Transferable Graph Neural Networks for Inferring Road Type Attributes in Street Networks

CHIDUBEM IDDIA NOZIE AND GAVIN M CARDELE
School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland
Corresponding author: Chidubem Iddianozie (iceanozie@gmail.com)

ABSTRACT In this paper, we study transferable graph neural networks for street networks. The use of Graph Neural Networks in a transfer learning setting is a promising approach to overcome issues such as the lack of good quality data for training purposes. With transfer learning, we can fine-tune a model trained on a rich sample of data before applying to a task with limited observations. Specifically, we focus on the open research problem of inferring the attributes of a street network as a node classification task. An attribute contains descriptive information about a street segment such as the street type. We propose and develop a neural framework capable of learning from multiple street networks using transfer learning to infer the semantics of another street network. Different from previous studies, we are the first to address this problem by learning from more than one street network using graph neural networks. We empirically evaluate our framework on multiple large real-world networks. Our evaluations show that while state-of-the-art methods can be negatively impacted by naive transfer learning, our framework consistently mitigates this phenomenon, with up to a 10% gain in mean transfer accuracy.

INDEX TERMS Transferable graph neural networks, street networks, semantics.

I. INTRODUCTION
Digital maps are a powerful technology and offer tremendous benefits in areas such as humanitarian disaster management, economic policy development and autonomous driving. However, creating and updating digital maps can be both time and labour intensive [1]. Hence, automating the mapping process could reduce the associated production and maintenance costs. Several attempts have been made to investigate the prospects of automated mapping, these attempts span from the use of GPS traces to infer geometries to predicting spatial attributes using contextual information [2]–[4]. However, these approaches are affected by a scarcity of good quality training data for street networks which impedes the performance of models [2], [4]. Transfer learning offers an opportunity to overcome this problem by using knowledge derived from domains with good quality training data to solve tasks in domains with limited training data [5], [6]. The typical transfer learning approach involves pre-training a model for a task on a source dataset and then fine-tuning that model before predictions are made for a target dataset. In this paper, we focus on inferring the attributes of street networks.

Graph Neural Networks (GNNs) have shown promise for inference tasks on unstructured data. Their ability to represent a graph object as a function of its neighbours, through message-passing or aggregation enables them to overcome some limitations of traditional machine learning methods. For example, [7] demonstrate that GNNs are able to overcome the limited receptive field problem of Convolutional Neural Networks in street networks by leveraging the spatial correlation that exists in street networks. It follows then that implementing transfer learning for GNN models could serve as a solution to the problem of automatically generating maps. However, naive transfer learning could negatively impact model performance [8]. This leads to the question: How do we effectively implement transfer of GNNs for inferring the attributes of street networks?

In this paper, we carry out an investigation through a systematic study. As part of our study, we propose a...
neural framework - Street Attribute Transfer Framework (SATF) - that learns to infer the attributes of a street network from other street networks (See Figure 1). The novelty of our framework lies in the transfer of knowledge from multiple source graphs for inference on a target graph. Different from past attempts, we are the first to investigate the use of knowledge from multiple networks for this particular task. We study the performance of our framework by carrying out experiments on multiple large real-world street networks. We compare against state-of-the-art GNN methods and our experiments show that naive GNN models could suffer from negative transfer. However, our framework is capable of improving transfer accuracy by up to 10%. We summarise our contributions: 1) A comprehensive demonstration, showing that naive transfer of GNNs could result in negative transfer for the task of inferring street network attributes. 2) A new framework capable of learning from multiple source graphs to attribute/label a target graph to mitigate naive transfer. 3) An evaluation to demonstrate that the framework mitigates the effects of naive transfer. The results are compared against state-of-the-art methods using real-world datasets.

II. RELATED WORK

There have been several attempts to infer the attributes of street networks. [2] develop approaches to label street networks using noisy aerial images. They develop methods that are capable of overcoming the noise in the data. [3] predict the street types on OpenStreetMap using their geometric properties such as length, turning angle, linearity. They model the street as a graph and learn to infer on the graph using a markov random field model. They use data from Boston and Cambridge, inferring semantics for eight road types and achieve 68% and 65% precision respectively. Their technique does not include contextual features which could be beneficial to model performance. [1] proposes weakly- and semi-supervised segmentation models for automated mapping. They focus on building detection and road segmentation and achieve improved performance by training on data collected from OpenStreetMap. Similarly, [7] develop a hybrid neural architecture composed of a Convolutional Neural Network (CNNs) and a Graph Neural Network (GNN) to infer the attributes of street networks. The GNN component of their architecture overcomes the receptive field limitation of CNNs. This observation agrees with calls for a move, away from traditional methods to methods that encode the inductive bias during the learning process for spatial data [5], [9]–[11].

