Predicting and Interpreting Myocardial Infarction using Equivariant Graph Neural Networks

Jacob Bamberger
January 19, 2022

1 Introduction

Coronary Artery Disease (CAD) is a leading cause of death worldwide. CAD involves the narrowing of one major artery supplying the heart. Identifying lesions that lead to myocardial infarction in the future is a challenging and time consuming daily task for experts. For more information about CAD, myocardial infarction and deep learning approaches see [3], since our work is a continuation thereof.

Much like [3], we propose a deep learning approach to detect myocardial infarction for vessels from stable CAD patients. However, instead of considering images as done in [3] we consider 3-dimensional (3D) models of arteries. We then apply graph neural networks and equivariant neural networks to detect whether a vessel leads to a Myocardial Infarction (MI). We find that our method leads to a data efficient and robust classification of lesions.

1.1 The Dataset

The dataset consists of personalized 3D meshes obtained from coronary angiograms of 80 patients, an example can be seen in Figure 2. The three main coronary arteries are the right coronary artery (RCA), the left circumflex artery (LCX) and the left anterior descending (LAD) as seen in Figure 1. For each patient, up to three of these vessels have been reconstructed leading to 188 data points (one data point is a vessel). The 3D reconstruction consists of a triangular mesh (or a 2-dimensional simplicial complex) that is, a graph $G = (V, E)$ and a set of triangles $T \in \mathcal{P}(E \times E \times E)$. In addition, each mesh comes with a $3D$ embedding $f_G : V \to \mathbb{R}^3$, meaning that each node has coordinates. Each data point is labelled as culprit or non-culprit, the classification task at hand is thus to predict whether a given mesh is a culprit or not.

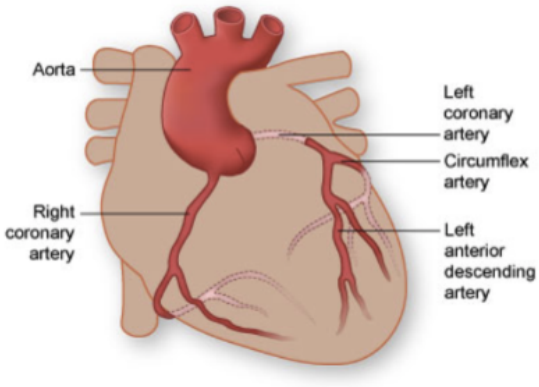


Figure 1: Heart model

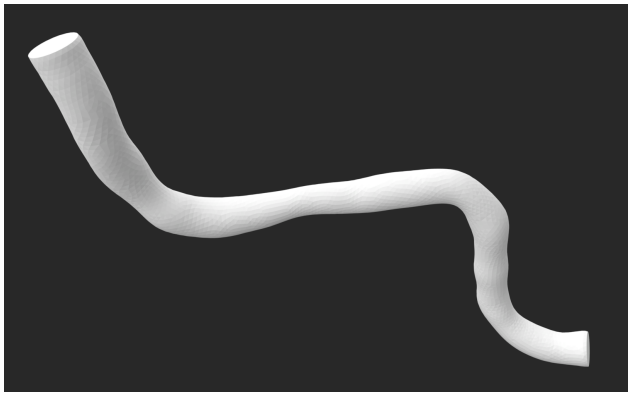


Figure 2: Example of the triangular mesh for a vessel

1.2 Symmetries in the Data

A first step in modern Geometric Deep Learning methodology [2] is identifying symmetries in the data, in order to encode them as inductive bias in the deep model architecture.

We identified two structures that can be encoded as inductive bias by available methods. The first is the graph structure of the mesh, which can be leveraged through the use of Graph Neural Networks. The second one comes from the realization that the classification should be invariant to rotations and translation of the surface, since such symmetries do not alter the surface's geometry. The group of symmetry at play is the Euclidean group $E(3)$ with the usual action on $\mathbb{R}^3$, so we conclude that our network need to be $E(3)$-invariant. This provides robustness to noise that can naturally be accumulated in the data collection: we cannot ensure that patients were lying at the same exact angle, or that arteries are embedded in bodies in canonical ways! It is moreover widely believe that such inductive biases are a way to fight against the curse of dimensionality. Another type of structure that we have not taken into account is the highly regular structure of the mesh. Locally, the mesh looks like a plane since it is a 2-dimensional surface, considering such structure leads to Gauge Equivariant networks as done in [6].

