**Transportation network perimeter identification**

Department of Computer and Electrical Engineering  
Technion

Lap : CRML

Main supervisor: Ayal Taitler

Students:  
Hadeel Rayan – 207731746  
Yasmeen khalid – 208035220

**Abstract**

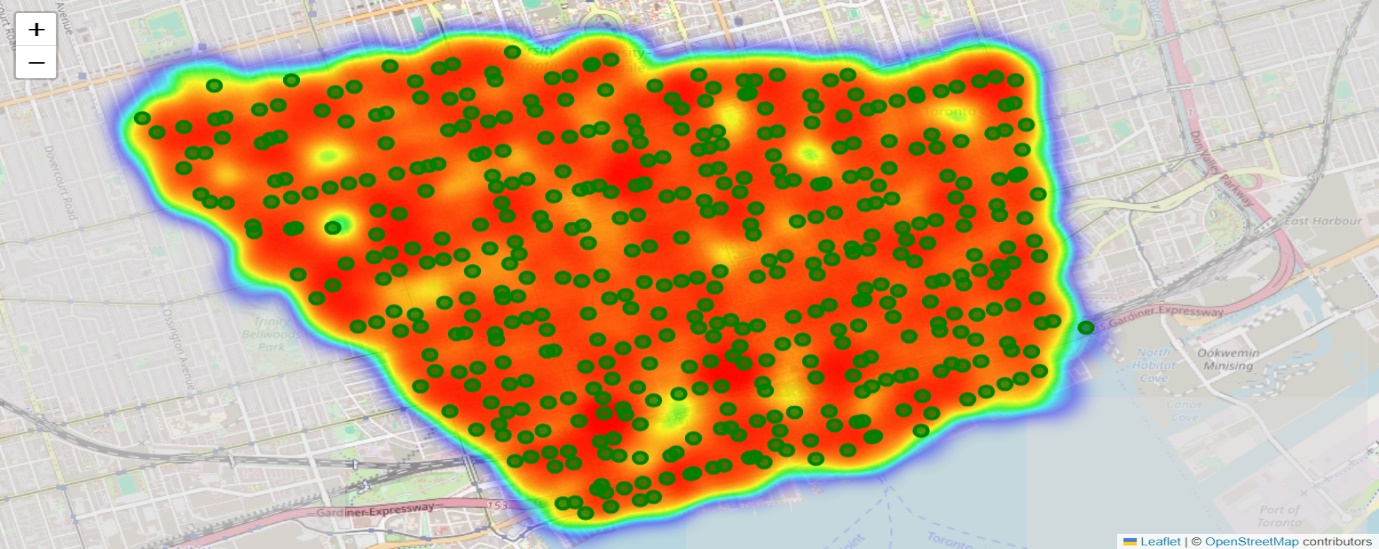
This project identifies high-interest zones within urban environments to improve traffic flow and optimize urban infrastructure planning. Our approach combines advanced heatmap analysis with K-Means clustering to outpoint activity hotspots and employes reinforcement learning to optimize computational efficiency.  
Preliminary results demonstrate efficient clustering of interesting and accurate identification of critical zones, providing actionable insights for urban planning. This framework has the potential to transform urban planning processes, offering a scalable and dynamic solution for managing rapidly evolving city landscapes.

**Introduction**

With the rapid growth of urban areas, identifying key zones of activity within cities has become essential for optimizing infrastructure, managing traffic, and creating safer, more vibrant environments. As of 2023, more than 56% of the global population resides in urban areas, a figure expected to reach 68% by 2050. This rapid urbanization presents significant challenges, including traffic congestion, increased pollution, and inefficient infrastructure use. Identifying high-interest zones within cities is critical to addressing these issues effectively and enabling sustainable urban development.  
To address these challenges, this project develops a dynamic framework for identifying and visualizing high-interest zones in urban environments. By leveraging geospatial traffic data from TomTom, a leading provider of navigation and traffic information, the framework integrates advanced heatmap analysis, K-Means clustering, and reinforcement learning techniques. The TomTom dataset, which provides precise data on GPS coordinates (latitude and longitude) , speed limit and average speed, forms the foundation for generating actionable insights and optimizing computational resources.  
The framework begins by generating geospatial heatmaps to visualize activity intensity. K-Means clustering is employed to group critical intersections, while reinforcement learning, powered by a neural network and the DQN algorithm, enhances computational efficiency and improves clustering accuracy. The DQN algorithm, a deep reinforcement learning technique, allows the framework to make adaptive and data-driven decisions by learning optimal policies for intersection clustering. Unlike traditional methods that rely on static traffic simulations or isolated heatmap analyses, this project introduces a dynamic, AI-driven approach that adapts to urban complexities.  
The insights generated by this framework have applications beyond traffic optimization. Urban planners can use these findings to enhance public transportation routes, improve pedestrian safety in high-density areas, and develop data-driven strategies for emergency response. By integrating neural networks and reinforcement learning into the analysis process, the framework provides a scalable and intelligent solution for managing dynamic city landscapes.  
This paper outlines the methodology, tools, and findings of the project. It begins with a discussion of data acquisition from TomTom and its processing into heatmaps. Subsequent sections present the analysis and findings, including heatmap visualizations, clustering results, and the role of the DQN algorithm in optimizing these processes. Finally, the paper explores the broader implications of this framework and highlights directions for future work.

