Link Prediction for Flow-Driven Spatial Networks

Bastian Wittmann
University of Zurich
bastian.wittmann@uzh.ch

Johannes C. Paetzold
Technical University of Munich
johannes.paetzold@tum.de

Chinmay Prabhakar
University of Zurich
chinmay.prabhakar@uzh.ch

Daniel Rueckert
Technical University of Munich
daniel.rueckert@tum.de

Bjoern Menze
University of Zurich
bjoern.menze@uzh.ch

Abstract

Link prediction algorithms aim to infer the existence of connections (or links) between nodes in network-structured data and are typically applied to refine the connectivity among nodes. In this work, we focus on link prediction for flow-driven spatial networks, which are embedded in a Euclidean space and relate to physical exchange and transportation processes (e.g., blood flow in vessels or traffic flow in road networks). To this end, we propose the Graph Attentive Vectors (GAV) link prediction framework. GAV models simplified dynamics of physical flow in spatial networks via an attentive, neighborhood-aware message-passing paradigm, updating vector embeddings in a constrained manner. We evaluate GAV on eight flow-driven spatial networks given by whole-brain vessel graphs and road networks. GAV demonstrates superior performances across all datasets and metrics and outperformed the state-of-the-art on the ogbl-vessel benchmark at the time of submission by 12% (98.38 vs. 87.98 AUC). All code is publicly available on GitHub.1

1. Introduction and Motivation

Networks (or graphs) can serve as efficient representations of real-world, ultra-complex systems and can be further classified into different categories. A prominent category is represented by undirected networks embedded in a Euclidean space constrained by geometry, called spatial networks [5]. In this work, we are focusing on spatial networks, where a form of physical exchange or flow can be used to describe characteristic functional properties of the underlying physical system. Examples include road networks, water bodies, and global exchange networks, but they can also be found in biology (e.g., vascular system, lymphatic system, and connectome). We will refer to such networks as flow-driven spatial networks (see Fig. 1).

Predominantly, network representations of physical systems originate from imaging methodologies, such as nanometer-scale microscopy in biology or regional to continental scale satellite remote sensing for road networks. The generation of compact network representations from these images is a multi-stage and imperfect process (segmentation, skeletonization, and subsequent graph pruning), which introduces noise and artifacts. However, for many relevant downstream tasks operating on flow-driven spatial networks, such as blood flow modeling [34] - a crucial component of investigating neurovascular brain disorders - or traffic forecasting [20], a flawless network representation is of utmost importance. In this context, the task of link prediction presents itself as a meaningful measure to identify absent connections and reduce local noise arising from the erroneous graph generation process [17, 32]. Link prediction algorithms tailored to flow-driven spatial networks, however, remain heavily under-explored.

Therefore, we bring a simplistic yet general definition of the principle of physical flow, characterized by a direction and magnitude, to link prediction in graph representation learning. Our hypothesis is that for flow-driven spatial networks, link prediction algorithms should heavily benefit from considering known functional properties, such as the aforementioned physical flow, which are defined by the structural properties of the network (e.g., bifurcation angles [35]). To this end, we propose the Graph Attentive...
Vectors (GAV) link prediction framework. GAV operates on vector embeddings representative of the network’s structural properties and updates them in a constrained manner, imitating simplified dynamics of physical flow in spatial networks (e.g., blood flow in the vascular system or traffic flow in road networks). We summarize our core contributions as follows:

1. We propose an attentive, neighborhood-aware message passing layer, called GAV layer, which updates vector embeddings, mimicking the change in direction and magnitude of physical flow in spatial networks.
2. We introduce a readout module that aggregates vector embeddings in a physically plausible way and thus facilitates the interpretability of results.
3. We prove the above-mentioned hypothesis by demonstrating state-of-the-art performance across all metrics in extensive experiments on eight flow-driven spatial networks, including the Open Graph Benchmark’s [17] ogbl-vessel benchmark2 (98.38 vs. 87.98 AUC).

2. Related Works

This section discusses previous work on link prediction algorithms and message-passing layers. Particular emphasis is placed on methods featured in our experiments.

