A Hyperparameters for PNA dataset.

In this section we provide the hyperparameters used for the different models on the PNA multitask benchmark. We train all models for 2000 steps and with 3 layers. The remaining hyperparameters for hidden size of each layer, learning rate, number of message passing steps (only valid for MPNN models), number of rotation matrices and same example frequency (when relevant) are provided in Table 6.

Table 6: Training hyperparameters for PNA dataset

Model	#Hidden size	L. rate	#MP steps	#Rotation matrices	#Same Examples
GAT	64	10^{-4}	-	-	-
GCN	64	10^{-4}	-	-	-
DGN	256	10^{-3}	-	-	-
MPNN	256	10^{-3}	2	-	-
ER GNN	128	10^{-3}	2	-	-
ER (node) embed.	64	10^{-3}	1	-	-
ER (edge) embed.	256	10^{-3}	2	-	-
ER (edge) embed.	256	10^{-3}	2	-	-
All ER features	256	10^{-4}	2	23	9
HT + ER (rand rot)	512	10^{-4}	2	23	4

B Details of MPNN Framework

As discussed previously, the architectures used in the experiments conform to the MPNN framework, which allows affinity measures to be added as additional node and edge features. We describe the details here for completeness.

Assume that our input graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, has node features $\mathbf{x}_u \in \mathbb{R}^n$, edge features $\mathbf{x}_{uv} \in \mathbb{R}^m$ and graph-level features $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^l$, for nodes $u, v \in \mathcal{V}$ and edges $(u, v) \in \mathcal{E}$. We provide encoders $f_n : \mathbb{R}^n \to \mathbb{R}^k$, $f_e : \mathbb{R}^m \to \mathbb{R}^k$ and $f_g : \mathbb{R}^l \to \mathbb{R}^k$ that transform these inputs into a latent space:

$$\mathbf{h}_u^{(0)} = f_n(\mathbf{x}_u) \qquad \mathbf{h}_{uv}^{(0)} = f_e(\mathbf{x}_{uv}) \qquad \mathbf{h}_{\mathcal{G}}^{(0)} = f_g(\mathbf{x}_{\mathcal{G}}) \tag{3}$$

Our *MPNN* then performs several message passing steps:

$$\mathbf{H}^{(t+1)} = P_{t+1}(\mathbf{H}^{(t)}) \tag{4}$$

where $\mathbf{H}^{(t)} = \left(\left\{ \mathbf{h}_u^{(t)} \right\}_{u \in \mathcal{V}}, \left\{ \mathbf{h}_{uv}^{(t)} \right\}_{(u,v) \in \mathcal{E}}, \mathbf{h}_{\mathcal{G}}^{(t)} \right)$ contains all of the latents at a particular processing step $t \geq 0$.

This process is iterated for T steps, recovering final latents $\mathbf{H}^{(T)}$. These can then be *decoded* into node-, edge-, and graph-level predictions (as required), using analogous decoder functions g_n, g_e and g_g:

$$\mathbf{y}_u = g_n(\mathbf{h}_u^{(T)}) \qquad \mathbf{y}_{uv} = g_e(\mathbf{h}_{uv}^{(T)}) \qquad \mathbf{y}_{\mathcal{G}} = g_g(\mathbf{h}_{\mathcal{G}}^{(T)}) \tag{5}$$

Generally, f and g are simple MLPs, whereas we use the MPNN update rule for P. It computes message vectors, $\mathbf{m}_{uv}^{(t)}$, to be sent across the edge (u, v), and then aggregates them in the receiver nodes as follows:

$$\mathbf{m}_{uv}^{(t+1)} = \psi_{t+1}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{h}_{uv}^{(0)}\right), \quad \mathbf{h}_u^{(t+1)} = \phi_{t+1}\left(\mathbf{h}_u^{(t)}, \sum_{u \in \mathcal{N}_v} \mathbf{m}_{vu}^{(t+1)}\right) \tag{6}$$

The message function ψ_{t+1} and the update function ϕ_{t+1} are both MLPs. All of our models have been implemented using the jraph library [16].

