Improving LLM Final Representations with Inter-Layer Geometry

Eyal Blyachman*
Tel Aviv University
blyachman1@mail.tau.ac.il

Tom Ulanovski*
Blavatnik School of Computer Science
Tel Aviv University
tomulanovski@mail.tau.ac.il

Maya Bechler-Speicher
Meta AI
mayabs@meta.com

Abstract

The standard in LLM-based prediction is to use the final-layer representation as the input to a downstream predictor. However, intermediate layers may encode complementary task-relevant signals. Existing approaches therefore either search for the best layer for each task or apply expensive attention-based mechanisms to learn inter-layer aggregation. In this work, we first show that such complexity is unnecessary: a lightweight Graph Neural Network over a fully connected graph of LLM layers is more efficient and achieves significantly stronger predictive performance than existing approaches. We then introduce the Cayley-Encoder, which further improves both efficiency and predictive performance by replacing the fully connected graph with a Cayley graph over $SL(2, \mathbb{Z}_n)$. These Cayley graphs provide a mathematically grounded topology that is sparse, regular by construction, and has low diameter. This enables effective communication across layers while constraining the aggregation structure and reducing the risk of GNN overfitting. In an evaluation of Cayley-Encoder across 13 tasks and 13 LLMs, Cayley-Encoder consistently outperforms baselines, achieving improvements of up to 48 percentage points in accuracy, while introducing at most 0.1% additional parameters relative to the LLM size. We further show that Cayley-Encoder is effective in few-shot regimes. Finally, we show that Cayley-Encoder outperforms LoRA fine-tuning while operating on the frozen LLM. We conclude with an explainability analysis showing that multiple layers contribute meaningfully to the final prediction, supporting our hypothesis.

1 Introduction

Large Language Models (LLMs) have become a standard backbone for learning transferable text representations across a broad range of downstream prediction tasks [Vaswani et al., 2017, Devlin et al., 2019, Brown et al., 2020]. A common efficient post-training approach is to use a frozen LLM as a feature extractor and train a lightweight prediction head on top of it. The dominant practice is to use the representation from the final layer of the LLM. This convention implicitly assumes that the last layer provides the most task-relevant representation.

A growing body of analysis shows that language models distribute information across depth. Lower layers tend to encode surface-level and lexical information, middle layers often capture syntactic

*Equal contribution. Listing order is random.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

structure, and upper layers are more strongly associated with semantic abstraction [Tenney et al., 2019, Jawahar et al., 2019, Clark et al., 2019]. Indeed, recent work has shown that intermediate layers can contain substantial task-relevant signal and may outperform final-layer representations depending on the task [Wallat et al., 2020, Fan et al., 2024, Skean et al., 2025]. There are approaches that focus on attention-based inter-layer aggregation to form a final representation based on information from all layers [ElNokrashy et al., 2024]. However, this design remains anchored to the final layer and introduces additional complexity. Other approaches modify the transformer architecture itself [Pagliardini et al., 2024, Xiao et al., 2025], but require training the architecture from scratch.

In this work, we show how inter-layer information from a frozen LLM can be aggregated in a way that is both more efficient and more effective than existing approaches, by introducing mathematically grounded geometry over the layers and formulating inter-layer aggregation as a graph learning problem. Our hypothesis is that introducing a structure that does not heavily rely on a single layer as in existing approaches is more beneficial for obtaining the final representation from the LLM. This view provides a direct mechanism for modeling interactions among layer representations without modifying the underlying LLM, in a way that is independent of the number of input tokens.

First, we show that even a simple fully connected layer graph, where every pair of layers can communicate directly through a lightweight GNN, substantially outperforms both attention mechanisms and selecting the single best-performing layer. We then show that both memory efficiency and predictive performance can be further improved by replacing the dense fully connected graph with a special sparse graph. While there are many ways to construct sparse graphs, prior work showed that GNNs tend to overfit the underlying graph structure when the structure does not carry information for the predictive task, and that regular graphs are more robust to this overfitting [Bechler-Speicher et al., 2024]. Moreover, arbitrary regular graphs may still introduce information bottlenecks and poor communication between nodes [Alon and Yahav, 2021, Wilson et al., 2025]. To address these limitations, we introduce the Cayley-Encoder, a geometric approach that replaces the dense fully connected inter-layer graph with a sparse graph with guaranteed advantages. Cayley-Encoder maps layers to a Cayley graph over the Special Linear group $SL(2, \mathbb{Z}_n)$, which is guaranteed to be sparse, regular, and low-diameter, enabling layers to communicate effectively through the GNN.

We evaluate Fully-Connected Encoder and Cayley-Encoder on 13 diverse downstream tasks, spanning intent detection, sentiment analysis, domain classification, and semantic similarity, across 13 pretrained LLMs from four families. Across this broad evaluation, Cayley-Encoder consistently outperforms baselines, reaching up to 48 percentage points improvement in accuracy, while adding at most 0.1% parameters relative to the base LLM. We further show that Cayley-Encoder remains effective in few-shot regimes. In addition, Cayley-Encoder outperforms LoRA fine-tuning [Hu et al., 2022] on most evaluated tasks while keeping the LLM frozen, showing that better use of existing representations can outperform direct adaptation of model weights.

Finally, we analyze the learned aggregation patterns and find that predictions rely on signals from multiple layers roughly evenly, rather than collapsing to a single dominant depth. This supports the benefit of drawing on the full depth of the LLM rather than relying on any single layer. Overall, our findings suggest that stronger representations can be obtained by allowing all layers to communicate effectively in a symmetric way.

Our main contributions are:

1. We formulate inter-layer aggregation in frozen LLMs as a graph learning problem and show that a lightweight GNN over a fully connected layer graph already substantially outperforms attention-based aggregation and best-layer optimization.

2. We introduce Cayley-Encoder, a sparse geometric inter-layer encoder based on Cayley graphs over $SL(2, \mathbb{Z}_n)$, providing a regular, sparse, and low-diameter structure for efficient communication across layers.

3. We evaluate Cayley-Encoder across 13 diverse downstream tasks and 13 LLMs, showing consistent gains over existing aggregation methods and LoRA fine-tuning while adding at most 0.1% parameters relative to the underlying LLM.

4. We show that effective inter-layer communication, rather than the identity or ordering of individual layers, is the primary driver of performance: all layers contribute roughly equally, performance is robust to layer-to-node assignment, and adding layer-position encodings does not help.

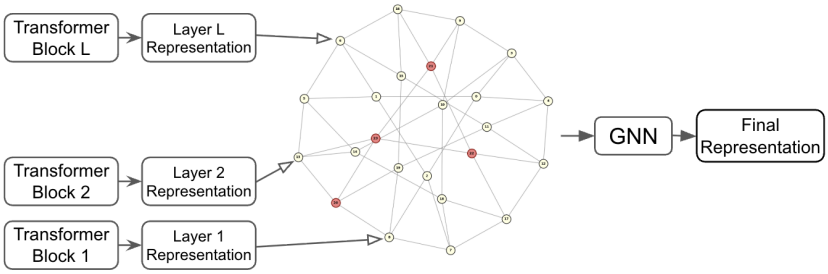


Figure 1: An overview of Cayley-Encoder. Layer representations are mapped into nodes in a Cayley Graph over the Special Linear group $SL(2, \mathbb{Z}_n)$, which is then fed into a GNN to learn the final inter-layer representation.

2 Related Work

2.1 Inter-Layer Representations in LLMs

Research on transformer depth has revealed that different layers encode different types of information. Lower layers tend to capture surface-level and lexical features, middle layers encode syntactic structure, and upper layers are associated with semantic abstraction [Jawahar et al., 2019, Tenney et al., 2019]. Importantly, intermediate layers can contain task-relevant signals that are absent from the final layer: Wallat et al. [2020] showed that intermediate layers capture factual knowledge better than the last layer, and Skean et al. [2025] demonstrated that intermediate representations consistently outperform final-layer ones across downstream tasks. These findings have motivated several approaches for aggregating information across layers. Peters et al. [2018] introduced task-specific scalar weights over BiLSTM layers in ELMo, but this restricts aggregation to a linear mixture that cannot capture non-linear inter-layer interactions. ElNokrashy et al. [2024] extended this to transformers with Depth-Wise Attention (DWAtt), which attends over layers using a query derived from the final-layer representation. While DWAtt models non-linear interactions, it anchors the aggregation to the last layer, treating earlier layers only as keys and values rather than as equal participants. Other work modifies the transformer architecture itself [Pagliardini et al., 2024, Xiao et al., 2025], but requires training from scratch. Methods such as LayerDrop [Fan et al., 2020] operate during training by randomly dropping layers, and are thus complementary to our post-training setting. Our work addresses these limitations by enabling structured inter-layer communication within a frozen LLM, without modifying the architecture or privileging any single layer (Figure 1).

