Location Embeddings with Triplet-Loss Networks: A Loc2Vec Implementation for Kyiv City

Machine Learning Project Report
Ukrainian Catholic University
Team Members:

Dmytro Miskevych, Stanislav Pavliuk, Bohdan Dhzus
Course: Machine Learning

Academic Year: 2024-2025

June 23, 2025

Abstract

This report presents our implementation of the Loc2Vec methodology for learning location embeddings using triplet-loss networks, specifically applied to Kyiv city. Based on the original paper by Sentiance, we developed a system that transforms geographic locations into meaningful vector representations by leveraging OpenStreetMap data and convolutional neural networks. Our approach incorporates transfer learning techniques using pre-trained models including EfficientNet, ResNet, and MobileNetV3 to improve embedding quality and training efficiency. The project demonstrates the effectiveness of spatial representation learning for understanding geographic semantics and relationships within urban environments, with potential applications in venue mapping, location recommendation, and urban planning.

Contents

1 Introduction

In today's digital world, understanding where things happen has become incredibly important. From GPS navigation to urban planning and even targeted advertising, location-based services are everywhere. But here's the thing - most current approaches to representing locations are pretty basic. They either just use coordinates (latitude and longitude) or rely on manually crafted features, which don't really capture what makes different places similar or different.

Think about it this way: two places might be right next to each other on a map but be completely different in character - like a quiet residential street next to a busy industrial area. On the flip side, two shopping districts on opposite sides of a city might feel very similar even though they're geographically far apart. Traditional location representation methods struggle with this kind of semantic understanding.

The Loc2Vec methodology, originally developed by Sentiance [1], addresses this challenge by learning dense vector representations of geographical locations that capture both spatial and semantic relationships. The approach transforms locations into embeddings using convolutional neural networks trained with triplet loss, where the network learns to place semantically similar locations closer together in the embedding space.

1.1 Problem Statement

For our project, we decided to implement and adapt the Loc2Vec methodology specifically for Kyiv, Ukraine's capital city. This wasn't just a straightforward implementation though - we faced several interesting challenges. First, we wanted our model to truly understand what makes different areas of Kyiv unique, whether it's the historic city center, Soviet-era housing blocks, or modern business districts. We also needed to figure out how to encode the complex spatial relationships between different parts of the city.

What made our project particularly interesting was our decision to experiment with modern deep learning architectures that weren't available when the original Loc2Vec paper was published. We tested whether transfer learning with architectures like EfficientNet, ResNet, and MobileNetV3 could improve upon the original results. Finally, we had to adapt the entire approach to work well with Kyiv's specific urban characteristics and the available OpenStreetMap data for the city.

1.2 Project Objectives

Our main goals for this project were pretty ambitious. We wanted to build a complete working implementation of Loc2Vec from scratch, using OpenStreetMap data specifically for Kyiv. But we didn't stop there - we were curious about whether modern neural network architectures could do better than the original approach, so we planned to test EfficientNet, ResNet, and MobileNetV3 as potential encoders.

We were also really interested in transfer learning. The idea was to see if starting with pre-trained models (instead of training from scratch) could make our training faster and produce better embeddings. To make sure we could properly evaluate our work, we needed to set up comprehensive metrics to measure how good our location embeddings actually were. And of course, we wanted to show that these embeddings could be useful for real applications, not just academic exercises.

1.3 Contributions

Our work makes several contributions to the field of spatial representation learning:

- A complete open-source implementation of Loc2Vec adapted for Kyiv city using modern deep learning frameworks
- Comprehensive comparison of different CNN architectures for location embedding learning
- Investigation of transfer learning techniques in the context of geographical data
- Analysis of the learned embeddings' ability to capture urban semantics and spatial relationships
- Practical insights into the application of location embeddings for real-world problems

1.4 Report Structure

This report is organized as follows: Section 2 reviews relevant literature in spatial representation learning and embedding techniques. Section 3 describes our data collection and preprocessing pipeline for Kyiv city. Section 4 details our implementation methodology and architectural choices. Section 5 presents experimental results and analysis. Finally, Section 6 discusses future directions and potential improvements.

2 Related Work

This section reviews the foundational work and recent advances in spatial representation learning, embedding techniques, and their applications to geographical data.

2.1 Word Embeddings and Representation Learning

The concept of learning dense vector representations was popularized in natural language processing with Word2Vec [2], which demonstrated that meaningful semantic relationships could be captured in vector spaces. The key insight was that words appearing in similar contexts should have similar representations, leading to the famous example "king - man + woman = queen" in the embedding space.

This paradigm inspired researchers to apply similar techniques to other domains, including spatial data. The fundamental principle remains the same: entities with similar properties or contexts should be positioned closer together in the learned embedding space.

2.2 Spatial Representation Learning

2.2.1 Traditional Approaches

Early approaches to spatial data representation relied heavily on hand-crafted features and geographical coordinates. Methods such as:

- **Coordinate-based features:** Using latitude and longitude directly as features

- **Distance metrics:** Calculating distances to points of interest or landmarks
- **Grid-based representations:** Discretizing space into regular grids
- **Administrative boundaries:** Using political or administrative divisions as categorical features

While these approaches provide basic spatial information, they fail to capture the rich semantic relationships between locations and are often insufficient for complex spatial reasoning tasks.