2 Related work

2.1 MI

MI from images Deep learning has recently been appplied to coronary angiograms in several works. In [3], the first (to our knowledge) use of deep learning method was used to predict MI. The difference between [3] and our work is that their dataset consisted of images, and contained roughly three times more data than ours.

LTS4 Summer 2021 project Our project was a continuation of a project by the same laboratory [8], which used GNNs to predict culprit stenosis. In addition to the coordinates, the data was enriched with computational fluid dynamics (CFD) data (such as Wall Shear Stress and Tsvi measurement) obtained from simulations which took as input the mesh and coordinates. Results were promising and showed that the CFD simulation data did not significantly increase predictive power. However, those results cannot be compared to ours because an error was found in the implementation showing that the deep learning algorithm was training on the validation and test set. Nonetheless, these experiment showed that GNN had the expressiveness to fit the training data neatly.

Hemodynamics and cardiovascular disease Computational fluid dynamics (CFD) has lately attracted much attention by the deep learning community, and recently methods from geometric deep learning have been applied to the study of hemodynamics in arteries. In [6], CFD parameters are estimated with very high accuracy on artificial data which clearly demonstrates that geometric deep learning is fruitful for understanding cardiovascular disease.

2.2 Graph Neural Networks

As mentioned previously, GNNs are extremely successful at processing data that lies on, or comes as, graphs. In this section we discuss relevant neural architectures.

Equivariant graph networks In order to train $E(3)$ invariance models, we used the E_GCL layer proposed in [5]. This is a layer that takes as input a graph G with a feature map $f_G : V \to \mathbb{R}^d$ and coordinate map $c_G : V \to \mathbb{R}^n$ and outputs similar maps into $\mathbb{R}^{d_2}$ and $\mathbb{R}^n$ such that the the coordinate map transformed $E(n)$-equivariantly. It is important to note that information flows from the coordinates to the features in an invariant way. This is achieved because the messages from node i to node j is $m_{ij} = ||c_G(i) - c_G(j)||_2^2$ which is invariant to $E(n)$. Moreover, the output coordinate map is obtained from a point-wise linear transformation of the input coordinates, making this part of the network equivariant and not invariant.

Mesh Convolution In [6], novel neural network architecture that operate on manifolds with tangential data (such as vector-fields) are developed and used to predict CFD quantities on artificial coronary data. The model architecture manages to use the neighborhood structure as an inductive bias. Although it is clear to us that such inductive bias is promising to our problem, it is not clear to us how to adapt the model to our problem since our problem does not use tangential data and combining their model to $E(3)$ equivariance would be a very interesting future work.

Graph Isomorphic Network One specific GNN architecture that was used in the last layer of our architecture is the GIN. This is a particularly expressive Graph Neural Network. However we did not tweek or learn the ϵ parameter.

3 Experiments and results

3.1 Equivariance

Need for invariance (CoordToCnc)
The first 'experiment' was a simple observation. We trained the model using only GIN layers (called CoordToCnc) and trained it as done in [8]. We then did rotational and translational invariance tests. That is, we compare the predictions of a mesh (G, c_G) with a transformed version of the same mesh (G, Tc_G) where the transformation T was either a

small rotation or a small translation. The result on 5 randomly sampled data points showed that a small rotation made the model change output classification for 2 data points, whereas the model seemed robust to translation (no points changed classification). Since rotational and translation noise can easily occur in the data collection pipeline, it seemed important to make sure the model learned roto-translational robustness.

Learning invariance (4RotCoordToCnc)
A first approach to robustness to group actions is simple data augmentation. This approach is vastly used in deep learning for images (and is also done in [3] through rotation and shiftscalerotate augmentation), and often appears as flipping and rotating images to augment datasets. Several data augmentation schemes were attempted. The more fruitful one was simply augmenting the data set by adding 4 copies of each data points rotated at angles of $\pm 9\,\mathrm{deg}$ and $\pm 18\,\mathrm{deg}$ around the z axis (which can be done by simple matrix multiplication). The architeture is the CoordToCnc architecture to which the data augmentation scheme was added and is what we refer to as the $4RotCoordToCnc$.

