Supplementary material

A Additional graph visualisations

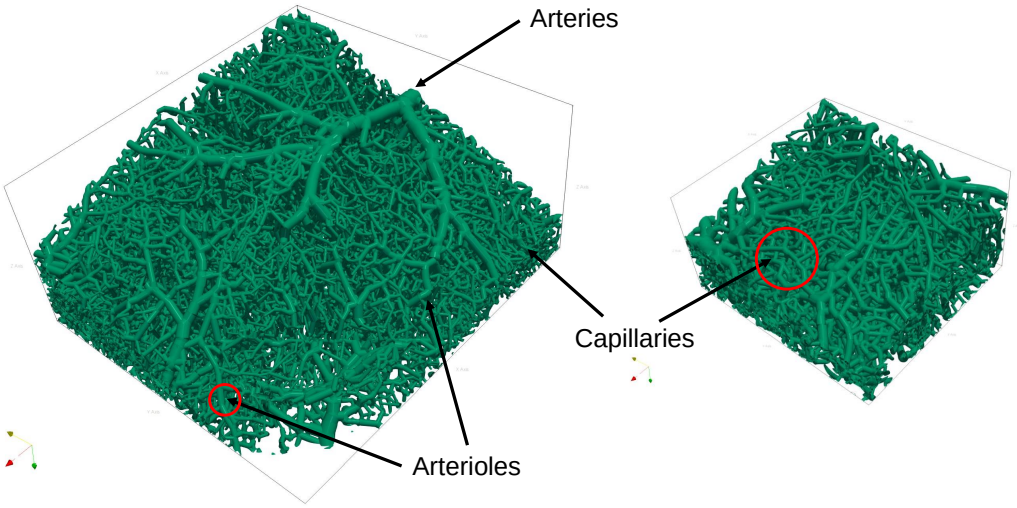

Figure 5: Graphical visualisations of the vessel graph and three different vessel types distinguishable by diameter.

B Code and dataset documentation

B.1 Hosting, licensing and author statement

All of our codes and baselines are available in a public github repository `https://github.com/jocpae/VesselGraph` licensed under an MIT License. All of our data is also freely available and can be downloaded from a university server following the links in `https://github.com/jocpae/VesselGraph#datasets`. The dataset is provided in CSV format. The dataset has the following DOI: `10.5281/zenodo.5301621`. All of our released data is licensed under an *Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license*. The authors confirm the CC licenses for the included datasets and declare to bear all responsibility in case of violation of rights. The authors declare no competing financial interests

B.2 Long term maintenance plan

The dataset and code has been permanently archived at Zenodo, guaranteeing long-term availability. We will update the dataset when novel segmentations of the whole brain vasculature become publicly available. Contributions will be solicited via GitHub pull request. Regarding maintenance, we will update the code repository for loading and processing the data; the links to the university server where the data is stored will also be kept up-to-date.

Additionally, we plan to incorporate our dataset into the Open Graph Benchmark[2]. Our dataset and dataloader are already compatible with the OGB data loader and platform. We believe that an integration into the OGB framework will further facilitate and simplify the usage of our data. We also provide an alternative dataloader for use with Pytorch Geometric.

B.3 Whole brain vessel imaging and segmentation

As discussed in the introduction, whole brain vascular imaging is an emerging field spanning different imaging techniques, among the first technologies were tissue clearing-based methods [9–12] which enable the fluorescent staining and clearing (that is, a chemical process which renders the organ transparent) of intact, whole

[2] `https://ogb.stanford.edu/`

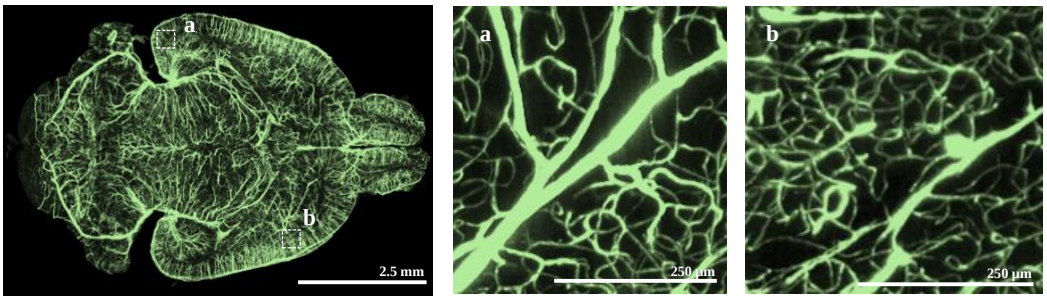

Figure 6: Local differences in vessel structure can be observed in different brain regions in a dataset of Todorov and Paetzold et al. [6].

brains with a subsequent imaging of the vessels with a 3D lightsheet microscope. Among the first imaging protocols were VesSAP [6], Tubemap [7] and the work by diGiovanna et al. [13]. Alternative imaging techniques are microCT[48, 49], magnetic resonance imaging [50] and optical coherence tomography [51], which however fail to achieve the spatial resolution to reliably image microcapillaries. Recently, a method based on synchrotron-based phase-contrast tomographic microscopy was developed, achieving an isotropic voxel size of 0.65 micrometers [52]. Other technologies, such as serial serial two-photon microscopic imaging are also developing rapidly with similar or even better resolution compared to the tissue clearing methods (e.g. $0.303 \times 0.303 \times 1.0$ resolution [1]), promising a widespread use/adoption of whole brain vascular imaging approaches in the future.

B.4 Individual datasets, licenses and animal experiments

The nine base datasets from the VesSAP paper [6] are available here: `http://discotechnologies.org/VesSAP/`. They are licensed under a Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). The animal experiments were carried out under approval of the institutional ethics review board of the Government of Upper Bavaria (Regierung von Oberbayern, Munich, Germany), and in accordance with European directive 2010/63/EU for animal research, details can be read here `https://doi.org/10.1038/s41592-020-0792-1`.