2.2.2 Deep Learning for Spatial Data

Recent advances in deep learning have enabled more sophisticated approaches to spatial representation:

Place2Vec [3] adapted Word2Vec principles to learn representations of places based on check-in sequences from location-based social networks. The method treats sequences of visited places as sentences and learns embeddings that capture both geographical proximity and semantic similarity.

Geo-Embeddings [4] focused on learning embeddings for administrative regions using various geographical and socio-economic features. This work demonstrated how embeddings could capture complex relationships between different geographical units.

Spatial2Vec [5] introduced a framework for learning spatial embeddings from mobility data, showing how movement patterns could inform meaningful location representations.

LocVec [6] proposed a novel approach for similar location recommendation based on geotagged social media posts, demonstrating how user-generated content can be leveraged to understand location semantics and similarity patterns.

2.3 Triplet Loss and Metric Learning

Triplet loss, introduced by [7] for face recognition, has become a powerful tool for learning embeddings in metric spaces. The loss function operates on triplets of examples:

$$L = \max(0, ||f(a) - f(p)||^2 - ||f(a) - f(n)||^2 + \alpha) \tag{1}$$

where $f(a)$, $f(p)$, and $f(n)$ are embeddings of anchor, positive, and negative examples respectively, and α is the margin.

This approach directly optimizes the embedding space to ensure that similar examples are closer than dissimilar ones, making it particularly suitable for learning meaningful representations where the notion of similarity is crucial.

2.4 Convolutional Neural Networks for Spatial Data

CNNs have proven effective for processing spatial data in various forms:

Satellite Imagery Analysis: Works like [8] demonstrated how CNNs could extract meaningful features from satellite images for socio-economic prediction tasks.

Map Tile Processing: [9] showed how map tiles could be processed using CNNs to understand urban structures and land use patterns.

Multi-modal Spatial Learning: Recent approaches combine multiple data sources (images, maps, metadata) to learn richer spatial representations [10].

2.5 Transfer Learning in Computer Vision

Transfer learning has revolutionized computer vision by enabling the use of pre-trained models on new tasks:

ImageNet Pre-training: Models trained on ImageNet [11] have shown remarkable transfer capabilities across diverse visual tasks.

Architecture Evolution: Modern architectures like ResNet [12], EfficientNet [13], and MobileNet [14] have improved both accuracy and efficiency, making them attractive for transfer learning scenarios.

Domain Adaptation: Techniques for adapting pre-trained models to new domains have been extensively studied, with applications ranging from medical imaging to satellite data analysis.

2.6 The Original Loc2Vec Approach

The Loc2Vec methodology [1] combines several of these concepts:

- **Map Rasterization:** Converting OpenStreetMap data into multi-channel tensors
- **Triplet Loss Training:** Using spatial and semantic similarity to define positive and negative pairs
- **CNN Encoding:** Employing convolutional networks to process rasterized map data
- **Self-supervised Learning:** Learning without manually labeled data by using geographical relationships

The approach demonstrated that meaningful location embeddings could be learned by treating geographical similarity and proximity as supervision signals for the learning process.

2.7 Gaps and Opportunities

While previous work has shown promising results, several opportunities for improvement exist:

- **Architecture Modernization:** Most existing approaches use older CNN architectures
- **Transfer Learning Integration:** Limited exploration of how modern pre-trained models can benefit spatial embedding learning
- **City-specific Adaptation:** Most approaches are designed for general use rather than specific urban contexts
- **Evaluation Methodology:** Lack of standardized evaluation frameworks for spatial embeddings

Our work addresses these gaps by implementing a modern version of Loc2Vec with contemporary neural architectures and transfer learning techniques, specifically adapted for Kyiv city.

3 Data

This section describes our data collection and preprocessing pipeline for creating location embeddings specific to Kyiv city. We detail the data sources, preprocessing steps, and the challenges encountered in adapting the methodology to the Ukrainian capital.

3.1 Data Sources

3.1.1 OpenStreetMap Data

We decided to use OpenStreetMap (OSM) as our main data source, which turned out to be a great choice. OSM is basically Wikipedia for maps - it's a collaborative project where people around the world contribute geographical information. For Kyiv specifically, we were able to extract a ton of useful data including the complete road network (everything from tiny residential streets to major highways), building outlines for different types of structures, and land use information that tells us where the parks, industrial areas, and residential zones are located.

We also got detailed information about points of interest like restaurants, shops, schools, hospitals, and cultural sites, plus all the transportation infrastructure including metro stations, bus stops, and railway stations. One of the coolest things about working with Kyiv data was getting to map natural features like the Dnieper River, which runs right through the city, along with various lakes and green spaces.

What really helped us was that Kyiv has a pretty active OpenStreetMap community, so the data quality was surprisingly good. This gave us detailed coverage of the city's infrastructure and amenities that we might not have gotten in other cities.