Invariance as an inductive bias (Equiv_GIN)
By far the most efficient and successful way of making the model invariant was to encode the invariance in the architecture in the same way [5] does it. An extensive grid search was done (details in the Appendix A.1) using 10-fold cross validation to find a proper combination and number of equivariant and GIN layers was done. The invariance is ensured by preceding GIN layers by equivariant layers, and then feeding the invariant part of the Equivariant layers in the GIN layer. Details can be found in the code. Other ways to encode the rotational invariance as an inductive bias was attempted by preprocessing the data and only feeding the magnitude (or l_2 norm) of the coordinate vectors into the network, which we call the MagToCnc model. Note that this does not ensure translational invariance, moreover, that model was not as good as the 4Rot_CoordToCnc model.

3.2 Data augmentation (non-symmetry inspired)

Data augmentation not inspired by group actions were attempted as well and are listed here.

5 Nearest Neighbor Augmentation (Equiv_GIN_KNN5) A promising direction from [8] was to augment the dataset through the constructing of a 5 nearest neighbor graph of each data point's point cloud, thus increasing the dataset size by a factor of 2. This model is refered to as $Equiv\_GIN\_KNN5$.

Gaussian Noise Augmentation (Equiv_GIN_Gaussian) We also attempted to add noise in the data at the coodinate level. This is similar to what is done in the image deep learning world. For each data point, we added Gaussian noise with mean 0 and standard deviation 0.001 into the coordinate. We did this to multiply the dataset size by a factor of 2 (to compare to KNN). This model is referred to as $Equiv\_GIN\_Gaussian$.

3.3 Results

The performance of each network is measured using four evaluation metrics (i) classification accuracy, which measures the performance of correctly predicted classes; (ii) sensitivity which measures the proportion of future culprit lesions that are correctly identified; (iii) specificity which measures the proportion of non-culprit lesions that are correctly identified; and (iv) F1 score, which is defined as the harmonic mean of precision and recall. Each model is evaluated on the same test set, and is trained on the same train validation split using 10-fold cross validation.

	Accuracy	Sensitivity	Specificity	f1-score
GIN	0.490 ± 0.047	0.473 ± 0.301	0.513 ± 0.397	0.377 ± 0.237
4Rot_GIN	0.632 ± 0.082	0.618 ± 0.127	0.650 ± 0.146	0.593 ± 0.096
Equiv_GIN_Gaussian	0.779 ± 0.096	0.845 ± 0.010	0.688 ± 0.140	0.721 ± 0.126
Equiv_GIN	0.805 ± 0.111	0.909 ± 0.115	0.663 ± 0.238	0.720 ± 0.183
Equiv_GIN_KNN5	0.816 ± 0.089	0.882 ± 0.071	$0.725 \pm 0.0.184$	0.758 ± 0.135

Table 1: Evaluation metrics on test set of the different models for the culprit/non-culprit classification task, all trained using hyper-parameters maximizing validation $f1$-score over a grid search on a 10 fold cross validation

3.4 Interpretability

In this section we apply the GNNExplainer method from [7] to attempt to interprete our 'Equiv_GIN' model. GNNExplainer is a successful method for explainability of graph neural networks for node classification, node prediction, and graph classification. In each case, GNNExplainer finds a subgraph of the computational graph of the objective. In the case of graph classification which we are interested in, GNNExplainer will find a subgraph of the full graph. GNNExplainer optimizes a function acting as a proxy for the value of

$$I(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S), \tag{1}$$

where Y is the prediction, G_S is the subgraph, X_S its features, I is the mutual information, and H is the Shannon entropy. In our case, Y is thought of as a random variable with two events: culprit and non-culprit. Similarly, G_S and X_S are also thought as random variable where each node and features appears with a given probability.

We apply the GNNExplainer method by training an explainer model for 200 epochs on our best performing Equiv_GIN model out of all the 10-fold cross validation ones. We then select the nodes that GNNExplainer deemed the most important for the prediction, indeed for all examples there is a clear cutoff of what most important means since a large percentage of nodes have almost equal scores except for some outliers, as seen in Figure 5c from Appendix A.3.

