# Weather and Weekday Prediction with London's Bike-Sharing Service using GNN

**Candidate Numbers:** <span style="background:black">████████████████</span>

## Abstract

This study introduced the use of Graph Neural Network (GNN) for predicting the weather and weekday of a day in London, from the dataset of Santander Cycles bike-sharing system as a graph classification task. The proposed GNN models newly introduced (i) a concatenation operator of graph features with trained node embeddings and (ii) a graph coarsening operator based on geographical contiguity, namely *Spatial Graph Coarsening*. With the node features of land-use characteristics and number of households around the bike stations and graph features of temperatures in the city, our proposed models outperformed the baseline model in cross-entropy loss and accuracy of the validation dataset.

**Keywords:** Graph Neural Network, Graph Classification, Graph Coarsening, Bike Sharing Service

## 1. Introduction

Urban transportation systems are essential for modern cities as they provide mobility to inhabitants, while significantly impacting the environmental, economic, and social aspects of cities. A form of transportation that is growing in popularity in recent years is biking. Bike-sharing services have become prevalent in cities worldwide and have emerged as a valuable tool for promoting sustainability as well as a healthier lifestyle. These services can also be viewed as socio-economic data sources that could provide useful insights into human activities in urbanised areas, with potential applications in fields such as Urban Economics and Social Network Analysis.

In parallel, since the Graph Neural Network (GNN) became prevalent for graph representation learning, a many algorithms have been invented, such as Graph Convolution Network (GCN) (Kipf and Welling, 2017). Since GNN-based models are permutation-invariant and free from the degree of the nodes for the neural network architecture, they have been applied to a variety of domains, such as molecular property prediction (Wang et al., 2023); and social network recommendation (Fan et al., 2019). In particular, demand prediction for bike-sharing and ride-sourcing services has been one of the notable topics of GNN-based model applications, for example in Beijing (Lin et al., 2018) and in New York (Ke et al., 2021). However, previous studies have focused only on the node regression of bike usage per station and edge regression of travel record counts, and graph prediction is still understudied. Moreover, the GNN-based models in the previous studies lacked the consideration of the geographical contiguity of bike stations as nodes, which is due to the graph representation of the dataset only as the relationship of travel records, not as the spatial proximity. A drawback could be the failure to capture the homophily of geographical neighbours be-

tween nodes in the GNN-based model. Lastly, any GNN-based prediction tasks has not been investigated yet on London's bike-sharing services.

In this study, we conduct a prediction of the weather and weekday in London from the usage of Santander Cycles (Transport for London, 2023), in order to demonstrate the importance of understanding the characteristics of a city. We use GNN to predict the weather and weekday of daily graphs as graph embeddings, with mapped land-use characteristics (Fleischmann and Arribas-Bel, 2022) and number of households in the neighbours by demographics (Office for National Statistics, 2022) as node features. Our main contributions are as follows (i) the introduction of a concatenation operator of graph features with trained node embeddings obtained after *Global Mean Pooling*; and (ii) the introduction of graph coarsening operator based on geographical contiguity, namely *Spatial Graph Coarsening*. These new proposals shall expand the employability of bike-sharing dataset for a better understanding of cities and humans living there.

## 2. Problem Formulation

### 2.1 Graph Neural Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph of the bike-sharing service dataset in a graph representation, where $\mathcal{V}$ and $\mathcal{E}$ denotes the bike stations used as nodes and travel records as edges on the day, and note that $|\mathcal{V}|$ and $|\mathcal{E}|$ stand for the number of nodes and edges of graph $\mathcal{G}$, respectively. Given $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, $\mathbf{H}^{(k)} \in \mathbb{R}^{|\mathcal{V}| \times d^{(k)}}$, $\mathbf{W}^{(k)} \in \mathbb{R}^{d^{(k+1)} \times d^{(k)}}$, and $\mathbf{B}^{(k)} \in \mathbb{R}^{|\mathcal{V}| \times d^{(k+1)}}$ which denote adjacency matrix of $\mathcal{G}$, node embeddings matrix of dimension $d^{(k)}$, and weight matrix and bias matrix at layer $k$, respectively, the node embeddings matrix at layer $k + 1$ as a result of messege passing of a GNN model is written as follows:

$$\mathbf{H}^{(k+1)} = \sigma \left( (\mathbf{I} + \mathbf{A}) \mathbf{H}^{(k)} (\mathbf{W}^{(k)})^\top + \mathbf{B}^{(k)} \right),$$