**Background**

Urban Analysis: Urban analysis is the study of city structures, dynamics, and interactions to better understand how urban areas function and evolve. By leveraging data from sources like TomTom, urban analysis helps uncover patterns in mobility, population density, and infrastructure usage. This understanding enables city planners, policymakers, and businesses to optimize resources, improve traffic flow, and enhance urban livability. Modern urban analysis often relies on advanced tools like heatmaps, clustering algorithms, and machine learning to visualize and process complex data, making it a critical component in shaping smarter and more sustainable cities.  
Urban analysis tools, such as geospatial mapping and clustering algorithms, are instrumental in identifying traffic bottlenecks, improving public transport networks, and planning pedestrian-friendly zones. This project extends these capabilities by integrating advanced heatmap visualizations and machine learning techniques to derive actionable insights.

Heatmap: A heatmap is a powerful visualization tool that represents the density or intensity of data over a geographical area using a gradient of colors. Warmer colors (such as red, orange, and yellow) typically indicate regions with higher activity, while cooler colors (like blue or green) signify lower levels of activity. In this project, the heatmap serves as a key component for analyzing urban environments by visualizing areas of high-interest activity based on traffic data.  
The heatmap in this work is generated using geospatial traffic data from TomTom, which includes traffic flow (speed limit and average speed), and intersection density across a specific urban area. Each point of interest, such as intersections or road segments, contributes to the overall activity level in a particular region. The heatmap highlights these high-activity zones, enabling the identification of traffic congestion areas or critical hotspots that require further analysis.   
In summary, the heatmap plays a crucial role in identifying high-activity zones by transforming complex geospatial traffic data into an intuitive, visual format. It provides a foundation for deeper analysis using clustering techniques and machine learning, ultimately supporting urban planning decisions that aim to optimize traffic flow and infrastructure usage.  
Clustering algorithm: Clustering algorithms are unsupervised machine learning techniques used to group data points based on their similarity or proximity. In the context of this project, clustering algorithms, specifically K-Means, are employed to group high-activity intersections identified on the heatmap. These algorithms analyze geospatial traffic data, such as vehicle density or intersection activity, and partition the data into clusters that represent regions with similar activity patterns.  
In this project, K-Means helps to reduce computational complexity by grouping closely related intersections or traffic points into clusters.   
This enables more efficient analysis of critical areas, as the algorithm focuses on meaningful regions rather than individual data points. By integrating clustering with heatmap visualization, the project identifies zones of high interest, optimizes urban infrastructure planning, and provides actionable insights for managing traffic congestion effectively.

Neural Network: A Neural Network is a machine learning model inspired by the human brain, designed to recognize patterns and learn relationships within data. It consists of layers of interconnected artificial neurons:   
an input layer that receives data  
hidden layers that process the data using weights, biases, and activation functions  
an output layer that provides the final predictions.   
Each neuron calculates a weighted sum of its inputs and applies a non-linear activation function to determine its output. During training, the network adjusts its weights and biases through a process called backpropagation, using optimization techniques like Gradient Descent to minimize errors. Neural networks excel at tasks such as classification, regression, and feature extraction, making them a powerful tool for complex problems like traffic analysis, clustering, and decision-making.

Reinforcement Learning: Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment to achieve a specific goal. The agent takes actions in the environment, observes the outcomes, and receives rewards based on how effective its actions are. Over time, the agent learns an optimal policy, a strategy for selecting actions that maximize cumulative rewards. In this project, RL is implemented using the DQN algorithm, which combines reinforcement learning with deep neural networks. By integrating RL, the project enhances the computational efficiency and accuracy of clustering processes, ensuring more intelligent and adaptive urban planning solutions.  
  
Deep Q-Network (DQN): The DQN is a reinforcement learning algorithm that combines Q-Learning with deep neural networks to solve decision-making problems in high-dimensional spaces. It is particularly effective for tasks where the environment is complex, and the state space or the possible conditions is too large to handle with traditional Q-Learning.

**Neural Network**

A Neural Network (NN) is a machine learning model inspired by the structure and functioning of the human brain. It is composed of artificial neurons organized into layers that process and learn from data. Neural Networks excel at solving problems such as pattern recognition, classification, regression, and decision-making.

The fundamental goal of a neural network is to approximate complex relationships within data by learning mappings between inputs and outputs.  
Structure of a Neural Network:  
A basic Artificial Neural Network (ANN) consists of three main layers:

1. Input Layer:   
   This layer takes in the raw input features like pixel values in images.  
   Each neuron in this layer represents a single input feature
2. Hidden Layers:  
   These layers perform the computation and transformations on the input data.  
   Each hidden layer consists of neurons (units) connected to neurons from the previous and next layers.  
   The number of hidden layers and neurons defines the network architecture.
3. Output Layer:  
   This layer provides the final predictions or outputs based on the data processing in the hidden layers.  
   For classification tasks, the output could be a probability for each class; for regression, it could be a numerical value.

Components of a Neural Network:

1. Neuron:  
   Each neuron performs the following steps:

* Weighted Sum: Combine the input values with learned weights.  
  where:  
   - Input values  
   - Weights associated with each input  
   - Bias term
* Activation Function: Apply a non-linear function to the weighted sum to introduce non-linearity. This allows the network to learn complex relationships.

1. Weights and Biases:

* Weights - determine the importance of each input feature. These are adjusted during training to minimize errors
* Biases - allow the activation of neurons to shift up or down, improving flexibility.