For each mesh, we then compare the selected nodes to an expert labelling of the vessel. The expert labelled each vessel by: 1 the portion of 3 diameters length upstream to the lesion, 2 the proximal part of the lesion, 3 the mid part of the lesion, 4 the distal part of the lesion, 5 the portion of 3 diameters length downstream to the lesion and 6 the remaining part of the vessel. We also illustrate this procedure in Figure 5 by adding some pictures of the importance mask, and the expert label of vessels.

No statistically significant pattern was identified among the small test set of 19 samples, but several things can be said. for many vessels such as the one in Figure 6b, nodes near the boundary of the vessel are often identified as important. This partly explains why for 15 out of the 18 properly classified vessels, an abnormal amount of important nodes are reported in remaining part of vessel. Out of those same vessels, 7 have abnormally many in the upstream region, 4 in the proximal region, 10 in the mid region, 3 in the distal and 2 in the downstream region. It thus seem that one of the most important region for the prediction is the mid part of the lesion.

4 Conclusion and future work

To conclude, we have demonstrated how graph neural networks and principles from geometric deep learning can provide robust and data efficient ways of solving problems that are hard and expensive for humans to do. We then interpreted the architecture showing that in some cases the model does base its prediction find the stenosis.

There are many interesting directions in which this project can go. We now summarize several of these directions which have been mentioned throughout the semester:

Mesh Convolution
A promising direction would be to implement a model that is a GNN, is $E(3)$-invariant, but is also gauge-(equi/in)variant as done in [6]. This seems to be a direction towards even more robust and data efficient models.

Interpretability patch
One failure of the interpretability aspect of the project was that GNNExplainer often deemed points close to the boundary to be important. We believe that this is because points near the boundary are anomalies: they do not have a fully planar neighborhoud. Therefore, when aggregating messages around these neighborhoods, only around 3 neighbours are summed, compared to 5 or 6 or 7 for a point in the middle of the surface. This provides one potential explanation for this problem. One easy solution to this is to forget about the nodes that are too close to the boundaries before feeding them into the classifier, indeed this would lessen their importance.

More Interpretability
There are other interpretability methods which could be applied to the project.

Computational Geometry perspective
An interesting but different approach would be to look at computational geometric properties of the surfaces. One algorithm that would be of great interest is one that manages to find the shortest path (with respect to the l_2 norm of R^3) around the vessel through each point. The length of such path could be added as a node feature. Note that this is both permutation and $E(3)$-invariant.

Circular graph RNN
Another architecture that was mentionned is to use an RNN on the sequences of 2D graph that correspond to the axial cuts, meaning using how the boundary of the vessel evolves along the center line of the vessel. This requires a computational geometry solution that consists of finding the coordinates of the centerline, and then defining the sequence of axial cuts.

Hemodynamics
A similar study to [8] can be done in the Equivariant framework to look at if enriching the data with CFD simulations is a promising approach.

Flow direction as feature
If the network is $E(3)$-equivariant, it cannot distinguish between the part of the vessel closest to the heart, and the part furthest to the heart. It could therefore be interesting to encode this information as a binary vector. This is done in [6] in the encoding of the flow direction.

References

[1] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[2] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[3] P. F. F. A. D. A. N. D. D. M. M. P. O. M. S. F. T. M. Dorina Thanou1 Ortal Senouf Omar Raita1, Emmanuel Abbé. Predicting future myocardial infarction from angiographies with deep learning. *preprint*, 2021.

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

[5] V. G. Satorras, E. Hoogeboom, and M. Welling. E(n) equivariant graph neural networks. *CoRR*, abs/2102.09844, 2021.

[6] J. Suk, P. de Haan, P. Lippe, C. Brune, and J. M. Wolterink. Mesh convolutional neural networks for wall shear stress estimation in 3d artery models. *CoRR*, abs/2109.04797, 2021.

[7] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894, 2019.

[8] D. T. L. Yunshu Ouyang, Ortal Senouf. Predicting myocardial infarction from personalized 3d geometries, 2021.

A Appendix

A.1 Hyperparameters

All our models had similar hyperparameters tuned differently.