2.2 Graph Neural Networks and Cayley Graphs

Graph Neural Networks (GNNs) are a family of neural networks designed to learn representations over graph-structured data. GNNs operate by iteratively propagating information between nodes through a process called message passing neural network (MPNN) [Gilmer et al., 2017]. Each MPNN layer consists of three operations: (1) each node computes messages to send to its neighbors, (2) each node aggregates the messages it receives, and (3) each node updates its representation by combining the aggregated messages with its previous state.

Many GNN variants have been proposed, differing mostly in how they aggregate information [Kipf and Welling, 2017, Xu et al., 2019]. See Gkarmpounis et al. [2024] for a recent survey on GNNs. Bechler-Speicher et al. [2024] showed that GNNs tend to overfit the underlying graph structure when the structure does not carry information for the predictive task, and that regular graphs are more robust to this overfitting. Another limitation of GNNs is over-squashing [Alon and Yahav, 2021], where information propagating through the graph is compressed into fixed-size node representations. Deac et al. [2022] introduced Expander Graph Propagation (EGP) to improve information flow in GNNs. They address over-squashing by leveraging Cayley graphs. Cayley graphs over $SL(2, \mathbb{Z}_n)$ with the Margulis generating set are regular and sparse yet highly connected, with logarithmic diameter, allowing any two nodes to communicate in $O(\log |V|)$ steps. Cayley graphs come in fixed sizes determined by the underlying group. Therefore, EGP truncates nodes to match the input graph size. However, Wilson et al. [2025] showed that truncation can reintroduce bottlenecks. They therefore

proposed Cayley Graph Propagation (CGP), which instead pads the input graph with virtual nodes to preserve the complete Cayley structure. Virtual nodes only propagate information through the graph and are excluded from the final aggregation.

Mantri et al. [2026] introduced GLOT, which applies a GNN over a token-similarity graph constructed from the last layer's representations. Unlike our method, GLOT operates within a single layer, requires constructing the graph per input via pairwise token similarity, and produces graphs that grow linearly with input length. Our graph is fixed and determined solely by the number of LLM layers, making it constant-size regardless of input.

3 Inter-Layer Geometry

In this section, we introduce the Fully-Connected and Cayley Encoders.

We consider LLMs with L layers and an input sequence of T tokens. Each layer $\ell \in \{1, \ldots, L\}$ produces a d-dimensional hidden representation for each token $\mathbf{H}_\ell \in \mathbb{R}^{T \times d}$. We form a *layer representation* $\mathbf{z}_\ell, \ell \in \{1, \ldots, L\}$ by averaging the hidden token representation of each layer:

$$\mathbf{z}_\ell = \frac{1}{T} \sum_{t=1}^{T} \mathbf{H}_\ell[t, :] \in \mathbb{R}^d$$

Mean-pooling over tokens is a standard approach for obtaining fixed-size sentence representations from LLMs [Reimers and Gurevych, 2019]. Here, it also ensures that the resulting layer graph has a fixed size determined solely by the number of LLM layers, independent of the input sequence length. We evaluate token-level aggregation as an extension in Section 4.5.

We then map each layer representation $\mathbf{z}_\ell$ into nodes of a graph $G = (V, E)$ with nodes V and edges E. Each node v_ℓ is associated with an initial representation $h_\ell^{(0)} = \mathbf{z}_\ell \in \mathbb{R}^d$. We then apply a GNN over G to produce one final representation h_G. At GNN layer k, the representation of node v is updated as follows:

$$m_v^{(k)} = \sum_{u \in \mathcal{N}(v)} M^{(k)} \left(h_v^{(k-1)}, h_u^{(k-1)} \right)$$

$$h_v^{(k)} = U^{(k)} \left(h_v^{(k-1)}, m_v^{(k)} \right)$$

where $m_v^{(k)}$ is the aggregated message received by node v at layer k, $\mathcal{N}(v)$ denotes the set of neighbors of node v, $M^{(k)}$ is the message function that computes the contribution from each neighbor, and $U^{(k)}$ is the update function that combines the node's previous representation with its aggregated messages. Both $M^{(k)}$ and $U^{(k)}$ are learnable functions. To obtain a graph-level representation, the final node representations are pooled; the pooling type, mean or sum, is selected as a hyperparameter.

Fully-Connected Encoder In the Fully-Connected Encoder (FC-Encoder) the layer representations are mapped to nodes in a fully connected graph, where every layer is connected to every other layer. This structure is dense, with $\binom{L}{2}$ undirected edges. Note that this corresponds to attention between layers only when the GNN computes attention weights over edges, as in GAT [Veličković et al., 2018]; the GNN architectures we evaluate, GIN [Xu et al., 2019] and GCN [Kipf and Welling, 2017] (Section 4), aggregate messages without such weights.

Cayley-Encoder In the Cayley-Encoder, layer representations are mapped to nodes of a Cayley Graph over $SL(2, \mathbb{Z}_n)$. A Cayley graph is a graph that describes the abstract structure of a mathematical group. Here we focus on the special linear group $SL(2, \mathbb{Z}_n)$, with elements that are defined as 2×2 matrices with integer entries modulo n and unit determinant. The resulting graph is 4-regular and has logarithmic diameter, enabling efficient global communication during message passing (see Appendix A for construction details). The number of nodes in the graph $|V_n|$ is determined by the formula:

$$|V_n| = n^3 \prod_{\text{prime } p \mid n} \left(1 - \frac{1}{p^2} \right)$$

To maintain the graph's symmetry and expansion properties, we choose the smallest n such that $|V_n| \geq L$. We map the L layer representations randomly to L nodes of the Cayley graph. When $|V_n| > L$, the remaining $|V_n| - L$ nodes are initialized as virtual nodes with zero vectors as their representations. Following Wilson et al. [2025], we use only the representations from the non-virtual nodes for the final hidden representation used for the prediction task.

While the graph is fixed for a given LLM, the encoder is trained separately for each task, so it does not assume that the roles of the layers are the same across tasks. The same connectivity can be used differently for different tasks. In the next section, we demonstrate the effectiveness of Cayley-Encoder and FC-Encoder through extensive evaluation.

4 Experiments

In this section we evaluate FC-Encoder and Cayley-Encoder[2]. We evaluate on 13 diverse tasks from the MTEB benchmark [Muennighoff et al., 2023], spanning intent detection, sentiment analysis, domain classification, and semantic textual similarity. Beyond standard full-data evaluation, we assess zero-shot transfer, few-shot learning with 1–1024 samples per label, scaling behavior across LLM sizes from 14M to 14B parameters in two LLM families, comparison with LoRA fine-tuning, token-level inter-layer aggregation, and an analysis of layer importance.

Datasets. We use 13 tasks from the MTEB benchmark [Muennighoff et al., 2023]. Five are classification tasks: Banking77 (77 intent classes) [Casanueva et al., 2020], Emotion (6 emotion categories) [Saravia et al., 2018], MTOPDomain and MTOPIntent (task-oriented dialogue) [Li et al., 2021], and Poem Sentiment (4 sentiment classes) [Sheng and Uthus, 2020]. For each, we train on the training split and evaluate on the test split. The remaining 8 are semantic textual similarity (STS) tasks. We train on STSBenchmark [May, 2021] and evaluate on its test split, then perform zero-shot transfer to STS12–16, BIOSSES, and SICK-R [Agirre et al., 2012, 2013, 2014, 2015, 2016, Soğancıoğlu et al., 2017, Marelli et al., 2014].

Setup. We compare both encoders against a diverse set of baselines: **Last-Layer**, the last-layer representation as in Skean et al. [2025]; **Best-Layer**, the best-performing layer selected by evaluating each layer independently; **Weighted**, a learned weighted sum of layer representations, similar to Peters et al. [2018]; **MLP Last-Layer**, a trained MLP over the last-layer representations; **DWAtt**, a projection to 256 dimensions followed by DWAtt [ElNokrashy et al., 2024], where the projection reduces the large parameter count of applying DWAtt directly to layer representations (results without the projection are in Appendix I); and **Deepset**, a Deepset over layer representations [Zaheer et al., 2017].

We evaluate on Pythia-410M (25 layers, 1024-dim) [Biderman et al., 2023], Gemma2-2B (27 layers, 2304-dim) [Team, 2024], Llama3-8B (33 layers, 4096-dim) [AI@Meta, 2024], and Qwen3-8B (37 layers, 4096-dim) [Yang et al., 2025]. In addition, Section 4.2 evaluates six additional Pythia sizes and three additional Qwen3 sizes, for a total of 13 LLMs. All base models remain frozen during training. For the first set of experiments, we use the layer representations obtained by mean-pooling over all token representations for each layer. These pooled representations serve as the input to all methods. In this setting the input size is constant and derived from the number of layers in the LLM. We further evaluate FC-Encoder and Cayley-Encoder on all token representations across all layers, making the input size dependent on the input sequence length to the LLM.

