Table 2. The best-performing hyperparameters for each GNN propagation rule in our experiments. The only experiment where the baseline model outperforms ECG is the SAGE propagation layer on `amazon-ratings`; hence, the hyperparameters k and p_{de} are irrelevant.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
ResNet					
L	2	1	5	5	1
d	512	512	512	512	512
GCN					
L	5	2	4	4	3
d	512	512	256	512	256
τ	MLPBGRL	MLP→GNN	MLP	MLP	BGRL
k	3	3	3	3	3
p_{de}	0.5	0.5	0.5	0.5	0.0
SAGE					
L	5	2	5	4	5
d	512	1024	256	256	256
τ	BGRL	Baseline	BGRL	BGRL	BGRL
k	10	—	20	20	10
p_{de}	0.5	—	0.5	0.5	0.0
GAT-sep					
L	5	2	5	5	4
d	512	512	256	256	256
τ	BGRL	MLP→GNN	MLP	BGRL	BGRL
k	10	3	20	20	10
p_{de}	0.5	0.5	0.5	0.5	0.5
GT-sep					
L	5	2	5	5	4
d	512	512	256	256	256
τ	BGRL	MLP→GNN	MLP	MLPBGRL	BGRL
k	20	3	20	20	10
p_{de}	0.5	0.5	0.0	0.0	0.5

A. Training information

We used the same experimental setup as presented in Platonov et al. (2023). Results are aggregated over ten random splits of the data, with each run taking 50% of the nodes for training, 25% for validation, and 25% for testing. The following hyperparameters are tuned for all models and baselines, using the average validation performance across the splits:

- Number of GNN/ResNet layers, $L \in \{1, 2, 3, 4, 5\}$.
- The dimensionality of the GNN/ResNet's latent embeddings, $d \in \{256, 512, 1024\}$.

Additionally, for the ECG models only, the following hyperparameters were swept:

- Embeddings used by ECG, $\tau \in \{\text{MLP}, \text{BGRL}, \text{MLPBGRL}, \text{MLP} \to \text{GNN}\}$, referring to:

MLP: Using the embeddings from a pre-trained ResNet;
BGRL: Using the embeddings from a pre-trained BGRL model;
MLPBGRL: Using the normalised concatenation of the ResNet and BGRL embeddings;
MLP→GNN: Using the embeddings from a pre-trained MLP-ECG model of the same type.