The base datasets from Ji et al. [1] are licensed under an open source (BSD 3-Clause) license. They are available upon email request from the authors, their code is available at `https://neurophysics.ucsd.edu/software.php`. The animal experiments followed the Guide for the Care and Use of Laboratory Animals and have been approved by the Institutional Animal Care and Use Committee, details can be found in the original paper: `https://doi.org/10.1016/j.neuron.2021.02.006`

The synthetic data was generated by the authors themselves following the approach by Schneider et al. [32], the same license applies as for the graph datasets presented here. The synthetic base data can be downloaded here `https://github.com/giesekow/deepvesselnet/wiki/Datasets`.

C Graph documentation

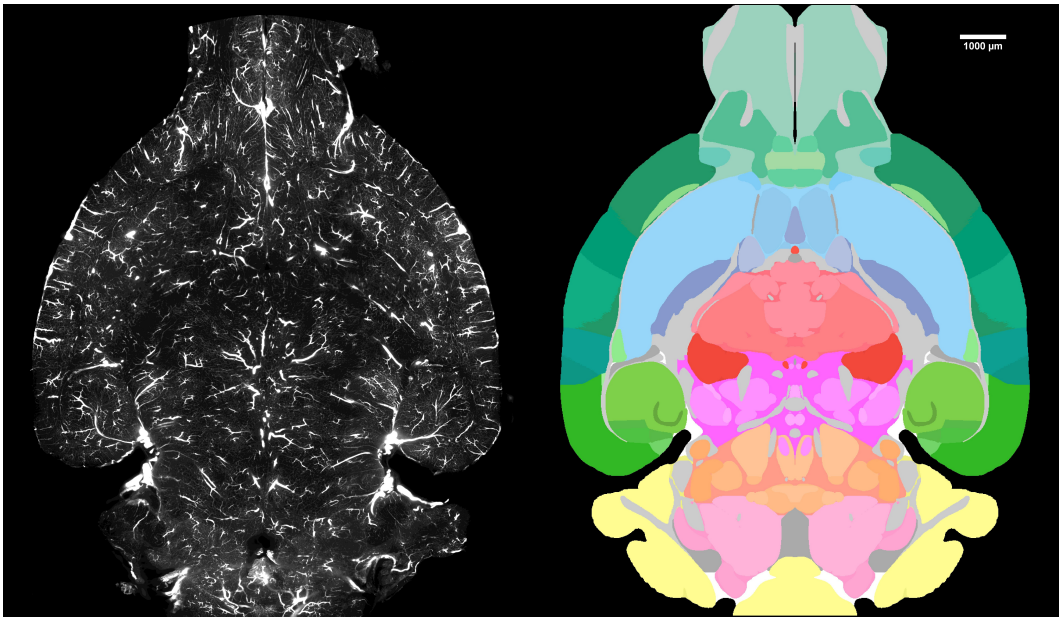

Figure 7: Rendering of the Allen brain atlas feature; on the left side a 2D image slice from the CD1-E-1 brain is depicted, on the right side a rendering of the Allen brain atlas regions corresponding to the coordinates of the image. A node n is assigned to a particular atlas region depending on the x_n, y_n, z_n coordinate of the node n .

C.1 Voreen parameters

In the following Sections, we discuss the details of the Voreen graph extraction pipeline which follows a four stage protocol introduced in Section 2. To recapitulate, the four stages are:

1. Skeletonization: The binary segmentation volume is reduced to a skeleton based representation by applying a standard topological thinning algorithm by Lee et al. [38].
2. Topology Extraction: memory efficient algorithms extract the vessel centerlines [39]. *Voreen* allows to store this intermediate representation in a combination with the graph.
3. Voxel-Branch Assignment: Computing of mapping between the so-called protograph (i.e. the initial graph) and the voxels of the binary segmentation.
4. Feature Extraction: On basis of the protograph and the mapping, several features can be computed from the foreground segmentation.

We chose the *Voreen* parameters in the following manner:

1. The *binarization threshold* is selected from the interval $[0, 1]$. This value is irrelevant for binary segmentation masks, e.g. VesSAP.
2. *Surface smoothing* is deactivated.
3. The relative *minimal bounding box diagonal* is set to 0.05.
4. The total *minimal bounding box diagonal* is set to 0 mm.
5. The *bulge size* is set to 3.0, see Figure 9.

Bulge Size As depicted in the figures below, the total number of nodes decreases when increasing the bulge size parameter. This is expected, as the bulge size describes the relation between parent vessel and branch, and the relation between vessel *bumpiness* and parent vessel [31]. The bulge size can be configured between $[0 < x < 10]$.

