Adaptive Message Passing: A General Framework to Mitigate Oversmoothing, Oversquashing, and Underreaching

Federico Errica [1] **Henrik Christiansen** [1] **Viktor Zaverkin** [1]
Takashi Maruyama [1] **Mathias Niepert** [1 2] **Francesco Alesiani** [1]

Abstract

Long-range interactions are essential for the correct description of complex systems in many scientific fields. The price to pay for including them in the calculations, however, is a dramatic increase in the overall computational costs. Recently, deep graph networks have been employed as efficient, data-driven models for predicting properties of complex systems represented as graphs. These models rely on a message passing strategy that should, in principle, capture long-range information without explicitly modeling the corresponding interactions. In practice, most deep graph networks cannot really model long-range dependencies due to the intrinsic limitations of (synchronous) message passing, namely oversmoothing, oversquashing, and underreaching. This work proposes a general framework that *learns to mitigate* these limitations: within a variational inference framework, we endow message passing architectures with the ability to adapt their depth and filter messages along the way. With theoretical and empirical arguments, we show that this strategy better captures long-range interactions, by competing with the state of the art on five node and graph prediction datasets.

1. Introduction

Complex systems, characterized by interacting entities and emergent behavior, are a cornerstone of research in many scientific disciplines. Mathematical models of such systems should consider the effects of both short and long-range interactions between entities, and the latter are often crucial to describe the system's behavior with the highest degree of precision. For instance, in computational physics, it is well-known that electrostatic and gravitational interactions decay slowly with distance (Campa et al., 2014); in computational chemistry and material sciences, the accurate modeling of non-local effects, such as non-bonded interactions in molecular systems, is necessary to estimate properties like the free energy (Shirts et al., 2007; Piana et al., 2012); in biology, disrupting long-range interactions in mRNA can inhibit slicing (Rüegsegger et al., 2001); in immunology, the distant interactions between a major histocompatibility complex and regions of the T-cell receptor molecule correlate with their binding process (Ferber et al., 2012).

Complex systems can be represented as graphs of interacting entities. Modeling long-range interactions often implies that the graph has dense connectivity, meaning the number of interactions is quadratic in the number of entities. Machine learning researchers tried to address these problems by relying on accurate surrogates for computationally demanding simulations (Sanchez-Gonzalez et al., 2020). Some of these methods rely on Deep Graph Networks (DGNs) (Bacciu et al., 2020b), deep learning models implement a *message passing* paradigm of computation. In message passing, nodes repeatedly exchange messages with each other to propagate information across the graph and compute their embeddings. More rounds of message passing increase the "receptive field" of each node.

Despite its long-standing history (Sperduti & Starita, 1997; Micheli, 2009; Scarselli et al., 2009; Bacciu et al., 2020b), research in graph representation learning has gained more traction in recent years, and there are still many open questions. For instance, it is well-known that most message passing architectures are ineffective at capturing long-range dependencies, thus reducing their impact in the scientific fields mentioned before. Researchers relate this problem to at least three others, namely *oversmoothing* (Li et al., 2018), *oversquashing* (Alon & Yahav, 2021; Rusch et al., 2023), and *underreaching* (Alon & Yahav, 2021). Briefly, oversmoothing means that the node embeddings of a DGN tend to converge to the same value as the depth increases. In contrast, oversquashing relates to the bottleneck of compressing a (possibly) exponential amount of information from neighboring nodes into a single node embedding. Fi-

[1]NEC Laboratories Europe [2]University of Stuttgart. Correspondence to: Federico Errica <federico.errica@neclab.eu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

nally, underreaching refers to DGNs' inability to propagate a node's information to more than K hops away.

This work provides a general framework for improving the ability of *any* message passing architecture to capture long-range dependencies; we extend the general message passing formulation to propagate relevant information across the graph. At the heart of our proposal is the idea to let DGNs learn *how many layers* of message passing to use and *when to send* specific messages. As a matter of fact, one typically observes oversmoothing and oversquashing when too many messages are propagated, hence learning which messages to discard is important. At the same time, solving underreaching requires a sufficient number of message passing rounds to be performed, and it is crucial to learn this information from that task rather than guessing it via expensive grid searches. In light of these characteristics, we call our approach Adaptive Message Passing (AMP).

Our contributions are multi-faceted. We extend a recent variational framework for unbounded depth networks (Nazaret & Blei, 2022) to the processing of graphs, and we introduce new families of distributions with specific properties to overcome previous limitations. We also propose a soft message filtering scheme to prune irrelevant information for the task at hand and favor the propagation of messages to distant regions of the input graph. Theoretically, we show how to propagate a message unchanged between any two connected nodes in the graph and that filtering decreases the upper bound on sensitivity (Topping et al., 2022); thus, underreaching and oversquashing can be mitigated. Empirically, AMP significantly and consistently improves the performances of message passing architectures on five well-known node and graph prediction datasets where long-range information is important. Qualitative analyses provide further evidence that AMP mitigates oversmoothing and oversquashing as well as learning the number of message-passing layers. Finally, we conduct an in-depth study of our approach via ablations and visualizations of the models' predictions.

2. Related Work

Due to space reasons, we provide an extended literature review in Appendix A.

Oversquashing. There are many methods that attempt to address the oversquashing problem with the goal of better capturing long-range dependencies (Alon & Yahav, 2021). Some works learn how a node should completely stop propagating a message in a fixed-depth architecture (Spinelli et al., 2020) or if it should only listen, isolate, or receive/broadcast its own message (Finkelshtein et al., 2024). Similarly, one can learn to sample edges at each message passing layer according to some learned parametrization (Hasanzadeh et al., 2020) or have a completely asynchronous message passing (Faber & Wattenhofer, 2023). Ordinary differential equation-based message passing approaches, instead, provably preserve information regardless of the depth in the network (Gravina et al., 2023; Heilig et al., 2025). On the other hand, most graph rewiring methods alter the graph connectivity to increase the node sensitivity (Topping et al., 2022), which has been theoretically linked to a form of "topological oversquashing". Some of them preserve locality and sparsity of the rewiring process (Barbero et al., 2024) or perform dynamic rewiring (Gutteridge et al., 2023). Probabilistic approaches to rewiring are based on sampled sub-graphs (Qian et al., 2024a). Recently, a critical perspective on the effectiveness of rewiring approaches has also been given (Tortorella & Micheli, 2022).

Oversmoothing. One practical mitigation to the oversmoothing problem is dropping edges, reducing the overall flow of messages and avoiding the convergence of all embeddings to the same value (Rong et al., 2020). Another possibility is to employ skip/residual connections (Kipf & Welling, 2017; Li et al., 2019). The concatenation of node representations across layers is yet another a way to contain oversmoothing, which has been adopted in neural and probabilistic models to improve the downstream performances on several node and graph-related tasks (Bacciu et al., 2018; Xu et al., 2018). Instead, an orthogonal research direction considers implicit neural networks for graphs that correspond to infinite-depth models and seem to be able to capture long-range dependencies (Poli et al., 2021; Liu et al., 2021). These models simulate synchronous message passing with a potentially infinite number of message-propagation steps, and some of them appear to be empirically robust to the oversmoothing problem.

Adaptive Architectures. This family of methods tries to learn the models' architecture during training. Our work is inspired by the unbounded depth networks (UDNs) of Nazaret & Blei (2022), who proposed a variational framework for learning the depth in multi-layer perceptrons and convolutional neural networks. In the graph domain, the first approach to learning the depth of a DGN was proposed by Micheli (2009), who applied the cascade correlation algorithm (Fahlman & Lebiere, 1989). Other works attempted to learn the "width" of each message passing layer by exploiting Bayesian non-parametric models (Castellana et al., 2022), which allows to save time and memory when building deeper probabilistic DGNs. Finally, it is important to notice that these works, including this manuscript, are all orthogonal to the popular field of neural architecture search (Zhou et al., 2022): The former attempts at dynamically modifying the architecture *during learning*, whereas most neural architecture search approaches find smarter ways to carry out a grid search or assume a maximum number of layers. An advantage of adaptive approaches is that they can greatly reduce time and computational costs to perform

a hyper-parameter search.

Contrary to all these approaches, AMP uses a variational framework to jointly learn both the depth of the DGN and filter messages passed between nodes in each of these layers.

3. Adaptive Message Passing

This section introduces the probabilistic framework of AMP, which we train with simple backpropagation to optimize a variational lower bound. This bound is composed of a predictive term and two (optional) regularizers over the parameters and the DGN's depth.

Definitions. We consider directed attributed graphs $g = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A})$, each consisting of a set of nodes $\mathcal{V} = \{1, \dots, n_g\}$ that are connected together via a set of oriented edges $\mathcal{E} = \{(u, v) | u, v \in \mathcal{V}\}$. When a graph is undirected, each edge is converted into two oriented ones, that is, (u, v) and (v, u). The set $\mathcal{X} = \{\mathbf{x}_v \in \mathbb{R}^d | v \in \mathcal{V}\}$ defines the d-dimensional attribute vector of each node in the graph, and similarly for the d'-dimensional edge attributes belonging to the set $\mathcal{A} = \{\boldsymbol{a}_{uv} \in \mathbb{R}^{d'} | (u, v) \in \mathcal{E}\}$. Finally, we define the neighborhood of a node v as the set of incoming edges $\mathcal{N}_v = \{u | (u, v) \in \mathcal{E}\}$. As outlined in previous works (Hammer et al., 2005; Bongini et al., 2018), each attributed graph can be seen as a realization of *some* random variable (*r.v.*) $\mathcal{G}$ with support in the graph domain. Similarly to classical machine learning, we do not have access to the data distribution $p(\mathcal{G})$, rather we are interested in modeling the conditional distribution $p(\mathcal{T} = Y_g | \mathcal{G} = g)$, where Y_g stands for the target value(s) to be predicted.

Multi-output Family of Architectures. AMP produces deep graph networks of *potentially infinite depth*, where **each layer** ℓ comprises a message passing operation MP **and** a readout mapping $\mathcal{R}$ (Bacciu et al., 2020b) from node embeddings to the desired output.

Without loss of generality, a message passing layer MP_ℓ can compute node embeddings $\boldsymbol{h}_v^\ell, \forall v \in \mathcal{V}$ as follows:

$$\boldsymbol{h}_v^\ell = \phi^\ell \left(\boldsymbol{h}_v^{\ell-1}, \ \Psi(\{\psi^\ell(\boldsymbol{h}_u^{\ell-1}, \boldsymbol{a}_{uv}) | u \in \mathcal{N}_v\}) \right), \quad (1)$$

where ϕ^ℓ and ψ^ℓ are learnable functions and Ψ is a permutation invariant function that aggregates the embeddings of v's neighbors computed at the previous layer. When $\ell = 1$, $\boldsymbol{h}_v^1$ is obtained by applying a learnable transformation of the node v's features $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$, and no neighbor aggregation is performed. Instead, the readout mapping $\mathcal{R}^\ell$ depends on the task: If one needs node-wise predictions, then the readout implements a learnable map $\hat{\boldsymbol{y}}_v^\ell = \rho^\ell(\boldsymbol{h}_v^\ell)$ from $\boldsymbol{h}_v^\ell$ to a node output $\hat{\boldsymbol{y}}_v^\ell$; On the other hand, in the case of whole-graph predictions, a global aggregation has to be performed first:

$$\hat{\boldsymbol{y}}^\ell = \rho_2^\ell \left(\Phi \left(\{\rho_1^\ell(\boldsymbol{h}_v^\ell) | v \in \mathcal{V}\} \right) \right), \quad (2)$$

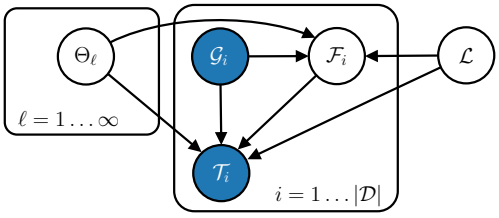

Figure 1. The graphical model of AMP, where white and blue circles denote, respectively, latent and observable random variables. Θ_ℓ is the *r.v.* over the parameters of layer k, $\mathcal{F}_i$ defines a distribution over the message filters, $\mathcal{L}$ implements a distribution over the layers of the architecture, while $\mathcal{G}_i$ and $\mathcal{T}_i$ are distributions over the (observable) input graph and the target label, respectively.

where ρ_1, ρ_2 denote learnable functions and Φ is a global pooling function that aggregates all node representations computed at a given layer ℓ. The learnable functions $\phi^\ell, \psi^\ell, \rho_1^\ell, \rho_2^\ell$ are typically implemented as 1-hidden layer MLPs parametrized by Θ_ℓ.

Variational Inference Given a dataset $\mathcal{D}$ of $|\mathcal{D}|$ *i.i.d.* graphs, we seek to maximize the log-likelihood

$$\ln p(Y|G) = \ln \prod_i^{|\mathcal{D}|} p(Y_{g_i} | g_i) = \sum_i^{|\mathcal{D}|} \ln p(Y_{g_i} | g_i). \quad (3)$$

One usually assumes the existence of a set of latent variables Z such that $p(Y|G) = \int p(Y, Z|G) dZ$. By designing a proper graphical model, which encodes conditional independence assumptions, we define how to compute $p(Y, Z|G)$; however, the integral to maximize often remains intractable. Therefore, one approach is to turn to variational inference (Jordan et al., 1999; Blei et al., 2017), which maximizes the Expected Lower Bound (ELBO) instead. In particular, by arbitrarily defining a distribution $q(Z|G)$, one has

$$\ln p(Y|G) \geq \mathbb{E}_{q(Z|G)} \left[\ln \frac{p(Y, Z|G)}{q(Z|G)} \right]. \quad (4)$$

Below, we show how we define the joint distribution $p(Y, Z|G)$ as well as the variational distribution $q(Z|G)$ in the specific context of AMP.