Graph Neural Networks are a class of inference methods that encode the inductive bias during model development. They can be broadly grouped into spectral and spatial approaches. Spectral GNNs borrow the general idea of neural convolutions [12] through operations on the graph laplacian matrix [13]–[16]. These operations can be computationally expensive, thereby affecting scalability [14]. Spatial GNNs compute representations directly on the graph [17], [18], [6] proposed an architecture for the transferrability of GNNs. Their experiments show that structural similarity between source and target graphs could be an indicator of successful transfer. This is in agreement with the work by [5] where the authors develop a statistical multi-measure to measure the effectiveness of transfer learning across domains. In [19], the authors propose a node importance sampling technique which is effective at addressing the limitations of Graph Neural networks on street networks. [8] study different strategies for the transfer of GNNs on graph classification tasks. They demonstrate that naive transfer learning for graph based tasks could lead to negative transfer. Our work seeks to mitigate negative transfer for the task of inferring the attributes of street networks.

III. PRELIMINARIES

In this section, we define important concepts and notations used in this paper and formulate our problem.

A. GRAPH NEURAL NETWORKS

A Graph Neural Network learns a representation of a graph using the structure of the graph such as the node or edge connectivity. Take a node from a graph for example, modern GNNs will derive a representation for this node from its neighbourhood over a number of iterations. This neighbourhood information is computed through message-passing or neighbourhood aggregation [20].

The general form of a GNN is a neural network that takes a graph \( G = (V, E) \) as input and a real-valued set of features \( H = \{\bar{h}_1, \ldots, \bar{h}_N\} \in \mathbb{R}^{N \times r} \) to produce a representation of node \( u_i \) as \( h'_i \). In this paper, we assume \( H \) is mapped to the nodes of \( G \). The GNN derives a representation for the nodes using information from the local neighbourhood of the node \( \mathcal{N}_i \). Our implementation performs neighbourhood aggregation of features for a node \( u_i \). We employ the graph attention mechanism from [18] to compute importance (or attention) values for the neighbours from the aggregated information. This way, our framework is able to quantify which neighbours to emphasize or not. An activation function is applied to the representations at each layer before passing to the next layer in the network. At the final layer of the network, we apply an activation function to derive class probabilities. We employ the softmax activation function [21] which is suitable for our multi-class node classification problem which involves inferring street types.
TABLE 1. Definition of notations used.

| Notation | Definition                      |
|----------|--------------------------------|
| $G_t$    | A target graph                |
| $S_G$    | A set of source graphs        |
| $\mathcal{V}$ | set of nodes in graph    |
| $\mathcal{E}$ | set of edges in graph       |
| $N$      | $|\mathcal{V}|$ number of nodes in $G$ |
| $H$      | set of features for nodes of $G$ |
| $\mathcal{Y}$ | set of class labels for nodes in $G$ |
| LeakyReLU | The Leaky ReLU activation function |
| $F(.)$   | Leart function                |
| $N_i$    | Neighbourhood of a node $v_i$ |

B. TRANSFER LEARNING

The general transfer learning problem considers two domains $D_s$ and $D_t$. Tasks on both domains can be denoted as $T_s$ and $T_t$ respectively. A function $f(Y|H)$ can be learnt for a given task, where $Y$ represents class labels and $H$ denotes the feature matrix. Transfer learning seeks to improve $f_t(Y|H)$ using knowledge derived from $D_s$. When, $H_s = H_t$, the problem is said to be a homogeneous transfer learning problem while $H_s \neq H_t$ denotes a heterogeneous transfer learning problem [22]. In this paper, we focus on homogeneous transfer learning.

