	0.3	0.5	0	
Pooling	True		False	
Batch Normalization	True		False	

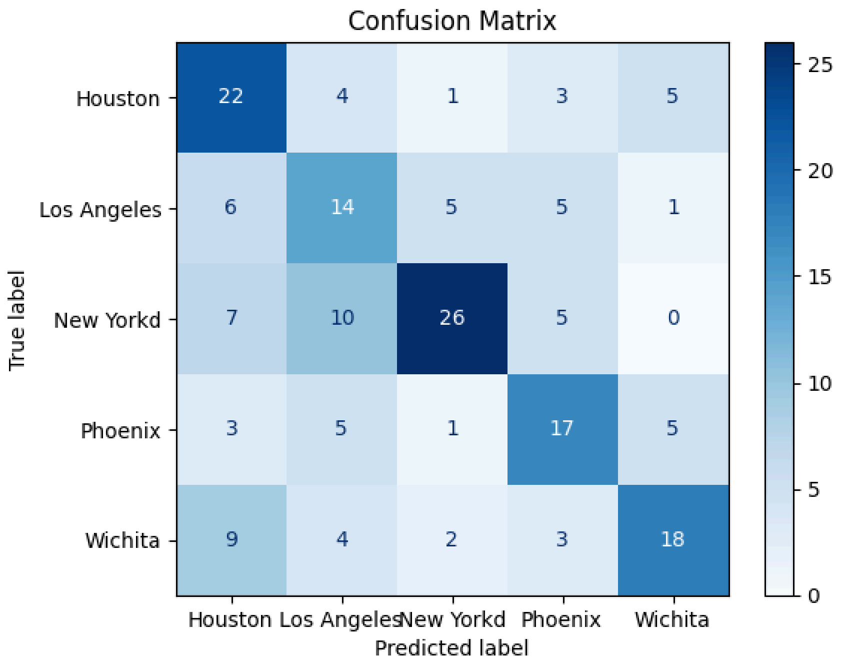
Updated Hyperparameter Tuning

Learning Rate	1e-1	1e-2	1e-3	1e-4
Weight Decay	1e-3	1e-4	1e-5	0
Batch Size	8		16	
Dropout Rate	0.5		0	
Pooling	False			
Batch Normalization	True			