The hyperparameters were the following:

- optimizer, which was set to Adam for all experiment because it seemed to be the most promising.
- optim_lr, learning rate of the optimizer. Very important hyperparameter for us.
- optim momentum, which was set to 0
- batch size, this was varied for the CoordToCnc models but fixed to 1 for the Equivariant models
- early stop, which indicates after how many epochs where the validation loss does not reach a minimal value.
- epochs, this was experimented with when not using early stop but that caused the model to overfit. After that it was set to 1000, but the models would almost always early stop.
- allow stop, this is the minimal number of epochs to train for and had priority over the early stop. It was useful in early versions of the equivariant and CoordToCnc models but eventually became obsolete because the validation loss would decrease early on making.
- model, this indicates the model name
- num_equiv, only useful for equivariant models. Determines number of equivariant layers
- num_gin, determines number of GIN layers after the equivariant layers.
- num_node_features, always 3 for the three coordinates for us.
- path_data, indicates what path to go fetch the data from. Only difference between CoordToCnc and 4Rot_CoordToCnc and between non augmented and data augmented models...
- path_mode, is just where to fetch the model
- physics, remains from [8] but not used in our project.
- seed, this determiens the train, val test splits and the cross validation (train, val) pairs. This was set to 0 for all our experiments. Confirming (or not...) our results with a different seed would be interesting.
- weighted_loss, this was set to 0.6 for all our experiments (because there 60% of the vessels are non-culprits), to balanced out the fact that the classes are not balance. We tried to tweek this hyperparameter by computing the actual class imbalance for each train, val pairs. But this did not lead to improvements.
- hidden dimension, all our blocks have 16 hidden channels, this parameter was not tuned.

A.2 Model descriptions and implementation details

CoordToCnc

The CoordToCnc architecture was chosen as in [8], but not removing edge pooling since it was not computationally efficient. The architecture design can be seen in Figure 3. These experiments were done earlier than the Equivariant ones, when the pipeline was not fully developed which explains why we haven't been able to keep track of the experiments as well.

We found that the best hyperparameters for our model were similar to that of [8] and were:

- batch size = 1
- early stop = 10
- allow stop = 0
- optim_lr = 0.0001
- epochs = 1000

Equiv_GIN

After some architecture search, we picked Equiv_GIN's architecture showed in 4, but we kept the number of equivariant and gin layers as a hyperparameter. We did a grid search on 414 combinations of num_gin, num_equiv, allow_stop, optim_lr and early_stop, leaving batch_size fixed to 1, optimizer to Adam with 0 momentum, and epochs to 1000. We ran the 3 first runs of a 10 fold cross validation and selected the model with the highest validation F1 score to conclude that the best hyper parameters were:

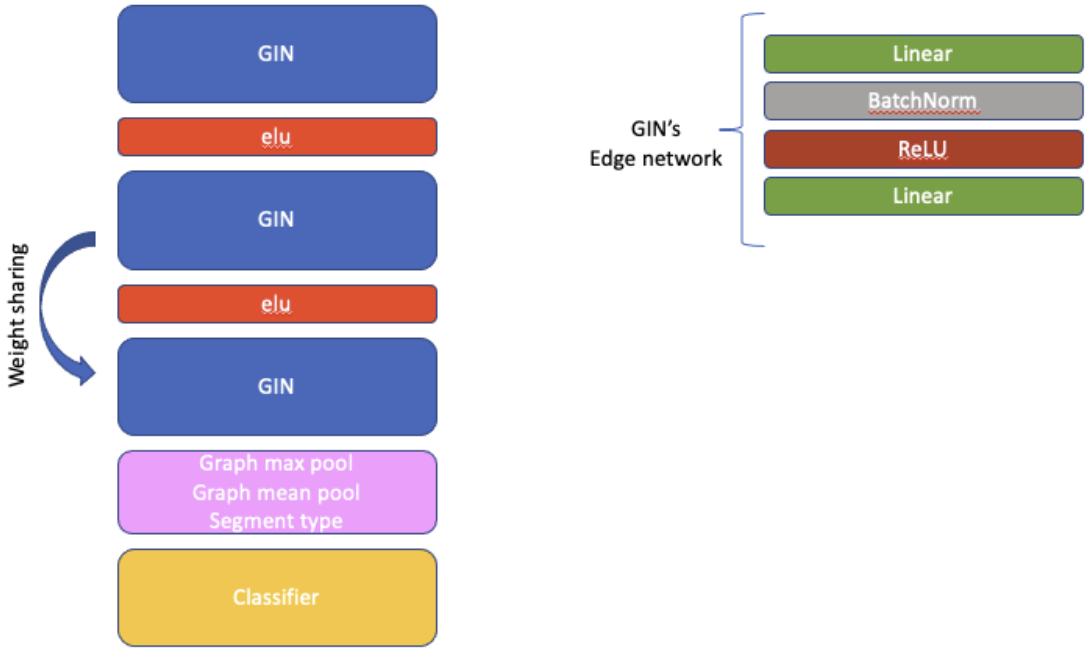


Figure 3: GIN's model architecture

- batch size = 1
- early stop = 50
- allow stop = 0
- optim_lr = 0.0001
- epochs = 1000
- num_equiv = 3
- num_gin = 1 .

For more information about hyper parameter see Appendix A.1. For more information about the grid search, all runs were recorded on wandb as explained in Appendix A.5 under group name: 'indiv-CV-run_Equiv_GIN_CV3outof10' and evaluated in group 'evaluate' and job type 'equiv_gin'.

Equiv_GIN_Gaussian
Similarly, a 10 fold cross validation gridsearch over 32 hyperparameter combination was done and all runs can be found on the wandb platform with group name 'run_Equiv_GIN_Gaussian2_std0.001_real'. The model with best validation F1 score was selected an evaluated in group 'evaluate'. The best hyperparameters were:

- batch size = 1
- early stop = 50
- optim_lr = 0.0001
- allow stop = 100
- epochs = 1000
- num_equiv = 2
- num_gin = 0.

Interestingly, the best architecture only had 2 equivariant layers, and no GIN layer, making it the simplest model.

Equiv_GIN_KNN5
Similarly, a 10 fold cross validation gridsearch over 24 hyperparameter combination was done and all runs can be found on the wandb platform with group name 'run_Equiv_GIN_KNN5'. The model with best validation F1 score was selected an evaluated in group 'evaluate'. The best hyperparameters were:

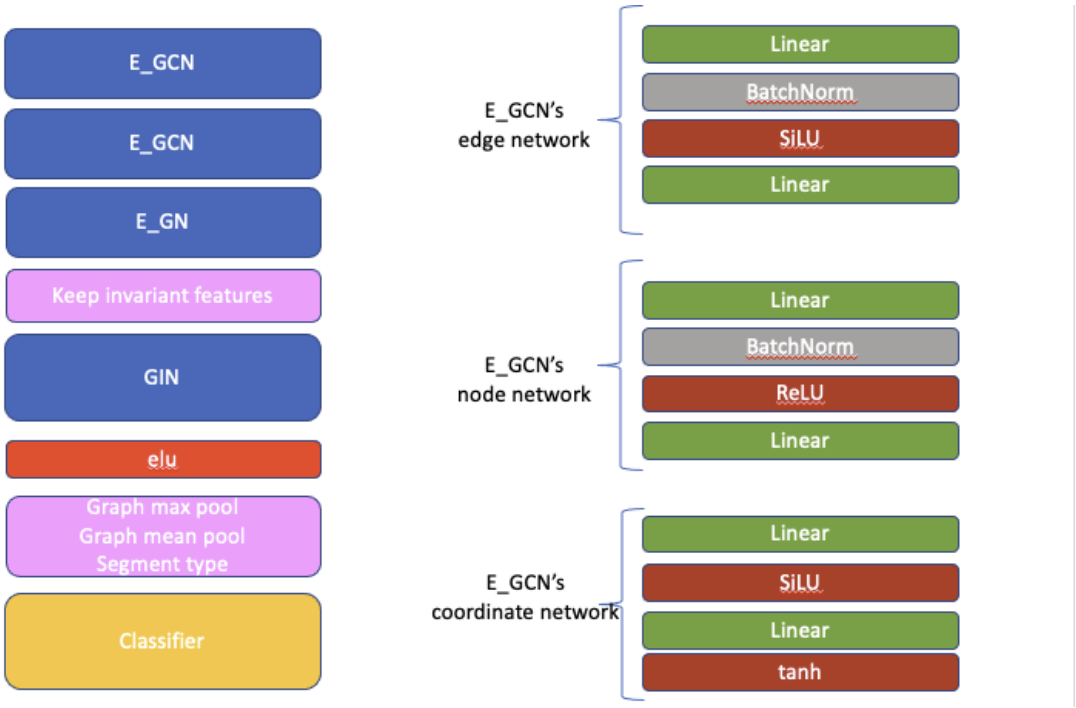


Figure 4: Equiv_GIN architecture

- batch size = 1
- early stop = 100
- optim_lr = 0.0001
- allow stop = 100
- epochs = 1000
- num_equiv = 3
- num_gin = 0

.