where $\mathbf{I} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is an identity matrix and $\sigma(\cdot)$ is an activation function. As our assumption, the weight matrix $\mathbf{W}^{(k)}$ can be shared between neighbours and self-nodes, since bike-sharing services are not limited to combination of different nodes (i.e. one can use a bike within Hyde Park). Regarding the adjacency matrix, we additionally use $\mathbf{A}_{\text{weighted}}$ as a weighted one, such that $[\mathbf{A}_{\text{weighted}}]_{ij} = n_{ij}$, where $n_{ij}$ denote the total count of travel records on a day between bike station $i$ and $j$. Moreover, $\mathbf{A}$ and $\mathbf{A}_{\text{weighted}}$ are used in the message passing after symmetrical normalisation as a process of *Graph Convolution* (Kipf and Welling, 2017) to balance out the influence of nodes, such that

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}};$$

$$\tilde{\mathbf{A}}_{\text{weighted}} = \mathbf{D}_{\text{weighted}}^{-\frac{1}{2}} \mathbf{A}_{\text{weighted}} \mathbf{D}_{\text{weighted}}^{-\frac{1}{2}},$$

where $\mathbf{D}$ and $\mathbf{D}_{\text{weighted}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ denote degree matrices of $\mathbf{A}$ and $\mathbf{A}_{\text{weighted}}$. A degree matrix is a diagonal matrix where the entires represent the degree of a vertex, i.e. the number of times an edge terminates at that vertex.

## 2.2 Global Mean Pooling and Graph Features Concatenation

In GNN architectures, graph pooling is a popular operation to produce summarised information of graphs for graph prediction tasks (Hamilton, 2020). This study commonly employs the *Global Mean Pooling* operator to obtain graph embeddings for following fully-connected layers. After *Global Mean Pooling*, a graph embedding with $d_{\text{graph}}$ components $\mathbf{z}_{\mathcal{G}} \in \mathbb{R}^{d_{\text{graph}}}$ is given by $\mathbf{z}_{\mathcal{G}} = \left( \sum_{v \in \mathcal{V}} \mathbf{z}_u \right) / |\mathcal{V}|$, where $\mathbf{z}_u \in \mathbb{R}^{d_{\text{node}}}$ is a vector of node embedding for node $u$ with $d_{\text{node}}$ components after graph convolutions. In addition to the obtained graph embeddings, we utilise graph features of temperature as a representation of seasonality. Let $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^3$ be a vector of graph features of mean, maximum and minimum daily temperatures. Then, we concatenate the graph embeddings and the graph features of temperatures to obtain $\mathbf{z}_{\mathcal{G}} \oplus \mathbf{x}_{\mathcal{G}} = \tilde{\mathbf{z}}_{\mathcal{G}} \in \mathbb{R}^{d_{\text{graph}}+3}$, where $\oplus$ denotes concatenation of two vectors.

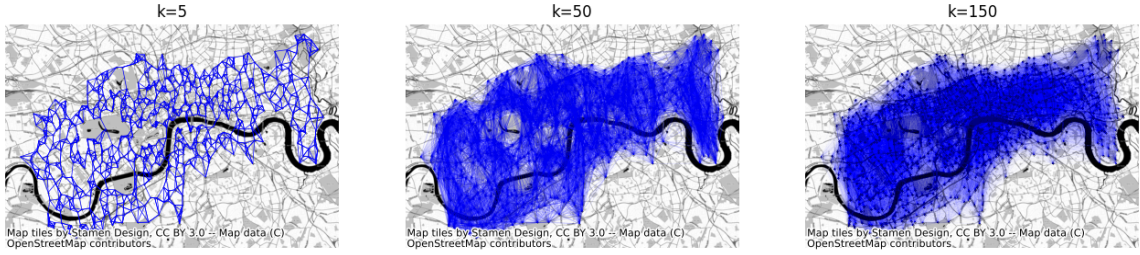## 2.3 Spatial Graph Coarsening

### 2.3.1 KNN GRAPH



Figure 1: KNN graph on bike stations of Santander Cycles in London. Edges were generated based on $k \in \{5, 50, 150\}$ nearest neighbours of geographic contiguity.