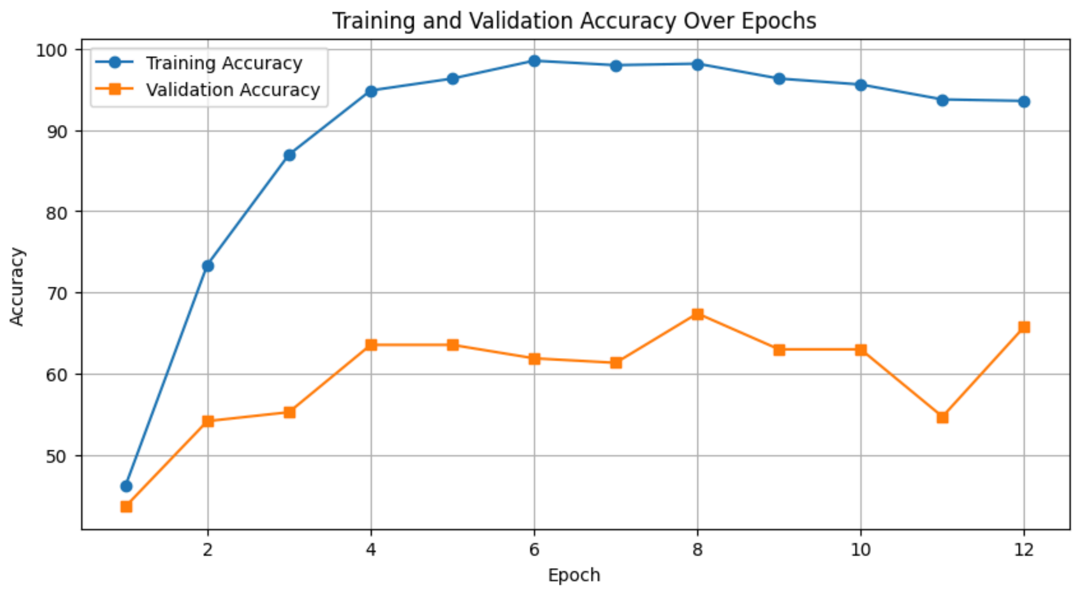
Results and Analysis

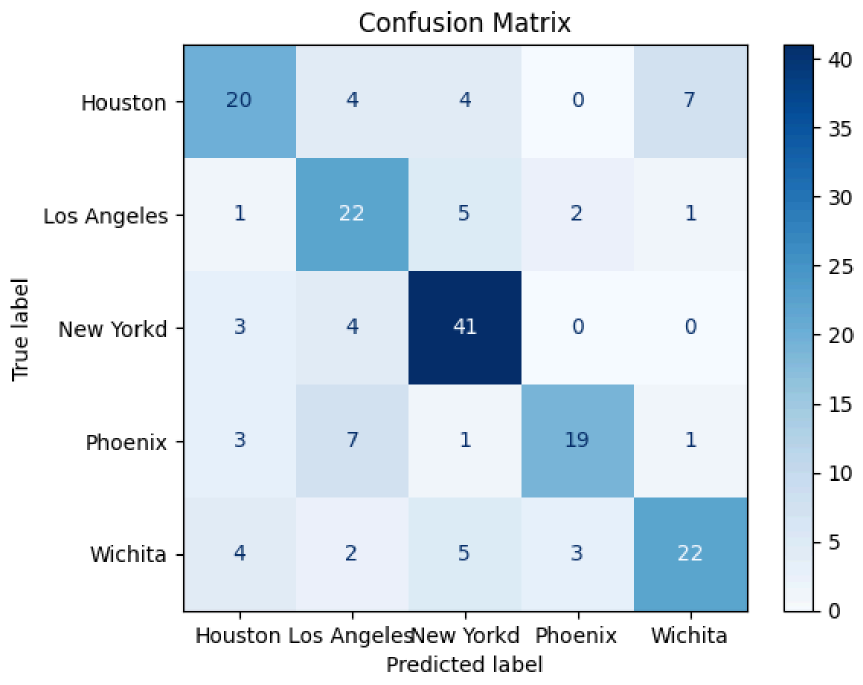
LayeredCNN Results





ResNet18 Results





	Houston	Los Angeles	New Yorkd	Phoenix	Wichita
Houston	20	4	4	0	7
Los Angeles	1	22	5	2	1
New Yorkd	3	4	41	0	0
Phoenix	3	7	1	19	1
Wichita	4	2	5	3	22

Analysis

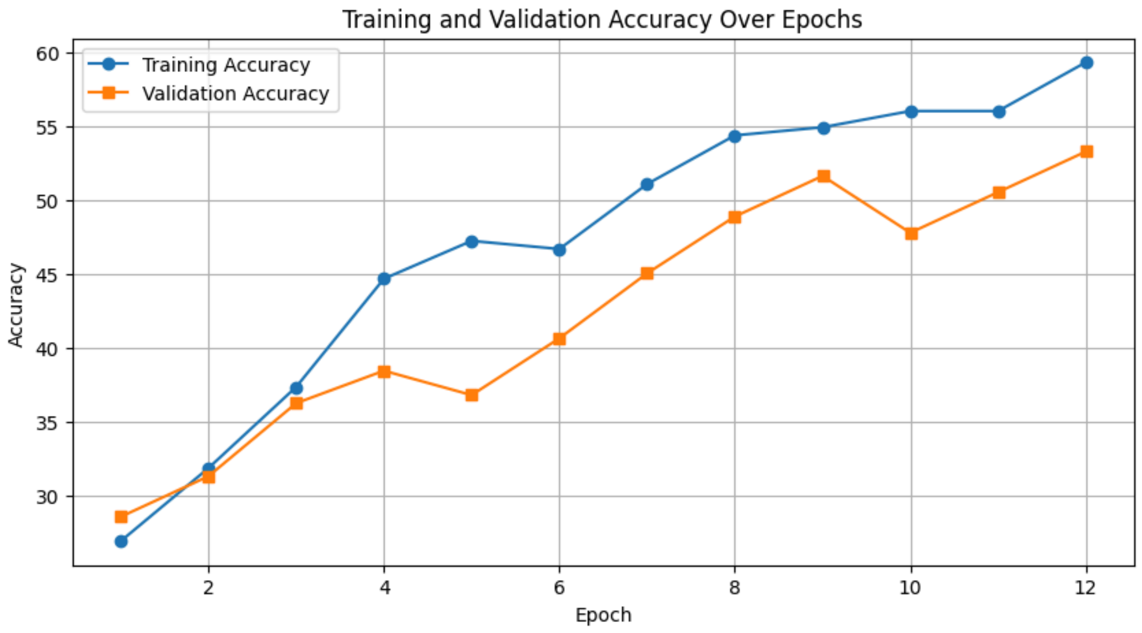
During training, the validation accuracy quickly reached about 60% but then stopped improving, showing that the model had trouble getting better after a certain point. We believe this was likely because the images from different cities weren't as unique as expected. Many cities shared similar features, like buildings or trees, which made it hard for the model to tell them apart. On top of that, we believe objects like cars in the images caused confusion. For example, cars are in almost all New York City street view images, causing the model to mistakenly think that images of other cities which also included cars were images of New York. These similarities made it harder for the model to focus on what actually makes each city different, resulting in a plateauing validation accuracy.