AMP Formulation. Figure 1 represents the graphical model associated with AMP, where white and blue circles represent latent and observed *r.v.*s, respectively. We extend the formulation of Nazaret & Blei (2022) to the domain of graphs by modeling the message filtering strategy: The variable Θ_ℓ follows a distribution over the parameters of layer ℓ of an infinite-depth network, and $\mathcal{L}$ follows a distribution over layers $L \in \mathbb{N}^*$ and is used to truncate the network to a finite depth L. For the i-th graph g_i, $\mathcal{F}_i$ follows a distribution over soft message filters $\boldsymbol{F}_i \in [0, 1]^{|\mathcal{V}| \times L \times d}$. In particular, given a node v and a layer ℓ, the d-dimensional vector

$\boldsymbol{F}_i(v, \ell)$ specifies how much of h_v^ℓ has to be propagated through the outgoing edges in the next message passing layer. The generative model is

$$\boldsymbol{\theta} \sim p(\Theta) = \prod_{\ell=1}^{\infty} p(\Theta_\ell) \tag{5}$$

$$L \sim p(\mathcal{L}) \tag{6}$$

$$\boldsymbol{F}_i | g_i, L, \boldsymbol{\theta} \sim p(\mathcal{F}_i | g_i, L, \boldsymbol{\theta}) \tag{7}$$

$$Y_{g_i} | g_i, \boldsymbol{F}_i, L, \boldsymbol{\theta} \sim p(\mathcal{T}_i;\ \Omega_L(g_i, \boldsymbol{F}_i, \boldsymbol{\theta})), \tag{8}$$

with Ω_L being the infinite DGN truncated at depth L whose output parametrizes the target distribution. This means that the joint distribution decomposes as

$$p(Y, \boldsymbol{\theta}, \boldsymbol{F}_i, L | G) = \\ = p(\boldsymbol{\theta}) p(L) \prod_{i}^{|\mathcal{D}|} p(Y_{g_i} | \boldsymbol{\theta}, \boldsymbol{F}_i, L, g_i) p(\mathcal{F}_i | g_i, L, \boldsymbol{\theta}) \tag{9}$$

Note that the independence of the priors is key for an efficient approximation of the posterior distribution (Nazaret & Blei, 2022).

In Figure 2, we visually represent the effect that message filtering has on the propagation of messages across DGN layers. A graph of seven nodes (a) is provided and the message filtering scheme (b) has been discretized in the interest of simplicity. For instance, node 1 will send its message only at message passing layer 1, nodes 2 & 3 will never send a message, and node 4 will send a message only at layer 2. Compared to the standard message passing (c), where all nodes send their messages at each layer, AMP implements a learnable filtering (d), where a subset of all possible messages is propagated at each layer in a way that depends on the task to be solved. In Section D, we discuss the implications of this adaptive message filtering scheme in mitigating the well-known issues of oversquashing, underreaching, and oversmoothing. Notably, message filtering does not introduce a significant computational burden since it has linear complexity in the number of nodes.

Choice of the Variational Distributions. We now need to define the learnable variational distribution $q(\boldsymbol{\theta}, L, \boldsymbol{F}_i, | g_i, Y_{g_i})$. We assume it factorizes as $q(\boldsymbol{\theta}|L; \boldsymbol{\nu}) q(L; \boldsymbol{\lambda}) q(\boldsymbol{F}_i | g_i, L, \boldsymbol{\theta})$, where $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$ are learnable parameters. We also assume that the variational posterior does not depend on Y_{g_i} (so we can drop the term) to allow for predictions on unseen graphs. Below, we describe how to compute each factor so that the computation of the ELBO is tractable.

The distribution $q(L; \boldsymbol{\lambda})$ has to belong to an *unbounded with bounded and connected members'* family (see Definition B.1 in Section B). In short, since the support of each distribution q in the family is bounded, we can compute its expectation

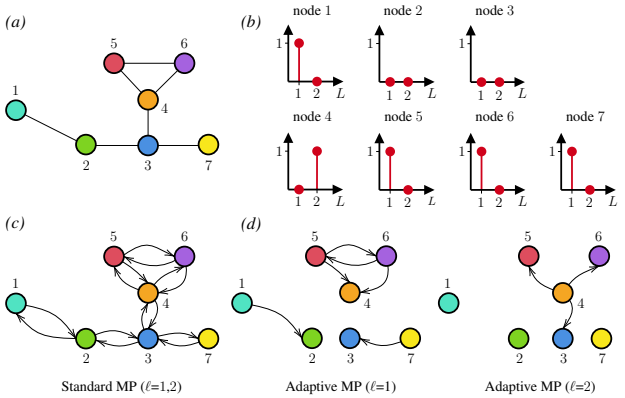

Figure 2. Given an input graph (a) and a discrete message filtering scheme (b), we observe how a L=2-layer standard message passing (c) differs from AMP (d) in terms of the number of messages sent. Please refer to the text for more details.

$\mathbb{E}_{q(L;\boldsymbol{\lambda})}[f(L)]$ as the sum $\sum_{\ell \in \text{support}(q)} q(\ell) f(\ell)$ for any function f. In Appendix B, we extend the original treatment of Poisson distributions to Gaussians and mixtures of distributions. Second, we define $q(\boldsymbol{\theta}|L; \boldsymbol{\nu})$ such that we cannot make any statement about the layers greater than L (Kurihara et al., 2007):

$$q(\boldsymbol{\theta}|L; \boldsymbol{\nu}) = q(\boldsymbol{\theta}_{1:L}; \boldsymbol{\nu}_{1:L}) \prod_{\ell=L+1}^{\infty} p(\boldsymbol{\theta}_\ell) \tag{10}$$

and $p(\boldsymbol{\theta}_\ell)$ can be, for instance, a Gaussian prior. We also fix $q(\boldsymbol{\theta}_{1:L}; \boldsymbol{\nu}_{1:L}) = \prod_{\ell=1}^{L} \mathcal{N}(\boldsymbol{\theta}_\ell; \boldsymbol{\nu}_\ell, \boldsymbol{I})$.

Finally, we define $q(\boldsymbol{F}_i | g_i, L, \boldsymbol{\theta})$ as a Dirac delta function $\delta_{\boldsymbol{F}_i}$ whose parameters $\boldsymbol{F}_i \in [0, 1]^{|\mathcal{V}| \times L \times d}$ are computed by a function $f(g_i)$. Choosing the delta function makes the computation of its expectation straightforward, but other choices can, in principle, be made. We propose two versions of the function $f(g_i)$, whose choice is left as a hyperparameter: the first computes $\boldsymbol{F}_i(v, \ell) = f_\ell(\boldsymbol{x}_v)$ and the second computes $\boldsymbol{F}_i(v, \ell) = f_\ell(\boldsymbol{h}_v^\ell)$, where f_ℓ is a Multi-Layer Perceptron (MLP) with sigmoidal activations. In other words, a node's outgoing messages will be filtered according to either the input features of that node or its embedding at layer ℓ. Given $\boldsymbol{F}_i \sim q(\boldsymbol{F}_i | g_i, L, \boldsymbol{\theta})$, we extend Equation 1 to apply such filtering:

$$\boldsymbol{h}_v^\ell = \phi^\ell(\boldsymbol{h}_v^{\ell-1}, \\ \Psi(\{\boldsymbol{F}_i(u, \ell-1) \odot \psi^\ell(\boldsymbol{h}_u^{\ell-1}, \boldsymbol{a}_{uv}) | u \in \mathcal{N}_v\})), \tag{11}$$

with $\odot$ being the element-wise product. Such message filtering is similar in spirit to many works (Franceschi et al., 2019; Spinelli et al., 2020; Finkelshtein et al., 2024), but our approach does not require gradient approximations caused by discrete operations and is fully differentiable.

Computation of the ELBO. Our choice of the variational distributions allows us to compute the ELBO efficiently and maximize it using backpropagation (Rumelhart et al., 1986). In particular, we write (the full derivation is in Appendix C)

$$\ln p(Y|G) \geq \sum_{\ell=1}^{\hat{L}} q(\ell;\boldsymbol{\lambda}) \left[\ln \frac{p(\ell)}{q(\ell;\boldsymbol{\lambda})} + \ln \frac{p(\boldsymbol{\nu})}{q(\boldsymbol{\nu}|\ell;\boldsymbol{\nu})} + \sum_{i}^{|\mathcal{D}|} \left[\ln \frac{p(\boldsymbol{F}_i)}{q(\boldsymbol{F}_i|g_i,\ell,\boldsymbol{\nu})} + \ln p(Y_{g_i}|\ell,\boldsymbol{F}_i,\boldsymbol{\nu},g_i) \right] \right], \quad (12)$$

where $\hat{L} = \text{support}(q(L))$, $p(L)$ is a prior over layers, such as a Poisson distribution, and $p(\boldsymbol{F}_i)$ is a prior over all possible message filtering schemes (*uninformative in this work*, so the term cancels). The second equivalence relies on the specific properties of the variational distributions and on the approximation of expectation $\mathbb{E}_{q(\boldsymbol{\theta}|L;\boldsymbol{\nu})}[f(\boldsymbol{\theta}_{1:L})])$, for a function f, at the first order[1] with $f(\mathbb{E}_{q(\boldsymbol{\theta}|L;\boldsymbol{\nu})}[\boldsymbol{\theta}_{1:L}]) = f(\boldsymbol{\nu}_{1:L})$ as in Nazaret & Blei (2022).

Akin to Nazaret & Blei (2022), AMP makes predictions about a new graph g_j as:

$$p(Y_{g_j}|g_j) \approx \mathbb{E}_{q(\boldsymbol{\theta},L,\boldsymbol{F}_j,|g_j)} \left[p(Y_{g_j};\Omega_L(g_j,\boldsymbol{F}_j,\boldsymbol{\theta})) \right] \quad (13)$$

$$\approx \sum_{\ell=1}^{\hat{L}} q(\ell;\boldsymbol{\lambda}) p(Y_{g_j};\Omega_\ell(g_j,\boldsymbol{F}_j,\boldsymbol{\nu})). \quad (14)$$

In other words, using the fact that $q(\ell;\boldsymbol{\lambda})$ has bounded support up to $\hat{L}$, we obtain the prediction as the weighted sum of the $\hat{L}$ output layers of the DGN, and the variational distribution $q(\ell;\boldsymbol{\lambda})$ over layers provides said weights.

We now show that there is a direct relation between the ability to filter out messages of Equation 11 and the upper bound on the Jacobian sensitivity discussed in Di Giovanni et al. (2023).

Theorem 3.1. *For* AMP *with m layers and $u, v \in \mathcal{V}$,*

$$\left\| \frac{\partial h_v^{(m)}}{\partial h_u^{(0)}} \right\|_{L^1} \leq d \left(\left(c_{\text{up}} \left(c_{\text{rs}} I + c_{\text{mp}} \left(c_F k_h + k_F \right) A \right) \right)^m \right)_{vu}$$

Here, MPNN is in the following form

$$h_v^\ell = \text{up}\left(\text{rs}(h_v^{\ell-1}) + \text{mp}(\sum_u A_{vu} F(h_u^{\ell-1}) \odot h_u^{\ell-1}) \right)$$

where up, rs, *and* mp *are Lipschitz functions as in Di Giovanni et al. (2023) with constants $c_{\text{up}}, c_{\text{rs}}, c_{\text{mp}}$, c_F is the upper bound of the entry-wise L^1 matrix norm of $\frac{\partial F}{\partial x}$ for the filtering function F, k_h is the maximal absolute value among entries of h, and similarly k_F for the output of F.*

Proof. The proof is provided in Appendix D. □

[1] First-order second-moment method (FOSM) of probability.

Note that $k_F \leq 1$; if we consider for simplicity a constant filtering function, namely $c_F = 0$ and we filter enough, meaning $k_F < 1$, then filtering will decrease the sensitivity's upper bound. At the same time, this helps to reduce the amount of messages that get squashed into a fixed size vector Alon & Yahav (2021), contradicting the widely accepted notion that "improving sensitivity mitigates oversquashing". Please consult Appendix D for a more detailed discussion on this matter.

Practical Considerations. The depth of AMP varies dynamically; in particular, the support of the distribution $q(\ell;\boldsymbol{\lambda})$ is obtained by truncating it as the quantile function evaluated at 0.99 (in our experiments). Whenever the quantile threshold shifts, we either grow or shrink the DGN by instantiating a new message-passing layer and increasing the output dimension of the function $f(g_i)$ that produces $\boldsymbol{F}_i$. When shrinking the DGN, we can retain the excess layers to account for future expansions or delete them; here we opt for the retention strategy. Importantly, the depth is not a hyper-parameter to be tuned anymore. While AMP requires choosing a family of truncated distributions $q(L)$ and a proper initialization, it is generally believed that this has a smaller effect on the final result (Goel & Degroot, 1981; Bernardo & Smith, 2009). Also, setting uninformative priors works well in our experiments but they are convenient way to penalize the computational costs of deeper networks.

We conclude with a theorem on AMP's ability to propagate a message unchanged from two connected nodes in a graph, which would not be possible on classical (synchronous) message-passing neural networks. However, achieving such behavior in practice might be difficult.

Theorem 3.2 (Short Version)**.** *For each graph g, a source node v and a destination node u, there exists a parametrization of AMP and a depth K such that $\boldsymbol{h}_u^K = \boldsymbol{h}_v^0 = \boldsymbol{x}_v$.*

Proof. The proof is provided in Appendix D, and Figure 8 sketches the process formalized in the theorem. □

3.1. Computational Considerations

The cost of filtering messages is $\mathcal{O}(|\mathcal{V}|)$. Therefore, the message passing operation is not altered significantly, since it has a cost of $\mathcal{O}(|\mathcal{V}|+|\mathcal{E}|)$. However, the additional burden introduced by AMP, compared to classical message-passing architectures, is the layer-wise readout that we implemented as an MLP. Classical MPNNs employ a single readout with cost $\mathcal{O}(|\mathcal{V}|)$ or $\mathcal{O}(1)$ depending on the task nature, whereas we use one per layer, so we have $\mathcal{O}(|\mathcal{V}|L)$ and $\mathcal{O}(|\mathcal{V}|L)$ respectively. In terms of training costs, we employ standard backpropagation with at most two light-weight additional regularizers, which is not so different from classical approaches.

4. Experimental Details

We evaluate AMP on two sets of tasks, both requiring the ability to capture long-range interactions.[2] Additional node classifcation results can be found in Appendix F.

Synthetic Datasets We consider the tasks of predicting the diameter, the single-source shortest paths (SSSP), and the node eccentricity on synthetic graphs (Corso et al., 2020). In particular, we closely follow the setup of Gravina et al. (2023) with graph sizes ranging from 25 to 35 nodes, topologies sampled from different graph generators, and each node has one random (sampled from a Normal distribution) feature attached. For SSSP, a binary feature is added to each node to indicate whether it is the source node in the graph or not. Each dataset amounts to 7040 graphs split into 5120 for training, 640 for validation, and 1280 for testing. The metric to be optimized is the $\log_{10}$ of the mean squared error (MSE). We apply early stopping on the validation MSE.
We have observed that the performance reported in Gravina et al. (2023) can be improved by a significant margin if we average results over 20 rather than four final (that is, after model selection) training runs and increase the patience of the early stopper from 100 to 300, giving models more time to converge to a good solution. Therefore, to ensure a more robust set of results, we re-evaluated all baselines[3] considering these changes, and in many cases, we improved the scores. We combine AMP with three message passing architectures, namely the Graph Convolutional Network (GCN) (Kipf & Welling, 2017), the Graph Isomorphism Network (GIN) (Xu et al., 2019), and the Anti-Symmetric DGN (ADGN) (Gravina et al., 2023), and in addition we compare against GAT (Velickovic et al., 2018), GraphSAGE (Hamilton et al., 2017), GCNII (Chen et al., 2020a), DGC (Wang et al., 2021), and GRAND (Chamberlain et al., 2021). Hyper-parameter details can be found in Appendix E.

