Bridging Graph Neural Networks and Large Language Models: A Survey and Unified Perspective

Anonymous authors
Paper under double-blind review

Abstract

Decoder-Transformers have achieved remarkable success and have laid the groundwork for the development of Large Language Models (LLMs). At the core of these models is the *self-attention matrix*, which allows different tokens to interact with each other. This process is remarkably similar to the message-passing mechanism used in Graph Neural Networks (GNNs), and as such decoder-Transformers suffer many of the optimization difficulties studied extensively in the GNN literature. In this paper, we present a unified graph perspective that bridges the theoretical understanding of decoder-Transformers and GNNs. We systematically examine how well-known phenomena in GNNs, such as **over-smoothing** and **over-squashing**, directly manifest as analogous issues like **rank collapse** and **representational collapse** in deep Transformer architectures. By interpreting Transformers' self-attention as a learned adjacency operator, we reveal shared underlying principles governing signal propagation and demonstrate how insights from one field can illuminate challenges and solutions in the other. We analyze the role of architectural components like residual connections, normalization, and causal masking in these issues. We aim to provide a framework for understanding how information flows through deep learning models that perform sequence *mixing* through an adjacency operator, and to highlight areas for cross-pollination of research, as well as to provide a comprehensive reference for researchers interested in the underpinnings of these architectures.

1 Introduction

The remarkable success of large language models (LLMs) in natural language processing, code generation, and multimodal reasoning has spurred intensive research into understanding their internal mechanisms and inductive biases. At the heart of modern LLMs lies the Transformer architecture (Vaswani et al., 2017), which leverages self-attention to capture both short and long-range dependencies in sequences. Despite their empirical success, little is understood about their learning dynamics, how their expressive power scales with depth and width (sequence length), and how to diagnose and mitigate different failure modes. In parallel, the field of graph neural networks (GNNs) has matured into a rich intersection of spectral graph theory, dynamical systems, and message-passing paradigms (Gilmer et al., 2017; Defferrard et al., 2016; Kipf & Welling, 2017). GNNs process relational data by propagating node features along graph edges. While these two architectures seem distinct, it has been argued (Joshi, 2025; Pappone, 2025) that Transformers are an instance of a GNN, which is due to the analogous way that tokens (or nodes) are mixed and iteratively updated through successive layers of the network. This similarity in information processing suggests that many insights and optimization-related problems from GNN theory, such as over-smoothing (Oono & Suzuki, 2020) and over-squashing (Topping et al., 2021), may shed light on analogous phenomena in decoder-Transformers, and hence, LLMs.

In this perspective and survey paper, we advocate for a *graph approach* to understanding LLMs, one that unifies sequence and graph modeling under the common umbrella of **mixing architectures**. We organize our exposition around the core idea of **information propagation**. In particular, we elaborate on how the normalized adjacency matrix – the key component of mixing architectures – affects information flow in *depth* (in the number of layers) and *width* (in the diameter of the graph or sequence). We examine how concepts like **over-smoothing** and **over-squashing**, well-studied in GNNs, have manifested as analogous

challenges named **rank collapse** and **representational collapse** in Transformer literature. By drawing these connections and citing the most relevant work in both bodies of literature, we aim to provide the reader with a cohesive perspective on the shared underlying principles governing the behavior of both GNNs and decoder-Transformers, highlighting how advancements in one domain can inform theoretical understanding and practical improvements in the other.

2 Background and Foundations

In this section, we aim to provide the reader with the relevant background in both sequence modeling and GNNs. We begin by introducing foundational sequence-modeling architectures – starting with Recurrent Neural Networks and their gradient-flow challenges, progressing to the attention mechanism, and culminating in the Transformer architecture. We then provide an overview of graph neural networks and their message-passing foundations, as well as their associated challenges.

2.1 Background on Sequence Modeling

Sequence modeling is a foundational concept in machine learning that deals with causally ordered data, such as text (Achiam et al., 2023), speech (Goel et al., 2022), or time series (Lim & Zohren, 2021). While such sequential tasks have often relied on handcrafted priors or simple linear models with closed-form solutions, the increased availability of compute power and data has led to the advent of deep learning approaches. Here, we will cover some of the most common frameworks and architectures in this domain.

2.1.1 Fundamentals of Sequence Modeling

Deep neural networks build intermediate representations by applying a series of iterated nonlinear transformations. This process enables the progressive abstraction of raw input data into higher-level features. Many architectures in this domain can be viewed as sequence-to-sequence mappings,

$$\mathbf{g}_{\boldsymbol{\theta}} : \mathbf{x}^{(0:t)} \mapsto \mathbf{y}^{(0:t)}, \tag{1}$$

where the input and output sequences up to time step t are defined as

$$\begin{aligned} \mathbf{x}^{(0:t)} &= (\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}), \\ \mathbf{y}^{(0:t)} &= (\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(t)}). \end{aligned} \tag{2}$$

Here, each input vector $\mathbf{x}^{(k)} \in \mathbb{R}^d$ represents the data at time step k, while each output vector $\mathbf{y}^{(k)} \in \mathbb{R}^m$ corresponds to the model's prediction at that time step. The parameter set $\boldsymbol{\theta}$ corresponds to all learnable weights of the model.

In sequence modeling, the inherent temporal (or causal) structure of the data is captured by having the *hidden state* $\mathbf{h}^{(t)}$ be a function of all previous information from the input sequence:

$$\mathbf{h}^{(t)} = \mathbf{g}_{\boldsymbol{\theta}}^{\text{enc}}\left(\mathbf{x}^{(0:t)}\right) \tag{3}$$

where $\mathbf{x}^{(0:t)}$ denotes the collection of all inputs from time 0 to t, and $\mathbf{g}_{\boldsymbol{\theta}}^{\text{enc}}$ is a learnable encoder function. The output at time step t is then generated by the decoder,

$$\mathbf{y}^{(t)} = \mathbf{g}_{\boldsymbol{\theta}}^{\text{dec}}\left(\mathbf{h}^{(0:t)}\right), \tag{4}$$

with $\mathbf{g}_{\boldsymbol{\theta}}^{\text{dec}}$ representing the learnable decoder function.

2.1.2 Recurrent Neural Networks

A Recurrent Neural Network is designed to handle sequential data by iteratively updating a hidden state $\mathbf{h}^{(k)} \in \mathbb{R}^{d_h}$ according to the input at each sequence position. Historically, RNN-based architectures have been extensively used in sequence modeling (e.g. Graves, 2013; Chung et al., 2014; Sutskever et al., 2014), achieving strong results on a variety of sequence modeling tasks. One of the simplest RNN variants is the Elman RNN (Elman, 1990), which updates its internal state according to

$$\mathbf{h}^{(t)} = \sigma\left(\mathbf{W}_h\, \mathbf{h}^{(t-1)} + \mathbf{W}_x\, \mathbf{x}^{(t)} + \mathbf{b}_h\right), \tag{5}$$

$$\mathbf{y}^{(t)} = \sigma\left(\mathbf{W}_y\,\mathbf{h}^{(t)} + \mathbf{b}_y\right), \tag{6}$$

where $\mathbf{h}^{(t)}$ represents the hidden state at time t, $\mathbf{y}^{(t)}$ is the output at that time, and $\mathbf{W}_h$, $\mathbf{W}_y$ and $\mathbf{W}_x$, are weight matrices with $\mathbf{b}_h$ and $\mathbf{b}_y$ as the bias vectors. The activation function $\sigma(\cdot)$ introduces nonlinearity to the state update. Although basic RNN models are expressive, they demonstrate certain pathologies during training, including vanishing and exploding gradients, motivating the development of more advanced architectures such as Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014). These issues are discussed more formally in the following subsection.