C. PROBLEM FORMULATION

We seek to learn a function $F(.)$, which given a target graph $G_t$ that contains insufficient features, learns to infer the classes ($Y$) from a set of source graphs $S_G$. Let $G_t = (\mathcal{V}, \mathcal{E})$ denote a target graph. The set of source graphs is denoted by $S_G = \{G_{s1}, \ldots, G_{sn}\}$ for $n$ number of graphs. Each graph $G$ represents a street network and is an unweighted, undirected, multi-class graph of size $N = |\mathcal{V}|$ with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$. The features of the nodes in $G$ are denoted by $H = \{h_1, \ldots, h_N\} \in \mathbb{R}^{N \times r}$, which holds an $r-$dimensional real vector representation of the features $\forall v \in \mathcal{V}$. The features are counts of spatial objects that exist within a specified context of the street segment. For example, the count of shopping malls within a 100 metre radius of a street. We discuss the feature generation process in section V-A. The class labels of $Y$ is given by $Y = \{y_1, \ldots, y_c\} \in \mathbb{R}^c$, where $c$ is the number of class labels that exist in $G$. In all the experiments carried out in this paper, we consider $G_t \not\in S_G$.

IV. METHODOLOGY

We propose a Graph Neural Network framework called Street Attribute Transfer Framework (SATF). SATF is a neural framework that learns to attribute a street network using multiple source graphs. The source graphs are representations of the street networks. We present a diagrammatic representation of our framework in Fig 2. We now describe the components of our framework.

A. NODE IMPORTANCE SAMPLING

The type of street networks we consider in this paper are multi-class and exhibit high class imbalance. Further, the distribution of node classes could be sparse and may not conform to assumptions of GNNs such as strong homophily [8]. Hence, we seek to handle this phenomenon in our framework by sampling important nodes used for training the neural networks. Intuitively, we can think of the node importance sampling as encoding the graph structure into the learning process. For the multi-class case of our graphs, we define the importance of a node as the level of influence it wields in the graph. Fig. 3 shows a simple description of node importance that could occur in multi-class graph structures. Here, we see that if we were training a GNN model on a budget, selecting the green node may be a good idea as it is influential in both structures. In this paper, we seek to sample important nodes in our framework in two ways. Both sampling approaches define node importance differently.

The first method (S1) for sampling nodes based on their importance is adopted from [19]. The authors showed that sampling nodes that encodes the global graph structural information could improve model performance. Their proposal was to sample nodes that are influential not only in their class but in the global structure of the graph. They modelled this using the intra-class and global-geodesic distances of nodes in a graph. The importance $\nu$ of a node is defined formally as $I = \frac{1}{2}(C, B)$. Here, $I(\nu)$ is a importance value of a node $\nu$, $C$ represents the intra-class distances and $B$ represents the global-geodesic distances. $C$ and $B$ are modelled using the closeness and betweenness centrality measures respectively [23]. We approximate $B$ using the approach described in [24]. Due to the multi-class nature of the graphs, $C$ is extended from its standard form to recognise class boundaries.

$$C(\nu^*_i) = \frac{N^*}{\sum_{\nu^*_i \in \mathcal{Y}^*} d(\nu^*_i, \nu^*_j)} \tag{1}$$

where, $d(\cdot)$ is the distance between two nodes, $\star$ refers to the class of $\nu_i$ and $N$. We consider $\mathcal{Y}^*$ of any class to be the largest component of connected nodes of $\star$.

The second sampling approach (S2) we employ in this paper is the eigen centrality [25]. Like the first sampling approach, the eigen value denotes the importance of node in a graph. The first sampling approach seeks to estimate this importance albeit recognizing the multi-class nature of the graphs. Whereas, the eigen centrality method computes a node’s importance in relation to the importance of it’s neighbours. A high eigen centrality denotes that a node is connected to many nodes or is connected to influential nodes or both [26]. We consider the eigen centrality method to be a class-agnostic sampling method. The eigen value $E_i$ of a node $\nu_i$ as defined in [25] is

$$E_i = \sum_j A_{ij}E_j \tag{2}$$

where $A_{ij}$ is an element of the adjacency matrix $A$. The nodes are updated as the eigen centrality is iteratively computed. After any number of iterations, the eigen centrality of a node
will equal the sum of the eigen centralities of its neighbours. In this paper, we compute the eigen centrality in 100 iterations.

B. NEURAL ARCHITECTURE

The next component of our framework is the neural network. We build upon the graph attention mechanism [18]. Using this mechanism, we are able to compute aggregate nodal information from neighbours based on their importance. Take into consideration that the attention mechanism ignores global graph structure which could impact model performance [18]. Intuitively, the node importance sampling and the attention mechanism uses graph structure and node features respectively to determine node importance.