2.1. Link Prediction

Link prediction algorithms are applied in various fields, such as social network analysis [10, 26, 30], bioinformatics [21, 25, 36], recommender systems [2, 18, 37, 53], and supply chain improvement [28]. Broadly speaking, different link prediction algorithms try to estimate link existence between two nodes either via heuristic or learned methods.

Heuristic algorithms employ predefined heuristics to encode the similarity between nodes. Some prominent candidates are represented by common neighbors, resource allocation [54], preferential attachment [4], Adamic-Adar [1], Jaccard [19], Katz [22], and average commute time [13]. However, all heuristic link prediction algorithms suffer from the same underlying issue. They exploit predefined, simple heuristics, which can not be modified to account for different network types. E.g., common neighbors has been developed for social networks and hence yields overwhelming results when applied to molecular graphs.

On the other hand, learned algorithms do not rely on pre-designed heuristics but rather learn a more complex, data-driven heuristic utilizing neural networks. Thus, learned algorithms can easily adapt to different network types while typically outperforming their heuristic counterparts. SEAL [49, 51] represents a prominent, learned link prediction framework, defining link prediction as a subgraph-level classification task by training a binary GNN-based classifier to map from subgraph patterns to link existence. To this end, SEAL first extracts a local subgraph around the link of interest, which is subsequently forwarded to DGCNN [50] for classification. Moreover, SEAL’s authors introduce an additional node labeling technique, known as labeling trick [51], to enhance the expressiveness of node features obtained from GNNs. SIEG [14] builds upon SEAL and introduces, inspired by Graphormer [48], a pairwise structural attention module between two nodes of interest to capture local structural information more effectively. This results in state-of-the-art performances and allows SIEG to simultaneously overcome Graphormer’s issue of exploding computational complexity when applied to ultra-large graphs. SUREL+ [46] introduces the use of node sets to represent subgraphs. To complement the loss of structural information, SUREL+ provides set samplers, structure encoders, and set neural encoders. SUREL+ outperforms the baseline SEAL on various link prediction benchmarks. It should be mentioned that Cai et al. [7] investigate the use of line graphs for link prediction. Please note that none of the above-mentioned methods are tailored to flow-driven spatial networks.

2.2. Message-Passing Layers

GNNs utilize the concept of message-passing to encode semantically rich features within network-structured data. Over time, multiple variations of message-passing layers have been proposed [11, 12, 16, 43]. For instance, GCN’s message-passing layer [24] weighs each incoming message with a fixed coefficient, the node degree, before aggregation. In contrast, GAT’s message-passing layer [6] learns aggregation weights dynamically based on attention scores. GraphSAGE’s message-passing layer [15] does not directly aggregate central node features with incoming messages. Instead, it distinguishes these two kinds of features and learns two different transformations, one on the central node and another on incoming messages. EdgeConv [41] aggregates the feature difference between the central node and its neighbors combined with the central node’s features. Thus, EdgeConv draws parallels to aggregating spatial vectors if the nodes embed spatial positions. However, our proposed GAV layer differs significantly from EdgeConv, as we explicitly constrain the update of vector embeddings to imitate the simplified dynamics of physical flow in flow-driven spatial networks. Importantly, only a few works tried to adapt the message-passing paradigm to spatial networks [9, 52].

3. The Graph Attentive Vectors Framework

Our proposed Graph Attentive Vectors (GAV) link prediction framework is depicted in Fig. 2. GAV represents a simple yet effective end-to-end trainable framework tailored to the task of link prediction for flow-driven spatial net-
works. It predicts the probability of link (or edge) existence between two nodes in a graph $G$ based on a binary classifier $F$, composed of a novel message-passing and readout module operating on vector embeddings (see Sections 3.2 and 3.4). To this end, $F$ should be able to differentiate between positive (real) and negative (sampled) links by assigning high probabilities of existence to plausible and low probabilities of existence to implausible links. Following most competitive approaches [14, 49, 51], we treat link prediction as a subgraph classification task. To determine the probability of existence of an individual target link between two target nodes, we, therefore, first extract an enclosing subgraph describing the target link’s local neighborhood in a subgraph extraction module (see Section 3.1). Subsequently, the subgraph is transformed into a line graph representation to construct vector embeddings and forwarded to $F$, resulting in an iterative link prediction scheme predicting the existence of target links one at a time. In the following, we elaborate extensively on GAV’s individual components (see Fig. 2). For ease of reference, we provide a notation lookup table in the supplementary (see Supp. A).