2.1.3 The Vanishing and Exploding Gradient Problem

While RNNs have been successfully trained and used in short sequences, long-range sequence modeling is a challenging problem as a consequence of the *exploding and vanishing gradients problem* (Bengio, 1994; Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013). This occurs because, during backpropagation through time (BPTT), the gradients of the hidden states are computed by repeatedly applying the chain rule across many time steps. As a result, the process involves multiplying a long sequence of Jacobians that are correlated to each other due to being the Jacobian of the same update rule. When this sequence is long, the accumulated product can either shrink rapidly toward zero or blow up uncontrollably, which leads to uncontrolled gradient propagation. In particular, following Pascanu et al. (2013), we let $\mathcal{L}^{(t)}$ denote the loss at time step t. The total loss over a sequence of length T is then given by

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}^{(t)}. \tag{7}$$

Accordingly, the gradient of the total loss with respect to the parameters $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T}\sum_{k=1}^{t}\left(\frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \boldsymbol{\theta}}\right). \tag{8}$$

As identified by Pascanu et al. (2013), a major issue arises from the product Jacobian,

$$\mathbf{J} = \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}} = \prod_{i=k+1}^{t} \mathbf{J}_i. \tag{9}$$

If $\|\mathbf{J}_k\|_2 \approx \lambda$ for all layers, then

$$\|\mathbf{J}\|_2 \leq \lambda^{t-k}. \tag{10}$$

Thus, it is necessary that $\lambda \approx 1$ for gradients to neither vanish nor explode—a condition often referred to as the *edge of chaos*. Specifically, if $\lambda < 1$ for certain terms in the gradient sum, these terms will disappear and will be absent from the gradient calculation. Conversely, if $\lambda > 1$, these gradients will explode and the gradient norm will increase greatly. RNN architectures such as LSTMs (Hochreiter & Schmidhuber, 1997) or GRUs (Cho et al., 2014) mitigate vanishing gradients via gated additive recurrence, but remain susceptible to exploding gradients and also struggle with long sequences. As such, a very relevant line of research has been dedicated to fixing this issue, examples of which include norm-preserving weight constraints (Arjovsky et al., 2016; Henaff et al., 2016; Chang et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019), initialization schemes (Tallec & Ollivier, 2018), and physics-inspired inductive biases (Erichson et al., 2020; Keller et al., 2023).

A fundamental limitation of RNNs is how information must propagate: at each time step, the network must compute its hidden state based on the output of the previous time step, resulting in a sequential bottleneck. This inherent temporal dependency means that computations cannot be parallelized across the sequence—each step must wait for the one before it to complete. As a result, training becomes slow, especially for long sequences. By contrast, Transformers enabled parallelization during training, and therefore were able to take advantage of modern parallel hardware (e.g., GPUs or TPUs). Recently, a strand of work

has adapted RNNs by replacing the non-linear recurrence with a linear counterpart followed by an MLP block, enabling efficient parallel computations during training, while maintaining the fast – compared to Transformers – inference speed typical to RNNs. The earliest instances of this framework are Structured State Space Models (SSMs), which rely on a parametrization derived from discretizing a continuous time linear dynamical system and rely on a deterministic initialization of the recurrent weights (known as the HiPPO matrix (Gu et al., 2020)). The effectiveness of these models has been linked to the eigenspectrum of the Jacobian matrix, as demonstrated in Orvieto et al. (2023) which replicates the earlier finding in the setting of linear RNNs while benefiting from fast training and inference through the use of parallel scan algorithms (Martin & Cundy, 2017). Interestingly, the design of Linear Recurrent Units (LRUs) in Orvieto et al. (2023) shares conceptual similarities with techniques from the reservoir computing literature (Jaeger, 2001; Sutskever et al., 2013), particularly in how recurrent dynamics are leveraged to guarantee stability (also known as the "echo-state property").[1]

2.1.4 Transformers

The introduction of the attention mechanism has led to the Transformer architecture (Vaswani et al., 2017). This model relies on the self-attention mechanism: in a multi-head Transformer, each self-attention head applies scaled dot-product attention to calculate similarity scores between different inputs in the sequence. Let $\mathbf{X} \in \mathbb{R}^{n \times f}$ denote a sequence of n feature vectors, each with dimension f. For the ith head we compute query, key and value matrices using projection matrices $\mathbf{W}_Q^i \in \mathbb{R}^{f \times d_k}, \mathbf{W}_K^i \in \mathbb{R}^{f \times d_k}$ and $\mathbf{W}_V^i \in \mathbb{R}^{f \times d_v}$ as

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_Q^i, \qquad \mathbf{K}_i = \mathbf{X}\mathbf{W}_K^i, \qquad \mathbf{V}_i = \mathbf{X}\mathbf{W}_V^i. \tag{11}$$

The scaled dot-product attention is then defined as

$$\mathbf{h}_i = \text{Attention}\left(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i\right) = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i, \tag{12}$$

Each head learns an attention function that captures complex dependencies within the input data. With multiple heads, the model jointly attends to different subspaces of the input sequence. The outputs from each head are concatenated to obtain a joint representation:

$$\text{MultiHead}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \text{Concat}(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_H)\mathbf{W}_O, \tag{13}$$

for output matrix $\mathbf{W}_O \in \mathbb{R}^{Hd_v \times d_o}$. Merging all heads' outputs through a linear function in this way produces a latent representation that encodes the most relevant information from the input features.

In autoregressive tasks, such as next-token prediction, a causal mask is essential to prevent a token from accessing future information. To achieve this, the standard attention computation is modified by adding a mask $\mathbf{M}$ to the scaled dot-products, as shown in

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}, \tag{14}$$

where the mask is defined by

$$M_{ij} = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i, \end{cases} \tag{15}$$

thereby ensuring that positions corresponding to future tokens obtain zero attention after the softmax. This mechanism is critical for maintaining the autoregressive property by ensuring that each token is computed solely on its preceding context, which preserves the sequential nature of the data.

Despite its strengths, self-attention runs in $\mathcal{O}(n^2 d + nd^2)$ time and requires $\mathcal{O}(n^2 + nd)$ memory, for dimension d (assuming $d = f = d_k = d_v$) and sequence length n. This arises since forming the score matrix

[1] It should be noted that the propagation of gradients through the network has also been amply studied in the feedforward and deep linear network regime. This includes the seminal work of Saxe et al. (2013); Schoenholz et al. (2016); Pennington et al. (2017), which characterized the dynamics of learning based on initialization and choice on nonlinearities using ideas from Random Matrix Theory (RMT).

$\mathbf{Q}\mathbf{K}^T$ costs $\mathcal{O}(n^2 d)$ and the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ projections cost $\mathcal{O}(nd^2)$. To cope with the computational challenges, these operations are mapped to huge, parallel batched matrix multiplications that GPUs handle with high arithmetic intensity, predictable memory access, and static tensor shapes that allow aggressive kernel fusion. This marriage between algorithm and hardware (also termed the "hardware lottery" (Hooker, 2021)) has led to the effective parameter scaling of attention-based models in practice (Achiam et al., 2023).