For classification tasks, we train the encoders jointly with a linear classifier head using cross-entropy loss. For STS tasks, we encode each sentence in a pair through the frozen LLM and our encoders, compute the cosine similarity between the resulting representations, and minimize Mean Squared Error (MSE) loss with respect to the ground-truth similarity score. We use the Adam optimizer [Kingma, 2014] with learning rate and weight decay selected via Optuna [Akiba et al., 2019] hyperparameter optimization on the validation set. For FC-Encoder and Cayley-Encoder, we evaluate GIN [Xu et al., 2019] and GCN [Kipf and Welling, 2017] architectures as hyperparameters. Trainable parameter counts and full hyperparameter details are in Appendix B and C. Following the standard MTEB evaluation protocol [Muennighoff et al., 2023], we report single-run results for each configuration.

[2]Code: https://github.com/eyali123/ILGE

4.1 Results

Tables 1 and 2 present the results. Cayley-Encoder achieves the best performance in 14 out of 20 classification configurations, 3 out of 4 supervised STS configurations, and 17 out of 28 zero-shot STS transfer configurations, making it the strongest method overall. For classification, Cayley-Encoder achieves average gains of 33 percentage points over Last-Layer and 25 percentage points over Best-Layer, which itself requires evaluating every layer independently. The largest gains appear on the Emotion and Poem Sentiment tasks, with improvements of up to 48 percentage points over Last-Layer. For STS, Cayley-Encoder achieves the highest score in 20 out of 32 configurations, with the largest margins on STS13 and STS14. Comparing to multi-layer aggregation approaches, Cayley-Encoder outperforms DWAtt on all 13 tasks while using 3–5$\times$ fewer parameters (Table 8). MLP Last-Layer achieves gains on classification but provides no consistent benefit for STS. Both encoders outperform Deepset in most configurations, indicating that inter-layer connectivity improves aggregation beyond permutation-invariant pooling. FC-Encoder, while on par with Cayley-Encoder in some configurations and particularly on smaller LLMs for classification, outperforms DWAtt in most configurations, demonstrating that structured aggregation can be sufficient without attention mechanisms.

Table 1: Performance comparison across 4 LLMs on 5 classification tasks. Cayley-Encoder achieves the best score in 14 out of 20 configurations, with its advantage most pronounced on larger LLMs. In all but three cases, FC-Encoder and Cayley-Encoder achieve the top two scores. We highlight in bold the best result and in blue the second best per column.

Base Model	Method	Banking77	Emotion	MTOPDomain	MTOPIntent	PoemSentiment
Pythia-410M	Last-Layer	61.17	33.48	80.88	66.97	42.40
	Best-Layer	66.67	35.02	83.78	71.18	45.67
	MLP Last-Layer	83.84	33.99	96.97	83.75	75.00
	Weighted	58.50	28.26	79.79	61.68	42.60
	DWAtt	83.23	58.60	98.03	91.66	70.87
	Deepset	84.23	47.89	97.59	92.21	73.37
	FC-Encoder	**90.65**	**75.61**	98.68	**95.04**	**75.77**
	Cayley-Encoder	89.43	73.83	**98.77**	94.72	70.87
Gemma2-2B	Last-Layer	62.16	26.89	80.79	68.02	35.67
	Best-Layer	72.31	31.27	87.09	75.36	42.79
	MLP Last-Layer	87.47	59.05	98.37	92.73	71.63
	Weighted	65.40	29.94	85.17	73.20	39.62
	DWAtt	88.82	61.22	98.39	90.93	67.79
	Deepset	90.03	78.77	98.92	94.37	78.56
	FC-Encoder	92.39	**79.90**	99.07	95.34	77.12
	Cayley-Encoder	**92.58**	69.60	**99.16**	**96.43**	**83.27**
Llama3-8B	Last-Layer	68.25	34.23	84.42	73.39	40.96
	Best-Layer	71.93	38.42	89.01	78.17	47.02
	MLP Last-Layer	86.70	67.67	98.58	92.09	75.00
	Weighted	66.63	27.94	83.85	71.78	37.79
	DWAtt	90.04	66.55	97.97	92.41	75.00
	Deepset	87.62	71.04	98.77	95.43	77.02
	FC-Encoder	92.38	71.64	98.99	96.19	77.98
	Cayley-Encoder	**92.85**	**73.43**	**99.03**	**96.46**	**79.04**
Qwen3-8B	Last-Layer	48.12	30.18	71.79	56.67	37.31
	Best-Layer	70.98	36.36	87.88	76.94	45.67
	MLP Last-Layer	81.37	56.95	98.09	87.10	68.65
	Weighted	60.51	28.52	81.71	53.63	43.08
	DWAtt	91.22	68.31	98.97	92.10	73.94
	Deepset	85.71	69.25	97.92	53.11	71.15
	FC-Encoder	91.90	69.66	**99.02**	95.60	80.48
	Cayley-Encoder	**92.34**	**75.70**	98.99	**95.99**	**85.38**

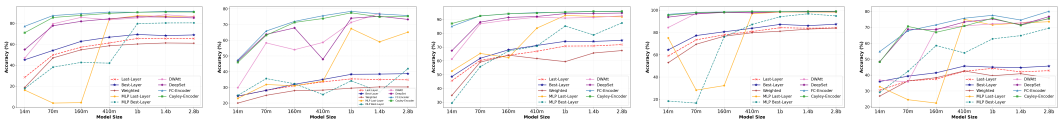
4.2 Effectiveness Across LLM Sizes

We evaluate whether the benefits of our encoders persist across LLM sizes. We conduct this evaluation on the classification tasks using the Pythia suite [Biderman et al., 2023]: 14M, 70M, 160M, 410M,

Table 2: Performance comparison across 4 LLMs on 8 STS tasks (Spearman correlation $\times 100$). Cayley-Encoder achieves the highest score in 20 out of 32 configurations. We highlight in bold the best result and in blue the second best per column.

Base Model	Method	STSBenchmark	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R
Pythia-410M	Last-Layer	39.12	46.96	47.00	41.45	49.32	50.37	67.30	52.55
	Best-Layer	53.53	50.62	59.27	51.61	65.59	58.02	**74.80**	**58.26**
	MLP Last-Layer	25.54	42.14	36.16	33.28	37.75	39.84	59.68	43.23
	Weighted	41.81	47.12	49.49	44.21	53.72	52.76	67.79	55.09
	DWAtt	49.40	52.51	44.57	47.46	52.51	44.80	45.29	52.43
	Deepset	39.48	52.11	42.68	43.99	48.32	40.48	44.72	44.84
	FC-Encoder	54.20	50.13	53.00	51.76	63.94	57.11	58.65	57.14
	Cayley-Encoder	**55.84**	**55.97**	**60.25**	**56.65**	**66.99**	**58.63**	56.59	55.34
Gemma2-2B	Last-Layer	36.82	33.70	45.60	37.58	50.22	51.40	58.44	43.01
	Best-Layer	52.97	43.02	58.56	52.43	65.49	58.89	**72.02**	57.24
	MLP Last-Layer	40.72	27.36	36.93	31.32	52.38	51.16	54.10	40.19
	Weighted	40.52	35.66	45.58	38.14	54.88	54.91	65.27	46.67
	DWAtt	53.37	51.86	48.14	55.29	67.73	51.11	57.08	55.36
	Deepset	36.67	43.97	38.97	35.05	50.51	47.34	31.64	42.22
	FC-Encoder	**62.86**	**61.83**	55.43	63.89	**74.68**	**62.96**	66.02	60.12
	Cayley-Encoder	61.62	60.38	**63.98**	**64.59**	73.94	62.69	64.07	**60.32**
Llama3-8B	Last-Layer	45.63	32.65	52.66	42.88	58.07	54.32	67.00	51.16
	Best-Layer	56.51	47.21	60.09	54.62	66.89	59.04	**72.83**	56.96
	MLP Last-Layer	48.91	34.88	54.80	44.08	61.19	52.55	55.35	46.62
	Weighted	45.77	32.86	59.46	54.90	67.30	58.97	52.82	56.91
	DWAtt	52.85	51.47	56.26	58.63	66.00	48.86	46.88	51.45
	Deepset	42.31	39.94	52.80	52.00	56.12	39.50	32.96	46.60
	FC-Encoder	59.33	55.19	53.51	59.88	73.69	57.25	62.81	57.51
	Cayley-Encoder	**63.05**	**65.31**	**63.53**	**70.17**	**76.96**	**63.13**	55.32	**60.74**
Qwen3-8B	Last-Layer	29.26	58.45	37.48	33.88	47.65	46.36	51.83	41.07
	Best-Layer	54.20	58.45	59.74	51.52	62.63	**60.03**	**69.67**	**58.11**
	MLP Last-Layer	26.36	45.20	35.55	31.01	47.06	36.56	53.64	40.93
	Weighted	38.45	52.03	45.05	38.63	57.53	49.71	59.45	50.34
	DWAtt	38.63	44.96	41.00	40.56	53.98	45.68	54.26	42.05
	Deepset	39.53	49.10	33.01	32.24	46.16	48.54	57.05	44.74
	FC-Encoder	54.24	**64.87**	49.30	59.51	69.34	57.47	65.11	56.20
	Cayley-Encoder	**58.69**	63.52	**64.81**	**68.85**	**73.69**	58.28	54.77	57.89