We now describe the use of node level features to implement an attention mechanism. Given a set of node features \( \mathbf{H} = \{ \hat{h}_1, \ldots, \hat{h}_N \} \) from a graph network \( G \), where \( \hat{h}_i \in \mathbb{R}^r \) and \( r \) is the number of features in the nodes \( v \in V \), we set \( \mathbf{H} \) to the attributes that correspond to the nodes returned after node importance sampling has occurred. First, we implement an attention mechanism \( a \) by performing a linear transformation \( \mathbf{W}h_i \) on the features of a node \( i \). If \( y_i = \mathbf{W}h_i \), then we can compute the attention value for any two pairs of nodes \( v_i, v_j \) that share an edge as

\[
e_{ij} = a(y_i, y_j) \tag{3}
\]

The attention values between any pair of nodes is normalized using the softmax function, thus:

\[
\alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \mathbf{a}^T (y_i \parallel y_j) \right) \right)}{\sum_{k \in N_i} \exp \left( \text{LeakyReLU} \left( \mathbf{a}^T (y_i \parallel y_k) \right) \right)} \tag{4}
\]

where \( \alpha_{ij} = \text{softmax}(e_{ij}) \), \( \mathbf{a} \) is a weight vector, \( \mathbf{a}^T \) is the transpose, \( \parallel \) is the concatenation operator and \( N_i \) is the neighbourhood of a node \( v_i \). In this paper, we consider \( N_i \) as a single hop. The normalized attention values are fed to the next layer of the neural network as a linear combination of the node features they represent, using a non-linear function \( \sigma \) expressed

\[
\tilde{h}_i' = \sigma \left( \sum_{j \in N_i} \alpha_{ij} \cdot y_j \right) \tag{5}
\]

For multiple independent attention mechanisms, Equation 5 can be extended

\[
\tilde{h}_i' = \| \sum_{k=1}^{K} \alpha_{ij}^k \cdot y_j^k \| \tag{6}
\]

where \( K \) is the number of independent attention mechanisms, \( \alpha_{ij}^k \) denote the normalized attention values returned by the \( k \)-th attention mechanism and \( y_j^k \) is the corresponding weight matrix. In our experiments, we set \( K \) to eight attention mechanisms for the implementation of our framework. Their output is averaged at the final layer before applying the non-linear softmax function (\( \sigma \)).

C. KNOWLEDGE TRANSFER

The ultimate goal of our framework is to produce models that leverage knowledge from multiple street networks to infer labels in a target graph representing the street network. We propose to achieve this through a policy that maximizes a
reward function by leveraging the similarity between source and target networks to predict node classes. Studies have shown that sufficient similarity between source and target distributions is a necessary condition for successful transfer learning [6], [7]. We borrow ideas from optimal transport theory and test for similarity between source and target graphs using the Wasserstein distance [27].

The Wasserstein distance, also called the earth mover’s distance is a distance function between probability distributions and can be interpreted as the minimum work required to match one distribution to the other. Given two probability distributions $A$ and $B$, the Wasserstein distance between them can be defined. The $p$-Wasserstein for $p \in [1, \infty)$ is given

$$W_p(A, B) = \left( \inf_{\omega \in \Omega(A, B)} \int_{M \times M} \|X - Y\|^p d\omega \right)^{1/p}$$

where $M$ is the metric space, $d$ is the ground distance and $\Omega(A, B)$ is the set of all joint probability measures over $M \times M$. In this article, we consider $p = 1$ and $d$ is the euclidean distance. We use node embeddings derived from the node features in place of probability distributions. It follows then that we define the similarity between graphs. The similarity between graphs is formalised

$$\text{sim}(G_i, S_G) = W_1(f(G_i), f(G'_i)) \quad \forall i \in S_G$$

where $G_i$ is a target graph, $S_G$ is a set of source graphs and $f$ is a function which computes an embedding of the graph. We compute the similarity between graphs at the node class level. The idea is that class-wise similarity between graphs would help to mitigate the influence of the class imbalances on the similarity score. This approach is supported by a repeated occurrence of high class balance across the street networks in different cities, see Fig. 4. Algorithm 1 is the pseudocode for computing the class-wise graph similarity.

For a target graph, this algorithm takes as input a set of source graphs, a target graph, class labels under consideration and the similarity function. The output of the process is a set of source graphs for each class in the target graph. The source graphs are determined by the similarity score, a smaller score is better. In our experiments, the data examples used to compute the similarities across the street networks is balanced through random down-sampling. The last portion of our knowledge transfer mechanism is a neural model ensembling mechanism which stacks the models for inference as a function of their similarity to the target graph. This procedure is analogous to the process of stacked ensembles where multiple models are trained in parallel for a single task [28]. However, instead of training as many models as possible, we simply train models for each class using the most suitable graph for that class. See Fig. 2 for a diagrammatic overview of this mechanism.