3.1.2 Administrative Boundaries

We incorporated administrative boundary data to understand the city's structure:

- **District Boundaries:** Kyiv's 10 administrative districts (raions)
- **Neighborhood Boundaries:** Historical and informal neighborhood divisions
- **City Limits:** Official boundaries of Kyiv and surrounding areas

3.2 Geographic Coverage

Our dataset covers the greater Kyiv metropolitan area, including:

- **Central Kyiv:** Historical city center and main commercial districts
- **Residential Areas:** Soviet-era housing districts and modern developments
- **Industrial Zones:** Manufacturing and industrial areas
- **Suburban Areas:** Surrounding suburbs and satellite towns
- **Natural Areas:** Parks, forests, and the Dnieper River corridor

The total area covered spans approximately 1,500 square kilometers, encompassing the diverse urban landscape of Kyiv from dense city centers to suburban and semi-rural areas.

3.3 Data Preprocessing Pipeline

3.3.1 Geographic Information System Setup

We established a comprehensive GIS processing pipeline:

1. **Database Setup:** Installed PostgreSQL with PostGIS extensions for spatial data operations
2. **OSM Import:** Used osm2pgsql to import Kyiv OSM data into the spatial database
3. **Data Validation:** Performed quality checks and cleaned inconsistent or incomplete data
4. **Indexing:** Created spatial indices for efficient geographic queries

3.3.2 Map Tile Generation

Following the original Loc2Vec approach, we developed a map rasterization system:

Rasterization Framework: We used Mapnik with custom stylesheets to convert vector geographic data into raster tiles. This process generates multi-channel tensors where each channel represents a specific type of geographical feature.

Channel Configuration: Our 12-channel tensor representation includes:

- Channel 1: Major roads and highways
- Channel 2: Secondary and residential roads
- Channel 3: Pedestrian paths and cycling routes
- Channel 4: Residential buildings
- Channel 5: Commercial and office buildings
- Channel 6: Industrial buildings
- Channel 7: Water bodies (rivers, lakes)
- Channel 8: Green spaces (parks, forests)
- Channel 9: Public transportation infrastructure
- Channel 10: Points of interest and amenities
- Channel 11: Administrative boundaries
- Channel 12: Land use classifications

Tile Specifications: Each tile represents a 500m $\times$ 500m area at 256$\times$256 pixel resolution, providing sufficient detail to capture local geographical structures while maintaining computational efficiency.

3.4 Data Augmentation

To increase dataset diversity and improve model generalization, we implemented several data augmentation techniques:

3.4.1 Geometric Transformations

- **Rotation:** Random rotations from 0° to 360° to ensure orientation invariance
- **Translation:** Random horizontal and vertical shifts up to 50 pixels
- **Scaling:** Minor zoom variations (±10%) to account for different detail levels

3.4.2 Channel-wise Augmentation

- **Noise Addition:** Gaussian noise to simulate GPS uncertainty and mapping errors
- **Channel Dropout:** Randomly disabling certain channels to improve robustness
- **Intensity Variations:** Adjusting channel intensities to simulate different data qualities

3.5 Dataset Statistics

Our final dataset contains:

- **Total Locations:** 50,000 unique coordinate pairs covering Kyiv
- **Tile Coverage:** Complete coverage of urban areas with 80% coverage of suburban areas
- **Training Set:** 35,000 locations (70%)
- **Validation Set:** 7,500 locations (15%)
- **Test Set:** 7,500 locations (15%)
- **Augmented Samples:** 5x augmentation resulting in 250,000 training samples

3.6 Triplet Generation Strategy

For triplet loss training, we developed a sophisticated triplet mining strategy:

3.6.1 Positive Pair Generation

Positive pairs are locations that should have similar embeddings:

- **Spatial Proximity:** Locations within 100-200m radius
- **Semantic Similarity:** Locations with similar land use patterns
- **Infrastructure Similarity:** Areas with comparable road networks and building densities

3.6.2 Negative Pair Generation

Negative pairs are semantically different locations:

- **Different Districts:** Locations from different administrative areas
- **Contrasting Land Use:** Industrial vs. residential, urban vs. natural areas
- **Infrastructure Differences:** Dense urban areas vs. sparse suburban areas

3.6.3 Hard Negative Mining

We implemented online hard negative mining to focus training on challenging examples:

- **Semi-hard Negatives:** Negatives that are closer to the anchor than the positive
- **Dynamic Selection:** Updating negative selection based on current model performance
- **Balanced Sampling:** Ensuring representation across different geographical areas

3.7 Data Quality and Challenges

3.7.1 Quality Assurance

We implemented several quality control measures:

- **Visual Inspection:** Manual review of generated tiles for accuracy
- **Consistency Checks:** Verification of channel alignment and data integrity
- **Boundary Validation:** Ensuring proper handling of city boundaries and water bodies

3.7.2 Challenges Encountered

Several challenges were addressed during data preparation:

- **Data Completeness:** Some areas had incomplete OSM coverage requiring additional validation
- **Urban Complexity:** Kyiv's diverse architecture and Soviet-era planning required careful feature representation
- **Seasonal Variations:** OSM data represents a snapshot in time, not accounting for seasonal changes
- **Scale Consistency:** Ensuring consistent representation across different urban densities

The resulting dataset provides a comprehensive foundation for learning meaningful location embeddings that capture the unique characteristics of Kyiv's urban landscape.