Chemical Datasets We also test AMP on real-world chemical graph prediction benchmarks, taken from the Long Range Graph Benchmark, called *peptides-func* and *peptides-struct* (Dwivedi et al., 2022). The first is an imbalanced multi-label graph classification dataset with ten total peptide functions, and we measure performances using the average precision (AP). The second is a multi-label graph regression task where we want to predict the peptides' properties based on their 3D information, and one evaluates the mean absolute error (MAE). Both datasets contain 15535 peptides with approximately 150 nodes each, and the data is split into 70 % for training, 15 % for validation, and 15 % for testing. We apply early stopping on the validation MAE.
We rely on the fair re-evaluation of Tönshoff et al. (2023a) that shows how simple baselines like a GCN can achieve very competitive performances when properly tuned. In addition, we follow previous works (Rampášek et al., 2022; Tönshoff et al., 2023a) and add random-walk structural encodings for *peptides-func* and Laplacian positional encodings for *peptides-struct*. For completeness, we include results from Dwivedi et al. (2022), its re-evaluation (Tönshoff et al., 2023a), and other results such as CRaWL (Tönshoff et al., 2023b), DRew (Gutteridge et al., 2023), Exphormer (Shirzad et al., 2023), GRIT (Ma et al., 2023), Graph ViT and G-MLPMixer (He et al., 2023), LASER (Barbero et al., 2024), CO-GNN (Finkelshtein et al., 2024), NBA (Park et al., 2024), PH-DGN (Heilig et al., 2025), GRED (Ding et al., 2024), PR-MPNN (Qian et al., 2024a) and IPR-MPNN (Qian et al., 2024b).

We evaluate AMP on GCN, GINE (Hu et al., 2020), and GatedGCN (Bresson & Laurent, 2017); our grid search follows the best hyper-parameter reported by Tönshoff et al. (2023a) (except the depth). As above, details on AMP's hyper-parameters can be found in Appendix E. Because the optimal depth ultimately depends on the task and the specific configuration of the model, we cannot impose arbitrary restrictions on the number of total parameters as done in Dwivedi et al. (2022); instead, we are interested in letting the model freely adapt and choose the best parametrization that maximizes the performance.

5. Results

5.1. Quantitative Results

Table 1 reports the test scores for the Diameter, SSSP, and Eccentricity datasets for all baselines and AMP versions. On all datasets, AMP always grants a reduction of the test error, with an average improvement of 63 % on Diameter, 72 % on SSSP, and 32 % on Eccentricity. These results show that learning the proper depth of a network and a policy for filtering messages exchanged between nodes is more effective than relying on a manually crafted grid search and a fully synchronous message passing behavior. Eccentricity is the most difficult task to solve, whereas one could claim that SSSP is almost solved for the graphs considered. We achieved the greatest reduction in error with respect to the GIN model, probably because the authors in Gravina et al. (2023) found that a 1-layer GIN was the best configuration across all tasks after tuning the depth. This stresses the positive impact of letting the model learn how and when to propagate messages. On the chemical datasets (Table 2), we observe a similar trend. Regardless of the base message passing architecture, AMP consistently improves its performance on classification and regression tasks. On *peptides-func*, we achieve an improvement of 2 to 3.4 % compared to the base models and a reduction of MAE on *peptides-struct* that positions all AMP versions at state-of-the-art levels (considering overlap of standard deviations). Our analyses

[2] https://github.com/nec-research/Adaptive-Message-Passing
[3] We received support from the authors of paper.

	Diameter	Rel Imp	SSSP	Rel Imp	Eccentricity	Rel Imp
GCN	0.6146 ± 0.0375		0.9132 ± 0.0051		0.7398 ± 0.0705	
GAT	1.4367 ± 0.3558		0.6070 ± 0.0375		1.0714 ± 0.0616	
GRAPHSAGE	0.6146 ± 0.0744		-1.0139 ± 0.0120		1.0859 ± 0.0001	
GIN	0.2408 ± 0.0154		-0.2648 ± 0.4437		0.9229 ± 0.0002	
GCNII	0.5057 ± 0.0309		-0.9172 ± 0.4396		0.7112 ± 0.0255	
DGC	0.5601 ± 0.0220		-0.0254 ± 0.0077		0.8051 ± 0.0017	
GRAND	0.9477 ± 0.2160		0.1909 ± 0.3103		0.7450 ± 0.1369	
ADGN	-0.4530 ± 0.0883		-3.5448 ± 0.2749		0.0547 ± 0.0732	
AMP_{GCN}	$-0.1072^{\dagger} \pm 0.0791$	-81%	$0.5440^{\dagger} \pm 0.0108$	-57%	$0.6054^{\dagger} \pm 0.0919$	-26%
AMP_{GIN}	$-0.4874^{\dagger} \pm 0.1111$	-81%	$-3.0628^{\dagger} \pm 0.3159$	-99%	$0.4093^{\dagger} \pm 0.0546$	-69%
AMP_{ADGN}	$-0.5891^{\dagger} \pm 0.0720$	-27%	$-3.9579^{\dagger} \pm 0.0769$	-61%	$0.0515^{\dagger} \pm 0.1819$	-1%
Avg Rel Imp		-63%		-72%		-32%

Table 1. Mean $log_{10}(\text{MSE})$ and standard deviation averaged over 20 final runs on Diameter, SSSP, and Eccentricity. A † indicates that AMP yields an improvement in the mean score compared to the base model.

	Method	peptides-func Test AP ↑	peptides-struct Test MAE ↓
LRGB	GCN	0.5930 ± 0.0023	0.3496 ± 0.0013
	GINE	0.5498 ± 0.0079	0.3547 ± 0.0045
	GATEDGCN	0.6069 ± 0.0035	0.3357 ± 0.0006
	TRANSFORMER	0.6326 ± 0.0126	0.2529 ± 0.0016
	SAN	0.6439 ± 0.0075	0.2545 ± 0.0012
	GPS	0.6535 ± 0.0041	0.2500 ± 0.0005
RE-EVAL	GCN	0.6860 ± 0.0050	0.2460 ± 0.0007
	GINE	0.6621 ± 0.0067	0.2473 ± 0.0017
	GATEDGCN	0.6765 ± 0.0047	0.2477 ± 0.0009
	GPS	0.6534 ± 0.0091	0.2509 ± 0.0014
OTHERS	CRAWL	0.7074 ± 0.0032	0.2506 ± 0.0022
	DREW_{GCN}	0.7150 ± 0.0044	0.2536 ± 0.0015
	$\text{DREW}_{\text{GATEDGCN}}$	0.6977 ± 0.0026	0.2539 ± 0.0007
	EXPHORMER	0.6527 ± 0.0043	0.2481 ± 0.0007
	GRIT	0.6988 ± 0.0082	0.2460 ± 0.0012
	GRAPH VIT	0.6942 ± 0.0075	0.2449 ± 0.0016
	G-MLPMIXER	0.6921 ± 0.0054	0.2475 ± 0.0015
	LASER	0.6440 ± 0.0010	0.3043 ± 0.0019
	CO-GNN	0.6990 ± 0.0093	-
	NBA_{GCN}	0.7207 ± 0.0028	0.2472 ± 0.0008
	$\text{NBA}_{\text{GATEDGCN}}$	0.6982 ± 0.0014	0.2466 ± 0.0012
	PH-DGN	0.7012 ± 0.0045	0.2465 ± 0.0020
	GRED	0.7041 ± 0.0049	0.2584 ± 0.0015
	PR-MPNN	0.6825 ± 0.0086	0.2477 ± 0.0005
	IPR-MPNN	0.7210 ± 0.0039	0.2422 ± 0.0007
	AMP_{GCN}	$0.7161^{\dagger} \pm 0.0047$	$0.2446^{\dagger} \pm 0.0026$
	AMP_{GINE}	$0.7065^{\dagger} \pm 0.0105$	$0.2468^{\dagger} \pm 0.0026$
	$\text{AMP}_{\text{GATEDGCN}}$	$0.6943^{\dagger} \pm 0.0046$	$0.2480^{\dagger} \pm 0.0012$

Table 2. Mean test scores and standard deviation averaged over 4 final runs on the chemical datasets. The † indicates that AMP yields an improvement compared to the base architecture.

also found that the number of hidden units is an important hyper-parameter to perform well on these tasks, and a larger value seems to correlate well with good performances. Combined with the above results, we argue that the parameter budget imposed by previous works (Dwivedi et al., 2022) might limit the future progress on these tasks, as deeper networks are probably needed (we provide an analysis of the depth found by AMP below). The average diameter of these peptides is 57, meaning that using ten layers as done in other works might not be enough to capture long-range dependencies (Tönshoff et al., 2023a). We provide a visual analysis of predictions in Appendix G.

5.2. AMP mitigates oversmoothing and oversquashing

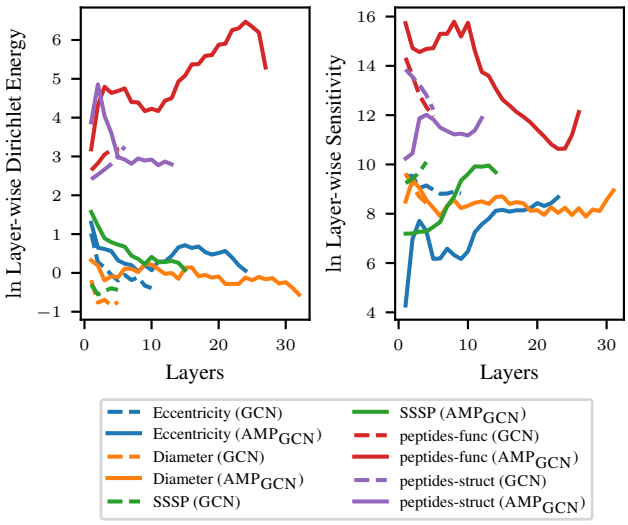

Figure 3. We show the Dirichlet energy (left) and the sensitivity (right) across layers for the GCN model and its AMP version.

We now comment on AMP's ability to mitigate oversmoothing and oversquashing, and we refer to Figure 3 for a qualitative analysis of the former (left) and of the latter (right). First, we computed the logarithm of the Dirichlet energy over embeddings of a trained GCN for different layers and datasets. This analysis reveals that the Dirichlet energy for AMP's variants is typically higher than the corresponding baselines and it can exhibit a stable, decreasing, or increasing behavior as the depth grows, in contrast to existing theoretical research on the GCN model where the Dirichlet energy constantly decreases and embeddings converge to the same value (Li et al., 2018; Rusch et al., 2023); note that we apply skip connections to the base GCN, so the energy does not immediately decreases. Therefore, it appears that our approach is indeed capable of *controlling* oversmoothing; we attribute this to the combination of message filtering and a layer-wise loss, which favors the propagation of gra-

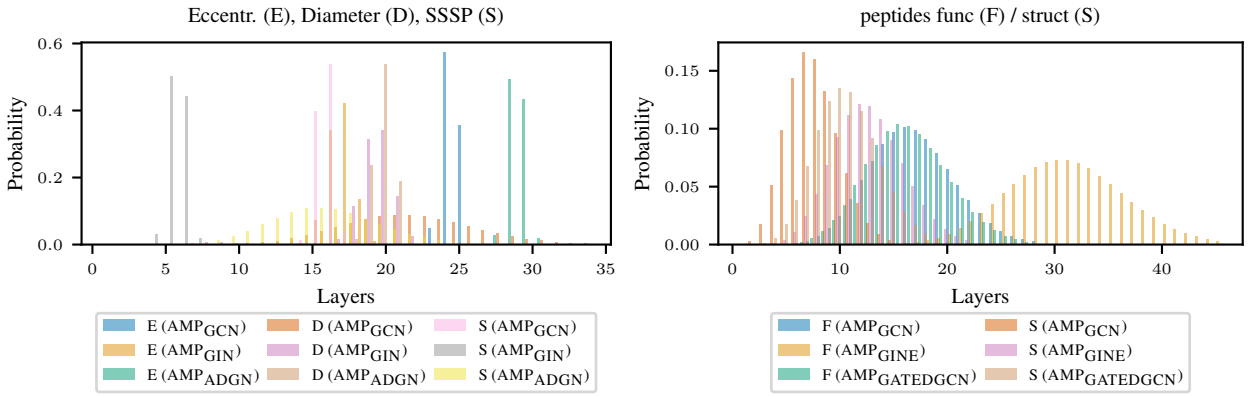

Figure 4. We show the distribution learned by the best configurations of each base model on the synthetic and chemical datasets.

dient to intermediate layers. We also report the layer-wise logarithm of node embeddings' sensitivity[4] as the gradient of the embeddings of the last layer L with respect to the ones of intermediate layers l: $\sum_{(v,u)\in\mathcal{V}} \left|\left| \frac{\partial \boldsymbol{h}_v^L}{\partial \boldsymbol{h}_u^\ell} \right|\right|_1$. Sensitivity provides insights into how pruning messages affects oversquashing; in fact, filtering messages might reduce said sensitivity with respect to the input. We report a quite heterogeneous picture in Figure 3 (right): the sensitivity of AMP can peak at the first or last layers, increase abruptly, or remain relatively stable. In all these cases, we have seen how AMP$_{\text{GCN}}$ substantially improves performances on tasks where it seems important to address oversquashing. This evidence also warns us against using sensitivity as the sole metric to measure oversquashing as a performance bottleneck.

5.3. Analysis of the AMP's Learned Depth

To understand how AMP mitigates underreaching, we inspect the learned distributions for synthetic datasets in Figure 4 (left). AMP uses more layers than the baselines (details in Appendix H) to achieve the best score on Eccentricity. Instead, it is found that about 20 layers are necessary to solve the task for the Diameter dataset, with all runs attaining a mean value between 17 and 22 layers. Finally, SSSP seems to be the task that requires fewer layers on average, with AMP$_{\text{GIN}}$ selecting less than ten layers to reach a very competitive score. Overall, it appears that folded normal and mixtures of folded normal distributions were selected more frequently as the best hyper-parameter for the synthetic tasks; in particular, the distributions for Eccentricity look sharply peaked, as if the models would need to use only the information computed at the very end of the deep architecture. It is worth remarking that this behavior is completely adaptive and guided by the task. Finally, we observe that the distributions learned on the chemical datasets are mostly Poisson ones, and AMP learns deeper networks than the corresponding baselines to reach better scores. These distributions peak at around ten layers for *peptides-struct*, which is more or less in line with what was reported in previous works (Tönshoff et al., 2023a). In any case, AMP enables training of very deep architectures thanks to its layer-wise output. However, this has a non-negligible cost in terms of number of readout parameters.