2.2 Background on Graph Neural Networks

Many real-world systems, from molecular structures and protein–protein interaction networks in biology to citation graphs and social networks in information science, naturally manifest as graphs, where nodes represent entities and edges encode relationships. Learning predictive models on such graph-structured data is crucial for tasks like node classification, link prediction, and graph-level regression, yet traditional machine learning methods that assume independent and identically distributed (i.i.d.) samples cannot exploit the rich relational inductive biases inherent in these domains.

Graph Neural Networks (GNNs) (e.g. Sperduti, 1993; Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2013; Defferrard et al., 2016) have emerged as a powerful framework for learning on graphs by combining spectral and spatial insights. In the spectral view, graph convolutions are defined via eigendecompositions of a graph shift operator (e.g., the graph Laplacian), enabling filtering in the frequency domain (Bruna et al., 2013; Defferrard et al., 2016); in the spatial view, methods directly aggregate information from each node's local neighborhood. Most contemporary architectures adopt a message-passing paradigm, wherein nodes iteratively exchange and transform "messages" with their neighbors—culminating in the Message-Passing Neural Network (MPNN) framework (Gilmer et al., 2017). Although these two lines of work have been developed independently, it is reasonable to use MPNNs as an umbrella framework to describe both. In this subsection, we will provide the necessary background on MPNNs and discuss related challenges.

2.2.1 Message Passing Neural Networks

Let a graph $\mathbf{G}$ be a tuple $(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of nodes and $\mathbf{E}$ is the set of edges. An edge from node $u \in \mathbf{V}$ to node $v \in \mathbf{V}$ is denoted by $(u, v) \in \mathbf{E}$. The graph's connectivity is captured by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where n is the number of nodes. We assume that $\mathbf{G}$ is undirected and that each node v is associated with a feature vector $\mathbf{h}_v \in \mathbb{R}^d$. A GNN can be broadly defined as a function

$$\mathbf{f}_{\boldsymbol{\theta}} : (\mathbf{G}, \{\mathbf{h}_v\}) \mapsto \mathbf{y}, \tag{16}$$

with parameters $\boldsymbol{\theta}$ learned via gradient descent, and $\mathbf{y}$ representing a prediction at the node or graph level. These functions typically adopt the message-passing paradigm, which in computing the latent representation at the kth-layer can be formulated as:

$$\mathbf{h}_u^{(k)} = \phi^{(k)}\Big(\mathbf{h}_u^{(k-1)}, \psi^{(k)}\big(\{\mathbf{h}_v^{(k-1)} : (u, v) \in \mathbf{E}\}\big)\Big), \tag{17}$$

for $k = 1, \ldots, K$. Here, $\psi^{(k)}$ is a permutation invariant aggregation function, and $\phi^{(k)}$ combines the incoming messages from a node's neighbors with its previous embedding to produce an updated representation. A common aggregation function is given by:

$$\psi^{(k)}\Big(\{\mathbf{h}_v^{(k-1)} : (u, v) \in \mathbf{E}\}\Big) = \sum_{\mathbf{v}} \tilde{\mathbf{A}}_{uv}\, \mathbf{h}_v^{(k-1)}, \tag{18}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A}\, \mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$.

A widely used instance of GNNs is the Graph Convolutional Network (GCN) (Kipf & Welling, 2017), which can be interpreted as an MPNN. Specifically, one can express the node features in matrix form $\mathbf{H}^{(k)} \in \mathbb{R}^{n \times d_k}$, and the GCN update equation is

$$\mathbf{H}^{(k)} = \sigma(\hat{\mathbf{A}}\, \mathbf{H}^{(k-1)}\, \mathbf{W}^{(k-1)}), \tag{19}$$

with

$$\hat{\mathbf{A}} = (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}}, \tag{20}$$

and $\sigma(\cdot)$ denoting a nonlinearity. Another popular MPNN instance is that of Graph Attention Networks (GATs) (Veličković et al., 2018), where a learned adjacency matrix replaces the fixed normalized adjacency to dynamically modulate connectivity while preserving key spectral properties. In particular, the aggregation is computed as

$$\mathbf{h}_i^{(l+1)} \;=\; \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)}\,\mathbf{W}\,\mathbf{h}_j^{(l)}\right), \tag{21}$$

where the attention coefficients are given by

$$\alpha_{ij}^{(l)} \;=\; \frac{\exp\big(\mathrm{LeakyReLU}\big(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i^{(l)} \,\|\, \mathbf{W}\mathbf{h}_j^{(l)}]\big)\big)}{\sum_{n\in\mathcal{N}(i)} \exp\big(\mathrm{LeakyReLU}\big(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i^{(l)} \,\|\, \mathbf{W}\mathbf{h}_n^{(l)}]\big)\big)}. \tag{22}$$

Multi-head attention is then introduced by computing K independent sets of coefficients $\{\alpha_{ij}^{(l),k}\}$ (with corresponding weights $\mathbf{W}^{(k)}$, $\mathbf{a}^{(k)}$) and either concatenating or averaging their outputs:

$$\mathbf{h}_i^{(l+1)} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l),k}\,\mathbf{W}^{(k)}\mathbf{h}_j^{(l)}\right) \quad \text{or} \quad \frac{1}{K}\sum_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l),k}\,\mathbf{W}^{(k)}\mathbf{h}_j^{(l)}\right). \tag{23}$$

Other notable MPNNs include GraphSAGE (Hamilton et al., 2017), Residual Gated Graph Convolutional Networks (Gated GCNs) (Bresson & Laurent, 2017), and Graph Isomorphism Networks (GIN) (Xu et al., 2018).

3 Signal Propagation in GNNs

In this section, we turn our attention to the mechanisms that govern information propagation in common GNN architectures. In Section 3.1 we introduce well-known issues around stacking GNN layers which lead to the notion of *over-smoothing* used to explain the degradation in performance for common GNN architectures, and in Section 3.2 we introduce the issues of *over-squashing* and long-range communication between distant nodes. We make use of the original definitions given to these terms, but also note recent work that highlights the role of vanishing gradients in these concepts (Arroyo et al., 2025) and discuss the interplay between them (Arnaiz-Rodriguez & Errica, 2025). To help unify nomenclature between GNN and Transformer literature, we will separate our discussion into considerations of *depth* (primarily addressed as the issue of over-smoothing in GNN literature) and *width* (addressed as over-squashing in GNN literature). These terms broadly relate to the computational tree, which follows a very similar structure between GNNs and Transformers, thus affording this unification of issues. We visualize an example of such a computational tree in Figure 1. We highlight here a potential nomenclature clash: in this work, "width" does not refer to hidden dimension size but refers approximately to the width of the computational tree, which relates to node receptive field and sequence length in GNNs and Transformers respectively.

3.1 The Depth Regime in GNNs: Over-Smoothing and Vanishing Gradients

Early Approaches: Over-Smoothing. GNNs are known to severely struggle and break down after the application of a few message-passing layers. Until very recently, this phenomenon has been attributed to the idea of *over-smoothing*. Over-smoothing refers to the phenomenon whereby the repeated application of message-passing operations in GNNs causes the node features to become increasingly similar, visualized in Figure 2. Some early works that analyze this theoretically are Oono & Suzuki (2020); Cai & Wang (2020). A common way to quantify this effect is via the *unnormalized Dirichlet energy*[2] of the node representations, which has been used in a number of works in the area (Rusch et al., 2022). Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with n nodes and node features $\mathbf{H} \in \mathbb{R}^{n\times d}$, the unnormalized Dirichlet energy is defined as

$$\mathcal{E}(\mathbf{H}) = \sum_{(u,v)\in\mathbf{E}} \|\mathbf{h}_u - \mathbf{h}_v\|^2, \tag{24}$$

[2]Note that other node similarity measures exist, including the normalized Dirichlet energy, used in the original works of (Cai & Wang, 2020) or other measures such as the one employed in (Wu et al., 2024).