V. EXPERIMENTS

In this section, we outline the details of our experiments. We train models to infer the street labels / road types in a street network. Specifically, we implement a framework that leverages knowledge from multiple source street networks (source graphs) to infer the attributes of a street network (target graph). The experiments carried out in this study seek to address the following questions:

- **Q1:** Does naive transfer learning impact model performance for the task of inferring street attributes?
- **Q2:** Can our framework mitigate negative transfer and improve transfer gain
- **Q3:** Are the model improvements from our framework, replicated across different street networks?

These questions dictate the experimental design and are discussed further in section VII.

A. DATA

The data - which consists of street networks - used in our experiments are collected from OpenStreetMap [29]. We consider the street networks of four cities - Los Angeles, Rome, Vancouver and New Delhi. The rationale behind choosing the cities were decided on because OpenStreetMap data is usually of high quality in big cities [30], [31]. Firstly, we derived the line graph of each street network such that the streets are represented by nodes and edges denote intersection between streets [32]. Consider $G$ and $G'$ to denote the original graph and the line graph respectively. This means that $N_{G'} = |V_{G'}| = |E_G|$. Where, $N_{G'}$ and $|V_{G'}|$ is the number of nodes in the line graph $G'$ and $|E|$ is the number of edges in the original graph $G$. Note that the graph representation of each street network is an undirected multigraph. We add a self-loop as an edge for every node in the graph in line with the procedure for the attention mechanism [18]. See Table 2 for summary statistics of the networks. We mapped features to the nodes using contextual information derived using the fixed multiple buffer approach as described in [33]. We consider spatial objects such as residential buildings, malls, schools etc. within a street network. Then using the buffer approach, we select a count of each object type within a specified distance from a particular street segment. For example, a node feature for a street will be the count of residential buildings within a 200 metre radius, another feature will be the count...
of residential buildings within a 500 metre radius and so on and so forth. All the graphs in our experiments are connected. We consider the street types as the attributes to be inferred. To remove any bias in the experiments, we used the intersection of features across all the street networks to train the models. The total number of features per node for all street networks was 176 while the number of classes is 9. The street classes considered are: residential, secondary, tertiary, primary, unclassified, trunk, motorway link, primary link and motorway. We chose to test the limits of our framework as long as the street network is still connected. These nine street classes represent the maximum number of classes that could satisfy this connectedness constraint for the cities under consideration. See Fig. 4 for the class distributions in the networks used.

**B. TRAINING THE MODELS**

The input to our framework before training commences is a target graph and a set of source graphs. We implement the transfer procedure by training models on a set of source graphs and then fine-tuning it for the target graph. All the models are trained for 100 epochs. The neural architecture of our framework is a 2-layer feed-forward network with $16$ hidden features. The activation function for the hidden layer. During training, we impose early-stopping with a patience of 20 epochs and save the best model. The models are validated at each epoch using the set of validation nodes. We implement this by down sampling to the smallest occurring class in each street network. We split the training, validation and testing nodes before model training. Also, we use the same set of training, validation and test nodes for all the experiments. The node importance sampling and graph similarity computations are performed on the training nodes. During training, we impose early-stopping with a patience of 20 epochs and save the best model. The models are validated at each epoch using the set of validation nodes.

**C. BASELINES**

We compare our framework against three state-of-the-art GNN approaches: GCN [14], GAT [18], GraphSAGE [17]. The cross-entropy loss is minimized using the Adam optimization rule [34]. For GAT, we use the same architecture as described in section IV. GCN is implemented as a 2-layer feed forward network with 16 hidden features. We use the RELU activation function for the hidden layer. During training, we use the $L_2$ regularization with $\lambda = 0.0001$. The softmax activation function is used on the final layer. GraphSAGE is set up as a 3-layer feed forward network with 16 hidden features. The activation function for the hidden layers is the RELU and softmax for the output layer. The neighbourhood aggregation scheme used is the mean. We set the learning rate $\lambda = 0.001$.