1. Forward Propagation:  
   Data moves forward through the network:  
   - Input data is passed to the input layer  
   - Each hidden layer processes the data using weights, biases, and activation functions.  
   - The output layer produces predictions.
2. Loss Function:  
   The loss function measures the difference between the predicted output and the true output. The network aims to minimize this loss during training.
3. Backpropagation:  
   - Backpropagation is the process of updating the weights and biases to reduce the loss  
   - The gradient of the loss function with respect to each weight is computed using the chain rule of calculus.  
   - Gradients are then used to adjust the weights.
4. Optimization:  
   Optimizers, such as Stochastic Gradient Descent (SGD) or Adam, are used to update weights and biases iteratively:

: Loss function.  
: Learning rate.  
​: Gradient of the loss with respect to weights.

Training a Neural Network:  
The training process of a neural network consists of the following steps:

1. Initialization: Initialize weights and biases randomly or with specific strategies.
2. Forward Propagation: Compute the output for given input data.
3. Loss Calculation: Compare predictions with true values using the loss function.
4. Backpropagation: Compute gradients and update weights using optimization algorithms.
5. Repeat: Repeat steps 2-4 for multiple epochs (iterations) until the network converges.

Advantages of Neural Network:  
Neural Networks are powerful because they can:

* Model Non-Linear Relationships: Activation functions allow them to capture complex relationships in data.
* Learn Automatically: They do not require explicit programming rules but instead learn from data.
* Generalize Across Domains: They work well for images, time series, natural language processing, and more.

**Reinforcement Learning with DQN**

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with an environment. The agent takes actions in the environment, observes the outcomes, and receives rewards based on the effectiveness of those actions. The goal of the agent is to learn an optimal policy, a strategy that maximizes the cumulative rewards over time.  
The RL process can be described as a Markov Decision Process (MDP), which includes:

1. States : The current condition or situation of the environment.
2. Actions : The decisions or moves the agent can take in each state.
3. Rewards : A numerical value indicating the outcome of an action.
4. Policy 𝞹: A mapping from states to actions
5. Transition Dynamics : The probability of moving from one state to another after taking a particular action.
6. Discount Factor γ: A value between 0 and 1 that determines the importance of future rewards.

Q-Learning is a value-based RL algorithm that learns the Q-value for each (state, action) pair where:

* represents the expected cumulative reward of taking action in state and following the optimal policy thereafter.  
  The Q-values are updated iteratively using the Bellman Equation:
* α is the learning rate
* is the reward received after taking action
* γ is the discount factor.
* is the maximum value of the next state .

Limitation of Q-Learning:  
When the state space is very large (e.g., urban traffic networks), Q-values cannot be stored in a table because the number of states and actions becomes impractically large. This is where DQN comes in.  
Overview:  
DQN combines Q-Learning with deep neural networks to approximate the Q-values. Instead of maintaining a Q-table, DQN uses a neural network to predict the Q-values for all possible actions in each state.  
How DQN Works:

1. Neural Network as a Q-Function Approximator:

* A deep neural network takes the current state as input and outputs the Q-values for all possible actions.
* The network approximates the mapping

1. Replay Buffer:

* DQN uses a replay buffer which is a memory bank to store past experiences as tuples
* During training, batches of these experiences are sampled randomly to break the correlation between consecutive experiences, leading to more stable learning.

1. Target Network:

* DQN uses two networks:   
  Main Q-Network: Learns the current Q-values.  
  Target Q-Network: Provides stable target Q-values during training
* The target network is updated periodically with the weights of the main Q-network to stabilize training.

1. Loss Function:

* The DQN algorithm minimizes the difference between the predicted Q-value and the target Q-value:
* is the Q-value predicted by the main network.
* is the Q-value from the target network

1. Epsilon-Greedy Exploration:

* To balance exploration and exploitation using known optimal actions, DQN uses an epsilon-greedy policy:  
  with probability ϵ take a random action.  
  with probability , take the action with the highest Q-value.

1. Training Process:

* The agent interacts with the environment and stores experiences in the replay buffer.
* A batch of experiences is sampled from the buffer.
* The Q-values are updated using the neural network and the target network.
* This process repeats until the agent learns an optimal policy.

Advantages of DQN:  
Scalability- DQN can handle large state spaces where traditional Q-Learning fails.  
Stability- The use of replay buffers and target networks improves the stability of training.  
Generalization- The neural network enables the agent to generalize learning across similar states.

**DQN in This Project**

The primary objective of the DQN implementation is to:  
Maximize the coverage of high-intensity zones (based on the heatmap).  
Minimize computational overhead by efficiently selecting intersections to form the convex hull.  
Encourage exploration of new configurations while penalizing redundant or unproductive actions.

State:   
The current clustering of intersections and activity zones derived from the heatmap, the state includes the following components:

* Heatmap Data: represents traffic intensity and critical zones visually.
* Convex Hull Mask: Encodes the current shape of the convex hull formed by selected intersections.
* Dot Indicators: Denotes the state (active or inactive) of each intersection.

Actions:  
Decisions related to toggling intersections in or out of the convex hull, Each action corresponds to selecting or deselecting a specific intersection, which modifies the shape and coverage of the convex hull.

Reward:   
The reward function is designed to balance the following objectives:

1. Encouraging Area Expansion:  
   the reward is proportional to the increase in the area covered by the convex hull. A positive reward is given when the addition of an intersection significantly increases the convex hull area.
2. Penalizing Redundant Actions:  
   repeated actions on already toggled intersections incur a heavy penalty.
3. Avoiding Insignificant Changes:  
   small or negligible changes in the convex hull area are penalized to discourage unproductive actions

Summary of Workflow:

* Generate heatmaps to identify activity intensity.
* Use K-Means clustering to group intersections.
* Train the RL agent using DQN to learn optimal clustering actions:  
  explore different clustering configurations.  
  learn from rewards to optimize clustering efficiency.
* Continuously update the model to adapt to real-world changes in traffic patterns.