Again, it was better not to use a GIN layer at the end.

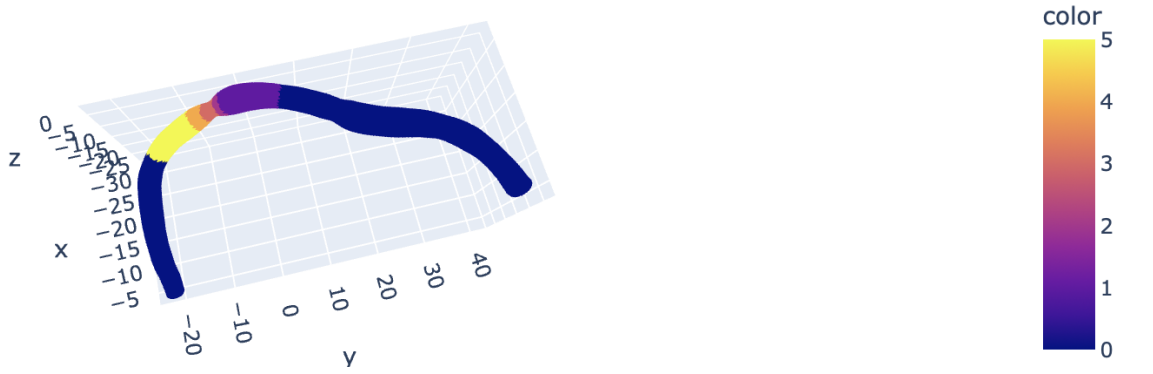
A.3 Interpretability

A.4 Code description

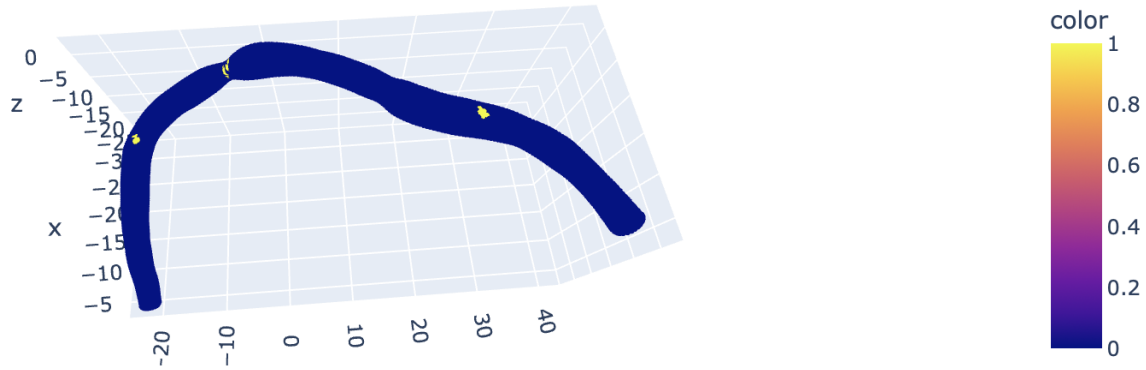
Making the code public and readable is crucial for fighting the reproducibility crisis in the computational science. This is why we shared what we believe to be clean code on https://github.com/jacobbamberger/MI-proj.git. Please refer to the readme for more information on how to use the code.