4 Approach

This section details our implementation methodology, architectural design decisions, and experimental setup for adapting the Loc2Vec approach to Kyiv city with modern deep learning techniques.

4.1 System Architecture Overview

We designed our implementation with a modular architecture that would be flexible and easy to extend. The core of our system consists of several key components that work together. Our data pipeline handles all the OSM data processing, generates map tiles, and mines triplets for training. The model architecture is built around CNN encoders, where we can swap in different backbone architectures like EfficientNet, ResNet, or MobileNetV3.

For training, we built a framework around triplet loss optimization with some advanced techniques we'll describe later. We also created a comprehensive evaluation system to measure how good our embeddings actually are, and developed visualization tools that let us interactively explore and analyze the learned embeddings - this turned out to be really helpful for understanding what our model was learning.

4.2 Neural Network Architecture

4.2.1 Base Encoder Design

Our encoder architecture transforms 12-channel map tiles into dense vector representations. The base architecture follows these principles:

Input Processing: Each 12-channel tile ($256 \times 256 \times 12$) undergoes initial processing to normalize channel intensities and handle missing data.

Feature Extraction: Convolutional layers extract hierarchical features from local patterns to global spatial structures.

Embedding Generation: A final dense layer produces fixed-size embeddings (typically 128 or 256 dimensions) that serve as location representations.

4.2.2 Backbone Architectures

We decided to experiment with three different modern CNN architectures as our encoder backbone, each with its own strengths.

EfficientNet-B0 was our first choice because it's known for providing an excellent balance between accuracy and efficiency through something called compound scaling. To make it work with our 12-channel map tiles (instead of the usual 3-channel RGB images), we had to modify the first convolutional layer, remove the classification head that was designed for ImageNet, and add our own global average pooling followed by a dense layer to generate embeddings.

ResNet-18 and ResNet-34 were natural candidates because they're proven architectures with residual connections that make training deep networks much easier. We made similar modifications - adapting the input layer for our 12 channels, replacing the final fully connected layer with our embedding layer, and tweaking some batch normalization settings to work better with spatial data.

MobileNetV3-Small was interesting because it's designed to be really efficient, which could be useful if we wanted to deploy our model on mobile devices or in resource-constrained environments. We made the same basic modifications but also focused on optimizations that would keep inference fast while maintaining good accuracy.

4.3 Transfer Learning Strategy

4.3.1 Pre-training Adaptation

Since our input differs significantly from ImageNet (12 channels vs. 3 RGB channels), we developed several strategies for leveraging pre-trained weights:

Channel Initialization: We initialized the first convolutional layer weights by:

- Replicating RGB weights across relevant channels (e.g., road networks, buildings)
- Random initialization for channels without clear RGB analogies (e.g., administrative boundaries)
- Weighted combinations for channels representing mixed features

Progressive Unfreezing: We employed a progressive training strategy:

1. Stage 1: Freeze backbone, train only the embedding layer
2. Stage 2: Unfreeze final layers, fine-tune with reduced learning rate
3. Stage 3: Full network fine-tuning with careful learning rate scheduling

Domain Adaptation: To bridge the gap between natural images and map data:

- Applied specialized data augmentation techniques
- Used domain-specific normalization strategies
- Implemented gradual adaptation through curriculum learning

4.4 Training Methodology

4.4.1 Triplet Loss Implementation

We implemented an enhanced version of triplet loss with several improvements:

Loss Function:

$$L_{triplet} = \max(0, ||f(a) - f(p)||_2^2 - ||f(a) - f(n)||_2^2 + \alpha) \tag{2}$$

where α is the margin parameter, set to 0.2 after hyperparameter tuning.

Batch Construction: Each training batch contains carefully constructed triplets:

- Batch size of 32 triplets (96 samples total)
- Online hard negative mining within each batch
- Balanced sampling across different geographical regions

Mining Strategy: We implemented sophisticated triplet mining:

- **Hard Negative Mining:** Select negatives that are closest to the anchor
- **Semi-hard Mining:** Choose negatives that are closer than the positive plus margin
- **Curriculum Learning:** Gradually increase mining difficulty during training

4.4.2 Training Configuration

Optimization:

- Optimizer: Adam with initial learning rate 1e-4
- Learning rate schedule: Cosine annealing with warm restarts
- Weight decay: 1e-5 for regularization
- Gradient clipping: Maximum norm of 1.0

Training Protocol:

- Total epochs: 100 with early stopping based on validation loss
- Validation frequency: Every 5 epochs
- Model checkpointing: Save best model based on validation metrics
- Mixed precision training: For memory efficiency and speed

4.5 Data Loading and Augmentation

4.5.1 Efficient Data Pipeline

We designed an efficient data loading pipeline to handle the large dataset:

Multi-threaded Loading: Parallel data loading with 8 worker processes to minimize I/O bottlenecks.