This integration of DQN ensures a dynamic, AI-driven framework for urban analysis, providing scalable and intelligent solutions for managing city landscapes. Over time, the system dynamically adapts to changes in traffic patterns, identifying and optimizing the most critical zones for urban infrastructure planning.

**Methodology**

**Data Acquisition:** Efficient data acquisition forms the foundation of the project, enabling accurate identification and analysis of high-interest zones within urban environments. The primary goal is to gather, preprocess, and structure geospatial traffic data for generating heatmaps and applying clustering techniques. The data acquisition process is tailored to ensure high-quality inputs for machine learning models and reinforcement learning algorithms.  
Sources of Data:

We used the source TomTom, its precision and comprehensiveness make it an ideal choice for traffic analysis. Geospatial Traffic Data from TomTom, Includes:  
GPS coordinates of road segments and intersections.  
Traffic flow and density data  
Speed limits and average speed measurements

Preprocessing Steps:

1. Data Cleaning
2. Filtering for Relevance
3. Feature Engineering
4. Spatial Data Transformation

Data Pipeline Automation:  
The data acquisition pipeline is automated to streamline the process using custom Python scripts:  
Extract JSON Data: Processes raw traffic data, cleans it, and generates an analysis-ready CSV (as implemented in extractJsonnData.py)​.  
Filter Relevant Data: Ensures only useful traffic points are passed into subsequent analysis stages.

Conclusion:  
The data acquisition process is a critical first step in creating a reliable framework for traffic analysis. By ensuring data quality and consistency, the project lays a strong foundation for generating accurate heatmaps, clustering intersections, and deriving actionable insights for urban planning.

**Heatmap Generation:**

Heatmaps are a vital visualization tool in this project, enabling the identification of high-interest zones by transforming complex geospatial traffic data into an intuitive, color-coded representation. The heatmap highlights traffic intensity across urban environments, with warmer colors (e.g., red) indicating areas of high activity and cooler colors (e.g., green or blue) representing lower activity levels.

This step bridges the raw data collected during the Data Acquisition phase with the analytical insights provided by clustering and reinforcement learning, laying the foundation for actionable urban planning decisions.  
Methodology:

1. Input Data  
   Geospatial traffic data includes:  
   Latitude and Longitude: To map intersections and road segments.  
   Inverted Speed Ratios: Derived from speed limits and average speed, emphasizing congestion levels.
2. Heatmap Metrics  
   Intensity -   
   The degree of activity or congestion at each point, calculated using:

Points with higher inverted ratios indicate areas of congestion.

Weighting -  
Assign weights to traffic points based on their importance, such as traffic density or proximity to intersections.

1. Heatmap Creation  
   A Python-based pipeline is used to generate heatmaps:

Data preprocessing ensures all traffic points are relevant (speed > 0)  
Geospatial points are plotted using Folium and the HeatMap plugin​​.  
A color gradient visualizes intensity, from blue (low activity) to red (high activity).

1. Dynamic Heatmap Updates  
   The pipeline is designed to support dynamic updates for real-time traffic analysis  
   By integrating streaming traffic data, the heatmap reflects current urban activity.

Implementation Details:

* Data Filtering:  
  Traffic data with invalid or missing fields (e.g., zero-speed segments) is removed.  
  Only high-activity points are retained, improving the clarity of the heatmap.
* Visualization Setup:  
  Folium Library:  
  A Python library used to create geospatial maps.  
  Heatmap Layer:   
  Represents traffic intensity over the area of interest, with adjustable parameters such as:  
  Radius: Controls the size of heatmap influence for each point  
  Blur: Smoothens the heatmap for better visual clarity.
* Integration with Clustering:  
  The generated heatmap is overlaid with clustered intersection points, helping identify zones that require deeper analysis.  
  Cluster centroids are visually represented as markers on the heatmap​.

Output and Interpretation:

* Visualizations:  
  Heatmaps are saved as interactive HTML files, allowing urban planners to explore traffic activity zones dynamically.  
  Combined maps (heatmaps + clustering markers) provide a holistic view of urban activity hotspots.
* Interpretation:  
  High-intensity areas (red zones) identify traffic bottlenecks or zones requiring infrastructure improvement.  
  Low-intensity areas (blue zones) suggest smooth traffic flow or less critical regions.

Conclusion:  
Heatmap generation transforms raw traffic data into actionable visual insights, enabling the identification of urban activity zones that demand intervention. By integrating heatmaps with clustering and reinforcement learning, the project provides a scalable and dynamic tool for optimizing urban infrastructure and managing traffic flows effectively.

**Clustering with K-Means Algorithm:**

Clustering is a vital technique in this project to group high-activity regions or intersections within urban environments. By reducing the complexity of large geospatial datasets, clustering simplifies the identification of critical zones, aiding in traffic management and infrastructure planning.

The project employs a hybrid approach:

* K-Means Clustering for grouping intersections based on geospatial proximity and traffic intensity.
* Reinforcement Learning (DQN) to dynamically optimize clustering parameters for real-time, adaptive analysis.

This combined method ensures high computational efficiency while maintaining accuracy in identifying zones of interest.