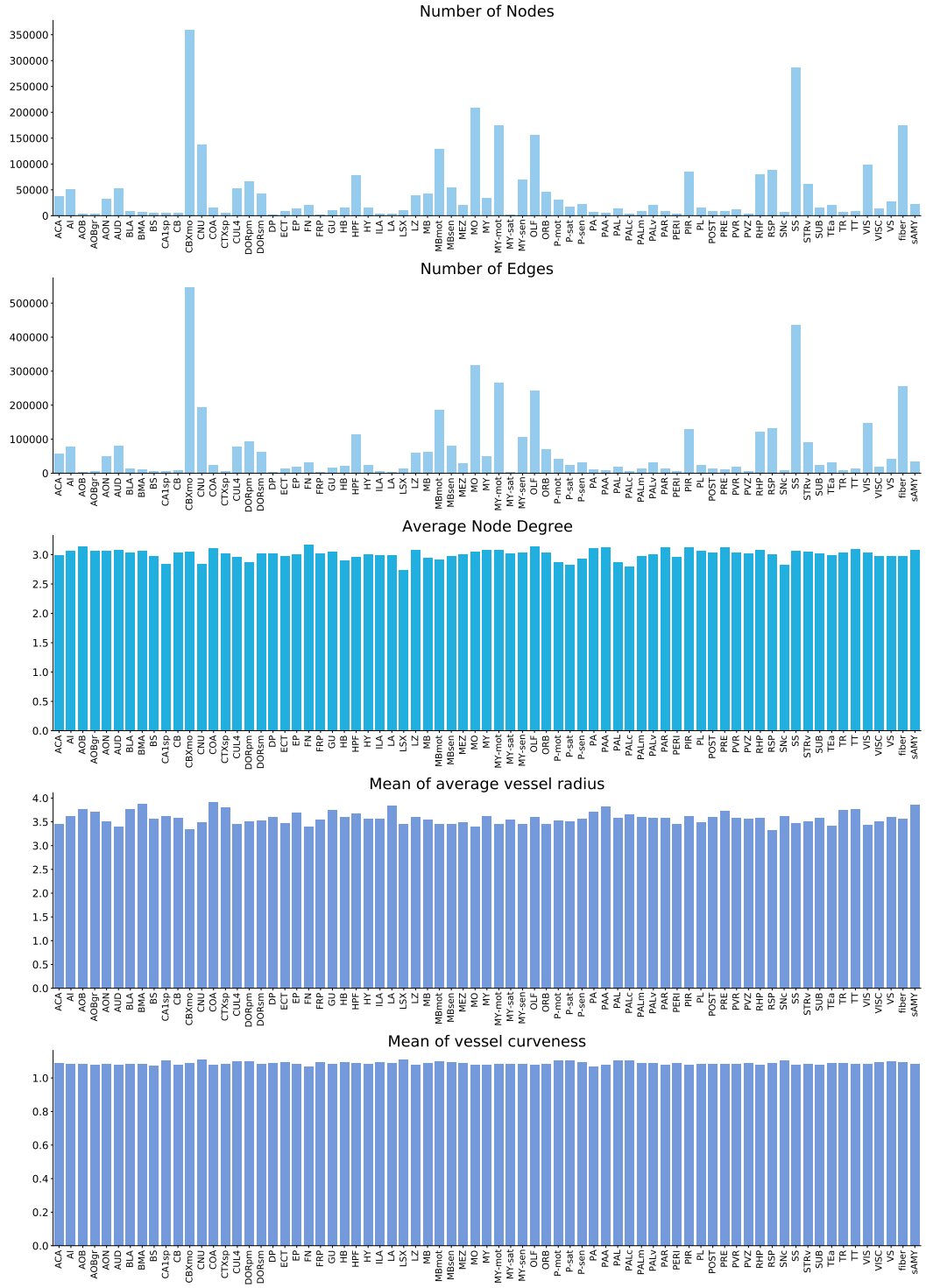

Figure 8: Different Node features of G plotted by the regions of the Allen brain atlas.

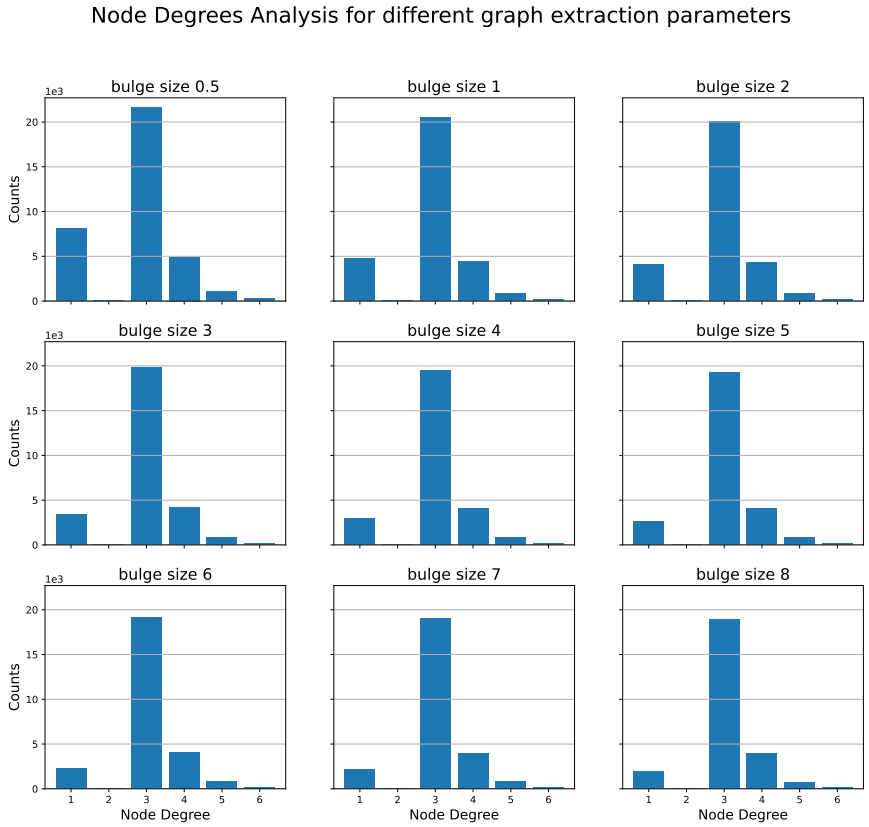

Figure 9: Bulge Size analysis in the *Voreen* pipeline for a parameter range of 0.5- 8.0.

For bulge _size $= 3.0$, we obtain a good relationship between capturing the essential vessels, while capturing the smoothness of the vascular trees. Following the recommendation of _size $= 3.0$ [31] for healthy vasculature, we keep _size $= 3.0$ as a maximum, to provide a reasonable _size $= 3.0$ value for future data with diseased animals.