[4]Averaged over 250 validation nodes due to prohibitive costs.

5.4. On the Effects of Message Filtering

Figure 5 visualizes the amount of information pruned at each layer by AMP$_{\text{GCN}}$ on all datasets, together with an ablation study about the benefits of message filtering. The amount of information pruned is computed by summing the message filters' activations and normalizing the result by the total number of messages exchanged at each layer. We can see how AMP$_{\text{GCN}}$ gradually increases the amount of information to be used for Eccentricity, whereas in *peptides-func*, this quantity is almost always below 50%. One can appreciate how, depending on the task, the behavior of the message filtering changes significantly.

The ablation study, on the other hand, provides evidence that message filtering is, in most cases, a good strategy for performance improvements. We see that, for each model and dataset, input-based and embedding-based filtering provide a positive improvement in scores compared to no-filtering. Points lying in the grey area correspond to no improvement. We conclude that the choice of which filtering strategy to use remains a matter of empirical investigations.

6. Conclusions

We have introduced Adaptive Message Passing, a probabilistic framework that endows message passing architectures with the ability to learn how many messages to exchange between nodes and which messages to filter out. Our ap-

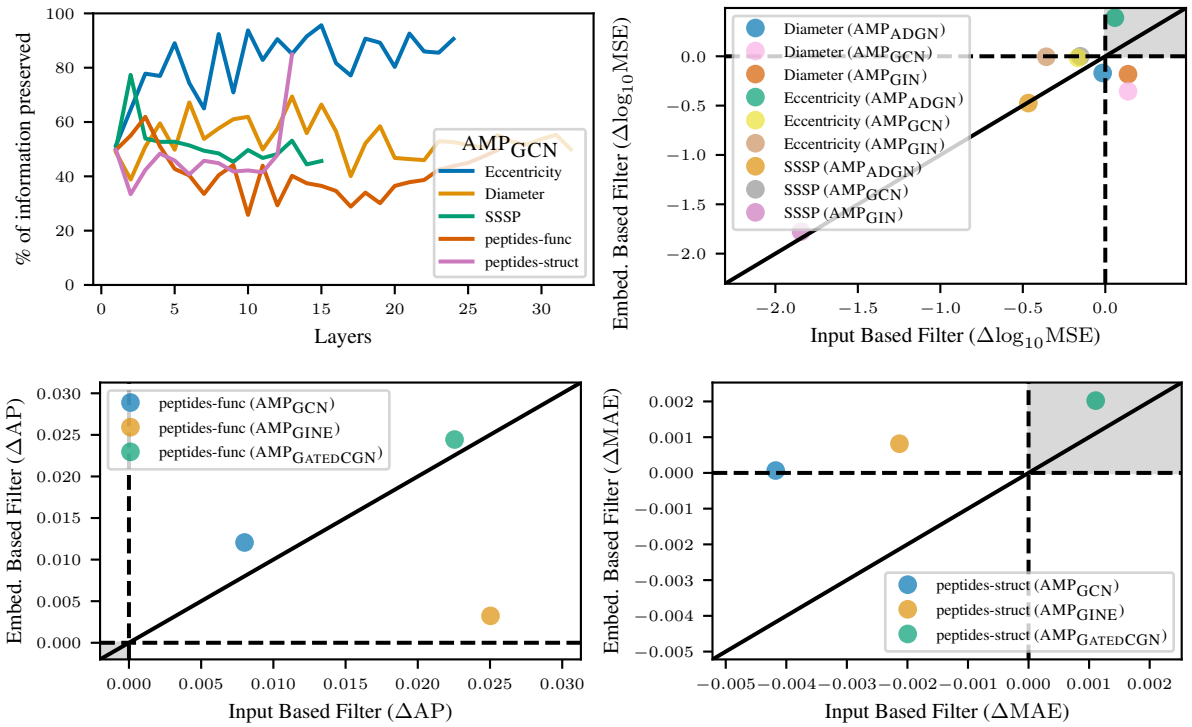

Figure 5. (Top right) we visualize the amount of information preserved in each layer by AMP$_{GCN}$. (Others) Ablation study of message filtering scheme: If a point lies in the area represented by the gray color, then filtering is not beneficial.

proach actively targets the long-range issue by relying on the observation that filtering messages mitigates oversmoothing and oversquashing, whereas learning depth can ideally solve underreaching. AMP achieves competitive results on long-range datasets without imposing strong inductive biases. Through qualitative analyses, our findings reveal how AMP learns very deep architectures if necessary for the task, and the amount of information propagated can greatly be reduced compared to classical message passing. Overall, our approach suggests that it might not be necessary to alter the initial graph structure, e.g., through rewiring, to improve the performances on long-range tasks. We hope Adaptive message passing will foster exciting research opportunities in the graph machine learning field and find successful applications in physics, chemistry, and material sciences.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Abboud, R., Dimitrov, R., and Ceylan, I. I. Shortest path networks for graph property prediction. In *The 1st Learning on Graphs Conference (LoG)*, 2022.

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *9th International Conference on Learning Representations (ICLR)*, 2021.

Bacciu, D., Errica, F., and Micheli, A. Contextual Graph Markov Model: A deep and generative approach to graph processing. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pp. 294–303, 2018.

Bacciu, D., Errica, F., and Micheli, A. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020a.

Bacciu, D., Errica, F., Micheli, A., and Podda, M. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 9 2020b.

Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., and Montúfar, G. Oversquashing in gnns through the lens of information contraction and graph expansion. In *Proceedings ot the 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2022.

Barbero, F., Velingker, A., Saberi, A., Bronstein, M. M., and Giovanni, F. D. Locality-aware graph rewiring in GNNs. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Bernardo, J. M. and Smith, A. F. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Bober, J., Monod, A., Saucan, E., and Webster, K. N. Rewiring networks for graph neural network training using discrete geometry. In *International Conference On Complex Networks And Their Applications*, 2023.

Bongini, M., Rigutini, L., and Trentin, E. Recursive neural networks for density estimation over generalized random graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 29(11):5441–5458, 2018.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Brockschmidt, M. Gnn-film: Graph neural networks with feature-wise linear modulation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Campa, A., Dauxois, T., Fanelli, D., and Ruffo, S. *Physics of long-range interacting systems*. OUP Oxford, 2014.

Castellana, D. and Errica, F. Investigating the interplay between features and structures in graph learning. In *MLG Workshop at ECML PKDD*, 2023.

Castellana, D., Errica, F., Bacciu, D., and Micheli, A. The infinite contextual graph Markov model. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pp. 2721–2737, 2022.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. Grand: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. PMLR, 2022.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

Chen, Y., Wu, L., and Zaki, M. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020b.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation. In *The 1st Learning on Graphs Conference (LoG)*, 2022.

del Castillo, J. and Pérez-Casany, M. Overdispersed and underdispersed poisson generalizations. *Journal of Statistical Planning and Inference*, 134:486–500, 2005.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., and Bronstein, M. M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Ding, Y., Orvieto, A., He, B., and Hofmann, T. Recurrent distance filtering for graph representation learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Errica, F. On class distributions induced by nearest neighbor graphs for node classification of tabular data. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Faber, L. and Wattenhofer, R. GwAC: GNNs with asynchronous communication. In *The 2nd Learning on Graphs Conference (LoG)*, 2023.

Fahlman, S. and Lebiere, C. The cascade-correlation learning architecture. In *Proceedings of the 3rd Conference on Neural Information Processing Systems (NIPS)*, 1989.

Fatemi, B., Abu-El-Haija, S., Tsitsulin, A., Kazemi, M., Zelle, D., Bulut, N., Halcrow, J., and Perozzi, B. Ugsl: A unified framework for benchmarking graph structure learning. In *Topology, Algebra, and Geometry in Machine Learning Workshop, ICML*, 2023.

Ferber, M., Zoete, V., and Michielin, O. T-cell receptors binding orientation over peptide/mhc class i is driven by long-range interactions. *PloS one*, 7(12):e51943, 2012.

Finkelshtein, B., Huang, X., Bronstein, M. M., and Ceylan, I. I. Cooperative graph neural networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., and Monti, F. Sign: Scalable inception graph neural networks. In *Graph Representation Learning and Beyond (GRL+) Workshop, ICML*, 2020.

Gabrielsson, R. B., Yurochkin, M., and Solomon, J. Rewiring with positional encodings for graph neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Gasteiger, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Goel, P. K. and Degroot, M. H. Information about hyperparamters in hierarchical models. *Journal of the American Statistical Association*, 76(373):140–147, 1981. ISSN 01621459.

Gravina, A., Bacciu, D., and Gallicchio, C. Anti-symmetric DGN: a stable architecture for deep graph networks. In *11h International Conference on Learning Representations (ICLR)*, 2023.

Gruber, L., Schäfl, B., Brandstetter, J., and Hochreiter, S. Processing large-scale graphs with g-signatures. In *ICML 2024 AI for Science Workshop*, 2024.

Gutteridge, B., Dong, X., Bronstein, M. M., and Di Giovanni, F. Drew: Dynamically rewired message passing with delay. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pp. 1024–1034, 2017.

Hammer, B., Micheli, A., and Sperduti, A. Universal approximation capability of cascade correlation for structures. *Neural Computation*, 17(5):1109–1159, 2005.

Harris, T., Yang, Z., and Hardin, J. W. Modeling underdispersed count data with generalized poisson regression. *The Stata Journal*, 12:736–747, 2012.

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., and Qian, X. Bayesian graph neural networks with adaptive connection sampling. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A generalization of vit/mlp-mixer to graphs. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Heilig, S., Gravina, A., Trenta, A., Gallicchio, C., and Bacciu, D. Port-hamiltonian architectural bias for long-range propagation in deep graph networks. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

Karhadkar, K., Banerjee, P. K., and Montufar, G. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *The 11th International Conference on Learning Representations (ICLR)*, 2023.

Kazi, A., Cosmo, L., Ahmadi, S.-A., Navab, N., and Bronstein, M. M. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:1606–1617, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Kurihara, K., Welling, M., and Teh, Y. W. Collapsed variational dirichlet process mixture models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

Leone, F. C., Nelson, L. S., and Nottingham, R. B. The folded normal distribution. *Technometrics*, 3:543–550, 1961.

Li, G., Muller, M., Thabet, A., and Ghanem, B. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Li, Y., Wang, Y., Huang, L., Yang, H., Wei, X., Zhang, J., Wang, T., Wang, Z., Shao, B., and Liu, T.-Y. Long-short-range message-passing: A physics-informed framework to capture non-local interaction for scalable molecular dynamics simulation. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Liu, J., Kawaguchi, K., Hooi, B., Wang, Y., and Xiao, X. Eignn: Efficient infinite-depth graph neural networks. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P. K., Coates, M., Torr, P., and Lim, S.-N. Graph inductive biases in transformers without message passing. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

Müller, L., Galkin, M., Morris, C., and Rampášek, L. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024.

Nazaret, A. and Blei, D. Variational inference for infinitely deep neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Park, S., Ryu, N., Kim, G., Woo, D., Yun, S.-Y., and Ahn, S. Non-backtracking graph neural networks. *Transactions on Machine Learning Research*, 2024.

Piana, S., Lindorff-Larsen, K., Dirks, R. M., Salmon, J. K., Dror, R. O., and Shaw, D. E. Evaluating the effects of cutoffs and treatment of long-range electrostatics in protein folding simulations. *PLoS One*, 7(6):e39918, 2012.

Poli, M., Massaroli, S., Rabideau, C. M., Park, J., Yamashita, A., Asama, H., and Park, J. Continuous-depth neural models for dynamic graph prediction. *arXiv preprint arXiv:2106.11581*, 2021.

Qian, C., Manolache, A., Ahmed, K., Zeng, Z., Broeck, G. V. d., Niepert, M., and Morris, C. Probabilistically rewired message-passing neural networks. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024a.

Qian, C., Manolache, A., Morris, C., and Niepert, M. Probabilistic graph rewiring via virtual nodes. In *The 38th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024b.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Roy, D. The discrete normal distribution. *Communications in Statistics - Theory and Methods*, 32:1871–1883, 2003. doi: 10.1081/sta-120023256.

Rüegsegger, U., Leber, J. H., and Walter, P. Block of hac1 mrna translation by long-range base pairing is released by cytoplasmic splicing upon induction of the unfolded protein response. *Cell*, 107(1):103–114, 2001.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Shirts, M. R., Mobley, D. L., Chodera, J. D., and Pande, V. S. Accurate and efficient corrections for missing dispersion interactions in molecular simulations. *Journal of Physical Chemistry B*, 111:13052–13063, 2007.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Sperduti, A. and Starita, A. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

Spinelli, I., Scardapane, S., and Uncini, A. Adaptive propagation graph convolutional network. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4755–4760, 2020.

Tönshoff, J., Ritzert, M., Rosenbluth, E., and Grohe, M. Where did the gap go? reassessing the long-range graph benchmark. In *The 2nd Learning on Graphs Conference (LoG)*, 2023a.

Tönshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023b.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *10th International Conference on Learning Representations (ICLR)*, 2022.

Tortorella, D. and Micheli, A. Leave graphs alone: Addressing over-squashing without rewiring. In *The 1st Learning on Graphs Conference (LoG)*, 2022.

Toth, C., Lee, D., Hacker, C., and Oberhauser, H. Capturing graphs with hypo-elliptic diffusions. In *The 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *6th International Conference on Learning Representations (ICLR)*, 2018.

Wang, C., Tsepa, O., Ma, J., and Wang, B. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

Wang, Y., Wang, Y., Yang, J., and Lin, Z. Dissecting the diffusion process in linear graph convolutional networks. In *The 35th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-I., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 5453–5462, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR)*, 2019.

Young, L. C. An inequality of the hölder type, connected with stieltjes integration. *Acta Mathematica*, 67:251–282, 1936. URL https://api.semanticscholar.org/CorpusID:122618935.

Yu, D., Zhang, R., Jiang, Z., Wu, Y., and Yang, Y. Graph-revised convolutional network. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD)*, 2020.

Zhao, T., Liu, Y., Neves, L., Woodford, O., Jiang, M., and Shah, N. Data augmentation for graph neural networks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Zhou, K., Huang, X., Song, Q., Chen, R., and Hu, X. Auto-gnn: Neural architecture search of graph neural networks. *Frontiers in Big Data*, 5:1029307, 2022.

Zhou, Z., Zhou, S., Mao, B., Zhou, X., Chen, J., Tan, Q., Zha, D., Feng, Y., Chen, C., and Wang, C. Opengsl: A comprehensive benchmark for graph structure learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

A. Extended Related Work Section

Below we provide a more detailed related work section that does not fit in the main paper.