Although the framework of GNN we introduced could capture the heterogeneity of weather and weekday, it lacks in considering the information of geographical contiguity between nodes. Thus, instead of directly pooling all the node embeddings into a single vector, we would like to propose spatial graph coarsening, which is based on clusters on $k$-Nearest Neighbour (KNN) graph of nodes by the geographical coordinates. By introducing spatial graph coarsening, the similarities among geographic neighbours could be aggregated by clusters. As shown in Figure 1, $k$ as the number of neighbours determines the connectivity in the KNN graph. Let $\mathcal{G}_{\text{KNN}}^{(k)} = (\mathcal{V}, \mathcal{E}_{\text{KNN}}^{(k)})$ a KNN graph based on $k$ nearest neighbours on the basis of Euclidian distance of the geographic coordinates, the component of its adjacency matrix $\mathbf{A}_{\text{KNN}}^{(k)} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is written as follows:

$$
[\mathbf{A}_{\text{KNN}}^{(k)}]_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{N}^{(k)}(i) \ \forall i, j \in \mathcal{V} \\ 0 & \text{otherwise.} \end{cases},
$$

where $i, j \in \mathcal{V}$ denote the nodes in the KNN graph, which is invariant to the value of $k$, and $\mathcal{N}^{(k)}(i)$ stands for the $k$ nearest neighbours of node $i$.

3

### 2.3.2 SPECTRAL CLUSTERING

To construct clusters on the KNN graph $\mathcal{G}_{\text{KNN}}^{(k)}$, we employ Spectral Clustering to identify strongly connected communities of nodes in graphs, by minimising Equation 1,

$$\text{NCut}(\mathcal{A}_1, \ldots, \mathcal{A}_K) = \frac{1}{2} \frac{\sum_{k=1}^{K} |(u,v) \in \mathcal{E} : u \in \mathcal{A}_k, v \in \bar{\mathcal{A}}_K|}{\text{vol}(\mathcal{A}_k)}, \tag{1}$$

through the generalised eigenvalue problem of symmetrically normalised graph Laplacian matrix (Shi and Malik, 2000), where $\mathcal{A}_k \quad \forall k \in [K]$ is a set of nodes in a graph which belongs to cluster $k$, and $\bar{\mathcal{A}}_k$ and $\text{vol}(\mathcal{A}_k)$ denote the complement and total sum of node degree of $\mathcal{A}_k$ respectively. By taking a normalisation with $\text{vol}(\mathcal{A}_k)$, rural bike stations with a small total node degree would be punished for clustering. For the KNN graph with $k$ nearest neighbours, the graph Laplacian $\mathbf{L}_{\text{KNN}}^{(k)}$ and the symmetric normalised Laplacian $\tilde{\mathbf{L}}_{\text{KNN}}^{(k)}$ are defined as

$$\tilde{\mathbf{L}}_{\text{KNN}}^{(k)} = \mathbf{D}_{\text{KNN}}^{(k)}{}^{-\frac{1}{2}} \mathbf{L}_{\text{KNN}}^{(k)} \mathbf{D}_{\text{KNN}}^{(k)}{}^{-\frac{1}{2}};$$
$$\mathbf{L}_{\text{KNN}}^{(k)} = \mathbf{D}_{\text{KNN}}^{(k)} - \mathbf{A}_{\text{KNN}}^{(k)},$$

where $\mathbf{D}_{\text{KNN}}^{(k)}$ is the degree matrix of KNN graph with $k$ nearest neighbours. The symmetrically normalised graph Laplacian matrix is then utilised for obtaining a cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times K}$ as follows (Hamilton, 2020):

1. Calculate the symmetrically normalised Laplacian $\tilde{\mathbf{L}}_{\text{KNN}}^{(k)}$ and obtain its second to $(K+1)$-th smallest eigenvalues and their corresponding eigenvectors;

2. Form a matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times K}$ where columns are $K$ eigenvectors of $\tilde{\mathbf{L}}_{\text{KNN}}^{(k)}$ from step 1;

3. Apply $k$-means clustering across $\mathbf{U}_{[u]} \in \mathbb{R}^K \,\forall u \in \mathcal{V}$ and obtain $\mathbf{s}_u \in \mathbb{R}^K$, where $\mathbf{U}_{[u]}$ is the $u$-th row of $\mathbf{U}$ which corresponds to the graph node $u$;

4. Form a cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times K}$, where $u$th row corresponds to $\mathbf{s}_u$ obtained in the step 3.