	length	distance	curveness	avgCrossSection	minRadiusAvg	minRadiusStd	avgRadiusAvg	avgRadiusStd	maxRadiusAvg	maxRadiusStd	roundnessAvg	roundnessStd	node1_degree	node2_degree	num_voxels	volume
length	1.00	0.96	0.50	0.15	-0.10	0.31	-0.07	0.37	-0.03	0.37	-0.10	0.17	-0.07	0.03	0.98	0.58
distance	0.96	1.00	0.28	0.18	-0.08	0.31	-0.05	0.36	-0.00	0.36	-0.10	0.17	-0.06	0.04	0.94	0.59
curveness	0.50	0.28	1.00	0.01	-0.07	0.18	-0.07	0.24	-0.06	0.23	-0.08	0.12	-0.09	-0.01	0.52	0.22
avgCrossSection	0.15	0.18	0.01	1.00	0.71	0.35	0.82	0.35	0.81	0.32	0.15	0.09	0.08	-0.02	0.16	0.73
minRadiusAvg	-0.10	-0.08	-0.07	0.71	1.00	0.11	0.93	0.02	0.74	-0.04	0.52	-0.18	0.12	0.01	-0.08	0.36
minRadiusStd	0.31	0.31	0.18	0.35	0.11	1.00	0.20	0.77	0.25	0.50	-0.13	0.50	0.05	0.09	0.30	0.41
avgRadiusAvg	-0.07	-0.05	-0.07	0.82	0.93	0.20	1.00	0.18	0.93	0.16	0.23	-0.17	0.14	0.01	-0.06	0.45
avgRadiusStd	0.37	0.36	0.24	0.35	0.02	0.77	0.18	1.00	0.31	0.87	-0.28	0.45	-0.02	0.05	0.36	0.42
maxRadiusAvg	-0.03	-0.00	-0.06	0.81	0.74	0.25	0.93	0.31	1.00	0.35	-0.07	-0.12	0.14	0.01	-0.03	0.48
maxRadiusStd	0.37	0.36	0.23	0.32	-0.04	0.50	0.16	0.87	0.35	1.00	-0.37	0.39	-0.04	0.03	0.36	0.39
roundnessAvg	-0.10	-0.10	-0.08	0.15	0.52	-0.13	0.23	-0.28	-0.07	-0.37	1.00	0.07	0.05	0.03	-0.08	0.01
roundnessStd	0.17	0.17	0.12	0.09	-0.18	0.50	-0.17	0.45	-0.12	0.39	0.07	1.00	0.01	0.09	0.17	0.13
node1_degree	-0.07	-0.06	-0.09	0.08	0.12	0.05	0.14	-0.02	0.14	-0.04	0.05	0.01	1.00	0.04	-0.11	0.03
node2_degree	0.03	0.04	-0.01	-0.02	0.01	0.09	0.01	0.05	0.01	0.03	0.03	0.09	0.04	1.00	-0.01	0.01
num_voxels	0.98	0.94	0.52	0.16	-0.08	0.30	-0.06	0.36	-0.03	0.36	-0.08	0.17	-0.11	-0.01	1.00	0.57
volume	0.58	0.59	0.22	0.73	0.36	0.41	0.45	0.42	0.48	0.39	0.01	0.13	0.03	0.01	0.57	1.00

Figure 10: Edge feature correlation plot.

Runtime of graph extraction: The *Voreen* pipeline used for graph extraction was run on a CPU cluster. We timed one exemplary graph extraction on this cluster using 8 logical CPUs. For one of our synthetic vessel datasets ($500 \times 500 \times 401$ pixels), the process described in Section 2 with 3 refinement iterations, resulting in 28538 nodes and 42727 edges required a total runtime of 3 minutes and 57 seconds. Importantly, the extraction times scale roughly linearly with the number of nodes which are extracted [31].

C.2 Post processing of graphs

In order to remove evident artifacts from the generated graphs we implemented rule based post processing or pruning steps.

Feature merging: In a small percentage of the extracted graphs ($< 1\%$), we obtain multiple edges e_{ij}^b for $b = 1 : B$ with the same vertices. This can be attributed to imaging artifacts and irregularities in the vessel staining process. For instance, holes in the segmentation mask, if not properly filled, can result in multiple edges spanning from the same source node n_i to the same target node n_j, respectively n_j to n_i, as the underlying graph is undirected. As our focus lies on biologically realistic graphs, in particular correct branching structures, we merge the edge features of two identically labeled edges by obtain approximations. As the greater percentage of irregularities have been already mitigated in various preprocessing steps, we are confident that approximations in a small number of edges should not affect the performance and generalization of the deep learning models. When we merge features we update them according to:

1. Length: $l_{ij} = \frac{1}{B} \sum_{b=1}^{B} l_{ij}^b$
2. Shortest Distance: $d_{ij} = \min_k \{d_{ij}^b\}$
3. Volume: $v_{ij} = \sum_{b=1}^{B} v_{ij}^b$
4. Number of Voxels: $nv_{ij} = \sum_{b=1}^{B} nv_{ij}^b$
5. Curvature: $\rho_{ij} = \frac{1}{B} \sum_{b=1}^{B} \rho_{ij}^b$
6. Mean Cross Section Area: $\alpha_{ij} = \sum_{b=1}^{B} \alpha_{ij}^b$

As radius features are highly shape-dependent, and highly variable in the vessel itself, we can only approximate the vessel attributes, as we cannot access the meta-information Voreen utilizes to calculate the minimum, average and maximum vessel radius and the corresponding standard deviations.

Consequently, we obtain the average radius by summing the average radius of all identically labeled edges. We calculate the standard deviation by obtaining the median of the mathematical relationship of standard deviation of average radius to average radius.

1. $\mu_{ij}^{r} = \mu_{ij}^{\bar{r}} = \mu_{ij}^{R} = \sum_{b=1}^{B} \mu_{ij}^{\bar{r}b}$
2. $\sigma_{ij}^{r} = \sigma_{ij}^{\bar{r}} = \sigma_{ij}^{R} = \sqrt{\sum_{b=1}^{B} (\sigma_{ij}^{\bar{r}b})^2}$

We fit a linear function $\mu^o = f(\mu^{\overline{r}}) = a \cdot \mu^{\overline{r}} + b$ to map the average radius $\mu^{\overline{r}}$ to the vessel roundness μ^o. Similarly to the radius approximations, we obtain the newly computed roundness standard deviation by obtaining the median of the mathematical relationship of standard deviation to roundness.

1. $\mu_{ij}^{o} = f(\mu_{ij}^{\overline{r}})$
2. $\sigma_{ij}^{o} = median \left[\left\{ \frac{\sigma_{ij}^{ob}}{\mu_{ij}^{ob}} \right\}_{b=1:B} \right] \cdot \mu_{ij}^{o}$

The following properties are computationally updated.