Oversquashing. There are many methods that attempt to address the oversquashing problem with the goal of better capturing long-range dependencies (Alon & Yahav, 2021; Li et al., 2024). There is agreement that modifying the message passing scheme leads to improved performances; in this sense, the graph structure does not match exactly the computational graph used to compute the node embeddings. Some works learn how a node should completely stop propagating a message in a fixed-depth architecture (Spinelli et al., 2020) or if it should only listen, isolate, or receive/broadcast its own message (Finkelshtein et al., 2024). Similarly, one can learn to sample edges at each message passing layer according to some learned parametrization (Hasanzadeh et al., 2020) or have a completely asynchronous message passing (Faber & Wattenhofer, 2023). Our work differs from these works as we apply a *learned* (soft) filtering to all existing messages.

Another idea is to modify message passing to avoid backtracking of messages back to the source node, to achieve less redundancy of information (Park et al., 2024). While this choice proves effective at several tasks, it is still an open question whether it is always the best choice for the task at hand. In attention-based approaches (Velickovic et al., 2018), an edge filter is computed using some non-linear relationship between the embeddings of the source and destination nodes. This can introduce a severe computational burden as the function needs to be applied to all edges. Similarly, GNN-FiLM (Brockschmidt, 2020) learns a feature-wise linear modulator that depends on the destination node and modulates the magnitude of all incoming messages.

On the other hand, rewiring approaches try to alter the graph connectivity rather than the message passing operation. This action is meant to increase the sensitivity (Topping et al., 2022) of a node with respect to another, and it has been theoretically linked to the oversquashing problem. Some recent works try to preserve locality and sparsity of the rewiring process (Barbero et al., 2024) or dynamically rewire the graph based on the layers (Gutteridge et al., 2023). In contrast, others take a probabilistic approach to rewiring based on sampled sub-graphs (Qian et al., 2024a). Recently, a critical perspective on the effectiveness of rewiring approaches has also been given (Tortorella & Micheli, 2022).
Finally, we mention ordinary differential equation-based message passing approaches, which are provably preserving information regardless of the depth in the network (Gravina et al., 2023) and have shown great results on datasets aimed at capturing long-range dependencies.

Oversmoothing. Oversmoothing is perhaps one of the first problems that emerged empirically and was then analyzed theoretically (Li et al., 2018; Oono & Suzuki, 2020; Rusch et al., 2023). Not surprisingly, one practical solution to oversmoothing is dropping edges to reduce the overall flow of messages and, thus, avoid the convergence of all embeddings to the same value (Rong et al., 2020). Another well-known solution to alleviate oversmoothing is to employ skip/residual connections (Kipf & Welling, 2017; Li et al., 2019), which consists of summing the representations learned at deeper layers with those of previous ones. Similarly to what is done in this work, the concatenation of node representations across layers is also a way to circumvent oversmoothing, which has been adopted in neural and probabilistic models to improve the downstream performances on several node and graph-related tasks (Bacciu et al., 2018; Xu et al., 2018; Bacciu et al., 2020a). Instead, an orthogonal research direction considers implicit neural networks for graphs that correspond to infinite-depth models and seem to be able to capture long-range dependencies (Poli et al., 2021; Liu et al., 2021). These models simulate synchronous message passing with a potentially infinite number of message-propagation steps, and some of them appear to be empirically robust to the oversmoothing problem.

Adaptive Architectures. The last part of this section is dedicated to works that try to learn the architecture of the model during training. Our work is inspired by the unbounded depth networks (UDNs) of Nazaret & Blei (2022), who proposed a variational framework for learning the depth in deep neural networks. In the graph domain, the first approach to learning the depth of a DGN was proposed by Micheli (2009), who applied the cascade correlation algorithm (Fahlman & Lebiere, 1989) to learn a proper depth for the task. In the field of graph representation learning, other works attempted to learn the width of the representation of each message passing layer by exploiting Bayesian non-parametric models (Castellana et al., 2022), which allows to save time and memory when building deeper probabilistic DGNs. Finally, it is important to notice that these works, including this manuscript, are all orthogonal to the popular field of neural architecture search (Zhou et al., 2022): The former attempts at dynamically modifying the architecture *during learning*, whereas neural architecture search approaches find smarter ways to carry out a grid search. An advantage of adaptive approaches is that they can greatly reduce time and computational costs to perform a hyper-parameter search.

Graph Rewiring, Structure Learning, and Transformer Models for Graphs. There has been a growing interest in adapting the input graph structure and, therefore, the message-passing operations of DGNs. These approaches can be roughly divided into graph structure learning and rewiring methods on one hand and transformer-based models on the other. There are several strategies of graph rewiring such as incorporating multi-hop neighbors (Gabrielsson et al., 2023) and nodes reachable through shortest paths (Abboud et al., 2022). Gutteridge et al. (2023) is most related to the filtering approach we introduce as it rewires the graph for every message-passing layer. The methods MixHop (Abu-El-Haija et al., 2019), SIGN (Frasca et al., 2020), and DIGL (Gasteiger et al., 2019) can also be considered as graph rewiring as these leverage different heuristics to reach further-away neighbors in a single message-passing layer. Deac et al. (2022); Shirzad et al. (2023) use the notion of expander graphs to alter the messages, while Karhadkar et al. (2023) resort to spectral techniques, and Banerjee et al. (2022) propose a random edge flip approach. There also exist rewiring heuristics based on particular metrics such as Ricci and Forman curvature (Bober et al., 2023). Finally, recent work has proposed an approach to rewire the input graph probabilistically and in a data-driven way (Qian et al., 2024a).

Graph structure learning (GSL) approaches are another line of work rewiring the input graphs. Here the focus is on node classification problems on a single graph. Typically, these methods maintain a learnable function that assigns prior scores to the edges, and based on these scores a subset of edges is selected from the original graph (Chen et al., 2020b; Yu et al., 2020; Zhao et al., 2021). To introduce discreteness and sparsity, Franceschi et al. (2019); Kazi et al. (2022); Zhao et al. (2021) use discrete categorical (sampled from using the Gumbel softmax trick) and Bernoulli distributions, respectively. Most existing GSL approaches typically use a k-NN algorithm, a simple randomized version of k-NN, or represent edges with independent Bernoulli random variables. For a comprehensive survey of GSL, see Fatemi et al. (2023); Zhou et al. (2023). In the context of node classification, there has been recent progress in a more principled understanding of the possible advantages of GSL in the fully supervised setting (Castellana & Errica, 2023; Errica, 2023).

Graph transformers (Dwivedi et al., 2022; He et al., 2023; Müller et al., 2024; Rampášek et al., 2022; Chen et al., 2022) adaptively change the message-passing operations for each layer by applying an attention mechanism among all nodes of the input graphs. Experimental results have shown that graph transformers have the ability to mitigate over-squashing (Müller et al., 2024). Due to their attention mechanism, however, transformer-based models have typically a quadratic space and memory requirement.

State space models have also been applied to graphs (Wang et al., 2024). Similarly, theoretically grounded sequence-processing frameworks (Toth et al., 2022; Gruber et al., 2024), leveraging randomized signatures, demonstrated promising potential in alleviating oversquashing effects in large graphs. The main differences with AMP, Wang et al. (2024) relies on a separate sequence model to develop a node selection mechanism whereas we work on the message passing itself; Toth et al. (2022) develops a new graph Laplacian that is better suited for long-range propagation; Gruber et al. (2024) converts a graph into a latent representation that can be passed to downstream classifiers.

Contrary to all these approaches, AMP, the adaptive message passing approach presented here, uses a variational framework to jointly learn both the depth of the DGN and a filtering of the messages passed between nodes in each of these layers.

B. Generalization to New Families of Truncated Distributions

The family of "truncated" Poisson distributions, proposed by Nazaret & Blei (2022) to learn unbounded depth networks, satisfies specific requirements that allow us to efficiently perform (variational) inference. In particular, by truncating the Poisson distribution at its quantile function evaluated at c, one can bound its support and compute expectations in finite time. However, the Poisson distribution suffers from equidispersion, meaning that the variance is equal to the mean; this is a particularly limiting scenario when learning distributions over the importance of layers. In fact, one might also want to model variances that are smaller or greater than the mean, which is referred to as under and over-dispersion, respectively, to learn a broader class of distributions (del Castillo & Pérez-Casany, 2005; Harris et al., 2012). To address this problem, in the following we introduce two families of distributions and prove that they also satisfy the requirements defined in Nazaret & Blei (2022); we formally recall such requirements below.

Definition B.1. A variational family $Q = q(\boldsymbol{\omega})$ over $\mathbb{N}^+$ is *unbounded* with *connected* and *bounded* members if

1. $\forall q \in Q, \text{support}(q)$ is bounded
2. $\forall L \in \mathbb{N}^+, \exists q \in Q$ such that $L \in \text{argmax}(q)$
3. Each parameter in the set $\boldsymbol{\omega}$ is a continuous variable.

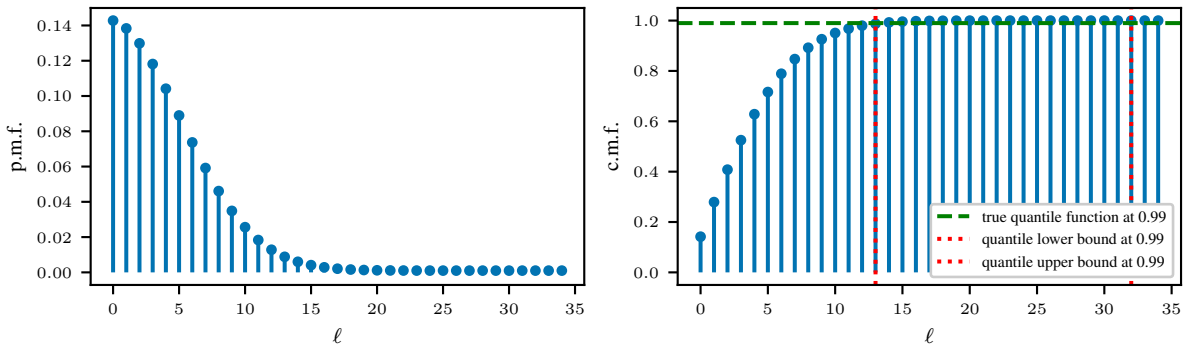

Figure 6. (Left) Probability mass function of a DFN distribution with $\mu = 1$ and $\sigma = 5$. (Right) The cumulative mass function of the distribution with an example of lower and upper bounds (vertical dashed lines) for the true quantile function evaluated at 0.99 of the corresponding FN distribution (horizontal dashed line).

Condition 1 is necessary to compute the expectation over $q(\ell; \boldsymbol{\lambda})$ in finite time, condition 2 ensures that we can give enough probability mass to each point in the support of q, and condition 3 is required for learning the distributions' parameters in a differentiable manner.

The Discrete Folded Normal Distribution. folded normal (FN) distributions (Leone et al., 1961) can model under-, equi-, and over-dispersion. They are parametrized by a mean parameter μ and a standard deviation σ. Its density is defined as

$$p_{\text{FN}}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x+\mu)^2}{2\sigma^2}}, \ \mu, \sigma \in \mathbb{R}, x \geq 0. \tag{15}$$

To use a FN distribution in AMP, the idea is to first define a discrete version of the folded normal (DFN) with the strategy highlighted in Roy (2003):

$$p_{\text{DFN}}(0; \mu, \sigma) = S_{\text{FN}}(1; \mu, \sigma) \tag{16}$$

$$p_{\text{DFN}}(x; \mu, \sigma) = S_{\text{FN}}(x+1; \mu, \sigma) - S_{\text{FN}}(x; \mu, \sigma), \forall x \in \mathbb{N}^+ \tag{17}$$

where $S_{\text{FN}}(x; \mu, \sigma)$ is the cumulative distribution function (c.d.f.) of the folded normal distribution evaluated at x.[5] It is also useful to notice the equivalence between the c.d.f of the DFN $S_{\text{DFN}}(x; \mu, \sigma)$ and that of a folded normal $S_{\text{FN}}(x; \mu, \sigma)$

$$S_{\text{DFN}}(x; \mu, \sigma) = \sum_{i=0}^{x} p_{\text{DFN}}(x; \mu, \sigma) = S_{\text{FN}}(x+1; \mu, \sigma), x \in \mathbb{N}, \tag{18}$$

which implies that $S_{\text{DFN}}(x; \mu, \sigma) \geq S_{\text{FN}}(x; \mu, \sigma)$. Figure 6 shows the probability mass function (p.m.f.) of a DFN distribution with $\mu = 1$ and $\sigma = 5$ and its cumulative mass function (c.m.f.).

Clearly, condition 3 of Definition B.1 is satisfied. It is also trivial to satisfy condition 2 by choosing a peaked distribution with a small value of σ. In what follows, we focus on lower and upper bounds to the quantile function evaluated at c of the DFN distribution so that we know we can truncate the distribution to the *finite* quantile threshold, meaning condition 1 is also satisfied.

Theorem B.2. *There exists lower and upper bounds to the quantile evaluated at c,* $0 < c < 1$*, for any DFN distribution with* $\sigma > 0$.

Proof. We first need to compute a lower bound to the quantile function of the FN distribution since there is no closed formula for it. To start, we note that the c.d.f. of the Gaussian distribution is greater or equal to that of a folded normal distribution:

$$\frac{1}{2}\text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) + \frac{1}{2} \geq \frac{1}{2}\text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) + \underbrace{\frac{1}{2}\text{erf}\left(\frac{x+\mu}{\sigma\sqrt{2}}\right)}_{\leq \frac{1}{2}} = S_{\text{FN}}(x; \mu, \sigma). \tag{19}$$

[5] We note that the support is defined over $\mathbb{N}$ and not on $\mathbb{N}^+$, but this is not an issue from a practical point of view.

where "erf" is the error function. This implies that the quantile threshold of the Gaussian x_G, which we know how to compute, is reached earlier than that of the FN x_{FN}, that is, $x_G \leq x_{\text{FN}}$, and in particular $\lfloor x_G \rfloor \leq \lfloor x_{\text{FN}} \rfloor$ are also lower bounds. It then follows from Equation 18 that $\lfloor x_G \rfloor$ - 1 is a lower bound for the DFN distribution.