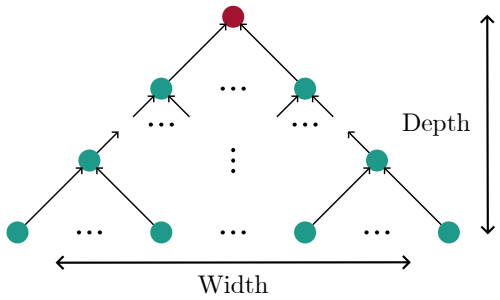

Figure 1: The computational tree for a given root node (in red) for GNNs and Transformers. For GNNs, the children of each node are typically its immediate neighbors in the underlying graph at the previous layer. For Transformers, the children of each node are typically the hidden representations of each (previous, in the case of a causal mask) sequence position at the previous layer. Depth refers to the depth of this tree (number of layers). Width loosely refers to the width of the tree: in GNNs this relates to the node receptive field, and in Transformers to sequence length.

where $\mathbf{h}_u$ and $\mathbf{h}_v$ denote the feature vectors for nodes u and v, respectively.[3] This energy measures the variability between features of adjacent nodes; a lower energy indicates smoother (i.e., more similar) node representations. In many GNN architectures, as more layers are stacked, the iterative aggregation tends to minimize $\mathcal{E}(\mathbf{H})$ to the extent that all node features converge toward a common value, as shown in Figure 3 (left). The common consensus in the literature has been that while some degree of smoothing is beneficial, excessive smoothing leads to a loss of discriminative power, thereby hindering performance in downstream tasks. Several works have aimed to prevent over-smoothing in GNNs, from simpler approaches (Rong et al., 2019; Zhao & Akoglu, 2019) to physics-inspired methods (Chamberlain et al., 2021; Bodnar et al., 2022; Di Giovanni et al., 2022; Bamberger et al., 2024). Other approaches that have emerged over the years include Chen et al. (2020); Fang et al. (2023); Lu et al. (2024); Maskey et al. (2024); Roth et al. (2024).

Figure 2: Illustration of the effect of feature collapse on a graph, commonly referred to as over-smoothing. From left to right more GNN layers are applied and node signals, here depicted by color, become more uniform over the graph. Nodes are sized by degree for illustration purposes.

Issues with Information Propagation and Optimization at Large Depth. Recently, the degradation in performance of GNNs with a large number of layers has also been associated with fundamental issues of signal propagation and optimization. In particular, Arroyo et al. (2025) highlight the importance of vanishing gradients and the *zero-collapse* phenomenon in GCNs and GATs. The central argument of this work is that while feature collapse is an issue, it should be possible for MPNNs to learn weights that counteract this smoothing effect: however, due to issues of vanishing (and exploding gradients), this is often not the case. This work shows that the same phenomena that gives rise to the over-smoothing effect (the product

[3]Note that the above assumes unitary edge weights.

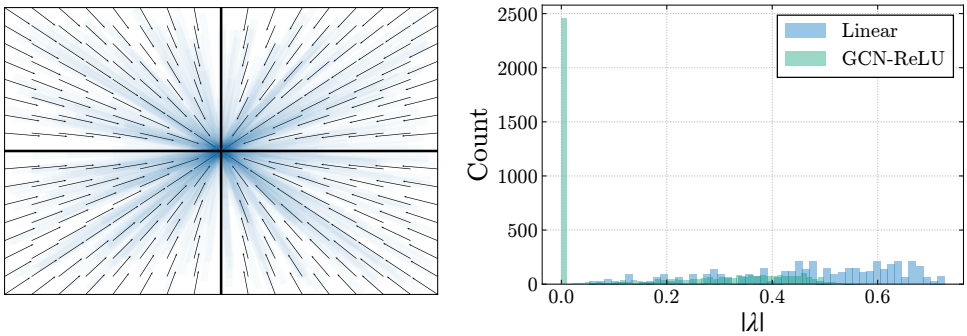

Figure 3: **Left:** 2-Dimensional random projection showing node feature evolution for a GCN on the Cora dataset. **Right:** Histogram of eigenvalue modulus of the Jacobian for linear, linear convolutional, and nonlinear GCN layers. Figures adapted from Arroyo et al. (2025).

Jacobian) also gives rise to vanishing gradients. Similar notions were explored in Arnaiz-Rodriguez & Errica (2025), who highlight both that over-smoothing is not an issue in all GNNs, and that over-smoothing does not necessarily cause performance degradation. We note that some of these topics were later explored in Keriven (2025).

Mathematically, the main reason why GCNs and GATs are in expectation more vulnerable to vanishing gradients than standard MLPs is the presence of an additional message-passing step using the normalized adjacency matrix: $\hat{\mathbf{A}}$ in the simplest case of GCN shown in Equation (19). From a dynamical systems perspective, this extra step induces a contraction in the spectrum of the Jacobian, which makes the chain of Jacobians in the backward pass dissipate information at a much higher rate in the backward pass of the network. This is illustrated in Figure 3 (right), together with an example of zero-collapsing of 2-dimensional random projection of node features on the Cora dataset evolved with a GCN. Figure 3 (left) additionally visualizes the eigenvalue distribution for a linear layer, a GCN without an activation function, and a GCN with ReLU activation; this demonstrates the effect of the normalized adjacency, causing most of the spectrum to shrink to near zero. To this end, we highlight a number of works that have employed ideas from vanishing gradients or dynamical isometry (Rusch et al., 2022; 2023; Epping et al., 2024; Scholkemper et al., 2024; Wang et al., 2025). While some of these works have attributed the success of their models to a superior inductive bias (e.g. through "physically-inspired methods"), it is more fundamentally tied to other optimization-related phenomena. We also highlight that the divergence effect reported in Arnaiz-Rodriguez & Errica (2025) can be understood from the perspective of optimization. This is due to the fact that the layer-wise Jacobians of some frequently used GNNs (e.g. GIN (Xu et al., 2018) or Gated-GCN (Bresson & Laurent, 2017)) are extremely unstable and will lead to gradient explosion at even moderate depths (see Arroyo et al., 2025).

3.2 The Width Regime in GNNs: Over-squashing in Graph Neural Networks

A complementary and highly explored topic in the GNN literature is that of *over-squashing*[4]. Over-squashing was originally introduced in Alon & Yahav (2021) as the compression of information from an exponentially growing receptive field of nodes into a fixed-size vector during the message-passing process. This phenomenon was later linked to a problem of sensitivity and mathematically quantified in Topping et al. (2021); Di Giovanni et al. (2023) through the Jacobian. In particular, this was done by considering an MPNN whose node representations are updated layer-by-layer. Let $\mathbf{h}_v^{(K)}$ denote the representation of node v after K layers, and $\mathbf{h}_u^{(0)}$ the initial representation of a distant node u. In Di Giovanni et al. (2023), the following bound on the sensitivity was derived:

[4]We note that the literature sometimes treats over-squashing as a problem of topological bottlenecks, and sometimes more generally as a problem of bottlenecks in the computational graph: we refer the reader to Arnaiz-Rodriguez & Errica (2025) for a discussion on this and other aspects of over-squashing.