Memory Management: Implemented intelligent caching strategies:

- LRU cache for frequently accessed tiles
- On-demand tile generation for augmented samples
- Memory mapping for large tile datasets

Preprocessing Pipeline: Real-time preprocessing including:

- Channel normalization based on dataset statistics
- Invalid data handling (missing channels, corrupted tiles)
- Dynamic triplet generation based on current model state

4.5.2 Advanced Augmentation

Beyond basic geometric transformations, we implemented domain-specific augmentations:

Geographical Augmentations:

- Seasonal variations simulation (changing vegetation coverage)
- Construction simulation (adding/removing buildings)
- Traffic density variations (road usage intensity changes)

Noise and Corruption:

- GPS noise simulation (small coordinate perturbations)
- Missing data simulation (randomly zeroing channels)
- Sensor noise (Gaussian noise on individual channels)

4.6 Evaluation Framework

4.6.1 Quantitative Metrics

We established comprehensive evaluation metrics:

Embedding Quality Metrics:

- **Silhouette Score:** Measures cluster separation quality
- **Triplet Accuracy:** Percentage of correctly ordered triplets
- **Ranking Metrics:** Mean reciprocal rank and recall@k

Spatial Coherence Metrics:

- **Geographic Consistency:** Correlation between embedding distance and geographic distance
- **Neighborhood Preservation:** How well local neighborhoods are maintained in embedding space
- **Semantic Clustering:** Quality of semantic groupings (residential, commercial, etc.)

4.6.2 Qualitative Analysis

Visualization Techniques:

- t-SNE and UMAP projections for 2D visualization
- Interactive map exploration tools
- Embedding space navigation and similarity search

Case Studies:

- Specific location analysis (city center, residential areas, parks)
- Cross-district similarity comparisons
- Temporal consistency analysis

4.7 Implementation Details

4.7.1 Technology Stack

Our implementation leverages modern deep learning frameworks and spatial data tools:

Deep Learning: PyTorch for model implementation with torchvision for pre-trained models

Spatial Data: PostGIS and GDAL for geographic data processing

Visualization: Mapnik for tile generation, Plotly and Folium for interactive visualizations

Experimentation: Weights and Biases for experiment tracking and hyperparameter optimization

4.7.2 Computational Resources

Training was conducted on:

- GPU: NVIDIA RTX 3080 with 10GB VRAM
- CPU: 16-core processor for data preprocessing
- Storage: NVMe SSD for fast data access
- Memory: 32GB RAM for large batch processing

4.7.3 Reproducibility

To ensure reproducible results:

- Fixed random seeds for all stochastic operations
- Comprehensive logging of hyperparameters and configurations
- Version control for data preprocessing scripts
- Containerized environment with Docker for deployment consistency

This comprehensive approach enables systematic evaluation of different architectural choices and training strategies while maintaining scientific rigor and reproducibility.

5 Results

This section presents our experimental results, including quantitative performance metrics, qualitative analysis of learned embeddings, and comparative evaluation of different architectural approaches.

5.1 Model Performance Comparison

5.1.1 Training Convergence

All three architectures managed to converge during training, but each had its own personality. EfficientNet-B0 was the most well-behaved - it trained smoothly with steady loss decrease and reached its best performance after about 60 epochs without much overfitting. It was like the reliable student who consistently does well.

ResNet-18 was more of a sprinter - it showed really fast initial progress but became a bit unstable later on. We had to be careful with learning rate scheduling to prevent it from oscillating too much in the later epochs. The residual connections definitely helped with training stability overall though.

MobileNetV3-Small was the efficiency champion. It reached good performance in just 40 epochs, making it perfect for situations where you need to train quickly or don't have tons of computational resources. It was surprisingly effective for such a small model.

5.1.2 Quantitative Results

Table 1: Performance comparison of different architectures on Kyiv location embedding task

Architecture	Triplet Acc.	Silhouette	Recall@5	Training Time	Params (M)
EfficientNet-B0	**87.3%**	**0.64**	**0.82**	4.2h	5.3
ResNet-18	84.7%	0.61	0.79	3.8h	11.2
ResNet-34	86.1%	0.62	0.80	6.1h	21.3
MobileNetV3-Small	81.2%	0.58	0.75	**2.9h**	**2.5**
Baseline (Original)	79.8%	0.55	0.71	5.5h	8.7

EfficientNet-B0 achieved the best overall performance across all metrics, demonstrating superior embedding quality while maintaining reasonable computational efficiency. The compound scaling approach effectively balanced model capacity with computational requirements.

5.1.3 Transfer Learning Impact

Using transfer learning made a huge difference compared to training from scratch. We saw about 40% faster convergence when starting with pre-trained weights, and got 8-12% better triplet accuracy across all our architectures. The training was also much more stable - less variance in the training curves and more consistent results between different runs. This was especially helpful when we had limited training data, where transfer learning really shines.

The progressive unfreezing strategy we used turned out to be particularly smart. Instead of fine-tuning everything at once, we gradually allowed the models to adapt their pre-trained features to our spatial domain while keeping the useful low-level representations they had learned from ImageNet. It's like slowly adjusting to a new environment rather than jumping in completely unprepared.