A.5 Experiment book-keeping

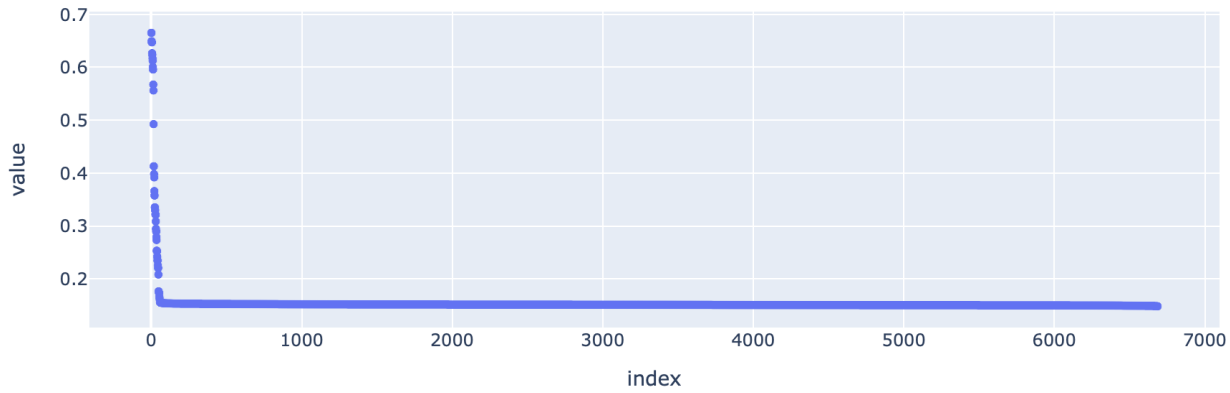
Experiment book-keeping has its role to play in the reproducibility crisis. In addition to sharing our code, we therefore also make all our expriments public on https://wandb.ai/yayabambam/mi-prediction, using the weights and biases platform [1]. The experiments were organized as best as possible using the 'Group' and 'Job Type' attribute on wandb. When navigating the database, we recommend to group by 'Group' and 'Job Type'. Once this is done there are two types of 'Groups': the 'evaluate' group, and the others. The 'evaluate' group consists of all the models we decided to evaluate on the test set, and each job type corresponds to one model (for example 'equiv_gin') and contains 10 runs each corresponding to one run in the 10-fold cross validation procedure. All the other groups have names that begin with 'indiv_CV_run' followed by the name of a model. Each Job Type withing such a group corresponds to one choice of hyper parameter, and the whole group can be thought of as a big grid search. Again, each Job Type will usually contain 10 runs (sometimes 3 to gain time) crresponding the the 10-fold cross validation procedure. Each Job Type, and



(a) Expert label of vessel 4

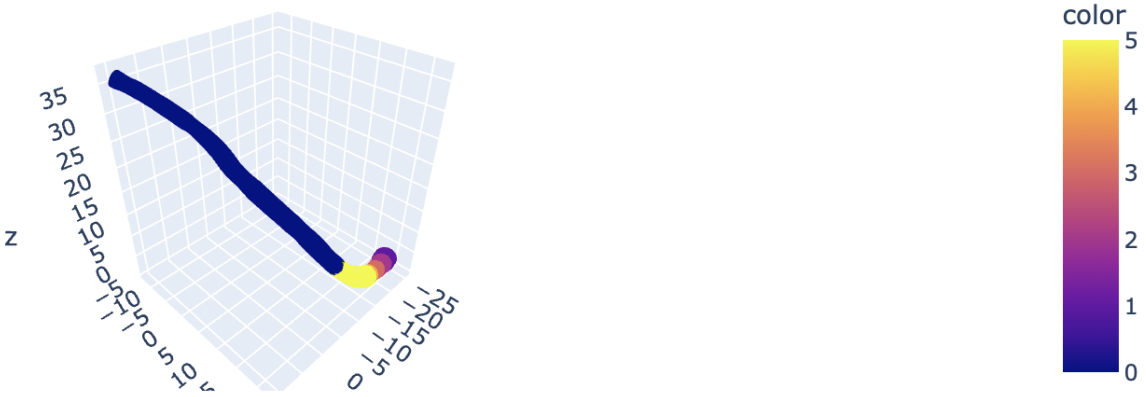


(b) Importance label of vessel 4



(c) Node importance distribution

Figure 5: Expert node label, node importance, and node importance distribution for vessel 4



(a) Lesion expert label

(b) Node Importance of vessel 0

Figure 6: Expert node label, node importance of vessel 0

each single runs have attributes which correspond to the hyper parameters of the model, this info can also be found in the 'overview' section of individual runs and looking at the Config section. Still looking at single runs, one can look at the logs to see the validation list, as well as the test set list (in the case of runs in the evaluate section).