Methodology:

1. Clustering with K-Means Algorithm:

* Objective:  
  We use K-Means because it is an unsupervised clustering algorithm that efficiently partitions spatial data based on proximity.
* Process:  
  Input geospatial traffic data (latitude, longitude).  
  Define k clusters (number of zones) based on traffic density.  
  Use K-Means to group data points into clusters
* Optimization:  
  Cluster centroids represent the high-interest zones, reducing the dataset size.  
  Results are saved in a simplified format for further processing (as demonstrated in generate2layersMap.py and extractIntersections.py)​​.
* Output:  
  Cluster centroids represent the most critical intersections.  
  The reduced dataset is saved in a simplified format for further processing
* Integration with Heatmap:

Clustering results are overlaid onto heatmaps generated from traffic intensity data.

The heatmap intensity (e.g., inverted speed ratios) is used to determine zones that require further analysis or clustering optimization.

1. Dynamic Optimization with Reinforcement Learning (DQN):

* Objective:  
  Traditional K-Means methods rely on static configurations and cannot adapt to real-time traffic changes.
* DQN Integration:  
  State: The current clustering of intersections and activity zones.  
  Action: Adjust clustering parameters (k) or modify existing clusters.  
  Reward: A numerical value evaluating the clustering efficiency, balancing:  
  Coverage of high-intensity zones and, Reduction of computational complexity.  
  Q-network: A neural network approximates the Q-values to optimize the clustering policy.
* Improvements in Clustering Process:  
  Adaptive clustering ensures the RL agent dynamically optimizes the number of clusters.  
  Redundant actions, such as selecting the same intersections repeatedly, are filtered through action memory mechanisms.

Evaluation:

1. Clustering Results:

Clustered intersections and high-activity zones are visualized on combined heatmap maps.  
Results demonstrate significant reduction in the number of intersections analyzed while preserving data accuracy.

1. Performance Metrics:  
   Inertia: Measures the compactness of clusters, lower values indicate better clustering.   
   Silhouette Score: Evaluates how well data points fit within their clusters.  
   Computational Efficiency: Clustering reduces the dataset size, leading to faster analysis and visualization.
2. Visualization:

Clustered points are represented as green markers on top of the heatmap, highlighting critical zones.  
Representative results are saved as HTML visualizations or images (e.g., combined\_heatmap\_intersections\_map.html).

Conclusion:  
The combination of K-Means clustering with reinforcement learning (DQN) offers a robust and adaptive solution for identifying critical zones in urban environments. This approach reduces computational overhead, dynamically adjusts to real-time traffic patterns, and provides valuable insights for urban infrastructure optimization.

**Neural Network Architecture**

The Neural Network (NN) used in the project plays a crucial role in optimizing the clustering of high-interest zones within urban environments. This network is based on the Deep Q-Network (DQN), which integrates reinforcement learning with neural networks to handle complex decision-making tasks like urban traffic clustering.  
The structure of the NN used in this project includes fully connected layers (Dense Layers), which are key components for approximating Q-values associated with the state-action pairs.

Layers Used in the Neural Network:  
The neural network consists of the following layers as implemented in the QNetwork class:

1. Input Layer:  
   Its purpose is to takes in the state representation of the environment, which includes processed heatmap data, convex hull masks, and dot indicators. The input is flattened into a single-dimensional tensor, representing all 3 channels (heatmap, convex hull, and dot indicators) of the state.  
   Dimensionality:
2. First Hidden Layer:  
   It is a fully connected layer, includes 128 neurons.  
   the activation function is ReLU (Rectified Linear Unit), that introduces non-linearity to the model, enabling it to capture complex relationships within the data. Its purpose is to extract high-level features from the input state.  
   The input is linearly transformed, followed by applying ReLU:
3. Second Hidde Layer:  
   It is a fully connected layer, includes 128 neurons.  
   the activation function is ReLU. Its purpose is to further refine the feature representation, allowing the network to learn intricate patterns in the data.
4. Output Layer:  
   It is a fully connected layer, the size is equal to the number of possible actions which corresponds to the intersections available for toggling.  
   Its purpose is to outputs the Q-values for each action, representing the expected cumulative reward for taking each action in the current state, there is no activation function, the raw Q-values are output directly as the network's predictions.

Advantages of the Network Design:

* Efficient Feature Extraction: The two hidden layers with ReLU activation allow the network to model complex traffic patterns effectively
* Scalability: The architecture is lightweight yet powerful, ensuring efficient learning while handling large input sizes (21168 features)
* Action-Specific Outputs: The output layer directly maps to all possible actions, simplifying the integration with reinforcement learning algorithms.

Integration with DQN:  
This neural network is a key component of the DQN Agent, which performs the following:

* Select Actions: The network predicts Q-values for each possible action, enabling the agent to choose the optimal action.
* Update Policy: The network is trained using the Mean Squared Error (MSE) loss to minimize the difference between predicted Q-values and target Q-values

Conclusion:  
By combining this architecture with the reinforcement learning algorithm, the system dynamically adjusts the clustering of high-interest zones, ensuring both computational efficiency and accuracy in urban traffic analysis​.

**Analysis and Findings**

**Challenges and Limitation**

**Conclusion**

**References**

[1] K. Ramamohanarao, H. Xie, L. Kulik, S. Karunasekera, E. Tanin, R. Zhang, and E. B. Khunayn, “SMARTS: Scalable Microscopic Adaptive Road Traffic Simulator,” ACM Transactions on Intelligent Systems and Technology, vol. 8, No. 2, Article 26, 2016, pp. 1–22.

[2] D. Igbe, N. Kalantery, S. Ijaha, and S. Winter, “An Open Interface for Parallelization of Traffic Simulation,” in Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications, 2003, pp. 158–163.