### 3.1. Subgraph Extraction Module

The undirected input graph $G := (V, E)$ is defined by a set of nodes $V$ and a set of corresponding edges $E$. While a node $n_i \in V$ contains a specific spatial position given by coordinates $(n_i, \in \mathbb{R}^{d_{spatial}})$, an edge $e_{ij} \in E$ indicates a connection between nodes $n_i$ and $n_j$.

Similar to most competitive approaches [14, 49, 51], we extract as a first step an $h$-hop enclosing subgraph $G^h_t$ around the nodes $\{n'_i, n'_j\}$ affiliated to the target link $e'_{ij}$ from the original graph representation $G$ (please note that we refer to the target in our notations as $t$). This results in an expressive and efficient representation of the target link’s local neighborhood, including relevant structural patterns necessary to determine link existence. Further, the subgraph extraction significantly reduces space complexity, which is crucial for link prediction in ultra-large graphs (see Table 1).

Since GAV operates on vector embeddings, we subsequently transform $G^h_t$ into a line graph representation $L(G^h_t) := (V', E')$. In $L(G^h_t)$, each node $n'_i \in V'$ represents an edge $e_{ij} \in E$ via an individual vector embedding, while an edge $e'_{ij} \in E'$ indicates adjacency between two edges in $G^h_t$ iff they are incident. To create vector embeddings, we encode edges as vectors between the involved nodes (e.g., $n'_i = n_j - n_i$). Therefore, $L(G^h_t)$’s node embeddings are defined as vectors (see Fig. 2) of unique length and direction representative of edges in $G^h_t$ (see Supp. C for implementation details). The line graph $L(G^h_t)$ formed around the target link $e'_{ij}$ is finally forwarded to the message-passing module.

### 3.2. Message-Passing Module and GAV Layer

To adjust link prediction to flow-driven spatial networks, we propose a novel message-passing layer, termed GAV layer. We perform $k$ iterations of message-passing among the nodes of the line graph $L(G^h_t)$, obtained from our subgraph extraction module, in our message-passing module. The GAV layer’s message-passing relies on a straightforward intuition inspired by principles of physical flow. To be precise, we treat nodes in $L(G^h_t)$ as vector embeddings and update them in a constrained manner, imitating simplified dynamics of physical flow in spatial networks. The detailed structure of a single GAV layer is visualized in Fig. 3.

In order to update an individual vector embedded in a node $n'_i \in \mathbb{R}^{d_{spatial}}$, we first project it together with a matrix $N_i \in \mathbb{R}^{N(n'_i) \times d_{spatial}}$, consisting of directly neighboring nodes and the node itself, into a higher dimensional space $d_{message}$; the projection is through a learnable func-
tor embeddings given by nodes in $L$ embeddings, preserving the network’s structural properties.

Subsequently, $\phi_0^{(1)}(n'_i)$ and $\phi_0^{(1)}(N_i)$ are forwarded to a multi-head attention operation, where $\phi_0^{(1)}(n'_i)$ represents a single query, while $\phi_0^{(1)}(N_i)$ represents the key and value sequence.

$\hat{n}'_i = \text{MultiHeadAttn}(\phi_0^{(1)}(n'_i), \phi_0^{(1)}(N_i), \phi_0^{(1)}(N_i))$ (1)

This results in an intermediate node representation $\hat{n}'_i \in \mathbb{R}^{d_{\text{message}}}$, which incorporates not only information of the node itself but also its local structural neighborhood via the concept of attention. In the next step, we apply a residual connection for increased gradient flow and forward the result to the learnable function $\phi_{\theta}(2) : \mathbb{R}^{d_{\text{message}}} \rightarrow \mathbb{R}$, followed by a tanh non-linearity.

$s_i = \tanh(\phi_{\theta}(2)(\hat{n}'_i + \phi_0^{(1)}(n'_i)))$ (2)

Finally, the scalar value $s_i \in (-1, 1)$ is utilized to update the original node representation $n'_i$ via scalar multiplication.