Table 3. Detailed breakdown of model performance on the datasets proposed by Platonov et al. (2023). ResNet, GCN, SAGE, GAT-sep and GT-sep are the baselines, while all the other models are variants of ECG. Red marks the best performance on each dataset for each of the considered GNN architectures and the corresponding ECGs. Accuracy is reported for `roman-empire` and `amazon-ratings`, and ROC AUC is reported for `minesweeper`, `tolokers`, and `questions`.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
ResNet	65.88 ± 0.38	45.90 ± 0.52	50.89 ± 1.39	72.95 ± 1.06	70.34 ± 0.76
GCN	73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27
MLP-ECG-GCN	83.55 ± 0.39	50.99 ± 0.64	92.63 ± 0.10	84.81 ± 0.25	76.25 ± 0.59
BGRL-ECG-GCN	80.59 ± 0.48	48.99 ± 0.28	92.35 ± 0.10	84.25 ± 0.22	77.50 ± 0.35
MLPBGRL-ECG-GCN	84.53 ± 0.26	50.11 ± 0.60	92.47 ± 0.50	84.73 ± 0.23	77.32 ± 0.31
MLP→GNN-ECG-GCN	84.39 ± 0.22	51.12 ± 0.38	92.56 ± 0.23	84.35 ± 0.31	75.16 ± 0.87
SAGE	85.74 ± 0.67	53.63 ± 0.39	93.51 ± 0.57	82.43 ± 0.44	76.44 ± 0.62
MLP-ECG-SAGE	85.82 ± 0.62	53.32 ± 0.39	94.10 ± 0.08	82.60 ± 0.23	76.13 ± 0.41
BGRL-ECG-SAGE	87.88 ± 0.25	53.12 ± 0.32	94.11 ± 0.07	82.61 ± 0.29	77.23 ± 0.36
MLPBGRL-ECG-SAGE	86.50 ± 0.34	52.34 ± 0.92	94.01 ± 0.07	82.55 ± 0.18	76.55 ± 0.33
MLP→GNN-ECG-SAGE	85.94 ± 0.57	53.45 ± 0.27	93.77 ± 0.12	82.52 ± 0.22	75.53 ± 0.64
GAT-sep	88.75 ± 0.41	52.70 ± 0.62	93.91 ± 0.35	83.78 ± 0.43	76.79 ± 0.71
MLP-ECG-GAT-sep	88.22 ± 0.36	52.98 ± 0.30	94.52 ± 0.20	83.91 ± 0.32	77.30 ± 0.47
BGRL-ECG-GAT-sep	89.62 ± 0.18	52.20 ± 0.57	94.24 ± 0.15	84.23 ± 0.25	77.38 ± 0.18
MLPBGRL-GAT-sep	88.73 ± 0.37	51.06 ± 0.73	94.39 ± 0.20	84.11 ± 0.23	76.97 ± 0.45
MLP→GNN-ECG-GAT-sep	88.04 ± 0.32	53.65 ± 0.39	93.97 ± 0.19	83.75 ± 0.30	75.61 ± 0.74
GT-sep	87.32 ± 0.39	52.18 ± 0.80	92.29 ± 0.47	82.52 ± 0.92	78.05 ± 0.93
MLP-ECG-GT-sep	88.56 ± 0.35	52.68 ± 0.65	93.62 ± 0.27	83.65 ± 0.29	77.82 ± 0.43
BGRL-ECG-GT-sep	89.56 ± 0.16	52.37 ± 0.30	93.55 ± 0.18	82.97 ± 0.26	78.12 ± 0.32
MLPBGRL-GT-sep	88.70 ± 0.30	52.29 ± 0.60	93.52 ± 0.25	84.00 ± 0.24	77.85 ± 0.45
MLP→GNN-ECG-GT-sep	88.62 ± 0.46	53.25 ± 0.39	92.69 ± 0.34	83.41 ± 0.44	75.50 ± 1.13

Note that, for methods requiring access to labels (such as MLP), a separate set of embeddings is computed for every dataset split (to avoid test data contamination). For self-supervised methods like BGRL, no labels are used, and hence a single set of embeddings is produced for all experiments.

- The number of neighbours sampled per node, $k \in \{3, 10, 20\}$.
- The DropEdge rate, $p_{de} \in \{0.0, 0.5\}$.

The model configuration with the best-performing average validation performance is then evaluated on the corresponding test splits, producing the aggregated performances reported in Table 1.

The best-performing hyperparameters for each model type on each dataset are given in Table 2. Each individual experiment has been executed on a single NVIDIA Tesla P100 GPU, and the longest training time allocated to an individual experiment has been six hours (on the `questions` dataset).

For convenience, and to assess the relative benefits of various ECG embedding sources, we provide in Table 3 an expanded version of 1, showing the test performance obtained by the tuned version of each ECG variant, for every embedding type.

For additional information, the anonymised code can be found at `https://anonymous.4open.science/r/evolving_computation_graphs-97B7/`.

A.1. Qualitative experiments

In Figure 2, we also verify how predictive of the node classes the graph topology is when obtained from the original data compared to when we build a complementary graph G^{ECG}. More precisely, we first build the graph G^{ECG} as presented in Step 1 of Algorithm 1, using pre-trained MLP embeddings. Then we use a randomly initialised GCN to compute node