**VI. RESULTS**

We present the results of our experiments in Tables 3, 4 and 5, comparing the performance of our transfer framework against three state-of-the-art approaches. Table 3 shows the mean accuracy and the standard deviations of our framework’s performance compared against naive transfer methods. In this scenario, a naive transfer method refers to any standard machine learning algorithm that is applied as-is without any modification for the task. Recall that a balanced set of nodes was used for training. Of the remainder of nodes for each street network, the validation and testing nodes were split evenly in the experiments. We use the F1 score (See Equation 8) and the accuracy measure to evaluate the model metrics. Table 4 shows the mean F1 Scores and the standard deviations of our framework’s performance compared against naive transfer methods.

$$F1\text{score} = 2 \times \frac{(precision \times \text{recall})}{(precision + \text{recall})}$$ (8)

The F1 score is the micro-averaged F1 score. The scores in Tables 3 and 4 are mean values derived after ten runs on the test sets. At each run, a fixed-size random set of the test nodes is used. We set the random seed to 100 for all test runs.

We perform an ablation study to understand the impact of the node importance sampling technique on model performance as presented in Tables 3 and 4. GCN, GAT and GraphSAGE denotes the results from our baselines. The row GNN-SATF denotes the results from our framework with...
TABLE 3. We show the results of our framework compared against the baselines. Results presented here are the averages of accuracy taken over multiple runs. The standard deviations are shown alongside rounded to the third decimal place. Best and second best results for each dataset are in bold.

| Method      | Los Angeles | Rome       | Vancouver  | New Delhi  |
|-------------|-------------|------------|------------|------------|
| GCN         | 0.124 ± 0.017 | 0.169 ± 0.015 | 0.133 ± 0.029 | 0.161 ± 0.049 |
| GAT         | 0.109 ± 0.019 | 0.143 ± 0.013 | 0.111 ± 0.037 | 0.178 ± 0.082 |
| GraphSAGE   | 0.118 ± 0.023 | 0.170 ± 0.013 | 0.128 ± 0.037 | 0.183 ± 0.007 |
| GNN-SATF    | 0.162 ± 0.009 | 0.071 ± 0.006 | 0.072 ± 0.037 | 0.217 ± 0.031 |
| GNN-SATF (S1) | 0.227 ± 0.017 | 0.063 ± 0.013 | 0.122 ± 0.044 | 0.172 ± 0.055 |
| GNN-SATF (S2) | 0.099 ± 0.015 | 0.186 ± 0.010 | 0.161 ± 0.032 | 0.205 ± 0.046 |

TABLE 4. We show the results of our framework compared against the baselines. Results presented here are the averages of the micro F1 score. The standard deviations are shown alongside rounded to the third decimal place. Best and second best results for each dataset are in bold.

| Method      | Los Angeles | Rome       | Vancouver  | New Delhi  |
|-------------|-------------|------------|------------|------------|
| GCN         | 0.135 ± 0.025 | 0.123 ± 0.013 | 0.083 ± 0.054 | 0.122 ± 0.035 |
| GAT         | 0.123 ± 0.019 | 0.135 ± 0.018 | 0.106 ± 0.071 | 0.228 ± 0.049 |
| GraphSAGE   | 0.112 ± 0.015 | 0.186 ± 0.020 | 0.128 ± 0.037 | 0.178 ± 0.051 |
| GNN-SATF    | 0.162 ± 0.009 | 0.078 ± 0.011 | 0.084 ± 0.039 | 0.222 ± 0.037 |
| GNN-SATF (S1) | 0.219 ± 0.002 | 0.061 ± 0.010 | 0.122 ± 0.044 | 0.189 ± 0.054 |
| GNN-SATF (S2) | 0.097 ± 0.017 | 0.182 ± 0.012 | 0.172 ± 0.041 | 0.206 ± 0.037 |

TABLE 5. We present the overall transfer gain, comparing our framework and the baseline models against their non pre-trained equivalents. We observe negative transfer on the baseline models while there is a clear gain from our framework for the attributing task.

| Method      | Los Angeles | Rome       | Vancouver  | New Delhi  |
|-------------|-------------|------------|------------|------------|
| GCN         | -0.3%       | 4.0%       | -2.3%      | 3.3%       |
| GAT         | -1.3%       | 5.2%       | -2.1%      | 2.1%       |
| GraphSAGE   | -1.7%       | 7.5%       | -1.6%      | 2.1%       |
| GNN-SATF    | 10.5%       | 9.5%       | 2.8%       | 6.1%       |

VII. DISCUSSION

In this section, we discuss the experimental results. In tables 3 and 4, we present the performance of the baseline models and that of our framework. Recall that for the baselines, we train multiple models for each target dataset using each of the others datasets, except the target dataset. The values shown in the tables are that of the best performing model for each target dataset. We observe that our framework shows the best or second-best performance in the majority of cases. We recognise that the variation in model performance with respect to the node importance sampling technique (S1 and S2) used suggests that certain sampling techniques may be more suited for a dataset (in our case the graph representation of different cities) over others. Next, we discuss the performance of our framework by addressing the questions posed in section V.