### 2.3.3 GRAPH COARSENING

Given a node embedding matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times \ell}$ with $\ell$ latent variables and symmetrically normalised adjacency matrix of KNN graph $\tilde{\mathbf{A}}_{\text{KNN}}^{(k)} = \mathbf{D}_{\text{KNN}}^{(k)}{}^{-\frac{1}{2}} \mathbf{A}_{\text{KNN}}^{(k)} \mathbf{D}_{\text{KNN}}^{(k)}{}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$, the pooled node embedding and adjacency matrices are computed, respectively, as

$$\mathbf{Z}^{\text{pool}} = \mathbf{S}^\top \mathbf{Z};$$
$$\mathbf{A}^{\text{pool}} = \mathbf{S}^\top \tilde{\mathbf{A}}_{\text{KNN}}^{(k)} \mathbf{S}.$$

The entry $z_{i,j}^{\text{pool}}$ in $\mathbf{Z}^{\text{pool}} \in \mathbb{R}^{K \times \ell}$ denotes the pooled components of node embedding $j$ of nodes in cluster $i$. $\mathbf{A}^{\text{pool}} \in \mathbb{R}^{K \times K}$ is a symmetric matrix, with entry $a_{i,j}^{\text{pool}}$ indicating the weighted sum of edges between clusters $i$ and $j$, and entries $a_{i,i}^{\text{pool}}$ indicating the weighted sum of edges between the nodes in cluster $i$ (Bianchi et al., 2020).

## 3. Proposed Solution

### 3.1 Dataset

#### 3.1.1 Node Features and Weighted Edges

We first obtained the dataset of bike stations and daily travel records using the Santander Cycles API (Transport for London, 2023) for the years 2021 and 2022, totalling 730 days. 798 stations were available as nodes and 13,483,777 travel records were available as edges. As seen in Figure 2 on the right side, there is a seasonal pattern of demands for the service, with a few outliers. To avoid confounded prediction, 7 samples which had more than 30,000 or less than 5,000 travel records were omitted. Daily travel records were then aggregated by the same origin and destination pair, yielding $723 (= 730 - 7)$ undirected graphs. We then mapped node features of 6 land-use characteristics (Fleischmann and Arribas-Bel, 2022) and the number of households in the smallest area unit by 10 family structures (Office for National Statistics, 2022) onto the nodes of bike stations.
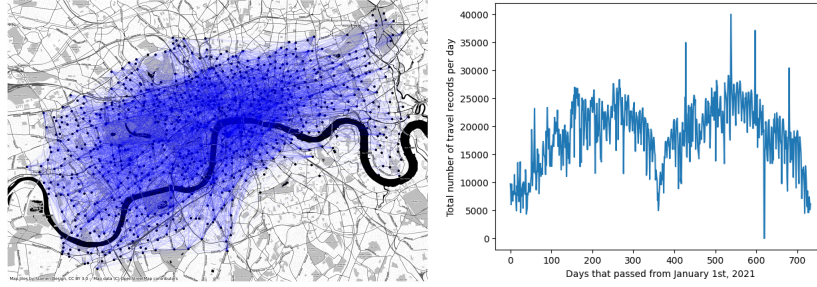


Figure 2: Visualised undirected graph of travel records from Santander Cycles on April 29th, 2021 (left), and the total number of travel records per day (right).

#### 3.1.2 Graph Features and Target Labels

For the graph features and target labels, the meteorological dataset the (European Climate Assessment & Dataset) observed daily at Heathrow was used. This can be considered as approximation to London's weather conditions. To capture seasonal patterns as seen in Figure 2, we consider the mean, maximum, and minimum daily temperatures after normalising to 0 mean and 1 standard deviation, as graph features. For the target labels, we created three categories of weather: (i) 'rainy', if precipitation is more than 0mm; (ii) 'cloudy', if daily cloud coverage is more than 20%; and otherwise (iii) 'sunny'. This is combined with a binary indicator of weekends or holidays, obtained from the Bank Holidays API (UK Government, 2023), yielding in total 6 categories. A summary is presented in Table 1.

| Name | Category | Data Type | Dimensions |
|---|---|---|---|
| Land-use Characteristics | Node Feature | Categorical | 6 |
| Number of Households by Demographics | Node Feature | Continuous | 10 |
| Temperatures | Graph Feature | Continuous | 3 |
| Weather $\times$ (Weekday or Weekend/Holiday) | Target Label | Categorical | $3 \times 2 = 6$ |

Table 1: Summary of node features, graph features, and target labels.