$\hat{n}'_i = s_i \cdot n'_i$ (3)

Hence, after one layer of message-passing, the updated, refined node representation is given by $\hat{n}'_i \in \mathbb{R}^{d_{\text{message}}}$. Importantly, $\hat{n}'_i$ preserves relevant structural properties of the original node representation $n'_i$. This is because scalar multiplication with $s_i \in (-1, 1)$ restricts the modification of vector embeddings given by nodes in $L(G'_h)$. In essence, our message-passing paradigm can be geometrically interpreted as a scaling combined with a potential flipping operation of vectors. These constraints imposed by our message-passing align with our principal idea of modeling simplified dynamics of physical flow in spatial networks via potential, constrained changes in the direction and magnitude of vector embeddings, preserving the network’s structural properties.

3.3. Labeling Trick

Following common practices [7, 14, 51], we apply a labeling trick to enable the message-passing module to distinguish between the relevance of different vector embeddings and, hence, learn an expressive structural representation of the target link’s local neighborhood. The labeling trick ensures that vector embeddings created from the target link $e'_t \setminus t$ and edges connected to the target nodes $\{n'_t, n'_j\}$ are identifiable by a distinct label. Labels generated by our interpretation of the labeling trick are shown in Fig. 4.

![Figure 4. Labels generated by the labeling trick for an exemplary line graph (h set to two). Our interpretation of the labeling trick assigns the label 0 to the vector embedding representing the target link (orange), the labels 1 and 2 to vector embeddings representing edges connected to the target nodes $n'_t$ and $n'_j$ (purple and blue), and the label 3 to remaining vector embeddings.](image_url)

The additional labels generated by the labeling trick are concatenated to $L(G'_h)$’s vector embeddings in the message-passing module (before message-passing).

3.4. Readout Module

Our proposed readout module consists of a custom node aggregation operation followed by a learnable function $\phi_{\theta}^{(3)} : \mathbb{R}^{2d_{\text{message}}} \rightarrow \mathbb{R}$ and processes refined vector embeddings obtained from the message-passing module. While the node aggregation operation aims to distill pertinent information from the refined graph representation in a fashion invariant to node ordering and quantity, $\phi_{\theta}^{(3)}$ predicts the probability of link existence with regard to the target link.

Equation 4 summarizes the readout module’s functionality, where $\mathcal{E}_{N(n'_t)}$ defines the set of refined vector embeddings originally created from edges adjacent to $n'_t$ (see Fig. 2), $\mathcal{E}_{N(n'_j)}$ the set of refined vector embeddings originally created from edges adjacent to $n'_j$ (see Fig. 2), $\parallel$ the concatenation operation, and $\hat{y}'_{ij}$ the predicted probability of target link existence.

$\hat{y}'_{ij} = \phi_{\theta}^{(3)}(\text{mean}(\mathcal{E}_{N(n'_t)})) \parallel \text{mean}(\mathcal{E}_{N(n'_j)}))$ (4)

Thus, our node aggregation operation constructs mean vector embeddings (see $\text{mean}(\mathcal{E}_{N(n'_t)})$ and $\text{mean}(\mathcal{E}_{N(n'_j)})$) representative of our simplified definition of flow in the two target nodes and intends to exploit their relationship to predict whether target nodes should be connected or not.
3.5. Loss Function

Since our approach represents a binary classifier $\mathcal{F}$ determining the probability of link existence with regard to the target link, we optimize a binary cross-entropy loss function during training. Here, $y_{ij} \in \{0, 1\}$ indicates whether the target links are negative (sampled) or positive (real).

$$
L_{\text{BCE}} = -\frac{1}{|\mathcal{E}|} \sum_{ij \in \mathcal{E}} y_{ij} \cdot \log(\hat{y}_{ij}) + (1 - y_{ij}) \cdot \log(1 - \hat{y}_{ij})
$$

We would like to highlight that our approach is trained in an entirely end-to-end manner, solely based on the information of link existence. Therefore, intermediate representations, such as the refined vector embeddings, are determined completely data-driven.