We incorporate edge-based affinity features (e.g., effective resistances and hitting times) in f_e and node-based affinity features (e.g., resistive embeddings) in f_n. Note that node-based affinity features may also naturally be incorporated as edge features by concatenating the node features at the endpoints.

Occasionally, the dataset in question will be easy to overfit with the most general form of message function (see (6)). In these cases, we resort to assuming that ψ factorises into an *attention mechanism*:

$$\mathbf{m}_{uv}^{(t+1)} = a_{t+1}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{h}_{uv}^{(0)}\right)\psi_{t+1}\left(\mathbf{h}_u^{(t)}\right) \tag{7}$$

where the attention function a is scalar-valued. We will refer to this particular MPNN baseline as a graph attention network (*GAT*) [44].

C Omitted Proofs

Lemma 3.2. *For any pair of nodes u, v, we have $\|\mathbf{r}_u - \mathbf{r}_v\|_2^2 = \mathsf{Res}(u, v)$.*

Proof.

$$\begin{aligned}
\|\mathbf{r}_u - \mathbf{r}_v\|_2^2 &= \|C^{1/2}BL_G^{-1}(\mathbf{1}_u - \mathbf{1}_v)\|_2^2 \\
&= (\mathbf{1}_u - \mathbf{1}_v)^T L^\dagger (B^T C B) L^\dagger (\mathbf{1}_u - \mathbf{1}_v) \\
&= (\mathbf{1}_u - \mathbf{1}_v)^T L^\dagger L L^\dagger (\mathbf{1}_u - \mathbf{1}_v) \\
&= (\mathbf{1}_u - \mathbf{1}_v)^T L^\dagger (\mathbf{1}_u - \mathbf{1}_v) = \mathsf{Res}(u, v).
\end{aligned}$$

□

Corollary 4.2. *For any fixed vectors $\alpha, \beta \in \mathbb{R}^n$, if we let $X := \sum_i \alpha_i x_i$, $\widehat{X} := \sum_i \alpha_i \widehat{x}_i$ and similarly $Y := \sum_i \beta_i x_i$, $\widehat{Y} := \sum_i \beta_i \widehat{x}_i$; then:*

$$\left|\langle X, Y\rangle - \langle \widehat{X}, \widehat{Y}\rangle\right| \le \frac{\epsilon}{2}\left(\|X\|^2 + \|Y\|^2\right).$$

Proof. Since $\langle X, Y\rangle = \frac{1}{4}\left(\|X+Y\|^2 - \|X-Y\|^2\right)$, we can bound $A = \left|\langle X, Y\rangle - \langle \widehat{X}, \widehat{Y}\rangle\right|$ from above as:

$$\begin{aligned}
A &= \left|\frac{1}{4}\left(\|X+Y\|^2 - \|\widehat{X}+\widehat{Y}\|^2 - \|X-Y\|^2 + \|\widehat{X}-\widehat{Y}\|^2\right)\right| \\
&\le \frac{1}{4}\left(\left|\|\widehat{X}+\widehat{Y}\|^2 - \|X+Y\|^2\right| + \left|\|\widehat{X}-\widehat{Y}\|^2 - \|X-Y\|^2\right|\right) \\
&\le \frac{1}{4}\left(\epsilon \cdot \|X+Y\|^2 + \epsilon \cdot \|X-Y\|^2\right) \qquad (8) \\
&= \frac{\epsilon}{4}\left(\|X+Y\|^2 + \|X-Y\|^2\right) \\
&= \frac{\epsilon}{2}\left(\|X\|^2 + \|Y\|^2\right),
\end{aligned}$$

where (8) follows from Lemma 4.1 with probability $1 - o(1)$, by our choice of k (as Lemma 4.1 guarantees that each of $\|\widehat{X}+\widehat{Y}\|^2 = (1 \pm \epsilon)\|X+Y\|^2$ and $\|\widehat{X}-\widehat{Y}\|^2 = (1 \pm \epsilon)\|X-Y\|^2$ holds with probability $1 - o(1)$, and one can take a union bound over the two events). □