1B, 1.4B, and 2.8B parameters. Using a single model family neutralizes confounding effects due to architectural differences between LLM families. Figure 2 shows performance across Pythia sizes. Both FC-Encoder and Cayley-Encoder consistently outperform all baselines across all Pythia sizes, with gains of 15–40 percentage points over Last-Layer and Best-Layer maintained throughout. Notably, smaller LLMs equipped with our encoders can match the performance of much larger ones: Pythia-14M already achieves 95–96% accuracy on MTOPDomain, just 2–4 percentage points below Pythia-2.8B. This suggests that our approach can be particularly valuable in resource-constrained settings, extracting strong performance from small models by better utilizing their existing layer representations. To verify that these trends are not specific to one family, we repeat the analysis on the Qwen3 family [Yang et al., 2025] with 1.7B, 4B, 8B, and 14B parameters (Appendix J). Both encoders outperform Last-Layer by 35–41 percentage points on average at all sizes. Cayley-Encoder outperforms FC-Encoder on average at every size, and its advantage is larger for the two largest LLMs (8B: +2.35, 14B: +1.47) than for the smaller ones (1.7B: +0.74, 4B: +0.31).

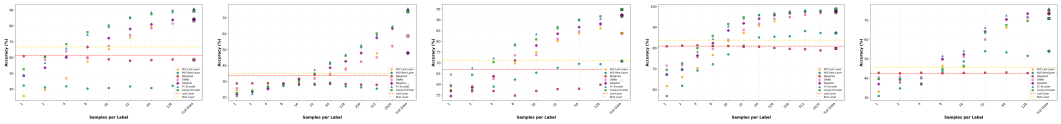


(a) Banking77 (b) Emotion (c) MTOPIntent (d) MTOPDomain (e) PoemSentiment

Figure 2: Performance across the Pythia models ranging from 14M to 2.8B parameters. Both encoders outperform Last-Layer and Best-Layer across all model sizes, with smaller LLMs achieving performance comparable to much larger ones.

4.3 Few-Shot Learning

We evaluate few-shot performance by restricting training to 1–1024 samples per label across all classification tasks. For Poem Sentiment, MTOPIntent, and Banking77, the available training data is limited and is fully covered at 128 samples per label. The results, presented in Figure 3, show that both encoders are highly data-efficient. On Banking77, MTOPDomain, and MTOPIntent, they outperform both Last-Layer and Best-Layer baselines with as few as 8 samples per label, and Cayley-Encoder achieves the highest accuracy on all three tasks at every training set size above 8. On Banking77, 32 samples per label suffice to surpass DWAtt trained on the full dataset. On Poem Sentiment, FC-Encoder approaches its plateau with 64 samples per label, performing comparably to other methods trained on more data. For Emotion, both encoders outperform Last-Layer and Best-Layer starting from 32 samples per label, and surpass all other baselines at larger training sizes. Emotion is the only task that exhibits consistent performance improvements as training data grows; the other tasks plateau between 128 and 512 samples per label. The overall trend suggests that both encoders extract strong representations even from very few examples, making them particularly valuable for practical applications where labeled data is scarce.



(a) Banking77 (b) Emotion (c) MTOPIntent (d) MTOPDomain (e) PoemSentiment

Figure 3: Few-shot learning analysis using Pythia-410M. Both encoders outperform all baselines with as few as 8 samples per label on most tasks. Cayley-Encoder consistently achieves the highest accuracy on Banking77 and the MTOP tasks across all training set sizes.

4.4 Comparison with LoRA

Although our encoders are designed as lightweight post-training modules that can be combined with fine-tuning approaches such as LoRA [Hu et al., 2022], we compare them directly against LoRA to demonstrate their standalone effectiveness. LoRA inserts trainable low-rank matrices into the LLM's attention layers, modifying the model weights themselves. Following Hu et al. [2022], we adapt all attention projections in every layer and keep the feed-forward modules frozen. We searched rank $r \in \{1, 2, 3, 16\}$ via Optuna and report the best-performing rank per model and task (see Appendix E for details). Table 3 presents the results for Pythia-410M, Gemma2-2B, and Llama3-8B. For classification, at least one of our encoders outperforms LoRA in 13 out of 15 configurations. For STS, Cayley-Encoder outperforms LoRA in 21 out of 24 configurations. Notably, these gains are achieved while keeping the LLM entirely frozen and using 6–12$\times$ fewer trainable parameters than LoRA at rank 16 (Table 11). This suggests that structured aggregation over existing layer representations can be more effective than direct weight adaptation, while being substantially more parameter-efficient.

Table 3: FC-Encoder and Cayley-Encoder vs. LoRA on classification tasks, measured by accuracy, and STS tasks, measured by Spearman correlation $\times 100$. At least one of our encoders outperforms LoRA in 13 out of 15 classification configurations, and Cayley-Encoder outperforms LoRA in 21 out of 24 STS configurations, while keeping the LLM frozen. We highlight in bold the best result per task.

Base Model	Method	Banking77	Emotion	MTOPDomain	MTOPIntent	PoemSentiment	STSBench.	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R
Pythia-410M	FC-Encoder	**90.65**	75.61	98.68	**95.04**	**75.77**	54.2	50.1	53.0	51.8	63.9	57.1	58.6	**57.1**
	Cayley-Encoder	89.43	73.83	**98.77**	94.72	70.87	**55.8**	**56.0**	**60.2**	**56.7**	**67.0**	**58.6**	56.6	55.3
	LoRA	82.96	**91.57**	96.64	88.25	70.96	54.1	44.6	51.7	52.3	64.8	45.8	**64.3**	52.0
Gemma2-2B	FC-Encoder	92.39	**79.90**	99.07	95.34	77.12	**62.9**	**61.8**	55.4	63.9	**74.7**	**63.0**	**66.0**	60.1
	Cayley-Encoder	**92.58**	69.60	**99.16**	**96.43**	**83.27**	61.6	60.4	**64.0**	**64.6**	73.9	62.7	64.1	**60.3**
	LoRA	56.77	70.89	89.65	80.83	61.92	58.7	49.1	57.8	60.4	69.9	51.8	58.8	54.2
Llama3-8B	FC-Encoder	92.38	71.64	98.99	96.19	77.98	59.3	55.2	53.5	59.9	73.7	57.2	62.8	57.5
	Cayley-Encoder	**92.85**	73.43	**99.03**	**96.46**	**79.04**	63.1	**65.3**	**63.5**	**70.2**	**77.0**	**63.1**	55.3	**60.7**
	LoRA	90.82	**91.98**	98.59	94.44	76.15	**65.3**	47.8	56.5	61.6	72.8	54.7	**69.6**	58.9

4.5 Token-Level Inter-Layer Aggregation

In all experiments above, each layer is represented by a single mean-pooled vector over tokens, resulting in a graph with L nodes (one per layer). A natural extension is to operate at the token level, constructing a graph with $L \times T$ nodes, one per token per layer, allowing the GNN to model interactions across both tokens and layers simultaneously. This setting is more expressive but substantially increases the graph size, making the choice of graph topology critical for scalability. We evaluate this extension using Gemma2-2B on all classification tasks. Table 4 presents the results. FC-Encoder exceeds GPU memory on Banking77, while Cayley-Encoder handles all tasks, using 12–34$\times$ less GPU memory. This demonstrates the practical advantage of the Cayley graph's sparse $O(n)$ edge complexity over the $O(n^2)$ cost of a fully connected graph as sequences grow longer. Beyond efficiency, token-level aggregation improves accuracy. Compared with the mean-pooled setting (Table 1), Cayley-Encoder with token-level representations achieves higher or comparable accuracy on all classification tasks for Gemma2-2B. Emotion improves from 69.60 to 89.70 and Poem Sentiment from 83.27 to 86.44. This indicates that jointly modeling information across tokens and layers captures complementary signals that are lost by mean-pooling, and suggests that token-level inter-layer aggregation is a promising direction, particularly when paired with a scalable graph topology such as the Cayley graph.

Table 4: Token-level aggregation with Gemma2-2B. Cayley-Encoder uses 12–34$\times$ less GPU memory while achieving comparable or higher accuracy. OOM: out of memory.