To find an upper bound, we apply Chernoff's Bound

$$p(X \geq x) \leq \frac{M_X(t)}{e^{tx}}, \ \ \forall t > 0 \tag{20}$$

where X is a *r.v.* that follows a folded normal distribution with mean μ and standard deviation σ, and $M_X(t) = \mathbb{E}[e^{tX}]$ is the well-known moment generating function of the X. To find an upper bound to the quantile threshold, we need that $(1-c) = \frac{M_X(t)}{e^{tx}}$ for some choice of t. Defining Φ as the normal *c.d.f.* $\Phi(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \geq 0$, we choose $t = \frac{1}{\sigma}$ and obtain

$$\frac{M_X(t)}{e^{tx}} = e^{-tx}\left(e^{\frac{\sigma^2 t^2}{2} + \mu t}\Phi\left(\frac{\mu}{\sigma} + 1\right) + e^{\frac{\sigma^2 t^2}{2} - \mu t}\Phi\left(-\frac{\mu}{\sigma} + 1\right)\right) \tag{21}$$

$$= e^{-\frac{x}{\sigma}} e^{\frac{1}{2}} e^{\frac{\mu}{\sigma}}\left(\Phi\left(\frac{\mu}{\sigma} + 1\right) + \Phi\left(-\frac{\mu}{\sigma} + 1\right)\frac{1}{e^{2\mu/\sigma}}\right) \tag{22}$$

$$= k e^{\frac{\mu - x}{\sigma}}, \tag{23}$$

$$k = e^{\frac{1}{2}}\left(\Phi\left(\frac{\mu}{\sigma} + 1\right) + \Phi\left(-\frac{\mu}{\sigma} + 1\right)\frac{1}{e^{2\mu/\sigma}}\right) > 0. \tag{24}$$

Therefore, the upper bound of the quantile threshold is given by

$$k e^{\frac{\mu - x}{\sigma}} = (1 - c) \tag{25}$$

$$\ln k + \frac{\mu - x}{\sigma} = \ln(1 - c) \tag{26}$$

$$\sigma \ln k + \mu - x = \sigma \ln(1 - c) \tag{27}$$

$$x = \mu + \sigma \ln k - \sigma \ln(1 - c). \tag{28}$$

Therefore, if the upper bound to the quantile of the FN is x, it follows from Equation 18 that $x - 1$ is also an upper bound of the DFN. □

Consequently, we can efficiently find the true quantile threshold by running a binary search between the lower and the upper bounds. Figure 6 (right) shows an example of lower and upper bounds (vertical dashed lines) as well as the true quantile threshold of the FN distribution.

A Mixture of Simpler Distributions. It is possible to learn more complex distributions $q(\ell; \boldsymbol{\omega})$ that satisfy the conditions of Definition B.1 by mixing simpler distributions like the DFN defined above. A mixture of C families of unbounded distributions $q_1(\ell; \boldsymbol{\omega}), \ldots, q_C(\ell; \boldsymbol{\omega})$ with bounded and connected members is defined as:

$$q_{\mathcal{M}}(\ell; \boldsymbol{\omega}) = \sum_{i=0}^{C} w_i q_i(\ell; \boldsymbol{\omega}) \tag{29}$$

where $0 \leq w_i \leq 1$ is mixture's i weight and $\sum_{i=0}^{C} w_i = 1$. Conditions 2 and 3 are again trivially satisfied (a mixture can always collapse to one of its distributions that satisfy said conditions), and below, we show that lower and upper bounds still exist.

Theorem B.3. *There exist lower and upper bounds to the quantile evaluated at c, $0 < c < 1$, for a mixture of C distributions that satisfy the conditions of Definition B.1, provided that lower and upper bounds exist for each distribution in the mixture.*

Proof. The c.m.f. of a mixture of discrete distributions can be written as a weighted sum of c.m.f.s:

$$S_{\mathcal{M}}(x; \boldsymbol{\omega}) = \sum_{i=0}^{C} w_i S_i(x; \boldsymbol{\omega}). \tag{30}$$

Let x^* be the greatest upper bound of the quantile threshold across all C components of the mixture, and let i^* be the associated component. It follows that, $\forall j, S_j(x^*;\boldsymbol{\omega}) \geq c$, and

$$S_{\mathcal{M}}(x^*;\boldsymbol{\omega}) = \sum_{i=0}^{C} w_i S_i(x^*;\boldsymbol{\omega}) \geq \sum_{i=0}^{C} w_i c = c. \tag{31}$$

Therefore, x^* is also an upper bound for the mixture of distributions. It is possible to prove that a lower bound of the mixture is the smallest lower bound of the quantile threshold across all C components using a similar approach. □

To summarize, we have shown how one can use more complex families of distributions in the context of AMP, allowing us to model under and over-dispersion. In this work, we will treat the choice of the family of distributions $q(\ell;\boldsymbol{\omega})$ to use as a hyper-parameter to be tuned.

C. ELBO Derivation

We now report the full derivation of our ELBO for a single sample:

$$\ln p(g_i, Y_{g_i}) \geq \mathbb{E}_{q(\boldsymbol{\theta},L,\boldsymbol{F}_i,|g_i,Y_{g_i})}\left[\ln p(Y_{g_i}, L, \boldsymbol{F}_i, \boldsymbol{\theta}|g_i) - \ln q(L, \boldsymbol{F}_i, \boldsymbol{\theta}|g_i)\right] \tag{32}$$

$$= \mathbb{E}_{q(L;\boldsymbol{\lambda})}\left[\ln \frac{p(L)}{q(L;\boldsymbol{\lambda})} + \mathbb{E}_{q(\boldsymbol{\theta}|L;\boldsymbol{\nu})}\left[\ln \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}|L;\boldsymbol{\nu})}\right]\right. \tag{33}$$

$$\left. + \mathbb{E}_{q(\boldsymbol{\theta}|L;\boldsymbol{\nu})q(\boldsymbol{F}_i|g_i,L,\boldsymbol{\theta})}\left[\ln \frac{p(\boldsymbol{F}_i)}{q(\boldsymbol{F}_i|g_i,L,\boldsymbol{\theta})} + \ln p(Y_{g_i}|L,\boldsymbol{F}_i,\boldsymbol{\theta}|g_i)\right]\right] \tag{34}$$

$$= \sum_{\ell=1}^{\hat{L}} q(\ell;\boldsymbol{\lambda})\left[\ln \frac{p(\ell)}{q(\ell;\boldsymbol{\lambda})} + \ln \frac{p(\boldsymbol{\nu})}{q(\boldsymbol{\nu}|\ell;\boldsymbol{\nu})} + \ln \frac{p(\boldsymbol{F}_i)}{q(\boldsymbol{F}_i|g_i,\ell,\boldsymbol{\nu})} + \ln p(Y_{g_i}|\ell,\boldsymbol{F}_i,\boldsymbol{\nu},g_i)\right], \tag{35}$$

The extension to the full dataset is identical except for the decomposition of the rightmost term into a product of conditional probabilities, one for each *i.i.d.* sample.

D. On Oversmoothing, Oversquashing, and Underreaching

In this section, we discuss AMP's implications on *oversmoothing*, *oversquashing*, and *underreaching*, all of which hamper the ability of DGNs to capture long-range interactions between nodes in the graph and are related in subtle ways.

Oversmoothing. Oversmoothing has been formally defined by Rusch et al. (2023) as the convergence of a node embeddings' similarity as the number of message passing layers increases. In other words, it formalizes the widely accepted notion that node embeddings tend to become identical after many layers of message passing. Different oversmoothing metrics have been proposed, and in this work, we consider the Dirichlet energy (Rusch et al., 2022; 2023) at layer ℓ defined as

$$E(\boldsymbol{H}^{\boldsymbol{\ell}}) = \frac{1}{|\mathcal{V}|}\sum_{u\in\mathcal{V}}\sum_{v\in\mathcal{N}_u} ||\boldsymbol{h}_u^{\ell} - \boldsymbol{h}_v^{\ell}||^2 \tag{36}$$

where we indicate with $\boldsymbol{H}^{\boldsymbol{\ell}}$ the set of node embeddings computed at layer ℓ.

There are at least two reasons why AMP alleviates oversmoothing. The first is that, in principle, the adaptive message filtering scheme reduces the synchronous exchange of *all* messages at a given layer, and message exchange will be different depending on the specific layer. The second is that the readout mapping of each layer directly propagates the gradient of the loss into the corresponding message passing operation, which encourages diversity of node representations of each layer ℓ as long as $q(\ell;\boldsymbol{\lambda})$ is large enough (that is, layer ℓ's output is important for the final prediction). In our experiments, we will show that AMP can generate architectures in which the Dirichlet energy does not decay exponentially and thus suffers less from oversmoothing than the baselines.

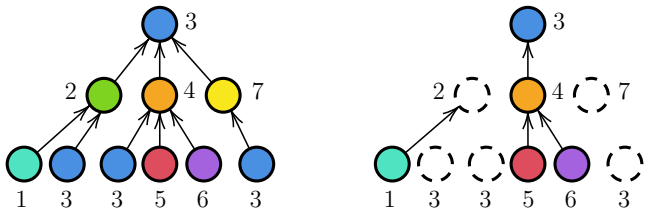

Figure 7. Comparison of the 2-hop computational tree necessary to compute $\boldsymbol{h}_3^2$ in the graph of Figure 2 for standard message passing (left) and AMP (right), where we discretized message filtering to simplify the concept. AMP can effectively prune/filter information in sub-trees to propagate only the relevant information for the task.

Oversquashing. The term oversquashing refers to the compression of an exponentially-growing amount of information (Micheli, 2009) into fixed-size node embeddings (Alon & Yahav, 2021; Di Giovanni et al., 2023), causing a possibly severe bottleneck that hampers DGNs' ability to effectively propagate task-specific information. An intuitive visualization is provided in Figure 7 (left), where node 3 of the graph defined in Figure 2 needs to compress information of its 2-hop neighborhood into a single node embedding. The literature on the topic is already vast despite its very recent introduction; some works address topological bottlenecks through rewiring of the original graph structure (Topping et al., 2022), while others preserve information by viewing the message passing operations through the lens of ordinary differential equations (Gravina et al., 2023; Heilig et al., 2025). By properly modifying the curvature of a graph (Topping et al., 2022), some graph rewiring approaches aim at increasing the *sensitivity* of a node's u embedding $\boldsymbol{h}_u^L$ with respect to the input $\boldsymbol{x}_v$ of another node v, that is $\left|\left|\frac{\partial \boldsymbol{h}_v^L}{\partial \boldsymbol{x}_u}\right|\right|_1$. Topping et al. (2022) argue that increasing the sensitivity can alleviate oversquashing and better capture long-range dependencies. Indeed, by rewiring two distant nodes with a new edge, the sensitivity of these two nodes will almost certainly increase.

While we do agree that long-range dependencies can be better captured, we argue that rewiring might make the computational bottleneck problem *worse* by adding extra information to be compressed into a node's embedding (assuming other edges are not removed). In contrast, the adaptive filtering scheme of AMP shown in Figure 7 (right) might decrease the overall sensitivity defined above, but at the same time it will reduce the number of messages that need to be compressed into node 3, hence *alleviating* oversquashing. Similarly, the synthetic datasets defined in Alon & Yahav (2021), which are meant to measure how well a model addresses oversquashing, require that *all* information is preserved to solve a task. This would certainly be a good test-bed for ODE-based models (Gravina et al., 2023), but other tasks might require propagating only a subset of the total information contained in the graph. The ability to *isolate* such information from the rest can be seen as a solution to the oversquashing problem, which is exactly the opposite goal of the synthetic tasks previously mentioned. In summary, the problem of "oversquashing" is clearly multi-faceted and requires great care regarding its evaluation. As such, it might be a good idea for the future to decompose over-squashing into simpler sub-problems, such as the ability to isolate the relevant information (which AMP can do), the ability to propagate all information, and the ability to increase the sensitivity between far-away nodes.

We complement this discussion with a Theorem, inspired by Di Giovanni et al. (2023), that shows how AMP can control the upper bound on the sensitivity by filtering messages.

Theorem 3.1 For AMP with m layers, embedding dimension d, and $u, v \in \mathcal{V}$,

$$\left|\left|\frac{\partial h_v^\ell}{\partial h_u^0}\right|\right|_{L^1} \leq d\left(\left(c_{\mathrm{up}}\left(c_{\mathrm{rs}} I + c_{\mathrm{mp}}\left(c_F k_h + k_F\right) A\right)\right)^\ell\right)_{vu}.$$

Here, MPNN is in the following form

$$(\mathbb{R}^d \ni)\ h_v^\ell = \mathrm{up}\left(\mathrm{rs}(h_v^{\ell-1}) + \mathrm{mp}(\sum_u A_{vu} F(h_u^{\ell-1}) \odot h_u^{\ell-1})\right)$$

where up, rs, and mp are Lipschitz functions as in Di Giovanni et al. (2023) with constants $c_{\mathrm{up}}, c_{\mathrm{rs}}, c_{\mathrm{mp}}$, c_F is the upper bound of the entry-wise L^1 matrix norm of $\frac{\partial F}{\partial x}$ for the filtering function F, k_h is the maximal absolute value among the entries of h, and similarly k_F for the output of F.