- ν_{ij}^1 is updated.
- ν_{ij}^2 is updated.

D Baseline experimentation and discussion

D.1 Link prediction

We implemented a set of baselines from the literature. All of these implementations, including documentation are available in our GitHub repository . The model architecture, learning rate, number of GCN layers, dropout and batch size are the hyperparameters we optimized for our training and document below. We selected our models according to the highest ROC AUC score on the validation set.

In order to select the values, we employed Grid Search, please see the tables below. Owing to the huge size of our dataset, hyperparameter tuning is challenging. To overcome this challenge, we subsample a region of the mouse brain in order to create a small graph. To ensure we are not introducing any bias, we measured the KL-Divergence [53] to ensure that our small graph is representative of the whole brain in its distribution of

Table 5: Details of the hyper-parameters search for Link prediction with the selected parameters for the final training of each of our baseline models.

Model	Parameter Range	Selected Parameters	Model Select.
Matrix Factorization	lr $\in \{1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{32, 64, 128, 256\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{-5}$ num of layers = 3 hidden channels = 64 dropout = 0.2	epochs 3000
MLP	lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{128, 256, 512\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{-5}$ num of layers = 4 hidden channels = 256 dropout = 0.2	epochs 3000
GCN [14]	lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{128, 256, 512\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{5}$ num of layers = 2 hidden channels = 256 dropout = 0.2	epochs 3000
GNN + N2Vec Embeddings [43]	lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{128, 256, 512\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{-5}$ num of layers = 2 hidden channels = 256 dropout = 0.2	epochs 3000
GNN + SAGE [17]	lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{128, 256, 512\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{-4}$ num of layers = 3 hidden channels = 16 dropout = 0.5	epochs 3000
GNN + SAGE [17] + N2Vec Embeddings [43]	lr $\in \{1 \cdot 10^{-02}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{128, 256, 512\}$ dropout $\in \{0, 0.2, 0.5\}$	lr = $1 \cdot 10^{-3}$ num of layers = 2 hidden channels = 16 dropout = 0.5	epochs 3000
SEAL [25]	lr $\in \{1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$ num of layers $\in \{2, 3, 4\}$ hidden channels $\in \{32, 64, 128\}$ dropout $\in \{0, 0.2, 0.5\}$ num of hops$\in \{1, 2, 3\}$ labeling $\in$ {drnl,de,de+,zo} model = DGCNN, GCN	lr = $1 \cdot 10^{-4}\}$ num of layers = 2 hidden channels = 32 dropout = 0.0 num of hops = 1 labeling = drnl model = DGCNN	epochs 100
N2Vec [43]	lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}, 1 \cdot 10^{-6}\}$ walk length $\in \{5, 10, 20\}$ walks per_node $\in \{5, 10, 20\}$ embedding_dim $\in \{16, 32, 64, 128, 256\}$	lr = $1 \cdot 10^{-6}$ walk length = 5 walks_per_node = 10 embedding_dim = 64	epochs 2

vasculature. We selected the best set of hyperparameters on the small graph and used it on the actual graph with small modifications if needed. We summarize our results in Table 5 and Table 6.

For Matrix Factorization, we tune the number of layers, hidden channels and dropout rate. To estimate the best hyperparameters, we employed grid search. We observe that the Matrix Factorization method is not sensitive to the choice of hyper-parameters as long as they are in a reasonable range (learning rate 1e-3 to 1e-5, dropout 0 to 0.5, hidden channels 32 to 256). We obtained the best results for a learning rate of 1e-4 with 3 layers, 64 hidden channels and a dropout rate of 0.2. When extending to the entire BALBc1 whole mouse brain dataset (5.35 million edges, 3.54 million nodes), we reduced the number of hidden channels to 64 owing to GPU memory constraints. We experimentally found out that the model with 64 channels was easily able to overfit to the training data, showing that the model is sufficiently complex.

The MLP model is trained with the Adam optimizer and a learning rate of 1e-4. This combination of hyperparameters provides us the best performance on the validation and test splits. We experimented with fully connected layers of 128 and 256 features, the latter resulting in better scores. Thus, our final model has 4 layers each with 256 channels.

For the Graph Neural Networks, we use two setups. A GCN setup without embeddings (which we refer to as GCN in the experiments) and a GCN with embeddings (which we refer to as GCN + Node2Vec Embeddings in the experiments, Tab. 5). Both models are trained with Adam and 1e-5 and 1e-7 learning rate, respectively.

We discover that GCN+node embeddings do not perform superior to the GCN which is trained only on the node features. Moreover, in our experiments, we find that the predictions made by the models result in scores close to chance (approx. 51 for both). The model is overfitting on the training data almost immediately and is not able to generalize to the unseen data. Reducing the model complexity by reducing the number of layers, hidden channels and increasing the dropout did not improve performance.

We observe poor generalization and immediate overfitting in all cases. We hypothesize that this behaviour is caused by the model not being able to distinguish symmetrically placed nodes while making the prediction (something that the SEAL algorithm achieves using the Labeling trick). Our experiments done using SEAL with a similar set of hyper-parameters substantiate this hypothesis. Please refer to the section on SEAL for more details below.

In our experiments, we find that the GraphSAGE models tend to outperform the GCN baseline. The *ROC AUC* increases to approximately 60 meaning a performance boost of almost 7% We initialized the nodes with the Node2Vec embeddings[43] (we refer to this model as GraphSAGE + Node2Vec Embeddings) and findthat the

inclusion of pretrained Node2Vec embeddings as features leads to deterioration in model performance. We observed the best model performance with a shallow network (2 hidden layers), a learning rate of $1e-4$ and 0.5 dropout. The hyperparameters were chosen according to the GCN baseline.

When we ran our model on the entire mouse brain graph, we observed behaviour similar to GCN and GCN+embedding. Although the GraphSAGE performance is slightly superior to the GCN baseline, it is still dwindling in the region of being random (approx 59%). The GraphSAGE algorithm tends to improve the update of node features by concatenating the features of the node and its aggregated neighbourhood features. As such, this may enable the network to assign different weights to the self nodes and its neighbours. However, this does not allow it to break the symmetry while performing the link prediction task. Indicating that these models suffer from the same issue in regards to node labeling as was described in the GCN discussion above. Further, inclusion of Node2Vec features decreases the performance resulting in an 53% ROC AUC value.