Lemma 4.3. *$H_{u,v} = 2M\langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{r}_v - \mathbf{p}\rangle$ where $\mathbf{p} := \sum_u \pi_u \mathbf{r}_u$.*

Proof. Consider the following expression of hitting times in terms of commute times by [43].

$$H_{u,v} = \frac{1}{2}\left[K_{u,v} + \sum_i \pi_i \left(K_{v,i} - K_{u,i}\right)\right]. \tag{9}$$

Dividing both sides of eq. (9) and using the relation $K_{u,v} = 2M\mathsf{Res}(u, v)$, we see that:

$$\begin{aligned}
\frac{H_{u,v}}{2M} &= \frac{1}{2}\left[\mathsf{Res}(u, v) + \sum_i \pi_i \left(\mathsf{Res}(v, i) - \mathsf{Res}(u, i)\right)\right] \\
&= \frac{1}{2}\left[\|\mathbf{r}_u - \mathbf{r}_v\|^2 + \sum_i \pi_i \left(\|\mathbf{r}_v - \mathbf{r}_i\|^2 - \|\mathbf{r}_u - \mathbf{r}_i\|^2\right)\right]. \qquad (10)
\end{aligned}$$

Let's focus on the inner summation. After expanding out the squared norms, we see that:

$$\begin{aligned}\sum_i \pi_i \left(\|\mathbf{r}_v - \mathbf{r}_i\|^2 - \|\mathbf{r}_u - \mathbf{r}_i\|^2\right) &= \sum_i \pi_i \left(\|\mathbf{r}_v\|^2 - \|\mathbf{r}_u\|^2\right) \\ &\quad - 2\sum_i \pi_i \langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{r}_i \rangle \\ &= \left(\|\mathbf{r}_v\|^2 - \|\mathbf{r}_u\|^2\right) \\ &\quad - 2\langle \mathbf{r}_v - \mathbf{r}_u, \sum_i \pi_i \mathbf{r}_i \rangle \\ &= \left(\|\mathbf{r}_v\|^2 - \|\mathbf{r}_u\|^2\right) - 2\langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{p} \rangle.\end{aligned}$$

Substituting this back into eq. (10), we can express $\frac{1}{2M} H_{u,v}$ as:

$$\begin{aligned}&\frac{1}{2}\left(\|\mathbf{r}_v - \mathbf{r}_u\|^2 + \|\mathbf{r}_v\|^2 - \|\mathbf{r}_u\|^2 - 2\langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{p}\rangle\right) \\ &\qquad = \|\mathbf{r}_v\|^2 - \langle \mathbf{r}_u, \mathbf{r}_v \rangle - \langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{p} \rangle = \langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{r}_v - \mathbf{p} \rangle. \qquad \square\end{aligned}$$

Lemma 4.4. $|\widehat{H}_{u,v} - H_{u,v}| \le 3\epsilon H_{\max}$.

Proof. Using Lemma 4.3, we see that

$$\begin{aligned}|\widehat{H}_{u,v} - H_{u,v}| &= 2M\,|\langle \widehat{\mathbf{r}}_v - \widehat{\mathbf{r}}_u, \widehat{\mathbf{r}}_v - \widehat{\mathbf{p}} \rangle - \langle \mathbf{r}_v - \mathbf{r}_u, \mathbf{r}_v - \mathbf{p} \rangle| \\ &\le \epsilon M \left(\|\mathbf{r}_v - \mathbf{r}_u\|^2 + \|\mathbf{r}_v - \mathbf{p}\|^2\right) \\ &\le 3\epsilon H_{\max},\end{aligned}$$

where we used Corollary 4.2 in the first inequality and Definition 3.4 in the last inequality. □

D Comparison: Effective Resistances vs. Shortest Path Distances

Given that effective resistance (ER) captures times associated with random walks in a graph, it is tempting to ask how effective resistances compare to shortest path distances (SPDs) between nodes in a graph. Indeed, for some simple graphs, e.g., trees, shortest path distances and effective resistances turn out to be identical. However, in general, effective resistances and shortest path distances behave quite differently.