Proof. **Case:** $\ell = 1$

The gradient of the α-th element of hidden vector after the first layer with respect to the β-th element of an input is written as

$$\frac{\partial h_v^{1,\alpha}}{\partial h_u^{0,\beta}} = \sum_{p=1}^{d} \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \left(\sum_{r=1}^{d} \frac{\partial \text{rs}^{1,p}}{\partial x_r} \frac{\partial h_v^{0,r}}{\partial h_u^{0,\beta}} + \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} \sum_z A_{vz} \frac{\partial \left(F(h_z^0) \odot h_z^0 \right)^q}{\partial h_u^{0,\beta}} \right).$$

Hereinafter, $\alpha, \beta \in [d]$ and for a scalar-valued function f, $\frac{\partial f}{\partial x_s}$ represents the derivative of f with respect to s-th element x_s of its vector input. We define a part of the second term in the right-hand side of the above equation as

$$M(z) := \frac{\partial \left(F_0(h_z^0) \odot h_z^0 \right)^q}{\partial h_u^{0,\beta}},$$

and then

$$\begin{aligned} M(z) &= \frac{\partial F^q(h_z^0)}{\partial h_u^{0,\beta}} h_z^{0,q} + F^q(h_z^0) \frac{\partial h_z^{0,q}}{\partial h_u^{0,\beta}} \\ &= \sum_{r=1}^{d} \frac{\partial F^q(h_z^0)}{\partial h_z^{0,r}} \frac{\partial h_z^{0,r}}{\partial h_u^{0,\beta}} h_z^{0,q} + F^q(h_z^0) \frac{\partial h_z^{0,q}}{\partial h_u^{0,\beta}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \left| \frac{\partial h_v^{1,\alpha}}{\partial h_u^{0,\beta}} \right| &= \left| \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \left(\sum_{r=1}^{d} \frac{\partial \text{rs}^{1,p}}{\partial x_r} \frac{\partial h_v^{0,r}}{\partial h_u^{0,\beta}} + \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} \sum_z A_{vz} M(z) \right) \right| \\ &\leq \left| \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \frac{\partial \text{rs}^{1,p}}{\partial x_\beta} \delta_{vu} \right| + \left| \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} \sum_z A_{vz} M(z) \right|. \end{aligned} \tag{37}$$

The right-hand side of (37) is further expanded to

$$\sum_{p=1}^{d} \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} \sum_z A_{vz} \sum_{r=1}^{d} \frac{\partial F^q}{\partial x_r} \frac{\partial h_z^{0,r}}{\partial h_u^{0,\beta}} h_z^{0,q} \tag{38}$$

$$+ \sum_{p=1}^{d} \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} \sum_z A_{vz} F^q(h_z^0) \frac{\partial h_z^{0,q}}{\partial h_u^{0,\beta}}. \tag{39}$$

Since this is the very first layer of MPNN, both terms can be reduced to simpler forms. Indeed, (38) can be reduced to

$$(38) = \sum_{p=1}^{d} \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \sum_{q=1}^{d} \frac{\partial \text{mp}^{1,p}}{\partial x_q} A_{vu} \frac{\partial F^q}{\partial x_\beta} h_u^{0,q}.$$

Using (generalized) Hölder's inequality, inequality (3.1) in Young (1936), we get

$$\begin{aligned} |(38)| &\leq A_{vu} \sum_{p=1}^{d} \left| \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \right| \sum_{q=1}^{d} \left| \frac{\partial \text{mp}^{1,p}}{\partial x_q} \right| \left| \frac{\partial F^q}{\partial x_\beta} \right| |h_u^{0,q}| \\ &\leq A_{vu} k_h \sum_{p=1}^{d} \left| \frac{\partial \text{up}^{1,\alpha}}{\partial x_p} \right| \sum_{q=1}^{d} \left| \frac{\partial \text{mp}^{1,p}}{\partial x_q} \right| \sum_{q=1}^{d} \left| \frac{\partial F^q}{\partial x_\beta} \right| \\ &\leq A_{vu} \cdot c_{\text{up}} \cdot c_{\text{mp}} \cdot c_F \cdot k_h. \end{aligned} \tag{40}$$

Similarly, the norm of (39) is also bounded from above

$$|(39)| \leq A_{vu} c_{up} c_{mp} k_F.$$

The left-hand term of (37) is also bounded by $c_{\text{up}}c_{\text{rs}}I_{vu}$ (see also the proof of Theorem B.1. in Di Giovanni et al. (2023),) and we get

$$\left|\frac{\partial h_v^{1,\alpha}}{\partial h_u^{0,\beta}}\right| \leq c_{\text{up}}\left(c_{\text{rs}}I + c_{\text{mp}}\left(c_F k_h + k_F\right)A\right)_{vu}.$$

Case: arbitrary ℓ.

The gradient of the hidden vector at the layer of arbitrary ℓ is written as

$$\frac{\partial h_v^{\ell+1,\alpha}}{\partial h_u^{0,\beta}} = \sum_{p=1}^{d}\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\left(\sum_{r=1}^{d}\frac{\partial \text{rs}^{\ell+1,p}}{\partial x_r}\frac{\partial h_v^{\ell,r}}{\partial h_u^{0,\beta}} + \sum_{q=1}^{d}\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\sum_z A_{vz}\frac{\partial\left(F(h_z^\ell)\odot h_z^\ell\right)^q}{\partial h_u^{0,\beta}}\right).$$

Define and expand the right-hand term of the above equation as follows:

$$\begin{aligned}(\#) &:= \sum_p \frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\sum_q\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\sum_z A_{vz}\frac{\partial\left(F(h_z^\ell)\odot h_z^\ell\right)^q}{\partial h_u^{0,\beta}}\\ &= \sum_p \frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\sum_q\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\sum_z A_{vz}\left(\sum_{r=1}^{d}\frac{\partial F^q}{\partial x_r}\frac{\partial h_z^{\ell,r}}{\partial h_u^{0,\beta}}h_z^{\ell,q} + F^q(h_z^\ell)\frac{\partial h_z^{\ell,q}}{\partial h_u^{0,\beta}}\right).\end{aligned}$$

Then, its norm is

$$\begin{aligned}|(\#)| &\leq \sum_p\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_q\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\sum_z A_{vz}\sum_{r=1}^{d}\left|\frac{\partial F^q}{\partial x_r}\right|\left|\frac{\partial h_z^{\ell,r}}{\partial h_u^{0,\beta}}\right|\left|h_z^{\ell,q}\right|\\ &\quad+\sum_p\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_q\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\sum_z A_{vz}\left|F^q(h_z^\ell)\right|\left|\frac{\partial h_z^{\ell,q}}{\partial h_u^{0,\beta}}\right|\\ &\leq \Bigg(\sum_p\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_q\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\sum_z A_{vz}\sum_{r=1}^{d}\left|\frac{\partial F^q}{\partial x_r}\right|\left|h_z^{\ell,q}\right|\\ &\qquad+\sum_p\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_q\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\sum_z A_{vz}\left|F^q(h_z^\ell)\right|\Bigg)\\ &\qquad\times c_{\text{up}}^{\ell+1}\left(\left(c_{\text{rs}}I + c_{\text{mp}}\left(c_F k_h + k_F\right)A\right)^\ell\right)_{vu}.\end{aligned} \tag{41}$$

The first term of (41), except the constant term, is bounded from above:

$$\begin{aligned}&\sum_{p=1}^{d}\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_z A_{vz}\left(\sum_{q=1}^{d}\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\left|h_z^{\ell,q}\right|\sum_r\left|\frac{\partial F^q}{\partial x_r}\right|\right)\\ &\leq \sum_{p=1}^{d}\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_z A_{vz}\left(\sum_{q=1}^{d}\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\right)k_h\left(\sum_{r,q}\left|\frac{\partial F^q}{\partial x_r}\right|\right)\\ &\leq c_{\text{up}}\cdot c_{\text{mp}}\cdot k_h\cdot c_F\cdot\sum_z A_{vz}.\end{aligned} \tag{42}$$

Note that the first inequality is derived by using Hölder inequality again. Similarly, the second term of (41) is

$$\begin{aligned}\sum_{p=1}^{d}\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_z A_{vz}\sum_{q=1}^{d}\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|\left|F(h_z^{\ell,q})\right| &\leq \sum_{p=1}^{d}\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\sum_z A_{vz}\left(\sum_{q=1}^{d}\left|\frac{\partial \text{mp}^{\ell+1,p}}{\partial x_q}\right|k_F\right)\\ &\leq c_{\text{up}}\cdot c_{\text{mp}}\cdot k_F\cdot\sum_z A_{vz}.\end{aligned} \tag{43}$$

Finally, define

$$\mathrm{UB}(\ell) := c_{\text{up}}^{\ell}(\widetilde{UB})^{\ell} := c_{\text{up}}^{\ell}\left(c_{\text{rs}}I + c_{\text{mp}}\left(c_F k_h + k_F\right)A\right)^{\ell},$$

and then the norm of $\frac{\partial h_v^{\ell+1,\alpha}}{\partial h_u^{0,\beta}}$ is

$$\begin{aligned}
\left|\frac{\partial h_v^{\ell+1,\alpha}}{\partial h_u^{0,\beta}}\right| &\leq \sum_{p=1}^{d}\left|\frac{\partial \text{up}^{\ell+1,\alpha}}{\partial x_p}\right|\left(\sum_{r=1}^{d}\left|\frac{\partial \text{rs}^{\ell+1,p}}{\partial x_r}\right|\left|\frac{\partial h_v^{\ell,r}}{\partial h_u^{0,\beta}}\right|\right) + ((42)+(43))\cdot \mathrm{UB}(\ell)_{zu} \\
&\leq c_{\text{up}}c_{\text{rs}}\cdot \mathrm{UB}(\ell)_{vu} + c_{\text{up}}c_{\text{mp}}k_h c_F\sum_z A_{vz}\mathrm{UB}(\ell)_{zu} + c_{\text{up}}c_{\text{mp}}k_F\sum_z A_{vz}\mathrm{UB}(\ell)_{zu} \\
&= c_{\text{up}}^{\ell+1}\left(c_{\text{rs}}\cdot \widetilde{UB}_{vu}^{\ell} + \sum_z c_{\text{mp}}(k_h c_F + k_F)A_{vz}\widetilde{UB}_{zu}^{\ell}\right) \\
&= c_{\text{up}}^{\ell+1}\left(\left(c_{\text{rs}}I + c_{\text{mp}}(k_h c_F + k_F)A\right)\widetilde{UB}^{\ell}\right)_{vu} \\
&= \mathrm{UB}(\ell+1)_{vu},
\end{aligned}$$

which completes the proof. □

Underreaching. Finally, underreaching is defined as the inability of standard message passing with K layers to capture interactions of range greater than K. Alon & Yahav (2021) addresses this problem by adding a message passing layer on a fully connected graph at the last layer of the architecture, which empirically improves the performances but does not fundamentally solve the problem. A solution to this problem is letting the model decide the right depth of the architecture for the task, which is exactly what AMP does.

Below, we provide the extended formulation and proof of our theorem about the propagation of a message unchanged through the graph.

Theorem 3.2 *Let us consider a graph $g = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V} = \{1, \dots, n_g\}$ is the set of nodes connected via oriented edges $\mathcal{E} = \{(u, v)|u, v \in \mathcal{V}\}$, and $\mathcal{X} = \{\mathbf{x}_v \in \mathbb{R}^d | v \in \mathcal{V}\}$ is the set of d node attributes. The neighborhood of a node v is defined as $\mathcal{N}_v = \{u|(u, v) \in \mathcal{E}\}$. Further, let us define the following message aggregation scheme, which produces vectors $\boldsymbol{h}_v^{\ell} \in \mathbb{R}^d$ for node v at iteration ℓ, up to a maximum iteration K:*

$$\begin{aligned}
\boldsymbol{h}_v^0 &= \boldsymbol{x}_v \\
\boldsymbol{h}_v^{\ell} &= \sum_{u\in\mathcal{N}_v}\boldsymbol{F}(u,\ell)\odot \boldsymbol{h}_u^{\ell-1} \quad (\ell \geq 1),
\end{aligned} \tag{44}$$

where $\boldsymbol{F}(u, \ell) \in (0, 1)^d, \forall u \in \mathcal{V}, \ell \in [1, K]$ and $\odot$ is the element-wise product. Let us also assume that g contains two (not necessarily distinct) nodes v and u, and a walk $((v, v_2), \dots, (v_K, u))$ of length $K \geq 1$ exists between them. Then, $\forall \epsilon > 0$ there exists a parametrization of $\boldsymbol{F}$ such that $\boldsymbol{h}_u^K$ belongs to the closed ball $\mathcal{B}(\boldsymbol{x}_v, d\epsilon, ||\cdot||_1)$ of radius $d\epsilon$ centered at $\boldsymbol{x}_v$ under norm $||\cdot||_1$.

Proof. We prove the statement by induction over K.

Base case ($K = 1$): In this case, the source node v is a neighbor of the destination node u. It holds that

$$\begin{aligned}
\boldsymbol{h}_u^1 &= \sum_{u'\in\mathcal{N}_u}\boldsymbol{F}(u',1)\odot \boldsymbol{h}_{u'}^0 \\
&= \boldsymbol{F}(v,1)\odot \boldsymbol{x}_v + \sum_{u'\in\mathcal{N}_u\setminus\{v\}}\boldsymbol{F}(u',1)\odot \boldsymbol{x}_{u'}.
\end{aligned}$$

We choose $\boldsymbol{F}(v,1) = \mathbf{1} - \boldsymbol{\varepsilon}_v$ and $\boldsymbol{F}(u',1) = \boldsymbol{\varepsilon}_{u'}$, for some $\boldsymbol{\varepsilon}_v, \boldsymbol{\varepsilon}_{u'}$ with all components > 0, therefore

$$\begin{aligned}\boldsymbol{h}_u^1 &= (\mathbf{1} - \boldsymbol{\varepsilon}_v) \odot \boldsymbol{x}_v + \sum_{u' \in \mathcal{N}_u \setminus \{v\}} \boldsymbol{\varepsilon}_{u'} \odot \boldsymbol{x}_{u'} \\ &= \boldsymbol{x}_v + \sum_{u' \in \mathcal{N}_u} (-1)^{\delta(u',v)} \boldsymbol{\varepsilon}_{u'} \odot \boldsymbol{x}_{u'}.\end{aligned}$$

At this point, we can always choose $\boldsymbol{\varepsilon}_{u'}, \forall u' \in \mathcal{N}_u$ such that, $\forall i \in [1, \dots, d]$ it holds

$$-\epsilon \leq \sum_{u' \in \mathcal{N}_u} (-1)^{\delta(u',v)} \varepsilon_{u'}[i] x_{u'}[i] \leq \epsilon$$

noting that the inequality is due to the fact that values of $\boldsymbol{x}_{u'}$ can be negative. Therefore, for every i we have that

$$x_v[i] - \epsilon \leq h_u^1[i] \leq x_v[i] + \epsilon \equiv |h_u^1[i] - x_v[i]| \leq \epsilon \tag{45}$$

which implies $\boldsymbol{h}_u^1 \in \mathcal{B}(\boldsymbol{x}_v, d\epsilon, ||\cdot||_1)$.

Inductive case ($K > 1$): Let us consider a random walk of length K, and assume our proposition holds for values up to $K-1$. With the same arguments as before, we can write

$$\begin{aligned}\boldsymbol{h}_u^K &= (\mathbf{1} - \boldsymbol{\varepsilon}_{v_K}) \odot \boldsymbol{h}_{v_K}^{K-1} + \sum_{u' \in \mathcal{N}_u \setminus \{v_K\}} \boldsymbol{\varepsilon}_{u'} \odot \boldsymbol{h}_{u'}^{K-1} \\ &= \boldsymbol{h}_{v_K}^{K-1} + \sum_{u' \in \mathcal{N}_u} (-1)^{\delta(u',v_K)} \boldsymbol{\varepsilon}_{u'} \odot \boldsymbol{h}_{u'}^{K-1}.\end{aligned}$$

By the inductive hypothesis we know that there exists a parametrization of $\boldsymbol{F}$ (up to iteration $K-1$) such that $\boldsymbol{h}_{v_K}^{K-1} \in \mathcal{B}(\boldsymbol{x}_v, d\frac{\epsilon}{2}, ||\cdot||_1)$. As before, we pick $\boldsymbol{\varepsilon}_{u'}, \forall u' \in \mathcal{N}_u$ such that, $\forall i \in [1, \dots, d]$ it holds

$$-\frac{\epsilon}{2} \leq \sum_{u' \in \mathcal{N}_u} (-1)^{\delta(u',v_K)} \varepsilon_{u'}[i] h_{u'}^{K-1}[i] \leq \frac{\epsilon}{2}$$

and therefore, noting that $x_v[i] - \frac{\epsilon}{2} \leq h_u^{K-1}[i] \leq x_v[i] + \frac{\epsilon}{2}$ by the inductive hypothesis,

$$x_v[i] - 2 * \frac{\epsilon}{2} \leq h_u^K[i] \leq x_v[i] + 2 * \frac{\epsilon}{2} \equiv |h_u^K[i] - x_v[i]| \leq \epsilon \tag{46}$$

meaning $\boldsymbol{h}_u^K \in \mathcal{B}(\boldsymbol{x}_v, d\epsilon, ||\cdot||_1)$. □

It is worth noting that the use of a differentiable sigmoidal activation makes it impossible to propagate the exact same value, but this holds in the limit of $\epsilon \to 0$. In light of the above discussion, this theorem hints at AMP indeed mitigating oversmoothing, oversquashing, and underreaching by being able to propagate a single message unchanged, which is reminiscent of asynchronous message passing (Faber & Wattenhofer, 2023).