4. Experiments and Results

In this section, we demonstrate the performance of our proposed GAV framework on the ogbl-vessel benchmark [17] and on additional datasets sourced from publicly available flow-driven spatial networks. We first elaborate on baseline algorithms and the experimental setup, followed by a detailed description of datasets utilized in our experiments. Finally, we report quantitative results, investigate our design choices by conducting extensive ablation studies, and discuss GAV’s interpretability in detail. Additional experiments on non-flow-based benchmarks and rototranslational invariance can be found in the supplementary.

4.1. Baselines and Experimental Setup

To evaluate GAV properly, we not only compare GAV to algorithms submitted to the ogbl-vessel benchmark but also propose a novel secondary baseline (SEAL+EdgeConv) combining the SEAL framework [49, 51] with the EdgeConv message-passing layer [41], following recent trends in graph-based object detection from point clouds [8, 40, 45, 47]. Extensive experiments with different baseline algorithms revealed that this provides us with an improved, highly competitive secondary baseline for link prediction in spatial networks. We, therefore, compare GAV on the additionally sourced datasets to SEAL’s original version (with tuned hyperparameters), which has shown to deliver results on par with or superior to the state-of-the-art on multiple link prediction benchmarks, and to our introduced secondary baseline SEAL+EdgeConv. Details on the configuration of our secondary baseline can be found in the supplementary (see Supp. H).

GAV was trained using the Adam optimizer [23] with a learning rate of 0.001 and a batch size of 32 on a single Tesla V100 GPU (32 GB) until convergence. An ablation study on the number of hops $h$ in the subgraph extraction module and the number of message-passing iterations $k$ indicates that setting both to one is sufficient (see Table 5). In the GAV layer, the number of heads of the multi-head attention operation is set to 4, while $\phi^{(1)}_\theta$ represents a single linear layer with an output dimension of $d_{\text{message}} = 32$, and $\phi^{(2)}_\theta$ is given by a two-layer MLP with a hidden dimension of 64. The GAV layer makes use of leaky ReLU non-linearities [29] to increase gradient flow and simplify weight initialization. The readout module’s learnable function $\phi^{(3)}_\theta$ is represented by a two-layer MLP with a hidden dimension of 128. All hyperparameters were tuned on the validation set of the ogbl-vessel benchmark.

4.2. Datasets

We experiment with multiple 2D and 3D flow-driven spatial networks to demonstrate the generalizability of our approach (see Table 1). In total, we conduct experiments on eight networks, including the ogbl-vessel benchmark and seven additionally sourced datasets, representing whole-brain vessel graphs of different mouse strains and road networks of various European countries. In this context, link prediction can be interpreted as predicting the probability of the existence of blood vessels and road segments. Whole-brain vessel graphs and road networks are graphically visualized in the supplementary (see Supp. E).