[3] R. Klefstad, Y. Zhang, M. Lai, R. Jayakrishnan, and R. Lavanya, “A Scalable, Synchronized, and Distributed Framework for Large-Scale Microscopic Traffic Simulation,” in The 8th International IEEE Conference on Intelligent Transportation Systems, 2005, pp. 813–818.

[4] K. Nagel and M. Rickert, “Parallel Implementation of the TRANSIMS Micro-Simulation,” Parallel Computing, vol. 27, No. 12, 2001, pp. 1611–1639.

[5] T. Potuzak, “Distributed/Parallel Genetic Algorithm for Road Traffic Network Division using a Hybrid Island Model/Step Parallelization Approach,” in 2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications, 2016, pp. 170-177.

[6] T. Potuzak, “Distributed-Parallel Road Traffic Simulator for Clusters of Multi-core Computers,” in 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, 2012, pp. 195–201.

[7] R. M. Fujimoto, Parallel and Distributed Simulation Systems, John Wiley & Sons, New York, 2000.

[8] M. J. Lighthill and G. B. Whitham, “On kinematic waves II: A theory of traffic flow on long crowed roads,” Proceedings of the Royal Society of London, Series A. Mathematical and Physical Sciences, vol. 229, No. 1178, 1955, pp. 317–345.

[9] A. T. Chronopoulos and C. M. Johnston, “A Real-Time Traffic Simula tion Using a Communication Latency Hiding Parallelization,” IEEE Tra nsactions on Vehicular Technology, vol. 51, No. 3, 2002, pp. 498–510.

[10] M. P. H. Raadsen and M. C.J. Bliemer, “Continuous-time general link transmission model with simplified fanning, Part II: Event-based algorithm for networks,” Transportation Research Part B, vol. 126, 2019, pp. 471–501.

[11] T. Potuzak and F. Kolovsky, “Parallelization of the B Static Traffic Assignment Algorithm,” ASEJ, vol. 13, No. 2, 101576, 2022.

[12] T. Potuzak, “Speedup of the Microscopic Road Traffic Simulation using Aggregated Vehicle Movement,” in 2015 4th Eastern European Regional Conference on the Engineering of Computer Based Systems, August 2015, pp. 111–118.

[13] W. Burghout, Hybrid microscopic-mesoscopic traffic simulation, Doctoral thesis, Royal Institute of Technology, Stockholm, 2004.

[14] V. A. Vu, G. Tan, X. Li, and X. Song, “Mesoscopic Traffic Simulation on CPU/GPU,” in Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, 2014, pp. 39–49.

[15] L. Nizzard, Combining Microscopic and Mesoscopic Traffic Simulators, Raport de stage d’option scientifique, Ecole Polytechnique, Paris, 2002.

[16] K. Nagel and M. Schreckenberg, “A Cellular Automaton Model for Freeway Traffic,” Journal de Physique I, 2, 1992, pp. 2221–2229.

[17] P. G. Gipps, “A behavioural car following model for computer simulation,” Transp. Res. Board, 15-B(2), 1981, pp. 403–414.

[18] D. Ni, “A Framework for New Generation Transportation Simulation,” in Proceedings of the 2006 Winter Simulation Conference, 2006, pp. 1508–1514.

[19] N. Cetin, A. Burri, and K. Nagel, “A Large-Scale Agent-Based Traffic Microsimulation Based on Queue Model,” Proceedings of 3rd Swiss Transport Research Conference, Monte Verita, 2003.

[20] T. Kiesling and J. Lüthi, “Towards Time-Parallel Road Traffic Simulation," in Workshop on Principles of Advanced and Distributed Simulation (PADS'05), 2005.

[21] H. Chen, K. Yang, S. G. Rizzo, G. Vantini, P. Taylor, X. Ma, and S. Chawla, “QarSUMO: A Parallel, Congestion-optimized Traffic Simulator,” in SIGSPATIAL '20: Proceedings of the 28th International Conference on Advances in Geographic Information Systems, 2020, pp. 578–588.

[22] Y. Xu and G. Tan, “An Offline Road Network Partitioning Solution in Distributed Transportation Simulation,” 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, 2012, pp. 210–217.

[23] D. Igbe, Dynamic Load Balancing of Parallel Road Traffic Simulation, Doctoral Thesis, University of Westminster, London, 2010.

[24] A. Rydzewski and P. Czarnul, “Recent advances in traffic optimisation: systematic literature review of modern models, methods and algorithms,” IET Intelligent Transport Systems, vol. 14, No. 13, 2020, pp. 1740–1758.

[25] R. Ding, N. Ujang & H. Bin Hamid, M. S. Abd Manan, R. Li, S. S. M. Albadareen, A. Nochian, and J. Wu, “Application of Complex Networks Theory in Urban Traffic Network Researches,” Networks and Spatial Economics, vol. 19, 2019, pp. 1281–1317.

[26] A. Buluc, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, “Recent Advances in Graph Partitioning," Algorithm Engineering - Selected surveys and Results, LNCS 9220, 2016, pp. 117–158.

[27] K. Schloegel, G. Karypis, and V. Kumar, “Graph partitioning for high performance scientific simulations,” Sourcebook of parallel computing, 2003, pp. 491–541.

[28] J. Kim, I. Hwang, Y.-H. Kim, and B.-R. Moon, “Genetic Approaches for Graph Partitioning: A Survey,” in Gecco'11, 2011, pp. 473-480

[29] S. A. Bhavsar, V. H. Patil, and A. H. Patil, “Graph partitioning and visualization in graph mining: a survey,” in Multimedia Tools and Applications, 2022.