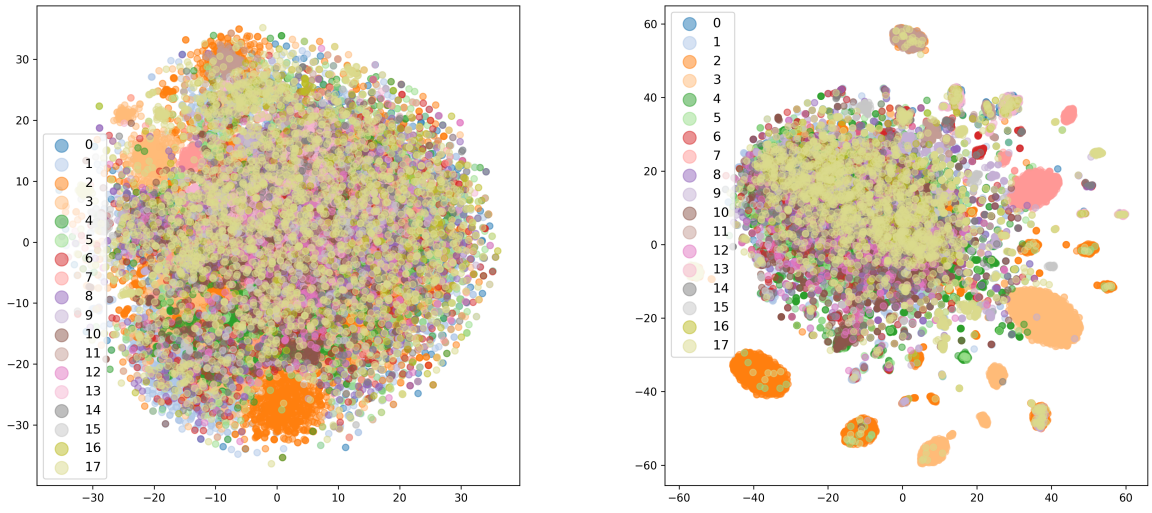

Figure 2. For `roman-empire`, we use a random GCN layer to obtain node embeddings based on the original graph G (left) or from the complementary graph G^{ECG} (right). The colours correspond to the ground-truth labels of the nodes.

embeddings on the input graph G, as well as on G^{ECG}. We visualise these two sets of node embeddings using t-SNE by projecting to a 2D space, attributing the colour of each point based on the node's ground truth label. We can observe that using the G^{ECG} topology leads to more distinguishable clusters corresponding to the classes even without any training, thus supporting the enhancements in performance when building ECG-GNN.

B. Homophily

One very popular metric, used by several studies, is *edge homophily*, which measures the proportion of homophilic edges:

$$\texttt{h-edge} = \frac{|(u,v) \in E : y_u = y_v|}{|E|} \tag{7}$$

while *adjusted homophily* was introduced to account for number of classes and their distributions:

$$\texttt{h-adj} = \frac{\texttt{h-edge} - \sum_{k=1}^{C} D_k^2/(2|E|)^2}{1 - \sum_{k=1}^{C} D_k^2/(2|E|)^2} \tag{8}$$

where $D_k = \sum_{u:y_u=k} d_u$, the sum of degrees for the nodes belonging to class k.

Additionally, *label informativeness* (LI) measures how much information about a node's label is gained by observing its neighbour's label, on average. It is defined as

$$\texttt{LI} = I(y_\xi, y_\eta)/H(y_\xi) \tag{9}$$

where $(\xi, \eta) \in E$ is a uniformly-sampled edge, H is the Shannon entropy and I is mutual information.

In Table 4, we analyse the properties of the complementary graphs G^{ECG} with $k = 3$ nearest neighbours. We note that this represents the graph used by ECG-GCN, which preferred lower values of k, while the optimal values of k for SAGE, GAT-sep and GT-sep were on the higher end, varying between 3, 10 and 20 depending on the dataset.

We observe that MLP-ECG confirms our hypothesis: taking the edges corresponding to the pairs of nodes marked as most similar by the ResNet results in a graph G^{ECG} with high homophily, especially compared to the original input graph. It is important to note that all of our MLP-ECG graphs were obtained with a relatively shallow ResNet, which, as it can be seen