## 3.2 Models

### 3.2.1 MODEL ARCHITECTURES

A total of 11 models are evaluated in this study. The baseline model (Model 0) uses 2 GCNConv layers (Kipf and Welling, 2017), a global mean pooling layer as described in Section 2.2 and 2 Linear (or Fully-Connected) layers, with rectified linear unit (ReLU) activation function $\text{ReLU}(x) = \max\{0, x\}$ and dropout layers (with probability $p = 0.5$) in between. All other models also consider the mean, minimum and maximum daily temperatures as input features. Models 3 to 10 additionally employ the GraphCoars operator as described in Section 3.1.2, followed by a GraphConv layer (Morris et al., 2021). The construction of $\mathbf{A}$ and $\mathbf{S}$ are different for each model and is summarised in Table 2. A simplfied diagram of the models with condensed notation is presented in Figure 3.
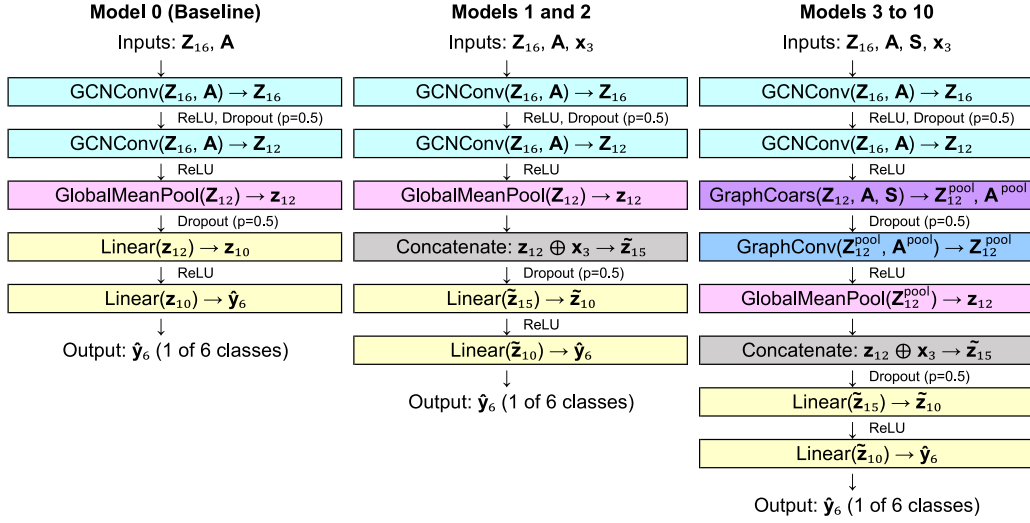


**Model 0 (Baseline)**
Inputs: $\mathbf{Z}_{16}$, $\mathbf{A}$
↓
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{16}$
↓ ReLU, Dropout (p=0.5)
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{12}$
↓ ReLU
GlobalMeanPool($\mathbf{Z}_{12}$) → $\mathbf{z}_{12}$
↓ Dropout (p=0.5)
Linear($\mathbf{z}_{12}$) → $\mathbf{z}_{10}$
↓ ReLU
Linear($\mathbf{z}_{10}$) → $\hat{\mathbf{y}}_6$
↓
Output: $\hat{\mathbf{y}}_6$ (1 of 6 classes)

**Models 1 and 2**
Inputs: $\mathbf{Z}_{16}$, $\mathbf{A}$, $\mathbf{x}_3$
↓
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{16}$
↓ ReLU, Dropout (p=0.5)
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{12}$
↓ ReLU
GlobalMeanPool($\mathbf{Z}_{12}$) → $\mathbf{z}_{12}$
↓
Concatenate: $\mathbf{z}_{12} \oplus \mathbf{x}_3 \to \tilde{\mathbf{z}}_{15}$
↓ Dropout (p=0.5)
Linear($\tilde{\mathbf{z}}_{15}$) → $\tilde{\mathbf{z}}_{10}$
↓ ReLU
Linear($\tilde{\mathbf{z}}_{10}$) → $\hat{\mathbf{y}}_6$
↓
Output: $\hat{\mathbf{y}}_6$ (1 of 6 classes)