		Bank.	Emot.	MT-D	MT-I	Poem
Acc.	Cayley	**92.6**	89.7	99.1	**96.3**	**86.4**
	FC	OOM	**90.4**	**99.2**	93.2	80.2
GPU (MB)	Cayley	**269**	**241**	**141**	**97**	**77**
	FC	OOM	8260	2978	1583	923

4.6 Understanding Inter-Layer Communication

To understand how different layers contribute to the final prediction in Cayley-Encoder, and what aspects of their communication are crucial, we conduct five complementary analyses.

Layer-to-node assignment does not matter. Since the Cayley graph may contain virtual nodes (when $|V_n| > L$), the specific layer-to-node assignment determines which layers are directly connected and which communicate only through virtual nodes. If the identity of communicating layers mattered, this assignment would significantly affect performance. To test this, we fix the model weights and hyperparameters for Cayley-Encoder and re-evaluate with 10 independent random layer-to-node permutations across all classification tasks using Pythia-410M, Gemma2-2B, and Llama3-8B. Table 5 shows that accuracy remains stable with minimal variance (standard deviations below 2 percentage points in all cases), indicating that the assignment has little effect.

Table 5: Robustness to layer-to-node assignment (mean $\pm$ std over 10 random permutations).

	Bank.	Emot.	MT-D	MT-I	Poem
Pythia	90.1±.3	74.9±.8	98.7±.1	94.6±.4	71.3±1.8
Gemma	92.4±.2	73.1±1.5	99.1±.1	96.2±.1	82.3±.5
Llama	92.9±.1	72.8±1.3	98.9±.1	96.4±.2	79.2±1.3

Virtual nodes do not dilute the representation. For small LLMs, virtual nodes can outnumber the layers. For example, Pythia-14M and Pythia-70M map 7 layer representations to a 24-node Cayley graph, so 71% of the nodes are virtual, and Cayley-Encoder still outperforms all baselines (Figure 2). To test this directly, we train Cayley-Encoder with Gemma2-2B on graphs larger than required, increasing the fraction of virtual nodes from 44% ($n = 4$, 48 nodes) to 78% ($n = 5$, 120 nodes) and 81% ($n = 6$, 144 nodes). STS performance does not decrease up to 78% and decreases only slightly at 81%, with 5 out of 8 tasks unaffected (Appendix H).

Layer identity is not a useful signal. We augment each node in the layer graph with a learnable positional encoding indexed by its original depth in the LLM, added to the node features prior to message passing. This provides the GNN with explicit information about which LLM layer each node represents. We train and evaluate on all classification tasks using Pythia-410M. Table 6 shows that adding positional encodings degrades performance in most cases,

Table 6: Effect of adding learnable layer-position encodings. Layer identity information degrades performance in most cases.

	Bank.	Emot.	MT-D	MT-I	Poem
Cayley	89.4	73.8	98.8	94.7	70.9
Cayley (PE)	90.3	75.5	98.7	94.6	69.2
FC	90.7	75.6	98.7	95.0	75.8
FC (PE)	86.3	72.1	98.4	94.1	68.4

for both Cayley-Encoder and FC-Encoder. This suggests that layer identity is not a useful signal for the aggregation.

Cayley Degree 4 vs 6 and 8. Our Cayley graphs are 4-regular, since they are generated by the Margulis generating set (Appendix A). To evaluate the effect of the degree, we construct Cayley graphs over $SL(2, \mathbb{Z}_n)$ of degree 6 and 8 by extending the generating set with additional generators and their inverses (Appendix G). We evaluate them using Qwen3-8B on all classification tasks. Table 7 shows that the degree-6 and degree-8 graphs do not improve over the degree-4 graph, which achieves the best result on 4 out of 5 tasks and the highest average. We therefore use the degree-4 graph, the sparsest of the three, throughout the paper.

Table 7: Effect of the Cayley graph degree (Qwen3-8B). The degree-4 graph used throughout the paper achieves the best result on 4 of 5 tasks and the best average.

	Bank.	Emot.	MT-D	MT-I	Poem	Avg
Cayley (deg 4)	**92.34**	**75.70**	**98.99**	**95.99**	85.38	**89.68**
Cayley (deg 6)	92.01	75.23	98.96	95.77	78.75	88.14
Cayley (deg 8)	92.31	66.28	98.74	94.76	**87.21**	87.86

All layers contribute. We apply GNNExplainer [Ying et al., 2019], a post-hoc attribution method for GNNs, to Cayley-Encoder trained on each classification task with Gemma2-2B. For each test instance, GNNExplainer identifies which nodes in the layer graph the model relied on most for its prediction. Figure 4 shows the mean node importance per layer, averaged over all test instances across all five tasks. All layers contribute roughly equally, with no single layer dominating. This suggests that Cayley-Encoder aggregates complementary information from across the full depth of the LLM rather than collapsing to a single layer.

Together, these results suggest that the primary driver of Cayley-Encoder's effectiveness is enabling all layers to communicate through an expressive graph topology, rather than the specific identity or ordering of the layers. The benefit lies in structured inter-layer communication itself, not in privileging particular layers or positions. This is consistent with the role of the graph in our method. We do not claim that any specific connectivity between layers carries task information. Rather, any aggregation over layers imposes some structure, and since this structure is independent of the target variable, we choose one that reduces the risk of overfitting to it while ensuring efficient communication between all layers. In line with this, replacing the Cayley graph with a random 4-regular graph, which does not guarantee these properties, degrades performance (Appendix F).

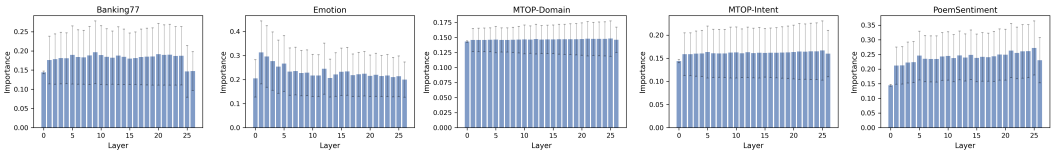


Figure 4: Per-layer contribution to the final prediction for Cayley-Encoder across classification tasks using Gemma2-2B. Estimated via GNNExplainer and averaged over all test instances. Error bars show standard deviation. All layers contribute roughly equally.

5 Conclusion

We introduced Cayley-Encoder, a lightweight method for aggregating representations from all internal layers of a frozen LLM through a structured graph topology. Across 13 diverse tasks and 13 LLMs, Cayley-Encoder consistently outperformed all baselines by large margins, including attention-based aggregation, best-layer selection, and LoRA fine-tuning, while adding at most 0.1% parameters. These gains persist in few-shot regimes and across LLM sizes from 14M to 14B parameters. Our analysis reveals that the benefit stems from enabling effective communication across all layers, rather than from the identity or ordering of individual layers. This suggests that the internal depth of LLMs contains complementary task-relevant signals that standard last-layer extraction fails to exploit, and that structured inter-layer aggregation provides a simple and efficient way to unlock them.

Several promising directions emerge. Scaling to larger architectures (e.g., 70B+ parameters) and generative settings remains to be explored. Additionally, integrating structured inter-layer communication directly into LLM pre-training, rather than as a post-training module, could yield models with improved information flow across depth.

Acknowledgments and Disclosure of Funding

This study was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: a pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, page 385–393, USA, 2012. Association for Computational Linguistics.

Eneko Agirre, Daniel Matthew Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *sem 2013 shared task: Semantic textual similarity. In *International Workshop on Semantic Evaluation*, 2013. URL https://api.semanticscholar.org/CorpusID:10241043.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91, 2014.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263, 2015.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In Steven Bethard, Marine Carpuat, Daniel Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1081. URL https://aclanthology.org/S16-1081/.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications, 2021. URL https://arxiv.org/abs/2006.05205.

Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In *Forty-first International Conference on Machine Learning*, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In Tsung-Hsien Wen, Asli Celikyilmaz, Zhou Yu, Alexandros Papangelis, Mihail Eric, Anuj Kumar, Iñigo Casanueva, and Rushin Shah, editors,

Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, pages 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828/.

Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pages 38:1–38:18, 09–12 Dec 2022. URL https://proceedings.mlr.press/v198/deac22a.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, June 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Muhammad ElNokrashy, Badr AlKhamissi, and Mona Diab. Depth-wise attention (DWAtt): A layer fusion method for data-efficient classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4665–4674, May 2024. URL https://aclanthology.org/2024.lrec-main.417/.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*, 2020.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*, 2024.

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*, pages 1263–1272. Pmlr, 2017.

Georgios Gkarmpounis, Christos Vranis, Nicholas Vretos, and Petros Daras. Survey on graph neural networks. *IEEE Access*, 12:128816–128832, 2024. doi: 10.1109/ACCESS.2024.3456913.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356/.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.257. URL https://aclanthology.org/2021.eacl-main.257.