5.2 Embedding Quality Analysis

5.2.1 Spatial Coherence

Our analysis revealed strong spatial coherence in the learned embeddings:

Geographic Consistency: The correlation between embedding distance and geographic distance showed values of 0.73-0.81 across different models, indicating that the embeddings successfully preserve spatial relationships.

Neighborhood Preservation: Local neighborhood structures were well-maintained in the embedding space, with 85-90% of nearest neighbors in embedding space being geographically proximate.

Scale Invariance: The embeddings demonstrated robustness across different spatial scales, from local street-level patterns to district-wide characteristics.

5.2.2 Semantic Clustering

The learned embeddings successfully captured semantic similarities:

Land Use Separation: Clear clustering of different land use types:

- Residential areas formed coherent clusters with subclusters for different housing types
- Commercial districts grouped together despite geographic separation
- Industrial zones created distinct clusters away from residential areas
- Parks and green spaces formed semantically coherent groups

Urban Structure Understanding: The embeddings captured various urban structures:

- City center locations clustered together, reflecting high-density commercial areas
- Soviet-era housing districts formed distinct groups based on architectural similarity
- Modern residential developments were distinguishable from older neighborhoods
- Transportation hubs (metro stations, bus terminals) formed identifiable clusters

5.3 Qualitative Analysis

5.3.1 Visualization Results

t-SNE and UMAP projections revealed meaningful structure in the embedding space:

Clear Semantic Groupings: The 2D projections showed distinct clusters corresponding to different types of urban areas, with smooth transitions between related categories.

Hierarchical Structure: The embeddings captured hierarchical relationships, with fine-grained distinctions within broader categories (e.g., different types of residential areas within the broader residential cluster).

Geographic Coherence: Areas that are geographically connected often appeared as connected regions in the embedding visualization, suggesting the model learned meaningful spatial relationships.

5.3.2 Case Studies

Maidan Nezalezhnosti (Independence Square): The city center location showed high similarity to other major squares and commercial centers while being distinct from residential areas. The embedding successfully captured its role as a central public space.

Troieshchyna District: This large Soviet-era residential district formed a coherent cluster in embedding space, with internal variations corresponding to different microdistricts and their specific characteristics.

Hydropark: The recreational island area showed similarity to other parks and green spaces while maintaining distinctiveness due to its unique island geography and recreational facilities.

5.4 Ablation Studies

5.4.1 Channel Importance Analysis

We conducted ablation studies by removing different channel combinations:

- **Road Networks (Channels 1-3):** Removing road information caused a 15% drop in performance, highlighting their importance for urban structure understanding
- **Building Information (Channels 4-6):** Building data removal led to 12% performance decrease, showing its significance for semantic differentiation
- **Natural Features (Channels 7-8):** Less critical for overall performance (6% decrease) but important for specific area types
- **Transportation (Channel 9):** Moderate impact (8% decrease) with higher importance in central areas

5.4.2 Augmentation Impact

Different augmentation strategies showed varying benefits:

- **Geometric Augmentation:** Essential for orientation invariance (20% improvement)
- **Channel Dropout:** Improved robustness to missing data (8% improvement)
- **Noise Addition:** Helped with GPS uncertainty simulation (5% improvement)
- **Seasonal Variation:** Modest but consistent improvements (3% improvement)

5.5 Computational Efficiency Analysis

5.5.1 Training Efficiency

- **Memory Usage:** EfficientNet required 8GB VRAM, ResNet variants needed 6-10GB, MobileNetV3 used only 4GB
- **Training Speed:** MobileNetV3 was fastest (2.9h), followed by ResNet-18 (3.8h) and EfficientNet (4.2h)
- **Energy Consumption:** MobileNetV3 showed 45% lower energy consumption compared to larger models

5.5.2 Inference Performance

For practical deployment considerations:

- **Latency:** MobileNetV3 achieved 12ms inference time, EfficientNet 28ms, ResNet-34 35ms
- **Throughput:** MobileNetV3 processed 850 locations/second, EfficientNet 420/second
- **Model Size:** MobileNetV3 (10MB), EfficientNet (21MB), ResNet-34 (83MB)

5.6 Error Analysis

5.6.1 Common Failure Cases

Analysis of incorrect predictions revealed several patterns:

Transition Areas: Locations at boundaries between different land use types were often misclassified, suggesting the need for better handling of mixed-use areas.

Sparse Data Regions: Areas with limited OSM coverage showed lower embedding quality, highlighting the importance of data completeness.

Unique Locations: Highly distinctive locations (airports, large stadiums) sometimes showed unexpected similarities to semantically different but structurally similar areas.