Table 4. Statistics of the original heterophilous graphs and of the evolutionary computation graph obtained from MLP and BGRL.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
edges	$32,927$	$93,050$	$39,402$	$519,000$	$153,540$
edge homophily	0.05	0.38	0.68	0.59	0.84
adjusted homophily	-0.05	0.14	0.01	0.09	0.02
LI	0.11	0.04	0.00	0.01	0.00
ECG($k=3$) edges	$67,986$	$73,476$	$30,000$	$35,274$	$146,763$
MLP-ECG edge homophily	0.73	0.66	0.79	0.79	0.97
MLP-ECG adjusted homophily	0.7	0.53	0.33	0.4	0.41
MLP-ECG LI	0.65	0.33	0.16	0.19	0.28
BGRL-ECG edge homophily	0.16	0.3	0.68	0.6	0.93
BGRL-ECG adjusted homophily	0.06	0.02	0.12	0.08	0.01
BGRL-ECG LI	0.1	0.03	0.05	0.03	0.03

in Table 1 lacks in performance compared to the graph-based methods. However, our method's success in conjunction with GNNs shows that even a weak classifier can be used to generate homophilic graphs that can improve performance when used to complement the information provided by the given input data.

C. Algorithm

Algorithm 1 Evolving Computation Graph for Graph Neural Networks: ECG-GNN

Input: Graph $G = (V, E)$; Node Feature Matrix $\mathbf{X}$; Adjacency Matrix $\mathbf{A}$.
Hyper-parameters: Value of k; Drop edge probability p_{de}; Number of layers L;
Output: Predicted labels $\hat{\mathbf{y}}$
begin
  /* Step 1: Extract embeddings */
  $\mathbf{H}_{\text{ECG}} \leftarrow \gamma(\mathbf{X}, \mathbf{A})$ /* Embeddings stored in matrix */
  /* Step 2: Construct ECG graph */
  $\mathbf{S} \leftarrow \mathbf{H}_{\text{ECG}}\mathbf{H}_{\text{ECG}}^{\top}$
  for $u \in V$ **do**
    for $v \in V$ **do**
      $\hat{s}_{uv} \leftarrow s_{uv}/(\|\mathbf{h}_{\text{ECG}_u}\|\|\mathbf{h}_{\text{ECG}_v}\|)$ /* Compute pair-wise cosine similarities */
    end
    $\mathcal{N}_u^{\text{ECG}} \leftarrow \text{top-}k_{v \in V}\, \hat{s}_{uv}$ /* Compute k nearest neighbours of u */
  end
  $E^{\text{ECG}} \leftarrow \{(u, v) \mid u \in V \wedge v \in \mathcal{N}_u\}$ /* Construct the ECG edges */
  /* Step 3: Running ECG-GNN with parallel processing of G and G^{ECG} */
  for $u \in V$ **do**
    $\mathbf{h}_u^0 \leftarrow \mathbf{x}_u$ /* Setting initial node features */
  end
  for $l \leftarrow 1$ **to** L **do**
    /* Message passing propagation with the two parallel processors on G and G^{ECG} respectively */
    $E_{(l)}^{\text{ECG}} \leftarrow \text{DropEdge}(E^{\text{ECG}}, p_{de})$ /* Randomly drop edges in the ECG graph */
    for $u \in V$ **do**
      $\mathbf{h}_{\text{INP}_u}^{(l)} \leftarrow \phi_{\text{INP}}^{(l)}\left(\mathbf{h}_u^{(l-1)}, \bigoplus_{(u,v)\in E} \psi_{\text{INP}}^{(l)}\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}\right)\right)$ /* GNN on G */
      $\mathbf{h}_{\text{ECG}_u}^{(l)} \leftarrow \phi_{\text{ECG}}^{(l)}\left(\mathbf{h}_u^{(l-1)}, \bigoplus_{(u,v)\in E_{(l)}^{\text{ECG}}} \psi_{\text{ECG}}^{(l)}\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}\right)\right)$ /* GNN on G^{ECG} */
      $\mathbf{h}_u^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{h}_{\text{INP}_u}^{(l)} + \mathbf{U}^{(l)}\mathbf{h}_{\text{ECG}_u}^{(l)}$ /* Updating the node representation */
    end
  end
  /* Predict node labels */
  for $u \in V$ **do**
    $p_u \leftarrow \text{softmax}(\mathbf{W}^{(c)}\mathbf{h}_u^{(L)})$ $\hat{y}_u \leftarrow \arg\max_{c \in C} p_c$ /* Predicted class label */
  end
  return $\hat{\mathbf{y}}$
end