Nevertheless, it is tempting to ask how effective resistance features compare to SPD features in GNNs, especially as there have been a number of recent model architectures that make use of SPD features (e.g., Graphormer [49], Position-Aware GNNs [52], DE-GNN [29]). We first note that the most natural direct comparison of our ER-based MPNNs with SPD-based networks does not quite make sense. The reason is that the analogous comparison would be to determine the effect of replace ERs with SPDs as features in our MPNNs. However, since our networks only use ER features along edges of the given graph, the corresponding SPD features would then be trivial (as the SPD between two nodes directly connected by an edge in the graph is 1, resulting in a constant feature on every edge)!

As a result, graph learning architectures that use SPDs typically either (a.) use a densely-connected network (e.g., Graphormer [49], which uses a densely-connected attention mechanism) that incurs $O(n^2)$ overhead, or (b.) pick a small set of *anchor nodes* or *landmark nodes* to which SPDs from all other nodes are computed and incorporated as node features (e.g., Position-Aware GNNs [52], DE-GNN [29]). We stress that the former approach generally modifies the graph (by connecting all pairs of nodes) and therefore does not fall within the standard MPNN approach, while the latter includes architectures that fall within the MPNN paradigm.

Furthermore, we note that DE-GNNs are arguably one of the closest proposals to ours, as they compute distance-encoded features. These features can be at least as powerful as our proposed affinity-based features *if* polynomially many powers of the adjacency matrix are used. However, for all but the smallest graphs, using this many powers will be impractical—in fact, [29] only use powers of A up to 3, which would not be able to reliably approximate affinity-based features. We also observe that the DE-GNN paper is concerned with learning representations of small sets of nodes (e.g., node-, link-, and triangle-prediction) and does not show how to handle graph prediction tasks, which the authors mention as possible future work. This makes a direct comparison of our methods with DE-GNNs difficult.

D.1 Empirical Results

In an effort to empirically compare the expressivity of ER features with that of SPD features, we once again perform experiments on the PNA dataset, picking the following baselines that make use of SPD features:

- The first baseline is roughly an MPNN with *Graphormer-based features*. More precisely, it is a densely-connected MPNN with SPDs *from the original graph* as edge features. In order to retain the structure of the original graph, we also use additional edge features to indicate whether or not an edge in the dense (complete) graph is a true edge of the original graph. We also explore the use of the *centrality encoding* (in-degree and out-degree embeddings) from Graphormer as additional node features.
- The second baseline is the Position-Aware GNN (P-GNN), which makes use of "anchor sets" of nodes and encodes distances to these nodes.

The results of these baselines are shown in Table 7. In particular, we note that our ER-based MPNNs outperform all aforementioned baselines.

Table 7: Results on the PNA dataset for MPNNs with Graphormer-based features (yellow) as well as SPD-based P-GNNs (orange). Here, CE refers to the *centrality encoding*, which is incorporated in the relevant MPNNs as additional node features. Similarly, SPD refers to *shortest path distance* features — in the relevant MPNNs, shortest path distances between all pairs of nodes in the graph are incorporated as edge features, along with an additional edge feature indicating whether an edge exists in the input graph. Therefore, the MPNN baselines are all variants of the same model with additional node/edge features. Similarly, P-GNN [52] uses SPD features with respect to a set of chosen *anchor nodes*. The average score metric is, as before, the average of the $\log(MSE)$ metric over all six tasks, as in Table 1.