**Models 3 to 10**
Inputs: $\mathbf{Z}_{16}$, $\mathbf{A}$, $\mathbf{S}$, $\mathbf{x}_3$
↓
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{16}$
↓ ReLU, Dropout (p=0.5)
GCNConv($\mathbf{Z}_{16}$, $\mathbf{A}$) → $\mathbf{Z}_{12}$
↓ ReLU
GraphCoars($\mathbf{Z}_{12}$, $\mathbf{A}$, $\mathbf{S}$) → $\mathbf{Z}_{12}^{\text{pool}}$, $\mathbf{A}^{\text{pool}}$
↓ Dropout (p=0.5)
GraphConv($\mathbf{Z}_{12}^{\text{pool}}$, $\mathbf{A}^{\text{pool}}$) → $\mathbf{Z}_{12}^{\text{pool}}$
↓ ReLU
GlobalMeanPool($\mathbf{Z}_{12}^{\text{pool}}$) → $\mathbf{z}_{12}$
↓
Concatenate: $\mathbf{z}_{12} \oplus \mathbf{x}_3 \to \tilde{\mathbf{z}}_{15}$
↓ Dropout (p=0.5)
Linear($\tilde{\mathbf{z}}_{15}$) → $\tilde{\mathbf{z}}_{10}$
↓ ReLU
Linear($\tilde{\mathbf{z}}_{10}$) → $\hat{\mathbf{y}}_6$
↓
Output: $\hat{\mathbf{y}}_6$ (1 of 6 classes)

Figure 3: A simplied diagram of the models. Each layer is written as Operator(Input) → Output. Subscripts denote the dimension (number of features).

| Model | $\mathbf{A}$ | $\mathbf{S}$ with 32 clusters |
|---|---|---|
| 0 | unweighted | n/a |
| 1 | unweighted | n/a |
| 2 | weighted | n/a |
| 3 | unweighted | unweighted $\mathbf{A}$ |
| 4 | unweighted | KNN ($k = 5$) |
| 5 | unweighted | KNN ($k = 50$) |
| 6 | unweighted | KNN ($k = 150$) |
| 7 | weighted | unweighted $\mathbf{A}$ |
| 8 | weighted | KNN ($k = 5$) |
| 9 | weighted | KNN ($k = 50$) |
| 10 | weighted | KNN ($k = 150$) |

Table 2: Model specific construction of $\mathbf{A}$ and $\mathbf{S}$.

### 3.2.2 Loss Function

$$\mathcal{L} = \sum_{u \in \mathcal{V}_{\text{train}}} - \log(\text{softmax}(\hat{\mathbf{y}}_{\mathcal{G}}, \ \mathbf{y}_{\mathcal{G}})) \tag{2}$$

All models were trained with the same loss function: cross-entropy loss (Equation 2). Cross-entropy loss measures the performance of the classification model where the prediction vector $\hat{\mathbf{y}}_{\mathcal{G}}$ forms a probability simplex. Cross-entropy loss increases as $\hat{\mathbf{y}}_{\mathcal{G}}$ diverges from $\mathbf{y}_{\mathcal{G}}$, hence we want to minimise this value. A perfect classifer has a cross-entropy loss of 0.

### 3.2.3 Computational complexity

Most of the graph neural networks suffer from the issue of time and space complexity. Graph convolution methods develop schemes for filtering a neighbourhood of a graph which are created on the top of the adjacency matrix. Essentially, a graph classification takes an adjacency matrix of size $\mathcal{O}(|\mathcal{V}|^2)$ as input. As the graph size increases, i.e., the number of graph nodes $|\mathcal{V}|$, increases, the graph convolution methods require the quadratic graph size for each layer (S. Liu, 2020). Moreover, a $L$-layer GCN model has time complexity $\mathcal{O}(L \cdot |\mathcal{V}| \cdot d^2)$ and space complexity $\mathcal{O}(L \cdot |\mathcal{V}| \cdot d + L \cdot d^2)$, where $d$ is the dimension (or number of channels). When compared to Models 0 to 2, the computational time for Models 3 to 11 is significantly larger, due to the added step of graph coarsening using spectral clustering. For the spectral clustering algorithm, the most computationally expensive step is the computation of the eigenvalues and eigenvectors for the Laplacian matrix. Given $|\mathcal{V}|$ data points, the algorithm forms an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and computes the eigenvectors for this matrix. This process has an overall time complexity of $\mathcal{O}(|\mathcal{V}|^3)$ (Li et al., 2011). Thus, we can state that the time complexity of spectral clustering-based graph coarsening is proportional to the time complexity of spectral clustering.