[30] N. M. Soudani, A. Fatemi, and M. Nematbakhsh, “An investigation of big graph partitioning methods for distribution of graphs in vertex centric systems,” Distributed and Parallel Databases, vol. 38, 2020, pp. 1–29.

[31] S. Schwartz, “An overview of graph covering and partitioning,” DiscreteMathematics, vol. 345, 2022.

[32] A. Ventresque, Q. Bragard, E. S. Liu, D. Nowak, L. Murphy, G. Theodoropoulos, and Q. Liu, “SParTSim: A Space Partitioning Guided by Road Network for Distributed Traffic Simulations,” in 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, 2012, pp. 202–209.

[33] D. Wei, F. Chen, and X. Sun, “An improved road network partition algorithm for parallel microscopic traffic simulation,” in 2010 International Conference on Mechanic Automation and Control Engineering, 2010.

[34] M. S. Ahmed, M. A. Hoque, “Partitioning of Urban Transportation Networks Utilizing Real-World Traffic Parameters for Distributed Simulation in SUMO,” in 2016 IEEE Vehicular Networking Conference (VNC), 2016.

[35] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” SIAM Journal on scientific Computing, vol. 20, No. 1, 1998, pp. 359–392.

[36] Y. Xu, W. Cai, D. Eckhoff, S. Nair, and A. Knoll, “A Graph Partitioning Algorithm for Parallel Agent-Based Road Traffic Simulation,” in SIGSIM-PADS ’17, 2017, pp. 209-219.

[37] D. Yin, S. Wang, and Y. Ouyang, “ViCTS: A novel network partition algorithm for scalable agent-based modeling of mass evacuation," Computers, Environment and Urban Systems, vol. 80, 2020.

[38] R. Fernandes, F. Vieira, and M. Ferreira, “Parallel Microscopic Simulation of Metropolitan-scale Traffic,” in ANSS 13: Proceedings of the 46th Annual Simulation Symposium, 2013.

[39] A. Acosta, J. Espinosa, and J. Espinoza, “Distributed Simulation in SUMO Revisited: Strategies for Network Partitioning and Border Edges Management," in SUMO User Conference 2016, vol. 30, 2016. 85 Authorized licensed use limited to: The University of Toronto. Downloaded on March 26,2023 at 09:22:02 UTC from IEEE Xplore. Restrictions apply. 2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)

[40] Y. Xu, W. Cai, H. Aydt, M. Lees, and D. Zehe, “Relaxing Synchronization in Parallel Agent-Based Road Traffic Simulation,” ACM Trans. Model. Comput. Simul., vol. 27, No. 2, 2017.

[41] D.-H. Lee and P. Chandrasekar, “A framework for parallel traffic simulation using multiple instancing of a simulation program,” ITS Journal, vol. 7, No. 3-4, 2002, pp. 279–294.

[42] W. Turek, “Erlang-based desynchronized urban traffic simulation for high-performance computing systems,” Future Generation Computer Systems, vol. 79, 2018, pp. 645–652.

[43] T. Potuzak, “Comparison of Road Traffic Network Division Based on Microscopic and Macroscopic Simulation,” in UKSim 2011 - UKSim 13th International conference on Computer Modelling and Simulation, 2011, pp. 409–414.

[44] T. Potuzak, “Utilization of a Genetic Algorithm in Division of Road Traffic Network for Distributed Simulation,” in ECBS-EERC 2011 - 2011 Second Eastern European Regional Conference on the Engineering of Computer Based Systems, 2011, pp. 151–152.

[45] T. Potuzak, “Methods for division of road traffic networks focused on load-balancing,” Advances in Computing, vol. 2, No. 4, 2012, pp. 42 53.

[46] T. Potuzak, “Parallelization Possibilities of a Genetic Algorithm for Road Traffic Network Division for Distributed/Parallel Environment,” in 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications, 2014, pp. 211–218.

[47] T. Potuzak, “Distributed/Parallel Genetic Algorithm for Road Traffic Network Division using Step Parallelization,” in 2015 4th Eastern European Regional Conference on the Engineering of Computer Based Systems, 2015, pp. 67–74.

[48] T. Potuzak, “Distributed/Parallel Genetic Algorithm for Road Traffic Network Division using a Hybrid Island Model/Step Parallelization Approach,” in 2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications, 2016, pp. 170–177.

[49] T. Potuzak, “Utilization of Graph Coarsening for Improving of Results of a Genetic Algorithm for Road Traffic Network Division,” in 2016 9th International Conference on Human System Interactions (HSI), 2016, pp. 28–34.

[50] T. Potuzak, “Division of Road Traffic Network for Distributed Simulation Performed on Heterogeneous Clusters,” in ECBS 2012 - 2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems, 2012, pp. 117–125.

[51] T. Potuzak, “Methods for division of road traffic network for distributed simulation performed on heterogeneous clusters,” ComSIS, vol. 10, No. 1, 2013, pp. 321–348.

[52] X. Song, Z. Xie, Y. Xu, G. Tan, W. Tang, J. Bi, and X. Li, “Supporting real-world network-oriented mesoscopic traffic simulation on GPU,” SIMPAT, vol. 74, 2017, pp. 46–63.