Krishna Sri Ipsit Mantri, Carola-Bibiane Schönlieb, Zorah Lähner, and Moshe Eliasof. Towards improved sentence representations using token graphs. *arXiv preprint arXiv:2603.03389*, 2026.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.

Philip May. Machine translated multilingual sts benchmark dataset. 2021. URL https://github.com/PhilipMay/stsb-multi-mt.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, 2023.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging, 2024. URL https://arxiv.org/abs/2402.02622.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, June 2018. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202/.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL https://aclanthology.org/D18-1404.

Emily Sheng and David C Uthus. Investigating societal biases in a poetry composition system. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 93–106, 2020.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models, 2025. URL https://arxiv.org/abs/2502.02013.

Gizem Soğancıoğlu, Hakime Öztürk, and Arzucan Özgür. BIOSSES: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, 33(14):i49–i58, 07 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx238. URL https://doi.org/10.1093/bioinformatics/btx238.

Gemma Team. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL https://www.kaggle.com/m/3301.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, July 2019. doi: 10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Jonas Wallat, Jaspreet Singh, and Avishek Anand. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 174–183, November 2020. doi: 10.18653/v1/2020.blackboxnlp-1.17. URL https://aclanthology.org/2020.blackboxnlp-1.17/.

JJ Wilson, Maya Bechler-Speicher, and Petar Veličković. Cayley graph propagation. In *Proceedings of the Third Learning on Graphs Conference*, volume 269 of *Proceedings of Machine Learning Research*, pages 8:1–8:20, 26–29 Nov 2025. URL https://proceedings.mlr.press/v269/wilson25a.html.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. *arXiv preprint arXiv:2502.12170*, 2025.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019. URL https://arxiv.org/abs/1810.00826.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

A Cayley Graph Construction Details

The Cayley graph over $SL(2, \mathbb{Z}_n)$ is constructed using the symmetric generating set $S = \{A, A^{-1}, B, B^{-1}\}$, where

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

with all operations performed modulo n. These are the standard Margulis generators [Deac et al., 2022], producing a 4-regular graph with logarithmic diameter. Each element of $SL(2, \mathbb{Z}_n)$ corresponds to a node in the graph, and two nodes are connected if one can be obtained from the other by left-multiplication with a generator. The resulting graph is vertex-transitive, meaning all nodes are structurally equivalent.

B Trainable Parameter Counts

Table 8 reports the number of trainable parameters introduced by each method. All base LLM parameters remain frozen throughout. FC-Encoder and Cayley-Encoder introduce at most 0.1% additional parameters relative to the base model, while DWAtt requires 3–5$\times$ more parameters.

Table 8: Trainable parameters added by each method. All base model parameters remain frozen. Percentages show overhead relative to base model size.

Method	Pythia-410M	Gemma2-2B	Llama3-8B	Qwen3-8B
Frozen Parameters	*410M*	*2.6B*	*8B*	*8B*
Weighted	25	27	33	37
MLP	394K (0.10%)	721K (0.028%)	1.18M (0.015%)	1.18M (0.014%)
Deepset	394K (0.10%)	721K (0.028%)	1.18M (0.015%)	1.18M (0.014%)
FC-Encoder	395K (0.10%)	722K (0.028%)	1.18M (0.015%)	1.18M (0.014%)
Cayley-Encoder	395K (0.10%)	722K (0.028%)	1.18M (0.015%)	1.18M (0.014%)
DWAtt	2.0M (0.49%)	2.46M (0.095%)	3.3M (0.041%)	3.6M (0.045%)

C Implementation Details

We provide full hyperparameter search spaces for all methods below. All methods use the Adam optimizer [Kingma, 2014] with hyperparameters selected via Optuna [Akiba et al., 2019] using MedianPruner (5 startup trials, 10 warmup epochs, checked every 5 epochs). The number of Optuna trials varied between 20 and 50 depending on the method.

FC-Encoder and Cayley-Encoder. Number of MPNN layers $\in \{1, 2\}$, dropout $\in \{0.0, 0.1, 0.2, 0.3\}$, number of MLP layers inside GIN aggregation $\in \{1, 2\}$, learning rate $\in \{10^{-4}, 10^{-3}\}$, weight decay $\in \{10^{-4}, 10^{-3}\}$. Hidden dimension is fixed to 256 and batch size to 64.

MLP Last Layer. Number of MLP layers $\in \{1, 2, 3\}$, dropout $\in \{0.0, 0.1, 0.2, 0.3\}$, learning rate $\in \{10^{-4}, 10^{-3}\}$, weight decay $\in \{10^{-4}, 10^{-3}\}$. Hidden dimension is fixed to 256.

Deepset. Pre-pooling layers $\in \{0, 1\}$, post-pooling layers $\in \{0, 1, 2\}$, pooling type $\in \{\text{mean}, \text{sum}\}$, dropout $\in \{0.0, 0.1, 0.2, 0.3\}$, learning rate $\in \{10^{-4}, 10^{-3}\}$, weight decay $\in \{10^{-4}, 10^{-3}\}$. Hidden dimension is fixed to 256.

DWAtt. Architecture is fixed to match the original paper: bottleneck ratio $\gamma = 0.5$, positional embedding dimension $d_{\text{pos}} = 24$, and hidden dimension 256. We search dropout $\in \{0.0, 0.1, 0.2, 0.3\}$, learning rate $\in \{10^{-4}, 10^{-3}\}$, and weight decay $\in \{10^{-4}, 10^{-3}\}$.

Weighted. The architecture has no tunable structure (only L scalar layer weights with softmax). We search learning rate $\in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and weight decay $\in \{0, 10^{-4}, 10^{-3}\}$.

LoRA. Rank $r \in \{1, 2, 3, 16\}$, alpha $\in \{8, 16, 32\}$, LoRA dropout $\in \{0.0, 0.1, 0.2\}$, learning rate $\in \{10^{-4}, 10^{-3}\}$, weight decay $\in \{10^{-4}, 10^{-3}\}$. Batch size is 32 and training runs for 20 epochs.

Table 9: Hyperparameter search spaces. All methods use Optuna with MedianPruner (5 startup trials, 10 epochs) and 20-50 trials per study.

(a) Shared training hyperparameters

Hyperparameter	Search Space
Learning rate	{1e-4, 1e-3}
Weight decay	{1e-4, 1e-3}
Dropout	{0.0, 0.1, 0.2, 0.3}
Batch size	64
Epochs	50 (cls) / 25 (STS)

(b) Cayley-Encoder / FC-Encoder

Hyperparameter	Search Space
GNN layers	{1, 2}
GIN MLP layers	{1, 2}
Hidden dim	256
Readout	{mean, sum}

(c) MLP

Hyperparameter	Search Space
MLP layers	{1, 2, 3}
Hidden dim	256

(d) Deepset

Hyperparameter	Search Space
Pre-pool layers	{0, 1}
Post-pool layers	{0, 1, 2}
Pooling type	{mean, sum}
Hidden dim	256

(e) DWAtt

Hyperparameter	Search Space
Bottleneck ratio	0.5
Pos. embed dim	24
Hidden dim	256

(f) Weighted

Hyperparameter	Search Space
Learning rate	{1e-4, 1e-3, 1e-2}
Weight decay	{0, 1e-4, 1e-3}

(g) LoRA

Hyperparameter	Search Space
Rank (r)	{1, 2, 3, 16}
Alpha	{8, 16, 32}
LoRA dropout	{0.0, 0.1, 0.2}
Batch size	32
Epochs	20

D Positional Encoding Ablation

We evaluate the effect of adding learnable layer-position encodings to FC-Encoder and Cayley-Encoder. Each node is assigned a learned embedding indexed by its original depth in the base LLM, added to the node features prior to message passing. We train and evaluate on both classification and STS tasks using Pythia-410M.

Table 10 presents the results. For classification, adding positional encodings yields comparable performance for Cayley-Encoder but consistently degrades FC-Encoder. For STS, positional encodings degrade Cayley-Encoder while providing marginal improvements for FC-Encoder. Overall, layer-position information does not add value, and in most cases hurts performance. This supports the finding from our main analysis that effective inter-layer communication, rather than layer identity, drives the encoders' effectiveness.

Table 10: Positional encoding (PE) ablation on Pythia-410M. Classification: accuracy (%), STS: Spearman correlation ($\times 100$). **Bold**: best per column within each encoder type.