Theorem 3.2.1 (Sensitivity bounds, (Di Giovanni et al., 2023)). *Consider a standard MPNN with k layers, where c_σ is the Lipschitz constant of the activation σ, w is the maximal entry-value over all weight matrices, and d is the embedding dimension. For $u, v \in V$ we have*

$$\left\| \frac{\partial \mathbf{h}_v^{(k)}}{\partial \mathbf{h}_u^{(0)}} \right\| \leq \underbrace{(c_\sigma w d)^k}_{model} \underbrace{(\mathbf{O}^k)_{vu}}_{topology}, \tag{25}$$

where $\mathbf{O} = c_r \mathbf{I} + c_a \hat{\mathbf{A}} \in \mathbb{R}^{n \times n}$ is the message-passing matrix adopted by the MPNN, and where c_r and c_a are the contributions of the self-connection and aggregation term.

Notice that the above formulation is composed of both a *model* term and a *topology* term. Many of the earlier works in the space explored rewiring, which explicitly engineered the graph's connectivity to enhance message propagation and mitigate bottlenecks (Gasteiger et al., 2019; Topping et al., 2021; Arnaiz-Rodríguez et al., 2022; Barbero et al., 2023; Gutteridge et al., 2023; Finkelshtein et al., 2024). On the other hand, a parallel strand of research concentrated almost exclusively on the model term itself, crafting non-dissipative update rules or otherwise taming the dynamics without touching the underlying graph (Gravina et al., 2023; 2025; Heilig et al., 2025). However, as pointed out in Arroyo et al. (2025), some of the most robust solutions in the literature combine both perspectives, jointly accounting for the graph topology for information propagation and the non-vanishing node-update dynamics. In particular, performing graph rewiring in isolation does not remove the zero collapse phenomenon and general optimization issues that characterize GNNs, while simply controlling node stability evolution will not enable long-range communication due to the modulation by the inverse square-root of the node degree typically performed when carrying out a message-passing step. We highlight that some of these problems are confined to the use of MPNN architectures, as recent work (Hariri et al., 2025) has also demonstrated the effectiveness of early spectral GNN designs in long-range information propagation. Although the depth terms in this bound mean that over-squashing can be accentuated in deep GNNs, we note too that "shallow" bottlenecks can cause issues; this is of particular relevance when considering long contexts in LLMs (Barbero et al., 2024). We refer readers to Arnaiz-Rodriguez & Errica (2025) for a discussion on over-squashing and depth, and to Blayney et al. (2025) who investigate over-squashing in shallow GNNs.

4 Intersections Between Transformers and GNNs

Having introduced the classical tools and problems in sequence and graph modeling, we are now ready to make connections between the two. Our analysis is based on the interpretation of sequence and Transformer models as message-passing architectures. This intersection can be articulated at several levels. First, at a conceptual level, the core operations of Transformers can be cast as a form of Graph Neural Network (GNN) operating on a fully-connected graph, a perspective originally popularized in Joshi (2025). Second, as a direct consequence of this conceptual similarity, both models suffer from analogous issues in information flow, such as signal degradation in deep models and information bottlenecks. Third, and most importantly for this discussion, the strategies developed to mitigate these issues share a common philosophy: preserving signal integrity and improving information flow through architectural changes like residual connections, normalization, and graph rewiring.

This section will unpack these points. We will first formalize the interpretation of Transformers as GNNs. We will then discuss the shared pathologies related to model depth and input width. Finally, we will dedicate a subsection to a direct comparison of the mitigation strategies that have emerged in both fields, highlighting their parallels.

4.1 Interpreting Transformers as GNNs

Even though Transformers and GNNs have traditionally been viewed as distinct architectures, they both interleave MLP blocks (building rich representations) with a "mixing" step that can be interpreted as a

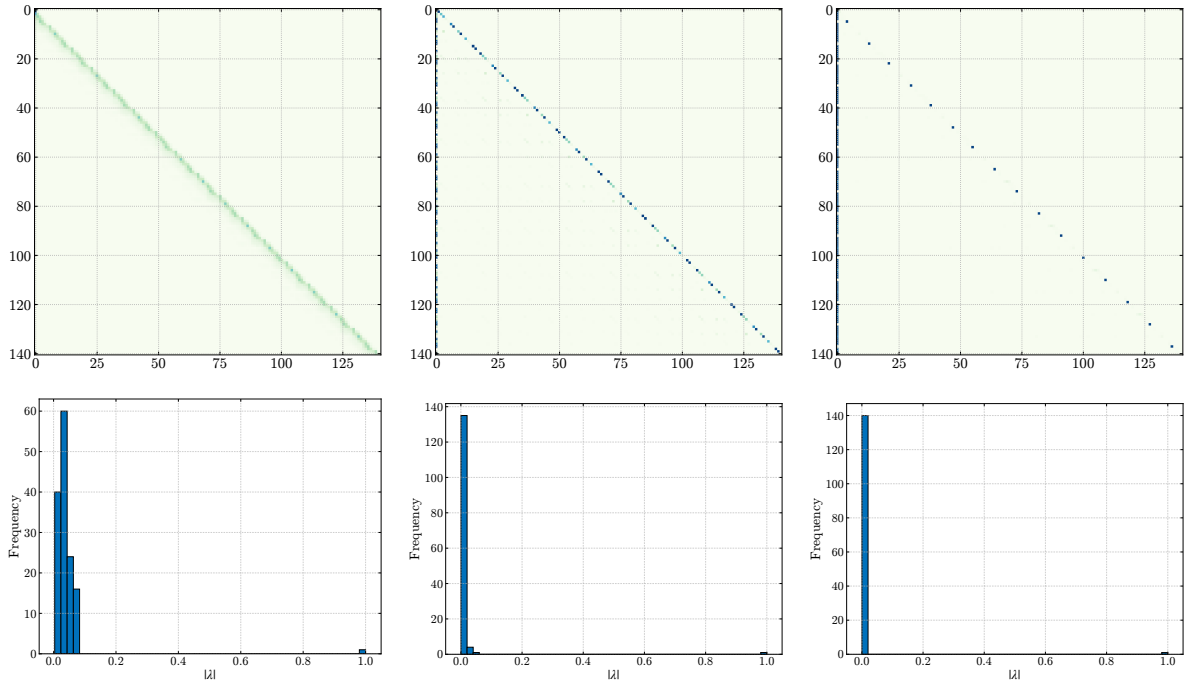

Figure 4: Attention matrices (top) and associated eigenvalue modulus histograms (bottom) for several heads in the early layers of the Gemma 7B model.

message-passing operation. Concretely, in a Transformer layer, we have

$$\mathbf{H}^{(k)} = \text{softmax}\left(\frac{\mathbf{Q}\,\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{26}$$

$$= \text{softmax}\left(\frac{\mathbf{H}^{(k-1)}\,\mathbf{W}_Q\left(\mathbf{H}^{(k-1)}\,\mathbf{W}_K\right)^\top}{\sqrt{d_k}}\right)\mathbf{H}^{(k-1)}\,\mathbf{W}_V \tag{27}$$

$$= \hat{\mathbf{A}}\,\mathbf{H}^{(k-1)}\,\mathbf{W}_V. \tag{28}$$

where $\hat{\mathbf{A}}$ is the corresponding attention matrix. From this, it is clear that Transformers and GNNs share a similar updating procedure where a learned adjacency matrix, $\hat{\mathbf{A}}$, is used to mix node (token) features. Therefore, MPNNs (especially GCNs and GATs) can be seen as a microcosm for how Transformers operate, and we should expect them to share similar pitfalls and learning dynamics. In particular, the spectral properties of the attention matrix will determine the behavior of the system in depth, while the characteristics of the `softmax` function will determine the properties of the graphs defined by this adjacency matrix. We will elaborate on both of these considerations in the next subsections.