Q1: Does naive transfer learning impact model performance for the task of inferring street attributes?

- In order to understand the impact of naive transfer learning on model performance, we need to compare the performance of our framework against transfer models trained naively. In table 5, we present the difference in performance from models trained naively versus using the target model itself. In essence, the performance difference will let us know if there is any benefit to using a transfer learning model in the first place. Looking at table 5, we notice that in all cases, the model performance is negatively impacted by naive transfer learning in all the datasets. For Los Angeles and Vancouver, we see clear occurrences of negative transfer from the baseline models. For Rome and New Delhi, we observe that the transfer gain achieved from using our framework is better than the baselines. Given these observation, we can
conclude that naive transfer learning will impact model performance.

Q2: Can our framework mitigate negative transfer and improve transfer gain?

- Similar to Q1, we need to compare the performance difference between naive transfer models and our framework to understand how much improvement is derived from using our framework. From table 5, we can see that our framework provides up to a 10.5% transfer accuracy improvement even in situations where state-of-the-art methods exhibit negative transfer. This shows that our framework mitigates negative transfer and improves transfer gain.

Q3: Are the model improvements from our framework replicated across different street networks?

- To answer the question, we need to evaluate our frameworks’ performance relative to multiple datasets. In table 5, we see that the improvements derived from our framework are evident across the different datasets. Our results suggest that the performance of our framework are not arbitrary as they consistently show the highest transfer gain across the street networks for all methods we compared against. So, to answer Q3, our framework shows consistent performance across the datasets.

In summary, our experimental results suggest that using our framework could benefit the inference of street attributes using transfer learning. We recognise that the results in tables 3 and 4 suggests that the models may not be suitable for real-world inference of street types. We would like to reiterate that there have been approaches to street type inference that have shown varied performance [1]–[3], [7]. Nonetheless, our work distinguishes itself from these approaches as the focus of this paper is on the potential of transfer learning and Graph Neural Networks to solve the data availability problem for the street type inference problem. Furthermore, we incorporate nine street classes, which coupled with the prevalent class imbalance could negatively impact model performance.

VIII. CONCLUSION

In this paper, we have carried out a study on the potential of transferable graph neural networks for street networks. As part of our study, we proposed a neural framework that learns from multiple source graphs to infer the attributes of a target graph to mitigate negative transfer. This is different from studies which consider single source and target graphs. Our framework achieves this through a class-wise graph similarity measure, a node importance sampling and a model ensembling mechanism. We evaluate our framework against state-of-the-art methods on four street networks. With as much as a 10% gain in mean accuracy, the results strongly suggests that our framework could be a good approach to transfer learning for inferring the attributes of street networks. Further, our framework can be generalised to address other inference tasks. As far we know, we are the first to investigate the transfer of knowledge from multiple networks for the task of inferring street network attributes.

Automatically inferring the attributes of street networks is an important problem with far-reaching applications. For example, it could facilitate the creation and update of maps for humanitarian assistance and autonomous driving. We believe that our framework will help to further investigations in this direction. For future work, we will research the extension of our framework to other spatial prediction tasks. An interesting application could be investigating transferable graph neural networks on heterogeneous representations of spatial networks [36]. Also, we will seek to understand the knowledge learnt by pre-trained models in order to improve the generalizability of our framework. We note the variation in the model performance in different sampling approaches and on different datasets. Understanding the context in which the sampling techniques work best is a further area for future work which is likely to lead to an improved performance overall.