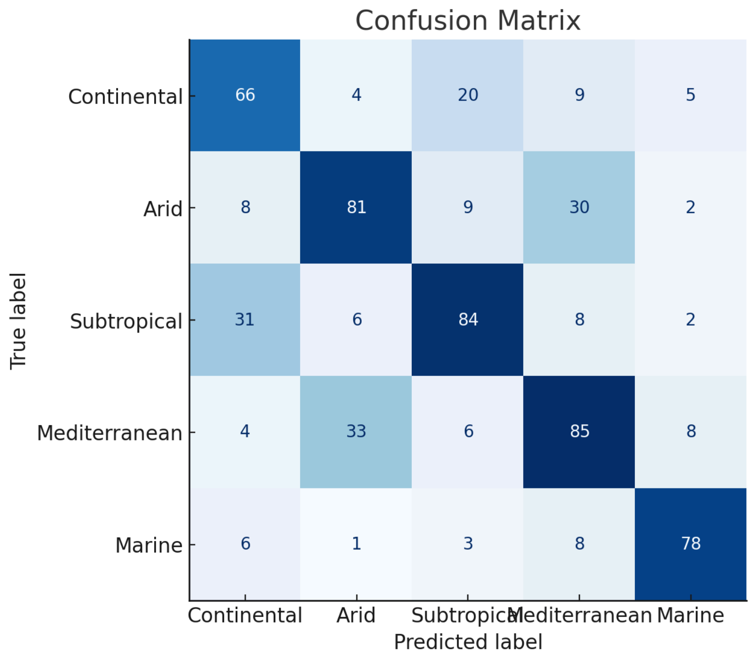
Another contribution to our poor results is that the dataset only had about 170 images per city, which may have affected the models' performances. With a small number of images, the model had limited examples to learn the unique features of each city. This can make it harder to generalize and accurately classify images. Which increased the chances of overfitting, where the model learns the specifics of the training data rather than general patterns. As seen above it is evident that we have severe overfitting. With more images, the model might have been able to better capture the differences between cities and improve its accuracy

Grouping by Climates

After the poor results we had with classifying individual cities, we decided to group the cities into the five climate categories: Continental, Arid, Subtropical, Mediterranean, and Marine. Based on Koppen Climate types. This required reassigning each city's images to its respective climate category while discarding cities that existed on the border of multiple climates or were in a unique climate, such as San Diego and Miami. By focusing on climates instead of individual cities, we aimed to reduce the confusion caused by shared features like cars or similar architecture. We hoped this grouping would allow the models to focus on broader landscape patterns unique to each climate improving classification accuracy.

ResNet Results





True label \ Predicted label	Continental	Arid	Subtropical	Mediterranean	Marine
Continental	66	4	20	9	5
Arid	8	81	9	30	2
Subtropical	31	6	84	8	2
Mediterranean	4	33	6	85	8
Marine	6	1	3	8	78

Conclusion

While this project successfully implemented and compared three CNN models for classifying cities using Google Street View images, the results were subpar. The models struggled to generalize, with validation accuracy plateauing early around 60%, revealing significant limitations in both the dataset and model design. The small dataset size, only ~170 images per city, made it difficult for the models to learn unique patterns. Cities often shared similar visual elements like cars, roads, and climate features, which caused confusion within the models. The attempt to group cities into climates improved focus but did not completely resolve the issue, as some climates (e.g. subtropical and continental) still shared many features(How different do Wichita and Releigh look in Summer?). The lack of distinct training data limited the models' ability to perform well on unseen data.

To improve the project and prevent overfitting, several changes can be considered. First, increasing the dataset size by collecting more images per city or climate would help the models generalize better. Applying data augmentation techniques like random cropping could expand the dataset and improve model robustness. Additionally, using transfer learning with deeper pretrained models, like ResNet50, may also yield better results, as those models are better at extracting meaningful features from complex data. Lastly, implementing early stopping based on validation performance could prevent the models from overfitting to the training data.