[53] Z. Shen, K. Wang, F.-Y. Wang, and C. L. Philp Chen, “GPU Based Genetic Algorithms for the Dynamic Sub-area Division Problem of the Transportation System,” in Proceedings of the 19th World Congress - The International Federation of Automatic Control, 2014, pp. 5115 5120.

[54] R. Saedi, M. Saeedmanesh, A. Zockaie, M. Saberic, N. Geroliminis, and H. S. Mahmassani, “Estimating network travel time reliability with network partitioning,” Transportation Research Part C, vol. 112, 2020, pp. 46–61.

[55] L. Ambühl, A. Loder, N. Zheng, K. W. Axhausen, and M. Menendez, “Approximative Network Partitioning for MFDs from Stationary Sensor Data,” Transportation Research Record, vol. 2673, No. 6, 2019, pp. 94 103.

[56] K. An, Y.-C. Chiu, X. Hu, and X. Chen, “A Network Partitioning Algorithmic Approach for Macroscopic Fundamental Diagram-Based Hierarchical Traffic Network Management,” IEEE Transactions on Intelligent Transportation Systems, vol. 19, No. 4, 2018.

[57] T. Anwar, C. Liu, H. L. Vu, and M. S. Islam, “Tracking the Evolution of Congestion in Dynamic Urban Road Networks,” in CIKM '16: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 2323–2328.

[58] P. Johnson, D. Nguyen, and M. Ng, “Large-scale network partitioning for decentralized traffic management and other transportation applications,” Journal of Intelligent Transportation Systems, vol. 20, No. 5, 2016, pp. 461–473.

[59] Q. Liu, Q. Wang, and S. Liu, “An Improved Sub-Networks Partitioning Method for Urban Traffic Networks,” in 2019 Chinese Control And Decision Conference (CCDC), 2019, pp. 6405–6410.

[60] Z. Zhou, S. Lin, and Y. Xi, “A fast network partition method for large scale urban traffic networks,” J Control Theory Appl, vol. 11, No. 3, 2013, pp. 359–366.

[61] Z. Zhou, S. Lin, and Y. Xi, “A Dynamic Network Partition Method for Heterogenous Urban Traffic Networks,” in 2012 15th International IEEE Conference on Intelligent Transportation Systems, 2012, pp. 820–825.

[62] H. Chen and Y. Hu, “Finding Community Structure and Evaluating Hub Road Section in Urban Traffic Network,” Procedia - Social and Behavioral Sciences, vol. 96, 2013, pp. 1494–1501.

[63] H. Guo, J. Cheng, Q. Peng, C. Zhu, and Y. Mu, “Dynamic Division of Traffic Control Sub-area Methods Based on the Similarity of Adjacent Intersections,” in 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 2208–2213.

[64] G. Yufeng, Z. Qingmei, W. Xiuqing, Z. Zhiling, and L. Pengpeng, “A Traffic Partition Method Based On Unsupervised Classification,” in 2010 International Conference On Computer Design And Appliations, vol. 3, 2010, pp. 445–449.

[65] Q. Yu, W. Li, D. Yang, H. Zhang, “Partitioning urban road network based on travel speed correlation,” International Journal of Transportation Science and Technology vol. 10, 2021, pp. 97–109.

[66] C. Withanage, D. Lakmal, M. Hansini, K. Kankanamge, Y. Witharanage, and U. Thayasivam, “A modified multilevel k-way partitioning algorithm for trip-based road networks,” in MATEC Web Conf., vol. 272, 2019.

[67] M. Cebecauer, E. Jenelius, and W. Burghout, “Spatio-Temporal Partitioning of Large Urban Networks for Travel Time Prediction,” in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 1390–1395

[68] F. Yan, M. Zhang, and Z.Shi, “Dynamic partitioning of urban traffic network sub-regions with spatiotemporal evolution of traffic flow,” Nonlinear Dyn, vol. 105, 2021, pp. 911–929.

[69] Y. Zheng, G. Zhao, and J. Liu, “A Novel Grid Based K-Means Cluster Method for Traffic Zone Division,” in CloudCom-Asia 2015, LNCS 9106, 2015, pp. 165–178.

[70] L. Dimitriou and P. Nikolaou, “Dynamic partitioning of urban road networks based on their topological and operational characteristics,” in 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017, pp. 457–462.

[71] H. Dong, M. Wu, X. Ding, L. Chu, L. Jia, Y. Qin, and X. Zhou, “Traffic zone division based on big data from mobile phone base stations,” Transportation Research Part C vol. 58, 2015, pp.278–291.

[72] S. Fang, K. Bian, H. Hong, K. Xie, and Y. Fu, “Using the Traffic Heterogeneity of Chinese Toll Highway Networks for Hierarchical Clustering,” in 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI), 2016, pp. 90–95.

[73] C. Li, Y. Xie, H. Zhang, and X. Yan, “Dynamic division about traffic control sub-area based on back propagation neural network,” in 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, 2010, pp. 22–25.

[74] Y. Y. Ma and X.-G. Yang, “Traffic sub-area division expert system for urban traffic control,” in 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA), 2008, pp. 589 593.

[75] Y. Wang,, J. H. van Schuppen, and J. Vrancken, “On-line distributed prediction of traffic flow in a large-scale traffic network,” in 19th ITS World Congress, 2012.

[76] Y. Wang,, J. H. van Schuppen, and J. Vrancken, “On-line distributed prediction of traffic flow in a large-scale road network,” SIMPAT, vol. 47, 2014, pp. 276–303.

**RESULT 1 : num of episodes 100, num of steps 100 :**

**RESULT 2 : num of episodes 100, num of steps 500 :**