REFERENCES

[1] D. Bonafilia, J. Gill, S. Basu, and D. Yang, “Building high resolution maps for humanitarian aid and development with weakly-and semi-supervised learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2019, pp. 1–9.
[2] V. Mnih and G. E. Hinton, “Learning to label aerial images from noisy data,” in Proc. 29th Int. Conf. Mach. Learn. (ICML), 2012, pp. 567–574.
[3] P. Corcoran, M. Jilani, P. Mooney, and M. Bertolotto, “Inferring semantics from geometry: The case of street networks,” in Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., Nov. 2015, p. 42.
[4] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk, “A comparison and evaluation of map construction algorithms using vehicle tracking data,” GeoInformatica, vol. 19, no. 3, pp. 601–632, 2015.
[5] C. Iddianozie and G. Mcardle, “A transfer learning paradigm for spatial networks,” in Proc. 34th ACM/SIGAPP Symp. Appl. Comput., Apr. 2019, pp. 659–666.
[6] J. Lee, H. Kim, J. Lee, and S. Yoon, “Transfer learning for deep learning on graph-structured data,” in Proc. AAAI, 2017, pp. 2154–2160.
[7] S. He, F. Bastani, S. Jagwani, E. Park, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and M. A. Sadeghi, “Roadtagger: Robust road attribute inference with graph neural networks,” in Proc. AAAI, 2020, pp. 10965–10972.
[8] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, “Strategies for pre-training graph neural networks,” 2019, arXiv:1905.12265.
[9] P. W. Battaglia et al., “Relational inductive biases, deep learning, and graph networks,” 2018, arXiv:1806.01261.
[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.
[11] K. Janowicz, S. Gao, G. McKenzie, Y. Hu, and B. Bhaduri, GeoAI: Spatially Explicit Artificial Intelligence Techniques for Geographic Knowledge Discovery and Beyond, Taylor & Francis, 2020.
[12] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” in The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10. Cambridge, MA, USA: MIT Press, 1995, p. 1995.
[13] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” 2013, arXiv:1312.6203.
[14] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907.
[15] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 2224–2232.
[16] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3844–3852.
W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1024–1034.

P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” 2017, arXiv:1710.10903.

C. Iddianozie and G. McArdle, “Improved graph neural networks for spatial networks using structure-aware sampling,” ISPRS Int. J. Geo-Inf., vol. 9, no. 11, p. 674, Nov. 2020.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in Proc. 34th Int. Conf. Mach. Learn., vol. 70, 2017, pp. 1263–1272.

R. A. Dunne and N. A. Campbell, “On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function,” in Proc. 8th Aust. Conf. Neural Netw., Melbourne, VIC, Australia, vol. 181, Jun. 1997, p. 185.

K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” J. Big Data, vol. 3, no. 1, p. 9, 2016.

L. C. Freeman, “Centrality in social networks conceptual clarification,” Soc. Netw., vol. 1, no. 3, pp. 215–239, 1979.

U. Brandes and C. Pich, “Centrality estimation in large networks,” Int. J. Bifurcation Chaos, vol. 17, no. 7, pp. 2303–2318, 2007.

P. Bonacich, “Power and centrality: A family of measures,” Amer. J. Sociol., vol. 92, no. 5, pp. 1170–1182, 1987.

M. Newman, Networks. London, U.K.: Oxford Univ. Press, 2018.

A. Ramdas, N. Trillos, and M. Cuturi, “On wasserstein two-sample testing and related families of nonparametric tests,” Entropy, vol. 19, no. 2, p. 47, Jan. 2017.

D. H. Wolpert, “Stacked generalization,” Neural Netw., vol. 5, no. 2, pp. 241–259, 1992.

G. Boeing, “OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks,” Comput. Environ. Urban Syst., vol. 65, pp. 126–139, Sep. 2016.

M. Haklay, “How good is volunteered geographical information? A comparative study of OpenStreetMap and ordnance survey datasets,” Environ. Planning B, Planning Design, vol. 37, no. 4, pp. 682–703, Aug. 2010.

P. Mooney, P. Corcoran, and A. C. Winstanley, “Towards quality metrics for OpenStreetMap,” in Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS), 2010, pp. 514–517.

R. Hemminger and L. W. Beineke, Line Graphs and Line Digraphs, Selected Topics in Graph Theory, W. B. Lowell and R. J. Wilson, Eds. New York, NY, USA: Academic, 1978.

C. Iddianozie, M. Bertolotto, and G. McArdle, “Exploring budgeted learning for data-driven semantic inference via urban functions,” IEEE Access, vol. 8, pp. 32258–32269, 2020.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, “Pytorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8026–8037.

C. Iddianozie and G. McArdle, “Towards robust representations of spatial networks using graph neural networks,” Appl. Sci., vol. 11, no. 15, p. 6918, Jul. 2021.