Model	**Average score**
*MPNN + CE	-2.728
*MPNN (dense) + SPD	-2.157
*MPNN (dense) + CE + SPD	-2.107
*P-GNN	-2.650
MPNN w/ resistive (edge) embeddings	**-2.789**
MPNN w/ all affinity measure features	**-3.106**

D.2 Theory: ER vs. SPD

In addition to experimental results, we would like to provide some theory for why effective resistances can capture structure in GNNs that SPDs are unable to.

We will call an initialization function $u \mapsto \mathbf{h}_u^{(0)}$ on nodes of a graph *node-based* if it assigns values that are independent of the edges of the graph. Such an initialization is, however, allowed to depend on node identities (e.g., for the single-source shortest path problem from a source s, one might find it natural to define $\mathbf{h}_s^{(0)} = 0$ and $\mathbf{h}_u^{(0)} = +\infty$ for all $u \neq s$).

Consider the task of computing "single-source effective resistances," i.e., the effective resistance from a particular node to every other node. We show that a GNN with a limited number of message passing steps cannot possibly learn single-source effective resistances, even to nearby nodes.

Theorem D.1. *Suppose we fix $k > 0$. Then, given any node-based initialization function $\mathbf{h}_u^{(0)}$, it is impossible for a GNN to compute single-source effective resistances from a given node w to any nodes within a k-hop neighborhood.*

More specifically, for any update rule

$$\begin{aligned}\mathbf{m}_{uv}^{(t+1)} &= \psi_{t+1}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, f_e(\mathbf{x}_{uv})\right) \\ \mathbf{h}_u^{(t+1)} &= \phi_{t+1}\left(\mathbf{h}_u^{(t)}, f\left(\{\mathbf{m}_{uv} : v \in \mathcal{N}(u)\}\right)\right),\end{aligned} \tag{11}$$

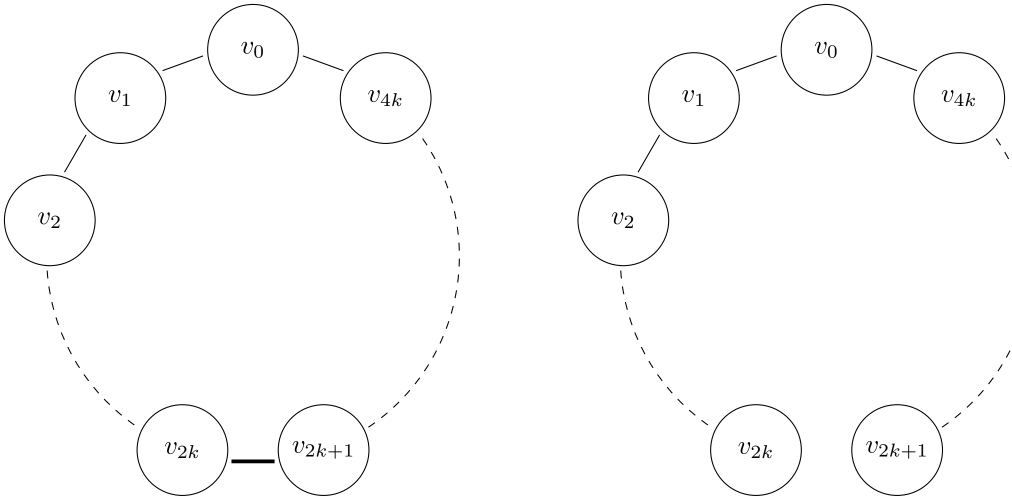
there exists a graph $G = (V, E)$ and $u \in V$ such that after k rounds of message passing, $h_v^{(k)} \neq \mathsf{Res}(u, v)$ for some $v \neq u$ within a k-hop neighborhood of u.

On the other hand, there exists an initialization with respect to which k rounds of message passing will compute the correct shortest path distances to all nodes within k-hop neighborhood.