E. Hyper-parameters Details

The set of hyper-parameters tried for the baselines and for AMP (except for the number of layers) is the same of Gravina et al. (2023) and Tönshoff et al. (2023a).

In particular, for the baselines and AMP on the synthetic tasks, we used an Adam optimizer with learning rate 0.003, weight decay $1e-6$, embedding dimension in $[10, 20, 30]$, tanh activation function and a number of layers (except for AMP) in $[1, 5, 10, 20]$.

Instead, results on the peptides-func dataset rely on the following hyper-parameters: learning rate 0.001, dropout 0.1, $6, 8, 10$ layers for GCN, GINE and GatedGCN respectively, and similarly embedding dimensions 235, 160 and 95. We use a readout with a depth of 3 layers, and RWSE positional encodings, a batch size of 200 and 250 maximum training epochs.

Finally, results on the peptides-struct rely on a similar set of hyper-parameters with the following exceptions: number of layers $6, 10, 8$ for GCN, GINE, and GatedGCN respectively, Laplacian positional encodings, and embedding dimensions $235, 145, 100$.

Below, we report the additional hyper-parameters values that we introduced when evaluating AMP.

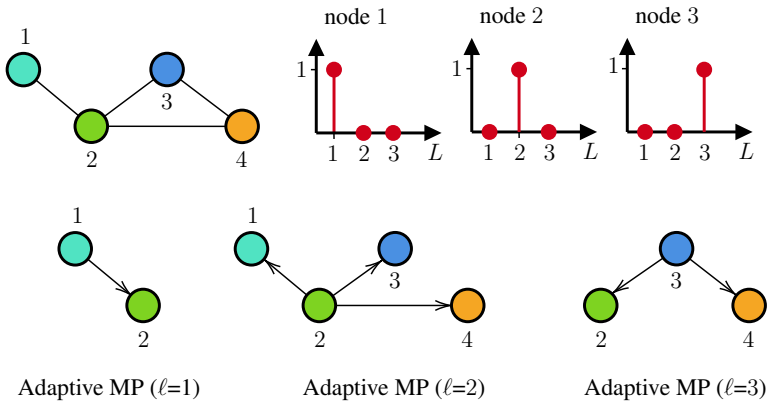

Figure 8. We sketch the idea of the proof of Theorem 3.2 for a graph of 4 nodes. When messages are filtered appropriately, a message can flow between node 1 and node 4 almost unchanged.

Synthetic Datasets To perform the grid search on AMP, in addition to the hyper-parameters range used for the base methods (with the exception of the depth), we tested four different distributions $q(\ell; \boldsymbol{\lambda})$: a Poisson with initial rate $\lambda = 10$, an FN with initial parameters $\mu = 10$ and $\sigma \in \{5, 10\}$, and a mixture of two folded normal distributions with initial parameters $\mu_1 = 5, \sigma_1 = 3, \mu_2 = 15, \sigma_2 = 3$. We fix the prior $p(\boldsymbol{\theta}^{\boldsymbol{\ell}}) = \mathcal{N}(\boldsymbol{\theta}^{\boldsymbol{\ell}}; \mathbf{0}, 10 * \mathbf{I})$, and we choose between three priors $p(L)$: an uninformative prior, a Poisson with rate 5, and a folded normal with parameters $\mu = 5$ and $\sigma = 10$. Finally, the message filtering function was chosen between one that does not filter at all, a function $f(\boldsymbol{x})$ acting on node features, and a function $f(\boldsymbol{h}^{\boldsymbol{\ell}})$ acting on node embeddings.

Chemical Datasets We tested three different distributions $q(\ell; \boldsymbol{\lambda})$: a Poisson with the initial rate $\lambda = 5$, a folded normal with initial parameters $\mu = 5$ and $\sigma = 1$, and a mixture of two folded normal distributions with initial parameters $\mu_1 = 1, \sigma_1 = 1, \mu_2 = 5, \sigma_2 = 1$. The tested message filtering functions and the number of layers are the same as the synthetic tasks, whereas $p(\boldsymbol{\theta}^{\boldsymbol{\ell}}) = \mathcal{N}(\boldsymbol{\theta}^{\boldsymbol{\ell}}; \mathbf{0}, 10 * \mathbf{I})$ for peptides-func and $p(\boldsymbol{\theta}^{\boldsymbol{\ell}}) = \mathcal{N}(\boldsymbol{\theta}^{\boldsymbol{\ell}}; \mathbf{0}, 1 * \mathbf{I})$ for peptides-struct. Also, following Tönshoff et al. (2023a), we use dropout in the output layers, after each activation function, and set its value to 0.1.

F. Node Classification Results

To demonstrate AMP's applicability on a broader range of tasks, we report node classification performances on different datasets with varying degrees of homophily, following the data split strategy of Zhu et al. (2020). We train a GCN by testing the following hyper-parameters: embedding size in $[8, 32, 64]$ and number of layers in $[2, 4, 6]$, Adam optimizer with learning rate 0.01, a maximum of 2000 training epochs and early stopping with patience 250 on the validation performance. The hyper-parameters were the same for AMP_{GCN} with the exception for the number of layers, which is replaced by the distributions tried for the chemical datasets.

Table 3 clearly shows that wrapping our framework around a GCN always grants a performance improvement. Only for reference, we report other results taken by (Zhu et al., 2020). However, please note that there an homophilic/heterophilic graph does not imply that the task is long-range, nor the converse is necessarily true.

G. Analysis of AMP's Predictions

We further delve into the predictions of AMPAMP on a few representative cases. In Figure 9 (left), we report the mean predictions of the best performing GCN and AMP_{GCN} runs on the Diameter dataset, where the shaded bands denote the minimal and maximal errors that both models make. Similarly, Figure 9 (right) shows the same plot but for ADGN and AMP_{ADGN} on Eccentricity. We can see how AMP generates an almost ideal average prediction on Diameter and is able to deal with higher eccentricity than the base model (despite an almost identical error being achieved in the latter case).

	Texas	Wisconsin	Actor	Squirrel	Chameleon	Citeseer	Pubmed	Cora
Hom. ratio h	**0.11**	**0.21**	**0.22**	**0.22**	**0.23**	**0.74**	**0.8**	**0.81**
#Nodes	183	251	7,600	5,201	2,277	3,327	19,717	2,708
#Edges	295	466	26,752	198,493	31,421	4,676	44,327	5,278
#Classes	5	5	5	5	5	7	3	6
H_2GCN-1	84.86±6.77	86.67±4.69	35.86±1.03	36.42±1.89	57.11±1.58	77.07±1.64	89.40±0.34	86.92±1.37
H_2GCN-2	82.16±5.28	85.88±4.22	35.62±1.30	37.90±2.02	59.39±1.98	76.88±1.77	89.59±0.33	87.81±1.35
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	76.04±1.30	88.45±0.50	86.90±1.04
GCN-Cheby	77.30±4.07	79.41±4.46	34.11±1.09	43.86±1.64	55.24±2.76	75.82±1.53	88.72±0.55	86.76±0.95
MixHop	77.84±7.73	75.88±4.90	32.22±2.34	43.80±1.48	60.50±2.53	76.26±1.33	85.31±0.61	87.61±0.85
GraphSAGE+JK	83.78±2.21	81.96±4.96	34.28±1.01	40.85±1.29	58.11±1.97 76.05±1.37	88.34±0.62	85.96±0.83	
Cheby+JK	78.38±6.37	82.55±4.57	35.14±1.37	45.03±1.73	63.79±2.27	74.98±1.18	89.07±0.30	85.49±1.27
GCN+JK	66.49±6.64	74.31±6.43	34.18±0.85	40.45±1.61	63.42±2.00 74.51±1.75	88.41±0.45	85.79±0.92	
GAT	58.38±4.45	55.29±8.71	26.28±1.73	30.62±2.11	54.69±1.95	75.46±1.72	84.68±0.44	82.68±1.80
GEOM-GCN*	67.57	64.12	31.63	38.14	60.90 77.99	90.05	85.27	
MLP	81.89±4.78	85.29±3.61	35.76±0.98	29.68±1.81	46.36±2.52	72.41±2.18	86.65±0.35	74.75±2.22
GCN	52.73±5.98	44.90±7.38	28.02±0.59	27.73±0.91	41.89±1.83	72.71±1.54	87.62±0.59	84.24±1.44
AMP_{GCN}	81.46±3.06	80.45±4.47	34.14±0.71	35.28±0.95	48.56±1.70	75.07±1.29	89.67±1.29	85.68±1.52

Table 3. Node classification results on heterophilic and homophilic graphs. Results are taken from Zhu et al. (2020).

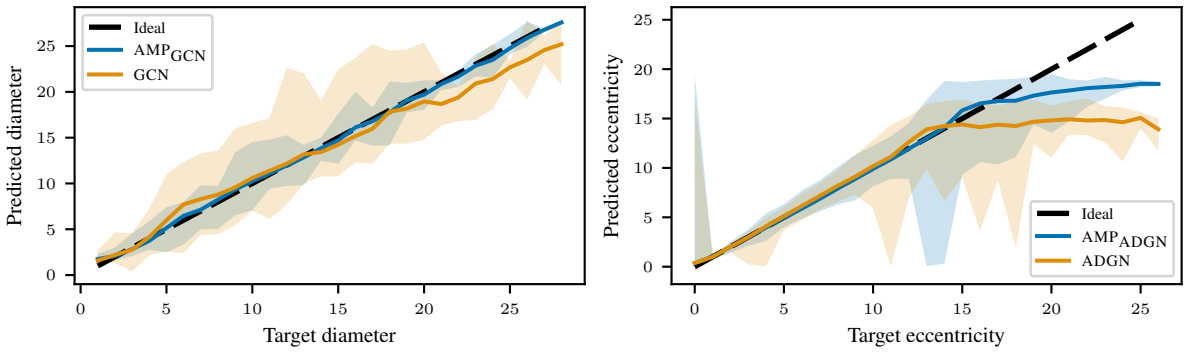

Figure 9. We report the average predicted *vs* target graph diameter and node eccentricities for the GCN and ADGN message passing architectures, respectively. Shaded bands denote minimal and maximal errors for each prediction.

H. Tuned Depth of Base Models

We report, for the base architectures we have tested within AMP, the number of layers selected by the hyper-parameter search in the original papers. For the synthetic datasets, we obtained this information directly from the authors (Gravina et al., 2023), whereas for the chemical datasets this information was already available in Tönshoff et al. (2023a).

I. Impact of Positional and Structural Encodings

For completeness, we investigate the impact of the positional and structural encodings on performances. Our findings show that, on peptides-func, the performance gain compared to AMP is marginal on GCN and GINE, but it becomes substantial on GatedGCN. In all cases, even without the additional encodings, AMP has better or comparable performance than the base models. We observe a similar trend on peptides-struct, with AMP_{GCN} suffering the most from the absence of extra information; here, the result is significantly worse than the base model.

J. Ablation Study on the Depth

Performing an ablation study about the depth of AMP implies that we perform model selection across a wide range of (fixed) depths and we learn the importance of each layer. This is an a-posteriori analysis, meaning that we the range of layers

	Diameter	SSSP	Eccentricity	peptides-func	peptides-struct
GCN	5	5	10	6	6
GIN	1	1	1	-	-
ADGN	10	20	20	-	-
GINE	-	-	-	8	10
GatedGCN	-	-	-	10	8

Table 4. Best number of layers selected during hyper-parameter tuning for the models considered within AMP. These values are taken from the original papers or provided by their authors.

	Method	peptides-func Test AP ↑	peptides-struct Test MAE ↓
With PE/SE	GCN	0.6860 ± 0.0050	0.2460 ± 0.0007
	GINE	0.6621 ± 0.0067	0.2473 ± 0.0017
	GatedGCN	0.6765 ± 0.0047	0.2477 ± 0.0009
	AMP_{GCN}	0.7161 ± 0.0047	0.2446 ± 0.0026
	AMP_{GINE}	0.7065 ± 0.0105	0.2468 ± 0.0026
	$\text{AMP}_{\text{GatedGCN}}$	0.6943 ± 0.0046	0.2480 ± 0.0012
W/O	AMP_{GCN}	0.7102 ± 0.0074	0.2524 ± 0.0035
	AMP_{GINE}	0.6994 ± 0.0098	0.2477 ± 0.0017
	$\text{AMP}_{\text{GatedGCN}}$	0.6725 ± 0.0113	0.2466 ± 0.0008

Table 5. Mean test scores and standard deviation averaged over 4 final runs on the chemical datasets.

to try is suggested by our previous results; the whole point of our contribution is that it may be difficult to find the exact range of message-passing layers that the task needs. The goal of this section is to understand if it is worth fixing the depth after a sensible range of layers has been found by AMP.

We use the information from Figure 4 to set up a reasonable range of fixed number of message passing layers to try, and then we ran the experiments again on the real-world chemical datasets. We also fix the other hyper-parameters to the best configuration found, for each model and dataset, by our model selection procedure.

	Method	peptides-func Test AP ↑	peptides-struct Test MAE ↓
	AMP_{GCN}	0.7161 ± 0.0047	0.2446 ± 0.0026
	AMP_{GINE}	0.7065 ± 0.0105	0.2468 ± 0.0026
	$\text{AMP}_{\text{GatedGCN}}$	0.6943 ± 0.0046	0.2480 ± 0.0012
Fixed Depth	AMP_{GCN}	0.7076 ± 0.0059	0.2497 ± 0.0009
	AMP_{GINE}	0.6999 ± 0.0041	0.2481 ± 0.0014
	$\text{AMP}_{\text{GatedGCN}}$	0.6750 ± 0.0029	0.2493 ± 0.0013

Table 6. Mean test scores and standard deviation averaged over 4 final runs on the chemical datasets.

It appears that fixing the depth does not allow to obtain better performances than the fully adaptive AMP.