## 4. Numerical Experiments

### 4.1 Expected Results

Table 3 shows the count for the weekend/holiday or weekday in the dataset used. It is evident that the dataset is imbalanced in terms of the count, and thus could potentially create some bias in the predictions, as the models could overfit and tend to predict the majority classes (Rainy Weekday or Cloudy Weekday) as the weather along with day type. With the baseline model, we expect the accuracy to be at least 35%. Another benchmark that can be set for the performance measures is the percentage of the majority class in the dataset. Cloudy weekday comprises of 32% of the dataset and we expect the measures to be higher than that.

| Weekend/Holiday | Rainy | Sunny | Cloudy |
|:---:|:---:|:---:|:---:|
| 0 | 222 | 50 | 226 |
| 1 | 98 | 24 | 103 |

Table 3: Count of Weather Type in the dataset

## 4.2 Experimental Results

All models were trained for 500 epochs with the same configuration: the data is split into 80% for training and 20% for validation/testing, with a batch size of 32. The Adam optimiser with learning rate $\gamma = 0.001$ and weight decay $\lambda = 0.0001$ (with $L_2$ regularisation) were used. To evaluate the performance of the models, we report the Accuracy $= \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\{y_i = \hat{y}_i\}$ for each of the models.

| Model | Training Loss | Validation Loss | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|
| 0 | $1.47 \pm 0.02$ | $1.49 \pm 0.04$ | $0.37 \pm 0.01$ | $0.36 \pm 0.01$ |
| 1 | $1.28 \pm 0.03$ | $1.40 \pm 0.07$ | $0.43 \pm 0.02$ | $0.41 \pm 0.06$ |
| 2 | $1.32 \pm 0.03$ | $1.44 \pm 0.07$ | $0.41 \pm 0.02$ | $0.41 \pm 0.05$ |
| 3 | $1.24 \pm 0.06$ | $1.24 \pm 0.11$ | $0.45 \pm 0.02$ | $0.43 \pm 0.04$ |
| 4 | $1.20 \pm 0.04$ | $\mathbf{1.14 \pm 0.06}$ | $\mathbf{0.46 \pm 0.02}$ | $\mathbf{0.48 \pm 0.03}$ |
| 5 | $1.27 \pm 0.04$ | $1.30 \pm 0.05$ | $0.45 \pm 0.02$ | $0.41 \pm 0.02$ |
| 6 | $\mathbf{1.19 \pm 0.04}$ | $1.22 \pm 0.09$ | $0.46 \pm 0.02$ | $0.45 \pm 0.04$ |
| 7 | $1.39 \pm 0.02$ | $1.62 \pm 0.03$ | $0.40 \pm 0.02$ | $0.35 \pm 0.01$ |
| 8 | $1.36 \pm 0.03$ | $1.54 \pm 0.04$ | $0.41 \pm 0.02$ | $0.39 \pm 0.02$ |
| 9 | $1.36 \pm 0.03$ | $1.60 \pm 0.04$ | $0.41 \pm 0.02$ | $0.37 \pm 0.02$ |
| 10 | $1.36 \pm 0.02$ | $1.57 \pm 0.04$ | $0.41 \pm 0.02$ | $0.38 \pm 0.02$ |

Table 4: Summary of results from all models where model numbers correspond to those on Table 2. (*mean $\pm$ standard deviation*) between 400 and 500 epochs are shown in each cell. The best *mean* score in each column is highlighted in bold.