	Classification						STS								
Method	Banking77	Emotion	MTOPDom.	MTOPInt.	PoemSent.	Avg	STS-B	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R	Avg
Cayley-Encoder	89.43	73.83	**98.77**	**94.72**	**70.87**	85.52	**55.8**	**56.0**	**60.2**	**56.7**	**67.0**	**58.6**	**56.6**	**55.3**	**58.3**
Cayley-Encoder (PE)	**90.25**	**75.52**	98.72	94.58	69.23	**85.66**	55.3	51.4	54.8	48.2	63.0	53.2	44.8	54.2	53.1
FC-Encoder	**90.65**	**75.61**	**98.68**	**95.04**	**75.77**	**87.15**	54.2	50.1	53.0	51.8	63.9	**57.1**	**58.6**	**57.1**	55.7
FC-Encoder (PE)	86.33	72.07	98.36	94.10	68.37	83.85	**55.3**	**52.5**	**55.9**	**54.2**	**65.0**	56.3	56.5	56.6	**56.5**

E Additional LoRA Results

LoRA adapters are inserted into all attention projections in every layer of the LLM: the fused query-key-value projection and the attention output projection for Pythia, and the query, key, value, and output projections for Gemma2 and Llama3. The feed-forward modules remain frozen, as in Hu et al. [2022].

Table 11 compares the trainable parameter counts of LoRA at ranks 3 and 16 against Cayley-Encoder. At rank 16, LoRA introduces 6–12$\times$ more trainable parameters than Cayley-Encoder, depending on the model size. Despite this parameter advantage, Cayley-Encoder outperforms LoRA on the majority of tasks while keeping the LLM entirely frozen.

Table 11: Trainable parameters for LoRA (ranks 3 and 16) compared to Cayley-Encoder. Percentages show overhead relative to base model size.

Method	Pythia-410M	Gemma2-2B	Llama3-8B
Frozen Parameters	*410M*	*2.6B*	*8B*
LoRA ($r = 3$)	442K (0.11%)	1.20M (0.046%)	2.56M (0.032%)
LoRA ($r = 16$)	2.36M (0.58%)	6.39M (0.25%)	13.6M (0.17%)
Cayley-Encoder	395K (0.10%)	722K (0.028%)	1.18M (0.015%)

F Cayley Graph vs. Random Regular Graph

To determine whether the performance of Cayley-Encoder stems from its algebraic structure or simply from regularity and sparsity, we replace the Cayley graph with a random 4-regular graph. The random graph matches the Cayley graph in degree, regularity, and sparsity, but does not guarantee its expansion properties. We evaluate across Pythia-410M, Gemma2-2B, and Llama3-8B on all classification and STS tasks, with hyperparameters tuned independently for each method.

Tables 12 and 13 present the results. The Cayley graph outperforms the random 4-regular baseline on 9 out of 15 classification and 22 out of 24 STS task–model pairs, with the largest gains on Llama3-8B. This indicates that the algebraic structure of the Cayley graph, not merely its regularity or sparsity, contributes meaningfully to performance.

Table 12: Ablation isolating the contribution of the Cayley-Encoder graph structure. Our Cayley-Encoder uses a 4-regular topology; we ask whether its gains over FC-Encoder come from sparsity alone or from the specific Cayley-Encoder connectivity. Replacing the Cayley-Encoder graph with a random 4-regular graph in classification tasks shows comparable results. Cayley-Encoder is better in 9 out of 15 cases. **Bold**: best per column.

Base Model	Graph Type	Banking77	Emotion	MTOPDomain	MTOPIntent	PoemSentiment
Pythia-410M	Random-Regular	**90.38**	**75.15**	**98.80**	**94.73**	70.77
	Cayley-Encoder	89.43	73.83	98.77	94.72	**70.87**
Gemma2-2B	Random-Regular	92.16	**76.50**	99.12	96.21	82.31
	Cayley-Encoder	**92.58**	69.60	**99.16**	**96.43**	**83.27**
Llama3-8B	Random-Regular	92.23	72.99	98.90	96.29	**79.81**
	Cayley-Encoder	**92.85**	**73.43**	**99.03**	**96.46**	79.04

Table 13: Ablation isolating the contribution of the Cayley-Encoder graph structure. Our Cayley-Encoder uses a 4-regular topology; we ask whether its gains over FC-Encoder come from sparsity alone or from the specific Cayley connectivity. Replacing the Cayley graph with a random 4-regular graph degrades performance on 22 of 24 task–model pairs across 3 LLMs and 8 STS benchmarks, showing that the expansion properties of the Cayley graph, not regularity alone, drive the improvement. **Bold**: best per column.

Base Model	Method	STSBenchmark	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R
Pythia-410M	Random-Regular	53.19	51.69	56.81	52.82	62.19	54.27	47.81	**55.66**
	Cayley-Encoder	**55.84**	**55.97**	**60.25**	**56.65**	**66.99**	**58.63**	**56.59**	55.34
Gemma2-2B	Random-Regular	59.85	52.57	58.25	58.73	72.40	61.11	62.62	58.34
	Cayley-Encoder	**61.62**	**60.38**	**63.98**	**64.59**	**73.94**	**62.69**	**64.07**	**60.32**
Llama3-8B	Random-Regular	54.46	44.55	48.92	46.02	65.47	55.55	**57.68**	53.51
	Cayley-Encoder	**63.05**	**65.31**	**63.53**	**70.17**	**76.96**	**63.13**	55.32	**60.74**

G Cayley Graph Degree

To study the effect of degree, we extend S (Appendix A) while keeping the group fixed. Since a Cayley graph's degree equals $|S|$ and S must be closed under inverses, we add generators in inverse pairs: $S_6 = S \cup \{AB, (AB)^{-1}\}$ and $S_8 = S_6 \cup \{AB^{-1}, (AB^{-1})^{-1}\}$, where

$$AB = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \qquad AB^{-1} = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}$$

Both have determinant 1 and hence lie in $SL(2, \mathbb{Z}_n)$. For $n \geq 3$ all generators are distinct, so the new resulting graphs are exactly 6 and 8-regular.

H Sensitivity to Virtual Nodes

When $|V_n| > L$, the Cayley graph contains virtual nodes that are initialized with zero vectors and excluded from the final representation. To test whether a large fraction of virtual nodes dilutes the learned representations, we train Cayley-Encoder with Gemma2-2B (27 layer representations) on Cayley graphs larger than required. Instead of the smallest n such that $|V_n| \geq L$ ($n = 4$, 48 nodes, 44% virtual), we use $n = 5$ (120 nodes, 78% virtual) and $n = 6$ (144 nodes, 81% virtual) with the same hyperparameters. We train on STSBenchmark and evaluate on all STS tasks. Table 14 shows that average performance does not decrease between 44% and 78% virtual nodes (63.95 and 64.50) and decreases slightly at 81% (62.01). This decrease is driven by STS12, STS13, and STS14, while the remaining 5 tasks are unaffected. Since we use the smallest n such that $|V_n| \geq L$, such high virtual-node fractions do not arise for the LLMs we evaluate, except for the smallest Pythia models (71%), where Cayley-Encoder still outperforms all baselines (Figure 2).

Table 14: Sensitivity to the fraction of virtual nodes. Cayley-Encoder is trained with Gemma2-2B on Cayley graphs larger than required ($n = 4$ is the default) and evaluated on the STS tasks (Spearman correlation $\times 100$). We highlight in bold the best result per column.

Configuration	STSBenchmark	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R	Avg
$n = 4$ (48 nodes, 44% virtual)	61.62	60.38	63.98	**64.59**	73.94	62.69	64.07	60.32	63.95
$n = 5$ (120 nodes, 78% virtual)	**61.96**	**61.08**	**64.19**	64.38	**74.76**	**64.13**	**64.50**	**61.02**	**64.50**
$n = 6$ (144 nodes, 81% virtual)	60.56	54.84	60.36	59.28	74.03	63.51	63.38	60.14	62.01

I DWAtt With and Without Projection

In our main experiments, DWAtt operates on layer representations projected to 256 dimensions. Without the projection, the attention matrices of DWAtt scale with the hidden dimension of the LLM, requiring 69–541$\times$ more trainable parameters than Cayley-Encoder (Table 15). To evaluate the effect of the projection, we also train DWAtt directly on the full-dimensional layer representations. Tables 16 and 17 present the results. The effect of the projection is task-dependent. On classification, the projection improves DWAtt on all four LLMs, and Cayley-Encoder outperforms DWAtt without

the projection on all tasks and LLMs. On STS, removing the projection improves DWAtt on all four LLMs, and Cayley-Encoder still achieves a higher average on 3 out of 4 LLMs. On Llama3-8B, DWAtt without the projection achieves a higher average by 1.3 points, while using 484× more trainable parameters than Cayley-Encoder.

Table 15: Trainable parameters of DWAtt with and without the 256-dimensional projection, compared to Cayley-Encoder.

Method	Pythia-410M	Gemma2-2B	Llama3-8B	Qwen3-8B
Cayley-Encoder	395K	722K	1.18M	1.18M
DWAtt (projection)	2.0M	2.46M	3.3M	3.6M
DWAtt (no projection)	27.4M	149.0M	571.1M	638.3M
No projection / Cayley-Encoder	*69×*	*206×*	*484×*	*541×*

Table 16: DWAtt with and without the 256-dimensional projection on STS tasks (Spearman correlation ×100). We highlight in bold the best result per column for each LLM.