Since Node2Vec is an unsupervised process we use the SGD optimizer with negative sampling, to maximize the log likelihood objective on the random walks in the graph. We perform a hyperparameter search on the learning rate, walk lengths, walks per node and embedding dimension.

We find the best combination to be a walk length of 5, 10 walks per node and an embedding dimension of 64. In our experiments with the small sub-graph, we observed that the loss quickly plateaus after 3 epochs. We use the same hyperparameter combination for the entire mouse brain graph. We observe that the inclusion of Node2Vec features does not improve performance and in most cases, leads to worse results. Empirically we observe that the inclusion of Node2Vec features is not useful for the link prediction task and as discussed above, the Labeling trick constitutes a much more important component.

SEAL: For the SEAL algorithm we implement a similar hyperparameter search. We discover a general pattern in the selection of different labeling tricks, and find the DE and DRNL to perform best on the training set.

Moreover, we find SEAL to perform very well at different rates. However, we do get a general trend of optimal performance for a learning rate of $1e-4$ and 2 hops. For the whole brain graph, we find Double-Radius Node Labeling (DRNL) to better capture the hierarchical structure [25]. Further, a DGCNN outperforms a GCN. When training on a small region of interest we observe a gradual improvement for up to 50 epochs, for the whole brain graph we discover that SEAL almost converges the first epochs while inducing significant inductive bias, making it the superior model for our spatial and hierarchical graph. The training quickly plateaus. In all our experiments, the SEAL method performs best amongst all the baselines. All performance measures are given in Table 3 in the main paper.

D.2 Node classification

Similar to the link prediction task, we employ a grid Search for hyperparameters. For each graph neural network, we explore the number of layers, hidden channels, learning rate and dropout ratio on the smaller sets validation and test set. For each model, if applicable, we select a set of hyper-parameters specific to the architecture and thus optimize. We selected our models according to the highest ROC AUC score on the validation set.

The Cluster GCN algorithm provides a memory efficient alternative to other graph learning algorithms and is specifically designed to handle large-scale graphs efficiently. The number of partitions constitutes the essential hyper-parameter. We find it to consistently outperform other models, irrespective of its number of partitions.

Further we implement the base GCN by Kipf et.al. and extend it to the GNN GCN and GNN SAGE (SAGE introduces convolutional operator). For all we vary the number of hidden channels, number of layers and dropout ratio. In direct comparison, the GNN SAGE outperforms the GNN GCN by high margins (>10%) for the ROC-AUC metric on the whole brain graph. Similar to the link prediction task, indicating that the base GCN is not suited for this task.

For the Node2Vec embeddings of the node classification task, we find similar trends for walk length, walks per node and embedding dimension as in the link prediction task. Please refer to Table 6 for the exact hyper-parameter selection.

For the MLP we select a comparative number of layers and obtain the best result for a high dropout ratio of 0.4. We observe that the the model is unable to generalize to unseen data, indicating the absence of an appropriate inductive bias.

For the MLP-CS, we selected similar values in comparison to the MLP implementation. Incorporating the Correct&Smooth methodology into the model increases the F1 score but does not improve the metrics which account for imbalanced datasets.

The SIGN applies subsampling techniques. Similar to the graph neural network baselines, we explore the hidden channels and number of layers. We obtain the best for a high model complexity with 4 layers and 512 hidden channels. The model converges very fast with a high learning rate but performs only on par with the base GCN.

Table 6: Details of the hyper-parameter search for node classification with the final parameters selected for the our baseline models.

Model	Parameter Range	Selected Parameters	Model Select.
Cluster-GCN	lr $\in \{2 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}\}$ number of layers $\in \{3, 4, 5\}$ hidden channels $\in \{128, 256, 512\}$ number of partitions $\in \{3, 6, 9\}$ dropout $\in \{0.0, 0.2, 0.5\}$	lr = $3 \cdot 10^{-3}$ number of layers = 4 hidden channels = 128 number of partitions = 9 dropout = 0.2	1500 epochs
GNN	lr $\in \{2 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}\}$ number of layers $\in \{3, 4, 5\}$ hidden channels $\in \{32, 128, 256, 512, 1024\}$ dropout $\in \{0.1, 0.4, 0.5\}$	lr = $3 \cdot 10^{-3}$ number of layers = 3 hidden channels = 256 dropout = 0.4	1500 epochs
GNN-SAGE	lr $\in \{2 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}\}$ number of layers $\in \{2, 3, 4\}$ hidden channels $\in \{32, 128, 256, 512, 1024\}$ dropout $\in \{0.0, 0.2, 0.4\}$	lr = $3 \cdot 10^{-3}$ number of layers = 4 hidden channels = 128 dropout = 0.4	1500 epochs
Graph-Saint	lr $\in \{1 \cdot 10^{-6}, 1 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 5 \cdot 10^{-3}, 1 \cdot 10^{-2}\}$ number of layers $\in \{2, 3, 4\}$ hidden channels $\in \{64, 256, 512, 1024\}$ walk length $\in \{3, 5, 7\}$ dropout $\in \{0.0, 0.35, 0.5\}$	lr = $5 \cdot 10^{-4}$ number of layers = 4 hidden channels = 64 walk length = 7 dropout = 0.35	1000 epochs
SIGN	lr $\in \{1 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 5 \cdot 10^{-3}\}$ number of layers $\in \{2, 3, 4\}$ hidden channels $\in \{32, 64, 128, 256, 512, 1024\}$ dropout $\in \{0.0, 0.1, 0.3, 0.5\}$	lr = $1 \cdot 10^{-3}$ number of layers = 3 hidden channels = 128 dropout = 0.1	1000 epochs
MLP	lr $\in \{2 \cdot 10^{-5}, 1 \cdot 10^{-4}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-2}\}$ number of layers $\in \{2, 3, 4\}$ hidden channels $\in \{32, 128, 256, 512\}$ dropout $\in \{0.0, 0.3, 0.4\}$	lr = $1 \cdot 10^{-3}$ number of layers = 3 hidden channels = 256 dropout = 0.0	1500 epochs
SpecMLP-W + C&S	lr $\in \{1 \cdot 10^{-5}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}\}$ number of layers $\in \{3, 4, 5\}$ hidden channels $\in \{128, 512, 1024\}$ dropout $\in \{0.0, 0.5\}$	lr = $1 \cdot 10^{-3}$ number of layers = 5 hidden channels = 128 dropout = 0.5	1500 epochs
N2Vec	lr $\in \{1 \cdot 10^{-5}, 1 \cdot 10^{-3}, 1 \cdot 10^{-2}\}$ walk length $\in \{16, 40\}$ walks per node $\in \{4, 10\}$ embedding dim $\in \{16, 128\}$ batch size $\in \{16, 128\}$ epochs $\in \{1, 5, 32\}$	lr = $1 \cdot 10^{-2}\}$ walk length = 40 walks per node = 10 embedding dim = 128 batch size = 128 epochs = 5	5 epochs
SpecMLP-W + C&S + N2Vec	lr $\in \{1 \cdot 10^{-5}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}\}$ number of layers $\in \{3, 4, 5\}$ hidden channels $\in \{128, 512, 1024\}$ dropout $\in \{0.0, 0.5\}$	lr = $1 \cdot 10^{-3}$ number of layers = 5 hidden channels = 128 dropout = 0.5	1500 epochs