Table 4 as a result presents several key insights about the performance of the different models. Firstly, when compared to the baseline model (Model 0), Models 1 and 2 showed improvement in both the cross-entropy and accuracy, due to the concatenation of graph features. This could be interpreted that the mean, maximum, and minimum temperatures of London could capture the pattern of seasonal patterns in demands for bikes, and additional features on graph level could help the prediction in general through the concatenation operation. Secondly, the implementation of graph coarsening further improved the models, among which Model 4 (GCN with graph coarsening based on a KNN graph with $k = 5$) scored the best with validation loss of 1.14 and accuracy of 0.48. Model 4 also outperformed Model 3 with validation loss of 1.24 and accuracy of 0.43, implying that the explicit capture of geographical contiguity succeeded in the efficient aggregation of node embeddings, as *Spatial Graph Coasening*. Furthermore, when comparing graphs with weighted and unweighted edges, graph coarsening had a greater impact on weighted graphs. While Models 7 to 10 did not outperform Model 2 in terms of the mean validation loss and accuracy, their standard deviations decreased (i.e. 0.05 of Model 2 $\rightarrow$ 0.02 of Models 8 to 10 for validation accuracy), implying the stabilising effect of graph coarsening while training the models. Finally, the best-performing models were achieved with KNN graphs with $k = 5$, for both weighted and unweighted edges. This suggests that increasing the complexity of KNN graphs do not necessarily improve the efficiency of aggregating node embeddings in neighbours. One of the possible reasons behind this could be that, as $k$ increases the graph would be closer to a complete graph, where no clusters could be obtained through Spectral Clustering.

## 5. Conclusion

This study compared 11 different models to determine which performs the best for predicting weather and day type on the basis of bike-sharing demand. We introduced two new architectures: the (i) concatenation operator of graph features; and (ii) spatial graph coarsening based on geographical contiguity. As expected, the model that implemented spatial graph coarsening in the training phase with 5-nearest neighbours performed the best. However, the best model could only reach up to the accuracy of 48%. This could be potentially because of certain limitations of the models. We only considered data for the years 2021 and 2022, totaling 723 days, which could have added some bias in the predictions as the dataset is relatively small. Given more resources, we could have deployed a larger dataset which would have led to better training and accuracy. Moreover, for the simplicity of predictions, we considered a cross-product of day type and weather type. With a reasonable amount of data, we could have created multiple categories with a combined loss function for the multi-label classification task. Also, the weather data considered was an approximation of London's weather. If a more granular dataset would have been available, such as hourly weather conditions differentiated by district, we could have split the travel records into finer time and geographic units. Lastly, we only used GCN for the message-passing framework. Given more time, we could have used alternative frameworks like GraphSAGE and Graph Attention Networks (GAT) (Hamilton, 2020) which could potentially improve the models. Accuracy may not be a proper measure for imbalanced datasets since it does not distinguish between the numbers of correctly classified examples of different classes and we could have used other metrics like $F_1$-score as it considers false positives and false negatives. However, the framework that we provide can be used as a base for further work and research. Lastly, the cluster size was set to 32 for the comparison between different spectral clustering. The choice of batch size makes a trade-off between the stability of the model and computational complexity. After experimenting it was found that smaller batch sizes tend to easily overfit the data.

# References

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling, 2020.

European Climate Assessment & Dataset. European climate assessment & dataset. URL https://www.ecad.eu/.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation, 2019.

Martin Fleischmann and Dani Arribas-Bel. Spatial signatures of great britain: Cdrc data, Jul 2022. URL https://data.cdrc.ac.uk/dataset/spatial-signatures-great-britain.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Jintao Ke, Xiaoran Qin, Hai Yang, Zhengfei Zheng, Zheng Zhu, and Jieping Ye. Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network. *Transportation Research Part C: Emerging Technologies*, 122:102858, 2021. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2020.102858. URL https://www.sciencedirect.com/science/article/pii/S0968090X20307580.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

Mu Li, Xiao-Chen Lian, James T. Kwok, and Bao-Liang Lu. Time and space efficient spectral clustering via column sampling. In *CVPR 2011*, pages 2297–2304, 2011. doi: 10.1109/CVPR.2011.5995425.

Lei Lin, Zhengbing He, and Srinivas Peeta. Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, 97:258–276, 2018. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.10.011. URL https://www.sciencedirect.com/science/article/pii/S0968090X18300974.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks, 2021.

Office for National Statistics, Jun 2022. URL https://census.gov.uk/census-2021-results.

J. Park S. Liu. Efficient and effective graph convolution networks. *SIAM International Conference on Data Mining (SDM20) Conference*, 2020.

Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.

Transport for London. Cycles open data api, 2023. URL `https://cycling.data.tfl.gov.uk/`.

UK Government. Bank holidays api, 2023. URL `https://www.api.gov.uk/gds/bank-holidays/#bank-holidays`.

Yuyang Wang, Zijie Li, and Amir Barati Farimani. Graph neural networks for molecules, 2023.