Whole-Brain Vessel Graphs Blood vessels represent fascinating structures forming complex networks that transport oxygen and nutrients throughout the human body. The vascular system is, therefore, intuitively represented as a flow-driven spatial network, where branching points of ves-
Table 2. Quantitative results. GAV outperforms the previous state-of-the-art across all metrics and datasets. We report mean and standard deviation values on the ogbl-vessel benchmark based on ten different random seeds (0 - 9). To compare GAV to existing baseline algorithms, we report quantitative results based on the area under the receiver operating characteristic curve (AUC). We additionally utilize the evaluation metric Hits@k as an additional, stricter performance measure. More details on evaluation metrics can be found in the supplementary (see Supp. G). Please note that the ogbl-vessel benchmark’s official evaluation metric is AUC. Therefore, Hits@k values of participating algorithms are not available. We highlight the respective three best quantitative values in shades of teal.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th># Params</th>
<th>AUC (%)</th>
<th>Hits@100 (%)</th>
<th>Hits@50 (%)</th>
<th>Hits@20 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ogbl-vessel</td>
<td>GCN [24]</td>
<td>396,289</td>
<td>43.53 ± 9.61</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>1,037,577</td>
<td>47.94 ± 1.33</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Adamic-Adar [1]</td>
<td>0</td>
<td>48.49 ± 0.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE [15]</td>
<td>396,289</td>
<td>49.89 ± 6.78</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SAGE+JKNet [44]</td>
<td>273</td>
<td>50.01 ± 0.07</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SGC [42]</td>
<td>897</td>
<td>50.09 ± 0.11</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>LRG [33]</td>
<td>26,577</td>
<td>54.15 ± 4.37</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SEAL [51]</td>
<td>172,610</td>
<td>80.50 ± 0.21</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>S3GRL (PoS*) [27]</td>
<td>2,382,849</td>
<td>80.56 ± 0.06</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SUREL+ [46]</td>
<td>56,353</td>
<td>84.96 ± 0.68</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>STEGI [14]</td>
<td>752,716</td>
<td>87.98 ± 1.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>97.53 ± 0.32</td>
<td>16.09 ± 10.48</td>
<td>9.37 ± 6.18</td>
<td>4.99 ± 4.24</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>98.38 ± 0.02</td>
<td>34.77 ± 0.94</td>
<td>28.02 ± 1.58</td>
<td>19.71 ± 2.31</td>
</tr>
<tr>
<td>c57-tc-vessel</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>78.21</td>
<td>0.12</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>97.23</td>
<td>16.71</td>
<td>10.39</td>
<td>5.01</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>98.24</td>
<td>33.26</td>
<td>26.89</td>
<td>21.32</td>
</tr>
<tr>
<td>cd1-tc-vessel</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>83.60</td>
<td>0.27</td>
<td>0.16</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>97.91</td>
<td>17.05</td>
<td>11.57</td>
<td>2.98</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>98.72</td>
<td>35.82</td>
<td>27.25</td>
<td>17.23</td>
</tr>
<tr>
<td>c57-cc-vessel</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>83.75</td>
<td>0.65</td>
<td>0.44</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>97.49</td>
<td>7.21</td>
<td>3.35</td>
<td>1.06</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>97.99</td>
<td>18.90</td>
<td>14.58</td>
<td>9.04</td>
</tr>
<tr>
<td>belgium-road</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>86.73</td>
<td>1.25</td>
<td>0.68</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>96.98</td>
<td>0.55</td>
<td>0.55</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>99.29</td>
<td>47.44</td>
<td>38.60</td>
<td>22.11</td>
</tr>
<tr>
<td>italy-road</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>90.07</td>
<td>0.32</td>
<td>0.16</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>90.24</td>
<td>0.26</td>
<td>0.17</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>99.41</td>
<td>28.49</td>
<td>20.08</td>
<td>11.99</td>
</tr>
<tr>
<td>netherlands-road</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>84.19</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>96.06</td>
<td>3.91</td>
<td>2.20</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>99.44</td>
<td>37.55</td>
<td>26.97</td>
<td>10.77</td>
</tr>
<tr>
<td>luxembourg-road</td>
<td>SEAL [51]</td>
<td>43,010</td>
<td>89.79</td>
<td>11.39</td>
<td>6.15</td>
<td>3.12</td>
</tr>
<tr>
<td></td>
<td>SEAL+EdgeConv (ours)</td>
<td>49,346</td>
<td>97.53</td>
<td>59.79</td>
<td>39.15</td>
<td>19.42</td>
</tr>
<tr>
<td></td>
<td>GAV (ours)</td>
<td>8,194</td>
<td>99.31</td>
<td>85.88</td>
<td>76.84</td>
<td>61.95</td>
</tr>
</tbody>
</table>

The evaluation metric Hits@k typically represents the number of links, where k determines the number of nodes resulting from the neighbors of a central node. Hence, Hits@k is used to measure the model's ability to predict the correct neighbors of a node, which is of particular interest for link prediction tasks.

Road Networks Further, we report results on diverse road networks representative of four European countries for a thorough evaluation of our proposed GAV framework’s generalizability. To this end, we adopt publicly available road networks introduced in the DIMACS graph partitioning and clustering challenge [3]. These road networks correspond to the largest connected components of OpenStreetMap’s [31] road networks and are vastly different in size (e.g., luxembourg-road constitutes roughly 100,000 edges, whereas italy-road has more than 7,000,000). In road networks, intersections and locations with stronger curvature represent nodes in the form of x- and y-coordinates, while connecting roads represent edges.