D.3 Graph explainability

We use the concept of the GNNExplainer [54] as an initial graph explainability approach. GNNExplainer aims to identify a compact subgraph with a small subset of node features which play a crucial role in a given GNN for node classification.

We decided to run an initial GNNExpainer experiment on a small but representative subgraph of roughly 16000 nodes and 18000 edges. We predict node 2290 with our trained SAGEConv based model for the node classification task on the Line Graph, see Figure 11. The visualization represents the nodes which our model considers important for the node prediction of 2290. One can observe, that the model relies heavily on its immediate neighbourhood (denoted by thick edges). In our plot, we have visualized a 5 hop neighbourhood, for a SAGEConv GCN with 4 layers. Naturally, the model does not consider any node beyond its 4 hop neighbourhood in the computational graph.

D.4 Computational resources

All of our neural network trainings were performed on an Nvidia Quadro RTX 8000 GPU with 48GB memory.

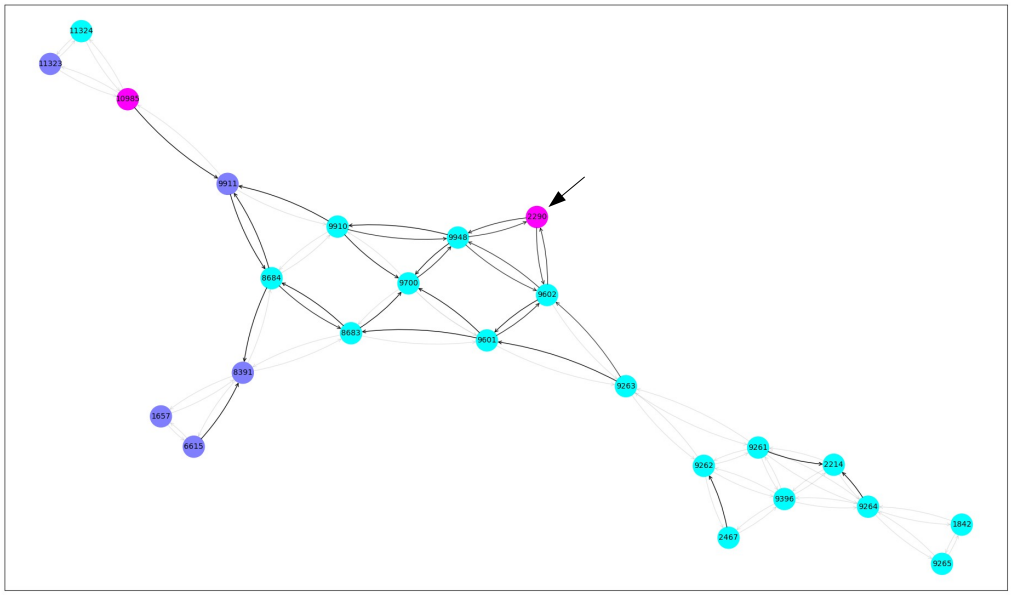

Figure 11: Graph explainability plot based on the GNNExplainer concept [54]. Our considered node of interest is node 2290. Pink nodes indicate nodes belonging artery/vein class, blue indicated arterioles/venules and green indicates capillaries. Thick edges represent a strong influence on the prediction. Please note that the considered graph is a spatial **Line Graph**.

E Datasheet for datasets

This description is an additional documentation intended to enhance reproducibility and follows the **Datasheets for Datasets**[3] working paper developed for the machine learning community.

- **For what purpose was the dataset created?** To foster research development in machine learning for graphs, in particular its application to neuroscience - specifically the brain vessel graph composition.
- **Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?** The dataset was created thorough a collaborative effort by neuro-scientists and computer-scientists at the Technical University of Munich and the Helmholtz Zentrum München (under the supervision of Ali Ertuerk, Bjoern Menze and Stephan Günnemann).
- **Who funded the creation of the dataset?** The creation of the dataset was funded only indirectly via the salaries of the scientists at the Technical University of Munich and the other corresponding affiliations of the authors.

E.1 Composition

- **What do the instances that comprise the dataset represent?** Our dataset represents graph representations of the whole brain vasculature. We are providing two alternative representations of the vascular graph. First, a representation where individual vessels are represented as edges in a Graph; and second, the corresponding Line graph were vessels are represented as nodes. One can interpret the graph of a single mouse brain as a single instance. Alternatively one can interpret each vessel (edge) and bifurcation (node) as a physical instance.
- **How many instances are there in total (of each type, if appropriate)?** By the instance definition of whole brain graphs as instances we are providing 17 graphs (with the option to generate the line graph) from 3 imaging sources as instances. In the future we plan to extend the dataset as soon as other whole brain vessel segmentations are made publicly available (open source).