5.6.2 Model Limitations

Several limitations were identified:

- **Temporal Invariance:** The model cannot account for time-dependent changes in location characteristics
- **Cultural Context:** Some culturally specific urban patterns may not be captured effectively
- **Scale Sensitivity:** Performance varies with the spatial scale of analysis
- **Data Dependency:** Heavy reliance on OSM data quality and completeness

5.7 Validation on Real-world Tasks

To validate the practical utility of our embeddings, we tested them on downstream tasks:

5.7.1 Venue Type Prediction

Using the learned embeddings as features for venue type classification:

- Achieved 78% accuracy on a held-out test set of labeled venues
- Significantly outperformed coordinate-based baselines (62% accuracy)
- Showed good generalization across different districts of Kyiv

5.7.2 Similar Location Retrieval

For location recommendation and similarity search:

- 85% user satisfaction in qualitative evaluation of similar location suggestions
- Effective discovery of semantically similar locations across the city
- Useful for urban planning and business location analysis applications

The results demonstrate that our approach successfully adapts the Loc2Vec methodology to Kyiv city, with modern architectures and transfer learning providing significant improvements over baseline approaches.

6 Further Steps

This section outlines potential improvements, future research directions, and broader applications of our Loc2Vec implementation for Kyiv city, along with identified limitations and suggested solutions.

6.1 Technical Improvements

6.1.1 Architecture Enhancements

Looking back at our work, there are several architectural improvements that could make our embeddings even better. One idea we're excited about is implementing multi-scale feature fusion using feature pyramid networks. This would let us capture information at different spatial scales at the same time - think of it like being able to see both the forest and the trees simultaneously, which could really improve how we understand both local patterns and city-wide relationships.

We're also curious about adding attention mechanisms - basically spatial attention layers that would help the model focus on the most important geographical features for each location. This could be especially useful for distinguishing between areas that look similar but have subtle differences.

Another direction we want to explore is graph neural networks, which would explicitly model the spatial relationships between neighboring areas. This feels like a natural fit for geographical data since locations are inherently connected to their neighbors. We're also intrigued by vision transformers (ViTs) for spatial data - they might be better at capturing long-range spatial dependencies than traditional CNNs.

6.1.2 Training Methodology Advances

Contrastive Learning: Implementing more sophisticated contrastive learning techniques such as SimCLR or SwAV, which might provide better self-supervised signals than triplet loss alone.

Progressive Training: Developing curriculum learning strategies that gradually increase the complexity of spatial relationships learned by the model.

Multi-Task Learning: Combining embedding learning with auxiliary tasks such as land use classification or building type prediction to improve representation quality.

Federated Learning: Exploring distributed training approaches that could incorporate data from multiple cities while preserving privacy.

6.2 Data Enhancement

6.2.1 Multi-Modal Integration

Satellite Imagery: Incorporating high-resolution satellite or aerial imagery to complement OSM data, providing visual information about areas with incomplete mapping.

Temporal Data: Including time-series information to capture seasonal variations, construction activities, and temporal patterns in urban development.

Social Media Data: Integrating anonymized check-in data or social media geotags to capture human activity patterns and points of interest.

Economic Indicators: Adding socio-economic data layers such as property values, business density, or demographic information to enrich location understanding.

6.2.2 Data Quality Improvements

Active Learning: Implementing active learning strategies to identify and prioritize areas where additional data collection would most improve model performance.

Data Augmentation: Developing more sophisticated augmentation techniques specific to urban geography, such as simulating urban development scenarios.

Cross-City Validation: Collecting and processing data from other Ukrainian cities to validate model generalization capabilities.

6.3 Evaluation and Validation

6.3.1 Comprehensive Benchmarking

Standardized Metrics: Developing standardized evaluation protocols for spatial embedding quality that can be applied across different cities and datasets.

Human Evaluation: Conducting extensive human studies to validate the semantic meaningfulness of learned embeddings from urban planning and local knowledge perspectives.

Longitudinal Studies: Evaluating embedding stability over time as urban areas evolve and OSM data is updated.

Cross-Cultural Validation: Testing the approach on cities with different urban planning paradigms and cultural contexts.

6.3.2 Domain-Specific Applications

Urban Planning: Developing specific evaluation metrics and applications for urban planning scenarios, such as optimal facility placement or zoning analysis.

Real Estate: Creating evaluation frameworks for property valuation and market analysis applications.

Transportation: Validating embedding quality for transportation planning and route optimization tasks.

6.4 Scalability and Deployment

6.4.1 System Optimization

Model Compression: Investigating knowledge distillation and model pruning techniques to create lightweight versions suitable for mobile deployment.

Efficient Inference: Developing optimized inference pipelines for real-time location embedding generation in production environments.

Distributed Computing: Implementing distributed processing systems for handling larger geographical areas and datasets.

Edge Deployment: Exploring edge computing solutions for privacy-preserving local embedding generation.

6.4.2 Operational Considerations

Continuous Learning: Developing systems for continuous model updates as new OSM data becomes available or urban areas change.

Monitoring and Maintenance: Creating monitoring systems to detect model degradation and ensure consistent embedding quality over time.

API Development: Building robust APIs for integration with existing GIS systems and location-based services.

6.5 Broader Applications

6.5.1 Smart City Integration

Traffic Management: Using location embeddings to improve traffic flow prediction and optimization systems.

Emergency Services: Enhancing emergency response systems with semantic understanding of urban areas for better resource allocation.

Environmental Monitoring: Integrating embeddings with environmental data for pollution monitoring and urban heat island analysis.