Preprocessing Link prediction datasets require positive (label 1) and negative links (label 0). Positive links correspond to existent edges in our datasets, whereas negative links represent artificially created, non-existent edges. As link prediction algorithms are commonly employed to improve the graph representation through the identification of...
absent connections and the reduction of local noise arising from graph generation, negative links should appear as authentic as possible. In light of the absence of negative links in our sourced datasets, we prepare our sourced datasets in a manner that aligns with the ogbl-vessel benchmark. Therefore, we sample negative links using a spatial sampling strategy. To be precise, we randomly connect nodes in close proximity, taking a maximum distance threshold of \( \delta = \bar{e}_{ij} + 2\sigma \) into account. Here, \( \bar{e}_{ij} \) denotes the average edge length estimated over the entire graph \( \mathcal{G} \) and \( \sigma \) the standard deviation. The number of negative, sampled links corresponds to the number of positive, real links across all datasets. We finally split positive and negative links into training, validation, and test sets (split 80%/10%/10%).

### 4.3. Quantitative Results

GAV demonstrates excellent, superior performances on the task of link prediction across all metrics and datasets, as can be observed in Table 2. We outperform the current state-of-the-art algorithm SIEG on the ogbl-vessel benchmark by 12\% (98.38 vs. 87.98 AUC) while requiring a significantly smaller amount of parameters (8,194 vs. 752,716). However, GAV not only drastically outperforms all algorithms submitted to the ogbl-vessel benchmark but also our introduced strong, secondary baseline, combining the SEAL framework with EdgeConv (SEAL+EdgeConv). The excellent performance and superiority of GAV is even more pronounced when considering the strict evaluation metric of Hits@\(k\). Quantitative results reported in Table 2 additionally indicate the strong performance of our introduced secondary baseline (see Section 4.1).

It is of note that the luxembourg-road dataset’s test set contains only 12,000 negative links. We, therefore, compare predictions of its positive links to 12,000 rather than 100,000 negative links (see supplementary, Supp. G). This explains the comparatively strong Hits@\(k\) performances on the luxembourg-road dataset.

### 4.4. Ablation Studies

To further validate GAV, we conduct detailed ablation studies on the validation set of the ogbl-vessel benchmark. Additional ablation studies on the GAV layer and its design choices can be found in the supplementary (Supp. F).

#### Table 3. Ablations on main design choices.

<table>
<thead>
<tr>
<th>Readout Module</th>
<th>Message-Passing</th>
<th>Labeling Trick</th>
<th>AUC ↑</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>98.39</td>
<td>–</td>
</tr>
<tr>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>98.28</td>
<td>-0.11</td>
</tr>
<tr>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>80.56</td>
<td>-17.83</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>96.00</td>
<td>-2.39</td>
</tr>
</tbody>
</table>

we exchange our readout module with a SortPooling layer followed by two convolutional layers and an MLP, resembling SEAL’s readout operation. We note that our readout module is more applicable to flow-driven spatial networks, as it leads to a modest AUC increase of 0.11. Second, we completely deactivate the message-passing module by forwarding \( \mathcal{L}(\mathcal{G}^t_{ij}) \) directly to the readout module. We observe a drastic AUC decrease of 17.83, indicating the importance of modifying the vector embeddings via our proposed GAV layer. Finally, we evaluate the impact of the labeling trick. Excluding the additional labels generated by the labeling trick results in an AUC decrease of 2.39, proving the significance of link identification via additional, distinct labels.

In a second ablation study, we experiment with different message-passing layers (see Supp. L), including EdgeConv, in our message-passing module. We report our findings in Table 4. Our proposed GAV layer outperforms the other message-passing layers across all metrics by a considerable amount, proving our rationale.

#### Table 4. Ablations with different message-passing layers.

<table>
<thead>
<tr>
<th>Message-Passing Layer</th>
<th>AUC ↑</th>
<th>Hits@100 ↑</th>
<th>Hits@50 ↑</th>
<th>Hits@20 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAV layer (ours)</td>
<td>98.39</td>
<td>34.46</td>
<td>26.30</td>
<td>19.81</td>
</tr>
<tr>
<td>EdgeConv [41]</td>
<td>97.43</td>
<td>17.30</td>
<td>5.97</td>
<td>0.78</td>
</tr>
<tr>
<td>GAT layer [6]</td>
<td>96.44</td>
<td>4.58</td>
<td>2.55</td>
<td>1.59</td>
</tr>
<tr>
<td>SAGE layer [15]</td>
<td>93.53</td>
<td>0.77</td>
<td>0.11</td>
<td>0.03</td>
</tr>
<tr>
<td>GCN layer [24]</td>
<td>89.31</td>
<td>0.39</td>
<td>0.22</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Lastly, we vary the number of hops \( h \) used to generate \( \mathcal{G}^t_{ij} \) and the number of message-passing iterations \( k \) (see Table 5). We observe that simultaneously increasing \( k \) and \( h \) results in no discernible differences in performance. This finding is in line with the \( \gamma \)-decaying theory [49], proving the approximability of high-order heuristics from locally restricted subgraphs.