4.2 Shared Pathologies: Information Flow Bottlenecks

The structural similarity between GNNs and Transformers leads to analogous failure modes related to information propagation, both in *depth of the model* and *width of the input*.

The Depth Regime: Over-smoothing and Rank Collapse Just as deep GNNs suffer from over-smoothing, where node representations converge to a uniform vector, naively stacking Transformer layers leads to rank collapse (Dong et al., 2021). In deep Transformers, the matrix of token embeddings tends to become low-rank, meaning the representations for all tokens collapse into a common subspace and lose their discriminative power. This phenomenon is a direct analogue of over-smoothing: in both cases, the iterative application of a mixing operator ($\hat{\mathbf{A}}$) is theoretically expected to yield increasingly similar node (or token) representations.

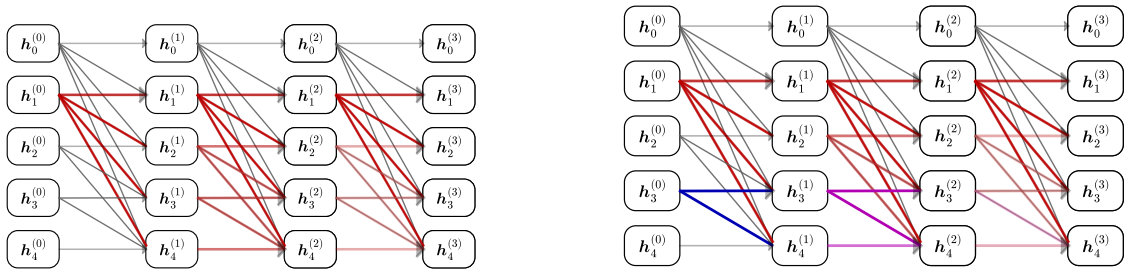

Figure 5: Illustration of information over-squashing in Transformers, adapted from similar figures in Barbero et al. (2024; 2025). **Left:** Information in a decoder-Transformer "overmixes", which results in performance degradation of the model. **Right:** The causal graph topology associated with a decoder-Transformer causes earlier tokens in the sequence to representationally "overwhelm" later ones.

The root cause of this shared pathology is the mixing operator $\hat{\mathbf{A}}$ itself, which is the very element that distinguishes these architectures from simple MLPs. This is not merely a representational issue; it has a critical effect on trainability. The same mechanism that drives over-smoothing (in the sense of zero collapse) is also responsible for *vanishing gradients*, making deep GNNs and Transformers notoriously difficult to train (Noci et al., 2022; Arroyo et al., 2025). The mathematical link is the *contractive nature* of the mixing matrix's spectrum, as shown in Arroyo et al. (2025). In MPNNs, the use of the normalized adjacency matrix is common, whose eigenspectrum satisfies $|\lambda_i| \leq 1$. In turn, modern LLMs, such as the Llama (Touvron et al., 2023a;b; Dubey et al., 2024) and Gemma (Team et al., 2024a;b; 2025) families, impose a triangular (causal) mask for next-token generation. This, together with the `softmax` function, makes it *row-stochastic*. This implies it can be analyzed as the transition matrix of a Markov chain, whose eigenvalues are similarly bounded by $|\lambda_i| \leq 1$. Recent work by Wu et al. (2024) has formally extended the notion of rank collapse to these causally masked attention matrices, with complementary random-matrix results in Naderi et al. (2024) employing a spectral approach to analyze their asymptotic behavior.

This creates a fundamental trade-off. On the one hand, contractive mixing causes training instability. On the other hand, empirical evidence from models like Gemma 7B shows that many learned attention heads are, in fact, highly contractive (Figure 4). This suggests the model finds these structures useful for its tasks. This presents a dilemma: to solve its task the model must learn contractive patterns, but these inherently inhibit the gradient flow required for deep, stable training. In practice, this is solved using the architectural solutions discussed in Section 4.3.

The Width Regime: Over-squashing and Representational Collapse Beyond the depth-wise issue, decoder-Transformers face "width-wise" issues, which manifest particularly in long-context scenarios. Some of these challenges are analogous to over-squashing in GNNs. In GNNs, the original definition of over-squashing (Alon & Yahav, 2021) stated that this phenomenon occurs when information from an exponentially growing number of nodes at a distance must be compressed into a fixed-size node embedding vector. Given that Transformers operate on a fully connected graph (Joshi, 2025), we expect this problem to be particularly pronounced. A key nuanced point needed to understand information propagation across the sequence in Transformers is the fact they operate over a *causal* graph (or Markov chain), as mentioned before.

One of the main consequences emerging from the causal structure of the attention matrix is a tendency of earlier tokens to overwhelm later ones, a problem which was formally introduced in Barbero et al. (2024). This can be interpreted as a form of the over-squashing problem, where in the limit in which the context size grows, only the first token will have an impact on the output of the model. Beyond this path-counting effect, Barbero et al. (2024) also prove the appearance of *representational collapse*, where for some distinct sequences (e.g., a string that ends with a certain token versus the same string with that token repeated at the end), the last-token representations become arbitrarily close as length grows, rendering the model effectively

unable to distinguish them under finite precision. This collapse is exacerbated by quantization/low-precision arithmetic and helps explain systematic failures on copying and counting in long, repetitive contexts.

Besides the issue of representational collapse, the fact that each token must integrate information from all previous tokens in the sequence leads to additional problems. In particular, we highlight the phenomenon of *over-mixing* outlined in Barbero et al. (2025), where the model thus loses the ability to distinguish individual tokens in the sequence and becomes more exposed to input perturbations. In the next subsection, we will present a number of attention patterns which large-scale models naturally learn through the gradient descent process to prevent this phenomenon. Illustrations of information over-squashing and over-mixing can be found in Figure 5.

4.3 Shared Mitigation Strategies: Preventing Collapse in Depth and Width

Previously, we have introduced the over-smoothing and over-squashing issues, and how they manifest in both decoder-Transformers and GNNs. In this subsection, we will emphasize how and why different architectural decisions can prevent or mitigate these phenomena, and what shortcuts models naturally learn during training to protect themselves against these issues.

Preventing Collapse in Depth: The Importance of the Transformer Block. As we have discussed, naively stacking message-passing or self-attention layers typically leads to trainability issues and problems with diversity of representations, even after a few layers. In the case of Transformers, self-attention layers are typically not stacked one after the other, but interleaved with normalization and residual layers to form a *Transformer block*, which is depicted in Figure 6. On the other hand, GNN layers were originally naively stacked one after the other without placing a similar emphasis on architectural optimization. While this choice was likely due to the locality present in some of the early GNN benchmarks, such as citation networks (McCallum et al., 2000; Sen et al., 2008), it could have been one of the culprits of the well-reported issue of over-smoothing in the GNN community.

We highlight that the specific design of the Transformer block is critical for mitigating the undesirable contraction properties that can characterize the self-attention matrix, and facilitates signal propagation through the network, and hence the overall training process. In the original Transformer architecture (Vaswani et al., 2017), the LN block was placed after the residual connection (also known as post-LN). However, in this original post-LN setup, large gradients at initialization force the use of a warm-up phase to prevent instability. As such, Xiong et al. (2021) proposed to reposition the LN inside the residual branch, (pre-LN), which led to well-behaved gradients that neither explode nor vanish even with a large initial learning rate, thus removing the necessity for a warm-up stage. Interestingly, the application of similar design principles in GNNs has yielded impressive, even state-of-the-art, performance (Luo et al., 2024; 2025). This is perhaps unsurprising given the conceptual similarities between Transformers and GNNs that we have illustrated. We note that the effect of different architectural components, such as residual connections and normalization, has been analyzed from a theoretical perspective in Dong et al. (2021) and more recently in Wu et al. (2024).