  By the definition of vessels and bifurcation points we have millions of instances for each. Please see Table 1 for detailed numbers.
- **Does the dataset contain all possible instances or is it a sample of instances from a larger set?** We are providing all available instances.
- **What data does each instance consist of?** In either case, please provide a description. By definition 1); each instance represents a whole mouse brains' vascular graph saved in the widely used CSV format. By definition 2) each node represents a bifurcation point and each edge a vessel.
- **Is there a label or target associated with each instance?** Yes, in case of the edge and node instances, the information from the extracted graphs (features) can be used as the instance labels. E.g. in our node classification benchmark we use the vessel radius binned in three classes as an instance label.
- **Is any information missing from individual instances?** No, all of the information has been provided.
- **Are relationships between individual instances made explicit?** In our dataset the instances (brain graphs) are independent.
- **Are there recommended data splits (e.g., training, development/validation, testing)?** For the benchmark we split one whole brain into a train, validation and test set of 80/10/10.
- **Are there any errors, sources of noise, or redundancies in the dataset?** Our graph extraction is based on experimental imaging and segmentation techniques. Therefore, errors and uncertainty are inherent. We discuss these in detail in our Limitations section in the conclusion.
- **Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?** The provided dataset is self-contained.
- **Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)?** No.
- **Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?** No.
- **Does the dataset relate to people?** No.

[3] `https://arxiv.org/abs/1803.09010`

E.2 Collection process

- **How was the data associated with each instance acquired?** The data was generated from a set of different publicly available datasets of whole murine brain images and segmentations. The specifics of the generation of each of these public segmentations are specified in the referenced literature and their licenses, see B.4.
- **What mechanisms or procedures were used to collect the data?** We use the *Voreen* framework [31, 30] to generate graphs from segmentations. *Voreen* is a software which runs on a CPU.
- **If the dataset is a sample from a larger set, what was the sampling strategy?** The dataset is complete.
- **Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?** Only researchers (co-authors) of the Technical University of Munich and the Helmholtz Zentrum München were involved in the data collection process.
- **Over what timeframe was the data collected?** Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)? The generation of the dataset, including dedicated research to gather the base segmentations and to optimize the graph extraction procedure took roughly one year.
- **Were any ethical review processes conducted (e.g., by an institutional review board)?** Our work is purely based on public and open sourced data. However, ethical review processes were carried out for each of these open sourced base segmentation sets:

  The three graphs from Ji et al. [1] are based on animal experiments, they followed the Guide for the Care and Use of Laboratory Animals and have been approved by the Institutional Animal Care and Use Committee, for details see `https://doi.org/10.1016/j.neuron.2021.02.006`.

  The animal experiments for the nine datasets from the VesSAP paper [6] were carried out under approval of the ethical review board of the government of Upper Bavaria (Regierung von Oberbayern, Munich, Germany), and in accordance with European directive 2010/63/EU for animal research, for details see `https://doi.org/10.1038/s41592-020-0792-1`.
- **Does the dataset relate to people?** No.

E.3 Preprocessing/cleaning/labeling

- **Was any preprocessing/cleaning/labeling of the data done ?** Yes, this actually constitutes a core contribution of our work, therefore please refer to Section 2 in the main paper and to Supplementary section C.
- **Was the 'raw' data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?** The raw data are the base segmentations. They are publicly available, the links are provided in Supplementary section B.4.
- **Is the software used to preprocess/clean/label the instances available?** The *Voreen* software used for the graph extraction is publicly available, see our github repo.

E.4 Uses

- **Has the dataset been used for any tasks already?** In its current size and level of labeling detailization, the dataset was not used before (besides for the presented link prediction and node classification in this work).
- **Is there a repository that links to any or all papers or systems that use the dataset?** If so, please provide a link or other access point. Yes, `https://github.com/jocpae/VesselGraph`.
- **What (other) tasks could the dataset be used for?** In the main paper we discussed two standard tasks in machine learning on graphs; we think that our dataset can serve as a starting point for many interesting research directions in machine learning research and neurovascular research.
- **Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?** No.
- **Are there tasks for which the dataset should not be used?** No.

E.5 Distribution

- **Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?** Our Dataset is open sourced under a CC Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License. Therefore all third parties can openly access it.

- **How will the dataset will be distributed (e.g., tarball on website, API, GitHub)?** Yes, our DOI is `10.5281/zenodo.5301621`
- **When will the dataset be distributed?** The dataset is available from the moment of submission.
- **Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?** Our Dataset is open sourced under a CC Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) License.
- **Have any third parties imposed IP-based or other restrictions on the data associated with the instances?** No.
- **Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?** No.

E.6 Maintenance

- **Who is supporting/hosting/maintaining the dataset?** The dataset is initially supported and maintained by the lead authors of this paper. The data is initially hosted on a university server and links are provided in the github repository `https://github.com/jocpae/VesselGraph`. In the long term we aim to incorporate our dataset into the open graph benchmark (OGB) initiative[4].
- **How can the owner/curator/manager of the dataset be contacted (e.g., email address)?** Of course via e-mail: `johannes.paetzold@tum.de` and via the github repository, see question above.
- **Is there an erratum?** At this stage no, but we are happy to track them in a dedicated file in our github repository.
- **Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?** Yes, we release the dataset on open platforms on which we plan to continuously update our dataset. Particularly to add novel whole brain vessel graphs to the dataset.
- **If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances?** The dataset does not relate to people.
- **Will older versions of the dataset continue to be supported/hosted/maintained?** When novel versions of the dataset will be released we will continue to host and maintain the old versions of the dataset.
- **If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?** We encourage other researches to exactly that. Depending on their contribution they can contribute to our github repository (in case of implementations) or reach out to us via e-mail in case they want to contribute graphs to the dataset. Our dataset and code are open sourced, see above.

[4] `https://ogb.stanford.edu/`