Base Model	Method	STSBenchmark	STS12	STS13	STS14	STS15	STS16	BIOSSES	SICK-R	Avg
Pythia-410M	DWAtt (projection)	49.4	52.5	44.6	47.5	52.5	44.8	45.3	52.4	48.6
	DWAtt (no projection)	46.2	**57.2**	51.3	52.6	60.8	53.9	**65.0**	**56.9**	55.5
	Cayley-Encoder	**55.8**	56.0	**60.2**	**56.7**	**67.0**	**58.6**	56.6	55.3	**58.3**
Gemma2-2B	DWAtt (projection)	53.4	51.9	48.1	55.3	67.7	51.1	57.1	55.4	55.0
	DWAtt (no projection)	54.1	54.7	56.0	59.3	68.8	57.7	63.9	54.7	58.7
	Cayley-Encoder	**61.6**	**60.4**	**64.0**	**64.6**	**73.9**	**62.7**	**64.1**	**60.3**	**63.9**
Llama3-8B	DWAtt (projection)	52.9	51.5	56.3	58.6	66.0	48.9	46.9	51.5	54.1
	DWAtt (no projection)	**66.3**	**66.4**	61.8	**71.1**	**79.5**	62.5	**58.9**	**62.1**	**66.1**
	Cayley-Encoder	63.1	65.3	**63.5**	70.2	77.0	**63.1**	55.3	60.7	64.8
Qwen3-8B	DWAtt (projection)	38.6	45.0	41.0	40.6	54.0	45.7	54.3	42.1	45.1
	DWAtt (no projection)	36.8	58.9	31.2	34.0	55.6	49.3	**59.5**	46.8	46.5
	Cayley-Encoder	**58.7**	**63.5**	**64.8**	**68.9**	**73.7**	**58.3**	54.8	**57.9**	**62.6**

Table 17: DWAtt with and without the 256-dimensional projection on classification tasks (accuracy). OOM: out of GPU memory; the corresponding average is over the remaining four tasks. We highlight in bold the best result per column for each LLM.

Base Model	Method	Banking77	Emotion	MTOPDomain	MTOPIntent	PoemSentiment	Avg
Pythia-410M	DWAtt (projection)	83.23	58.60	98.03	91.66	**70.87**	80.48
	DWAtt (no projection)	84.81	57.98	98.14	88.26	61.83	78.20
	Cayley-Encoder	**89.43**	**73.83**	**98.77**	**94.72**	**70.87**	**85.52**
Gemma2-2B	DWAtt (projection)	88.82	61.22	98.39	90.93	67.79	81.43
	DWAtt (no projection)	63.20	59.11	96.06	85.42	40.10	68.78
	Cayley-Encoder	**92.58**	**69.60**	**99.16**	**96.43**	**83.27**	**88.21**
Llama3-8B	DWAtt (projection)	90.04	66.55	97.97	92.41	75.00	84.39
	DWAtt (no projection)	OOM	59.03	84.05	86.91	75.19	76.30
	Cayley-Encoder	**92.85**	**73.43**	**99.03**	**96.46**	**79.04**	**88.16**
Qwen3-8B	DWAtt (projection)	91.22	68.31	98.97	92.10	73.94	84.91
	DWAtt (no projection)	85.39	59.92	97.31	73.86	63.56	76.01
	Cayley-Encoder	**92.34**	**75.70**	**98.99**	**95.99**	**85.38**	**89.68**

J Effectiveness Across Qwen3 Sizes

To complement the Pythia analysis in Section 4.2 with a second LLM family, we evaluate Last-Layer, FC-Encoder, and Cayley-Encoder on all classification tasks using Qwen3 [Yang et al., 2025] with 1.7B, 4B, 8B, and 14B parameters. Table 18 presents the results. Both encoders outperform Last-Layer by 35–41 percentage points on average at all sizes. Cayley-Encoder outperforms FC-Encoder in 16 out of 20 configurations and on average at every size, with a larger advantage for the two largest LLMs (8B: +2.35, 14B: +1.47) than for the smaller ones (1.7B: +0.74, 4B: +0.31).

Table 18: Classification accuracy across Qwen3 sizes. Cayley-Encoder outperforms FC-Encoder on average at every size, with a larger margin for the two largest LLMs. We highlight in bold the best result per column for each LLM.

Base Model	Method	Banking77	Emotion	MTOPDomain	MTOPIntent	PoemSentiment	Avg
Qwen3-1.7B	Last-Layer	53.23	31.88	72.98	58.12	39.52	51.15
	FC-Encoder	91.56	68.96	**98.94**	94.91	79.33	86.74
	Cayley-Encoder	**92.27**	**70.01**	98.87	**95.36**	**80.87**	**87.48**
Qwen3-4B	Last-Layer	55.39	31.45	74.61	62.54	39.13	52.62
	FC-Encoder	91.39	**76.05**	98.60	95.51	79.62	88.23
	Cayley-Encoder	**91.78**	75.91	**98.83**	**95.90**	**80.29**	**88.54**
Qwen3-8B	Last-Layer	48.12	30.18	71.79	56.67	37.31	48.81
	FC-Encoder	91.90	69.66	**99.02**	95.60	80.48	87.33
	Cayley-Encoder	**92.34**	**75.70**	98.99	**95.99**	**85.38**	**89.68**
Qwen3-14B	Last-Layer	47.79	29.96	69.32	58.10	41.35	49.30
	FC-Encoder	92.41	73.54	98.81	**96.34**	80.19	88.26
	Cayley-Encoder	**92.52**	**78.68**	**99.15**	96.16	**82.12**	**89.73**

K Hardware Details

All experiments were conducted on an HPC cluster with access to NVIDIA A6000, A100, V100, and GeForce RTX 3090 GPUs. We do not report training time comparisons, as experiments were distributed across different GPU types and scheduling queues, making direct timing comparisons unreliable.

L Asset Licenses and Terms of Use

All models and datasets used in this work are publicly available and used in accordance with their respective licenses.

L.1 Model Licenses

The following large language models served as the frozen backbones for our experiments [Biderman et al., 2023, Team, 2024, AI@Meta, 2024, Yang et al., 2025] :

- **Pythia Suite (14M–2.8B)**: Licensed under the **Apache License 2.0**.
- **Gemma2-2B**: Released under the **Gemma Community License**.
- **Llama3-8B**: Released under the **Meta Llama 3 Community License Agreement**.
- **Qwen3 (1.7B–14B)**: Licensed under the **Apache License 2.0**.

L.2 Dataset Licenses (MTEB Benchmark)

The datasets used for evaluation were accessed via the Massive Text Embedding Benchmark (MTEB) [Muennighoff et al., 2023] which is licensed under the **Apache License 2.0**. Individual dataset licenses are summarized below:

Task Category	**Dataset**	**License**
Classification	Banking77	**CC BY 4.0**
	Emotion	**CC BY 4.0**
	MTOP (Domain/Intent)	**CC BY-SA 4.0**
	Poem Sentiment	**Apache License 2.0**
STS	STSBenchmark	**CC BY-SA 4.0**
	STS12–16	**CC BY-SA 4.0**
	BIOSSES	**CC BY-NC 4.0**
	SICK-R	**CC BY-NC-SA 4.0**

Table 19: Verified licenses for the datasets used in this work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We show the claims in the Experiments and Results section of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations as future work in section 5 - Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The relevant theory regarding this paper is contained in the cited papers about Cayley Graph Propagation. Here we make use of this method for internal LLM layers. This is experimental only.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed protocol of training and includes the hyperparameters search space for all experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include a GitHub repository link in a footnote in Section 4. The repo includes all our code and experiments that can be reproduced. The only disclaimer is the resources and GPU machines available. The experiments require GPU and might not run properly if the GPU is not sufficient (possibly get OOM).

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The training protocol is specified in Experiments section and the full hyperparameter search space is included in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 5 reports mean and std over 10 random layer-to-node mappings for Cayley-Encoder (stds: 0.05–1.83). Figure 4 reports per-layer importance with error bars representing standard deviation across test instances. Appendix F shows Cayley-Encoder wins 22/24 STS comparisons against a matched random 4-regular baseline.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix K describes the compute resources used. In addition, in the github repo, the generated jobs to execute experiments specify amount of memory and time needed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors read the Code of Ethics and verified the research followed it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper suggests a new method for using all information from LLM layers for downstream tasks. There is no focus on societal impact and no direct way to exploit it.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper doesn't include released models but suggests a new way to incorporate all information from LLM layers.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets are cited in the References section. Licenses are in Appendix L

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the code used to reproduce all experiments in a public repository linked in the main paper (footnote in Section 4).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development does not involve LLMs as important, original or non standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.