Note that the assumption on the initialization function in the above theorem is reasonable because enabling the use of arbitrary, unrestricted functions would allow for the possibility of precomputing effective resistances in the graph and trivially incorporating them as node features, which would defeat the purpose of computing them using message-passing.

Proof. Consider the following set of graphs, each on $4k+1$ nodes:

Figure 2: Both of the above graphs are on $4k+1$ vertices, labeled $v_0, v_1, \ldots, v_{4k}$. The only difference is a single edge, i.e., the graph on the left has an edge between v_{2k} and v_{2k+1}, while the one on the right does not have this edge.



Let $V = \{v_0, v_1, \ldots, v_{4k}\}$. The first graph $G = (V, E)$ is a cycle, while the second graph $G' = (V, E')$ is a path, obtained by removing a single edge from the first graph (namely, the one between v_k and v_{k+1}). Suppose the edge weights are all 1 in the above graphs.

Let $w = v_0$ be the source and let $\{\mathbf{h}_v^{(0)} : v \in V\}$ be a "local" node feature initialization. Note that for any GNN (i.e., update and aggregation rules in (11), add the formal update rule somewhere), the computation tree after k rounds of message passing is identical for nodes $v_0, v_1, \ldots, v_k, v_{3k+1}, v_{3k+2}, \ldots, v_{4k}$ (i.e., the nodes within the k-hop neighborhood of v_0) in both G and G'. This is because the only difference between G and G' is the existence of the edge between v_{2k} and v_{2k+1}, and this edge is beyond a k-hop neighborhood centered at any one of the aforementioned nodes. Therefore, we will necessarily have that $\mathbf{h}_{v_i}^{(k)}$ is identical in both G and G' for $i = 1, \ldots, k, 3k+1, 3k+2, \ldots, 4k$.

However, it is easy to calculate the effective resistances in both graphs. In G, we have $\mathsf{Res}_G(v_0, v_i) = \frac{i(4k+1-i)}{4k+1}$, while in G', we have $\mathsf{Res}_{G'}(v_0, v_i) = \min\{i, 4k+1-i\}$. Therefore, $\mathsf{Res}_G(v_0, v_i) \neq \mathsf{Res}_{G'}(v_0, v_i)$ for all $i = 1, 2, \ldots, k, 3k+1, 3k+2, \ldots, 4k$.

It follows that for any $i = 1, 2, \ldots, k, 3k+1, 3k+2, \ldots, 4k$, the execution of k message passing steps of a GNN cannot result in $\mathbf{h}_{v_i}^{(k)} = \mathsf{Res}(v_0, v_i)$ for both G and G', which proves the first claim of the theorem.

For the second part (regarding single-source shortest paths), observe that single-source shortest path distances can, indeed, be realized via aggregation and update rules for a message passing network. In particular, for k rounds of message passing, it is possible to learn shortest path distances of all nodes within a k-hop neighborhood. Specifically, for a source w, we can use the following setup: Take $\mathbf{h}_w = 0$ and $\mathbf{h}_u = \infty$ for all $u \neq w$. Moreover, for any edge (u, v), let the edge feature $\mathbf{x}_{uv} \in \mathbb{R}$ simply be the weight of (u, v) in the graph. Then, take the update rule (11) with f_e, ψ_{t+1} as identity functions and

$$\begin{aligned} f_e(\mathbf{x}_{uv}) &= \mathbf{x}_{uv} \\ \psi_{t+1}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, f_e(\mathbf{x}_{uv})\right) &= \mathbf{h}_u^{(t)} + \mathbf{x}_{uv} \\ f(S) &= \min_S\{s \in S\} \\ \phi_{t+1}(a, b) &= \min\{a, b\}. \end{aligned}$$

It is clear that the above update rule simply simulates the execution of an iteration of the Bellman-Ford algorithm. Therefore, k message passing steps will simulate k iterations of Bellman-Ford, resulting in correct shortest path distances from the source w for every node within a k-hop neighborhood. $\square$