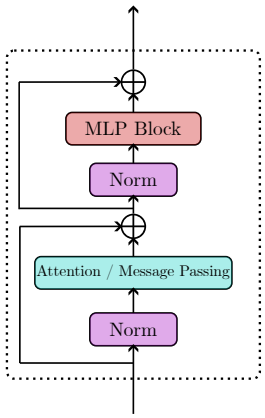

Figure 6: Illustration of "Transformer-style" blocks in both Transformers and GNNs, where either Attention or message-passing acts as a "mixing" step.

Despite the success of the block structure in Transformers, we highlight that it is also possible to achieve good signal propagation without relying on this specific structure. An example of this is He et al. (2023), where the use of bias matrices, location-dependent rescaling, and kernel-preserving initializations that let *vanilla* Transformers (*without* skip connections or normalization) maintain unit-variance signal all the way through hundreds of layers, enabling effective signal propagation. Their follow-up study (He & Hofmann, 2023) goes further by pruning value projections, merging the attention and MLP sequence into a single operation, and discarding LayerNorm, and they obtain blocks that are 15% faster and 15% smaller yet

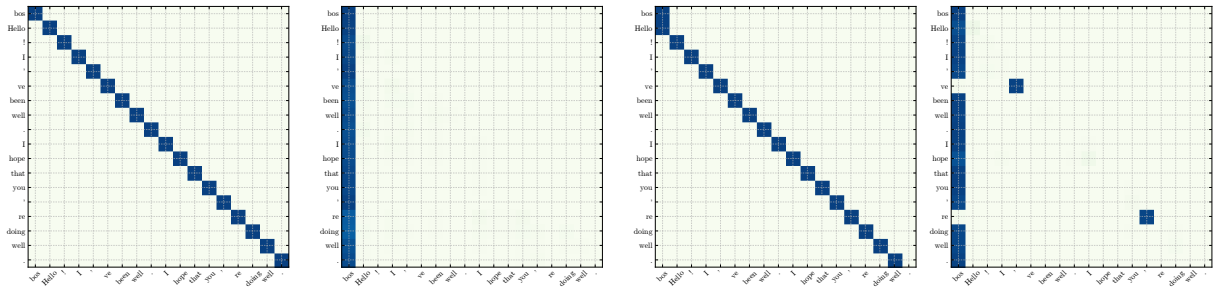

Figure 7: Example of sharp attention matrices in the Gemma 7B model. **Left:** An identity head. **Middle Left:** A BoS head. **Middle Right:** A previous token head. **Right:** An apostrophe head.

match the per-update training speed and final accuracy of standard pre-LN stacks. We also highlight the recent work of Zhu et al. (2025), who replace layer normalization with the *Dynamic Tanh (DyT)* activation function in the transformer block while achieving comparable performance.

Preventing collapse in width: Sharpness and gating help with over-squashing. One of the most effective strategies Transformers can learn to prevent this representational bottleneck is by developing specialized, *sharp* attention heads. Instead of forming a dense, fully-connected graph, these heads learn sparse patterns that route information in a more structured way, effectively preventing over-mixing. To this end, language models empirically tend to construct attention heads that either attend to very specific patterns in the sequence (previous token, spaces, or apostrophes), or attend only to the *BoS* (Beginning of Sequence) token despite not having any semantic meaning. This can be clearly seen in the example heads from the Gemma 7B model shown in Figure 7, which have also been reported empirically in Barbero et al. (2025); Queipo-de Llano et al. (2025). This "attention sink" phenomenon was originally reported in Xiao et al. (2023), and analyzed in more detail in Gu et al. (2024); Barbero et al. (2025). We note that recent work (Qiu et al., 2025) also prevented the appearance of attention sink by incorporating *gating* into the attention mechanism, which can be seen as an alternative (and more direct) way of preventing over-mixing. While these ideas have been heavily explored in the context of recurrent architectures (Hochreiter & Schmidhuber, 1997; Gu & Dao, 2023; Beck et al., 2024; De et al., 2024), it appears that similar approaches can mitigate the undesirable effects of over-mixing in decoder-Transformers.

The ability of decoder-Transformers to reliably learn and maintain these desirable sharp patterns is also fundamentally constrained by the choice of attention function. To this end, recent work (Veličković et al., 2024) argue that the standard `softmax` will fail in settings that require generalization to sequences longer than those seen during training. In these cases, the nature of softmax can cause attention scores to bleed and diffuse, making it difficult for a head to remain sharply focused on a single token. We highlight other works which have further analyzed the softmax function (Masarczyk et al., 2025) as well as other variants (Saratchandran et al., 2024; Ramapuram et al., 2024), including ones that prevent the formation of attention sinks (Zuhri et al., 2025). We note that Over-squashing in GNNs is also sometimes addressed through methods that reduce bottlenecks in the computational tree. One recent method to do this has been *adaptive message-passing* (Errica et al., 2023; Finkelshtein et al., 2024), which can also be seen as a more explicit way to enforce "sharpness".

5 Conclusion

In this survey, we have explored the conceptual and mathematical parallels between decoder-Transformers and GNNs, advocating for a unified graph-based perspective on information propagation in these seemingly disparate architectures. We have shown how fundamental challenges in GNNs, such as **over-smoothing** and **over-squashing**, find direct analogues in Transformers as **rank collapse** and **representational collapse**.

Our analysis revealed that the iterative message-passing operations in GNNs, which can lead to node feature homogenization, mirrors the depth-wise rank collapse observed in Transformers where token representations

lose their distinctiveness. The role of residual connections and normalization techniques, crucial for mitigating these issues in both architectures, underscores a shared need for mechanisms that preserve signal diversity and prevent information dissipation or concentration. Furthermore, the sensitivity bounds derived for GNNs provide a lens through which to understand how information from distant parts of a sequence can be "squashed" in Transformers, highlighting the importance of managing receptive field and information flow.

By interpreting the self-attention mechanism as a learned adjacency matrix, we bridge the gap between spectral graph theory and the dynamics of attention. This perspective offers a powerful framework for understanding how the spectral properties of attention matrices influence signal propagation and why architectural choices, like the Transformer block, are essential for robust learning. The burgeoning research into Transformers without traditional normalization or residual connections, driven by insights into parameterization and initialization, further reinforces the notion that controlled information flow is paramount.

Ultimately, this unified view not only enhances our theoretical understanding of these powerful models but also paves the way for cross-pollination of ideas. Solutions developed for performant GNNs, particularly those addressing vanishing gradients and information bottlenecks, could inspire novel architectural designs or training strategies for Transformers, and vice-versa. As the fields of graph representation learning and large language models continue to advance, a shared vocabulary and conceptual framework will be valuable for developing more robust, efficient, and interpretable deep learning architectures.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pp. 1120–1128. PMLR, 2016.

Adrian Arnaiz-Rodriguez and Federico Errica. Oversmoothing," oversquashing", heterophily, long-range, and more: Demystifying common beliefs in graph machine learning. *arXiv preprint arXiv:2505.15547*, 2025.

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lov\'asz bound. *arXiv preprint arXiv:2206.07369*, 2022.

Álvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in gnns: Bridging recurrent and graph learning. *arXiv preprint arXiv:2502.10818*, 2025.

Jacob Bamberger, Federico Barbero, Xiaowen Dong, and Michael M Bronstein. Bundle neural networks for message diffusion on graphs. *arXiv preprint arXiv:2405.15540*, 2024.

Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni. Locality-aware graph-rewiring in gnns. *arXiv preprint arXiv:2310.01668*, 2023.

Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João GM Araújo, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers need glasses! information over-squashing in language tasks. *arXiv preprint arXiv:2406.04267*, 2024.

Federico Barbero, Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do llms attend to the first token? *arXiv preprint arXiv:2504.02732*, 2025.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

Yoshua Bengio. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Hugh Blayney, Álvaro Arroyo, Xiaowen Dong, and Michael M Bronstein. glstm: Mitigating over-squashing by increasing storage capacity. *arXiv preprint arXiv:2510.08450*, 2025.

Cristian Bodnar, Francesco Di Giovanni, Benjamin P. Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs, 2022.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein. GRAND: Graph neural diffusion. In *International Conference on Machine Learning (ICML)*, pp. 1407–1418. PMLR, 2021.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3844–3852, 2016.

Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Graph neural networks as gradient flows: understanding graph convolutions via energy. *arXiv preprint arXiv:2206.10991*, 2022.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pp. 7865–7885. PMLR, 2023.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pp. 2793–2803. PMLR, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Bastian Epping, Alexandre René, Moritz Helias, and Michael T Schaub. Graph neural networks do not always oversmooth. *arXiv preprint arXiv:2406.02269*, 2024.

N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2020.

Federico Errica, Henrik Christiansen, Viktor Zaverkin, Takashi Maruyama, Mathias Niepert, and Francesco Alesiani. Adaptive message passing: A general framework to mitigate oversmoothing, oversquashing, and underreaching. *arXiv preprint arXiv:2312.16560*, 2023.

Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 4267–4275, 2023.

Ben Finkelshtein, Xingyue Huang, Michael M Bronstein, and Ismail Ilkan Ceylan. Cooperative graph neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In *International conference on machine learning*, pp. 7616–7633. PMLR, 2022.

M Gori, G Monfardini, and F Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Alex Graves. Generating sequences with recurrent neural networks. *ArXiv*, abs/1308.0850, 2013.

Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *The 39th Annual AAAI Conference on Artificial Intelligence*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *arXiv preprint arXiv:2410.10781*, 2024.

Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. DRew: dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267. PMLR, 2023.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Ali Hariri, Álvaro Arroyo, Alessio Gravina, Moshe Eliasof, Carola-Bibiane Schönlieb, Davide Bacciu, Kamyar Azizzadenesheli, Xiaowen Dong, and Pierre Vandergheynst. Return of chebnet: Understanding and improving an overlooked gnn on long range tasks. *arXiv preprint arXiv:2506.07624*, 2025.

Bobby He and Thomas Hofmann. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906*, 2023.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L Smith, and Yee Whye Teh. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. *arXiv preprint arXiv:2302.10322*, 2023.

Simon Heilig, Alessio Gravina, Alessandro Trenta, Claudio Gallicchio, and Davide Bacciu. Port-hamiltonian architectural bias for long-range propagation in deep graph networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mikael Henaff, Arthur Szlam, and Yann LeCun. Recurrent orthogonal networks and long-memory tasks. In *International Conference on Machine Learning*, pp. 2034–2042. PMLR, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

Chaitanya K Joshi. Transformers are graph neural networks. *arXiv preprint arXiv:2506.22084*, 2025.

T Anderson Keller, Lyle Muller, Terrence Sejnowski, and Max Welling. Traveling waves encode the recent past and enhance sequence learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Nicolas Keriven. Backward oversmoothing: why is it hard to train deep graph neural networks? *arXiv preprint arXiv:2505.16736*, 2025.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pp. 3794–3803. PMLR, 2019.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Weigang Lu, Yibing Zhan, Binbin Lin, Ziyu Guan, Liu Liu, Baosheng Yu, Wei Zhao, Yaming Yang, and Dacheng Tao. Skipnode: On alleviating performance degradation for deep graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *arXiv preprint arXiv:2406.08993*, 2024.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Unlocking the potential of classic gnns for graph-level tasks: Simple architectures meet excellence. *arXiv preprint arXiv:2502.09263*, 2025.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. *arXiv preprint arXiv:1709.04057*, 2017.

Wojciech Masarczyk, Mateusz Ostaszewski, Tin Sum Cheng, Aurelien Lucchi, Razvan Pascanu, et al. Unpacking softmax: How temperature drives representation collapse, compression, and generalization. *arXiv preprint arXiv:2506.01562*, 2025.

Sohir Maskey, Raffaele Paolino, Aras Bacho, and Gitta Kutyniok. A fractional graph laplacian approach to oversmoothing. *Advances in Neural Information Processing Systems*, 36, 2024.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Alireza Naderi, Thiziri Nait Saada, and Jared Tanner. Mind the gap: a spectral analysis of rank collapse and signal propagation in transformers. *arXiv preprint arXiv:2410.07799*, 2024.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022.

Kenta Oono and Taiji Suzuki. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *International Conference on Learning Representations*, 2020.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Francesco Pappone. Beyond attention as a graph. *Blogpost*, 2025.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28, pp. III–1310, 2013.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

Enrique Queipo-de Llano, Álvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in llms are two sides of the same coin. *arXiv preprint arXiv:2510.06477*, 2025.

Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, et al. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint arXiv:2409.04431*, 2024.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Andreas Roth, Franka Bause, Nils M Kriege, and Thomas Liebig. Preventing representational rank collapse in mpnns by splitting the computational graph. *arXiv preprint arXiv:2409.11504*, 2024.

T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pp. 18888–18909. PMLR, 2022.

T Konstantin Rusch, Benjamin Paul Chamberlain, Michael W Mahoney, Michael M Bronstein, and Siddhartha Mishra. Gradient gating for deep multi-rate learning on graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

Hemanth Saratchandran, Jianqiao Zheng, Yiping Ji, Wenbo Zhang, and Simon Lucey. Rethinking softmax: Self-attention with polynomial activations. *arXiv preprint arXiv:2410.18613*, 2024.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Michael Scholkemper, Xinyi Wu, Ali Jadbabaie, and Michael T Schaub. Residual connections and normalization can provably prevent oversmoothing in gnns. *arXiv preprint arXiv:2406.02997*, 2024.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Alessandro Sperduti. Encoding labeled graphs by labeling raam. *Advances in Neural Information Processing Systems*, 6, 1993.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*, 2018.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024a.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. softmax is not enough (for sharp out-of-distribution). *arXiv preprint arXiv:2410.01104*, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, 2018.

Keqin Wang, Yulong Yang, Ishan Saha, and Christine Allen-Blanchette. Understanding oversmoothing in gnns as consensus in opinion dynamics. *arXiv preprint arXiv:2501.19089*, 2025.

Xinyi Wu, Amir Ajorlou, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. On the role of attention masks and layernorm in transformers. *arXiv preprint arXiv:2405.18781*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Zheng Xiong, Luisa Zintgraf, Jacob Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14901–14911, 2025.

Zayd MK Zuhri, Erland Hilman Fuadi, and Alham Fikri Aji. Softpick: No attention sink, no massive activations with rectified softmax. *arXiv preprint arXiv:2504.20966*, 2025.