#### Table 5. Ablations on \( k \) and \( h \).

<table>
<thead>
<tr>
<th>( k ) &amp; ( h )</th>
<th>AUC ↑</th>
<th>Hits@100 ↑</th>
<th>Hits@50 ↑</th>
<th>Hits@20 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.39</td>
<td>34.46</td>
<td>26.30</td>
<td>19.81</td>
</tr>
<tr>
<td>2</td>
<td>98.39</td>
<td>34.00</td>
<td>26.93</td>
<td>21.21</td>
</tr>
<tr>
<td>3</td>
<td>98.39</td>
<td>34.25</td>
<td>25.99</td>
<td>17.84</td>
</tr>
</tbody>
</table>

### 4.5. Interpretability and Analysis of Results

In Fig. 5, we visualize the behavior of our proposed GAV layer, attempting to facilitate interpretability and provide a clearer insight on how it creates superior representations for link prediction in flow-driven spatial networks. An analysis based on a toy example and additional visualizations similar to Fig. 5 are provided in the supplementary (Supp. D & B). Scalar multiplication via \( s_i \) enables GAV to flip individual vector embeddings (mimicking change in direction of flow). We observe that this results in physically implausible, learned sink/source flow between the two target nodes (red and green) of implausible formations (see Fig. 5, second
Figure 5. Visualization of the effect of our GAV layer on vector embeddings (ogbl-vessel). We visualize subgraph representations $G_t$ (set to one) of four positive, plausible (first row) and four negative, implausible target links (second row) together with the GAV layer’s predicted scalar values $s_i \in (-1, 1)$. The scalar values $s_i$ used to update vector embeddings in $L(G_t)$ have been projected to $G_t$ to provide an interpretable visualization. The directionality of edges already incorporates potential shifts in direction enforced by our GAV layer. Please note that following the color coding scheme of Fig. 2, the target link $e_{ij}$ is depicted in orange, whereas the two target nodes $n_i$ and $n_j$ are displayed in red and green. We additionally report the angle $\angle$ between vector embeddings aggregated around the two target nodes represented by mean $\langle E_N(n_i) \rangle$ and mean $\langle E_N(n_j) \rangle$, constructed in our readout module (see Section 3.4) and the predicted probability of link existence $\hat{y}_{ij}$. The last column contains more challenging cases.

In this work, we propose the simple yet effective Graph Attentive Vectors (GAV) link prediction framework. GAV relies on the idea of modeling simplified physical flow by updating vector embeddings in a constrained manner, which intuitively models the underlying physical process and, therefore, presents a strong inductive bias for link prediction in flow-driven spatial networks. This allows GAV to outperform the previous state-of-the-art by an impressive margin on all metrics across multiple whole-brain vessel and road network datasets while requiring a significantly smaller amount of trainable parameters. GAV’s imitation of the dynamics of physical flow, however, represents a simplified concept, which is not entirely representative of physical principles from, e.g., fluid dynamics (see Fig. 5). Future work should, therefore, aim to further investigate GAV’s parameters and extend its assumptions by incorporating different physical principles, such as conservation of mass and momentum, resulting in vector embeddings highly representative of physical flow in flow-driven spatial networks.
References


[3] David A Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 10th dimacs implementation challenge-graph partitioning and graph clustering, 2011. 5, 6


[31] OpenStreetMap contributors. Planet dump retrieved from planet.osm.org. openstreetmap.org, 2017. 6


[34] Franca Schmid, Giulia Conti, Patrick Jenny, and Bruno Weber. The severity of microstrokes depends on local vascular topology and baseline perfusion. Elife, 10:e60208, 2021. 1