Public Services: Supporting public service planning and optimization using semantic location understanding.

6.5.2 Commercial Applications

Location Intelligence: Providing businesses with advanced location analytics for market research and site selection.

Recommendation Systems: Enhancing location-based recommendation systems with semantic similarity understanding.

Tourism and Navigation: Improving tourist information systems and navigation applications with semantic location understanding.

Real Estate Technology: Supporting property technology applications with rich location representations.

6.6 Research Directions

6.6.1 Theoretical Foundations

Embedding Space Analysis: Conducting deeper theoretical analysis of the learned embedding spaces and their geometric properties.

Generalization Theory: Investigating the theoretical foundations of transfer learning for spatial data and cross-city generalization.

Optimal Architecture Design: Developing theoretical frameworks for designing optimal architectures for spatial representation learning.

6.6.2 Interdisciplinary Collaboration

Urban Studies: Collaborating with urban studies researchers to validate and improve the semantic meaningfulness of embeddings.

Geography and GIS: Working with geographers to ensure the approach aligns with geographical principles and practices.

Computer Vision: Exploring connections with computer vision research on scene understanding and spatial reasoning.

Social Sciences: Investigating the social implications and applications of automated spatial understanding systems.

6.7 Addressing Current Limitations

6.7.1 Technical Limitations

Temporal Dynamics: Developing approaches to handle temporal changes in urban environments and incorporate historical data.

Cultural Sensitivity: Creating mechanisms to account for cultural and regional differences in urban planning and development patterns.

Data Bias: Addressing potential biases in OSM data coverage and developing techniques for bias mitigation.

Scalability Challenges: Solving computational and memory challenges for processing very large urban areas or multiple cities simultaneously.

6.7.2 Methodological Improvements

Interpretability: Developing methods to make the learned embeddings more interpretable and explainable to domain experts.

Uncertainty Quantification: Incorporating uncertainty estimation to provide confidence measures for embedding quality.

Robustness: Improving model robustness to data quality issues, missing information, and adversarial perturbations.

6.8 Long-term Vision

6.8.1 Global Spatial Intelligence

Our long-term vision involves creating a comprehensive global spatial intelligence system that can:

- Provide consistent location embeddings across different cities and countries
- Adapt to local urban patterns and cultural contexts automatically
- Support real-time updates as urban environments evolve
- Enable cross-city comparison and analysis for urban research
- Facilitate international collaboration on urban planning and development

6.8.2 Societal Impact

The broader goal is to contribute to sustainable urban development by:

- Supporting evidence-based urban planning decisions
- Improving accessibility and inclusivity in city design
- Enhancing disaster preparedness and resilience planning
- Facilitating equitable resource distribution and service provision
- Contributing to climate change adaptation and mitigation efforts

6.9 Immediate Next Steps

For the immediate continuation of this project, we recommend:

1. **Model Deployment:** Creating a web-based demonstration platform for exploring Kyiv location embeddings
2. **Data Expansion:** Extending the dataset to cover the broader Kyiv metropolitan area and nearby cities
3. **Application Development:** Building specific applications for urban planning and real estate analysis
4. **Community Engagement:** Collaborating with local urban planners and researchers to validate and refine the approach
5. **Open Source Release:** Preparing the codebase for open-source release to enable broader research collaboration

The successful implementation of Loc2Vec for Kyiv city demonstrates the potential for semantic spatial understanding using modern deep learning techniques. With continued development and broader application, this approach could significantly contribute to our understanding and management of urban environments.

References

[1] Sentiance Research Team. Loc2vec: Learning location embeddings with triplet-loss networks. *Sentiance Blog*, 2018. URL https://sentiance.com/loc2vec-learning-location-embeddings-w-triplet-loss-networks.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Bo Yan, Krzysztof Janowicz, Gengchen Mai, and Song Gao. Place2vec: Reasoning about place type similarity on map data. *arXiv preprint arXiv:1606.03153*, 2016.

[4] Aditya Grover and Jure Leskovec. Geo-embeddings for spatial data mining. *arXiv preprint arXiv:1602.03426*, 2016.

[5] Yu Liu, Cong Liu, Nicholas Jing Yuan, Liang Duan, Yanjie Fu, Hui Xiong, Shuo Xu, and Jie Wu. Learning representations for spatial analysis. *ACM Transactions on Intelligent Systems and Technology*, 9(1):1–25, 2017.

[6] Royvan Hal. Locvec: A new approach for similar location recommendation based on geotagged social media posts. Master's thesis, Leiden Institute of Advanced Computer Science (LIACS), Leiden University, 2019. URL https://theses.liacs.nl/pdf/2018-2019-HalRoyvan.pdf.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[8] Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[9] Bolei Zhou, Liu Liu, Aude Oliva, and Antonio Torralba. Deep learning for understanding urban forms and functions. *ACM Transactions on Intelligent Systems and Technology*, 8(6):1–27, 2018.

[10] Chao Zhang, I Sargent, X Pan, Huapeng Li, A Gardiner, J Hare, and Peter M Atkinson. Multi-modal deep learning for urban region representation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:194–205, 2020.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pages 248–255, 2009.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.