GAMER: Graphs As Multi-modal intErmediate Representations; An exploration of a high-resolution, graph-based approach for multi-modality

Daniel Volkov **Dekel Yehezkel** **Shahar Glam** **Neri Nigberg**

{danielvolkov, dekely1, shaharglam, nerinigberg}@mail.tau.ac.il

Abstract

The current State-of-the-Art when considering multi-modal tasks often leverages cross attention in transformers in order to implement multi-modal fusion, often ignoring useful information about the nature of the different modalities. We introduce a novel approach, considering graphs as an intermediate abstraction for multi-modal input, and using a parameter-efficient GNN to learn multi-modal fusion. We present evidence that on the VQA task, our approach successfully implements multi-modal text-image fusion, and performs better when operating with un-aligned embeddings rather than aligned ones. Our implementation of our approach is available at our GitHub repository[1].

1 Introduction

The rise of transformers [24] established a new State-of-the-Art (SotA) standard for generating high-quality embeddings; through their self-attention mechanism, these models excel at capturing both intra-modal relationships [5, 6, 9] as well as cross-modal relationships [18, 13, 19].

However, recent papers may hint these techniques are not without fault. Evidence shows modern SotA models have difficulty with fine-grained tasks [30], provide sub-human results in visual-spatial tasks [29] and even struggle with basic reasoning [23]; meaning these models fail to learn fundamental interactions between different modalities.

In this paper, we explore an alternative approach to representing multi-modal inputs. We hypothesize that modern techniques often fail to capture the *relationship* between modalities, and examine the potential of capturing these relationships using graphs, utilizing GNNs [21] as our neural network architecture.

We compare our proposed multi-modal graph representation against a Cayley graph baseline, leveraging the GraphGPS architecture [20] with various SAGPool [10] configurations in our experimental framework. Our empirical evaluation demonstrates that our proposed representation yields better results than the modality-agnostic Cayley graph approach. We conclude by analyzing the factors contributing to these improvements and identifying promising directions for future research that could advance cross-modal representation learning.

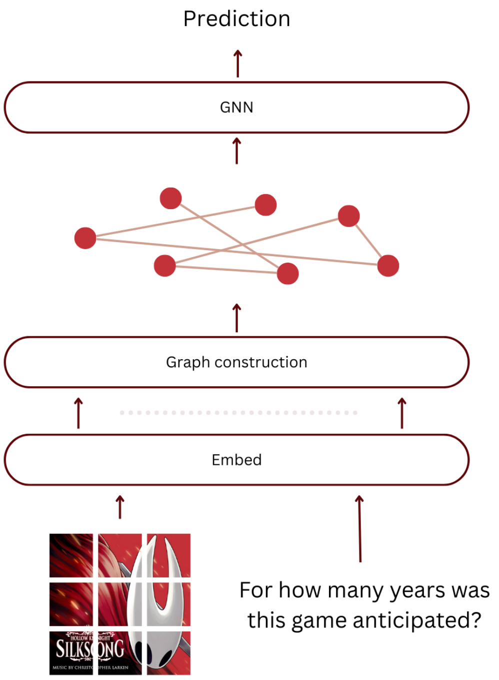


Figure 1: A high-level illustration of our methodology. Each modality is broken down into smaller atomic units (text → tokens, image → patches) which are then embedded. A graph is constructed with these embeddings as it's node features. A GNN is trained to produce predictions conditioned on these graphs.

2 Prior work

[28] surveys popular approaches to multi-modal learning, highlighting how self-attention is applied to capture cross-modal interactions; one can observe that most methods may be reduced to variants of summation, concatenation, or transformer-based fusion over uni-modal embeddings. These

[1] https://github.com/D4niel0s/GAMER

methods, while powerful, generally treat tokens in a modality-agnostic way, assuming all modalities can be represented with the same structure.

Considering research in the field of graph-based modality representation – GNNs have been explored mainly for niche and narrow domains. For instance, [11] models emotion recognition in conversations by constructing graphs where each node represents a modality-specific remark and is connected both temporally and cross-modally. Similarly, multi-modal GNNs have been applied to recommender systems [17] to model interactions between users and items, though in both cases the design encodes strong inductive biases about input structure.

For cases where the graph isn't so easily deduced, strides have been made in the realm of topology-prediction [7, 4] - where the structure of the graph is learned via a separate down-stream task, reminiscent of meta-learning. However these methods are both computationally demanding and are often task-specific themselves (for example, topology-prediction for node-classification tasks)

Ultimately, to the best of our knowledge, no prior work explicitly models modality-specific structures and their cross-modal interactions.

3 Method

To achieve high-resolution multi-modal modeling, we choose to break down each modality into smaller atomic units – tokens for the text modality, and patches for the image modality. As illustrated in figure 1, our approach consists of 3 main components: 1) The embedding used for the atomic units. 2) The graph construction method, used to construct graphs from the resulting embeddings. 3) The GNN used for prediction.

While our approach could be generalized to a wide array of multi-modal tasks, we provide a proof-of-concept, and focus our experimentation on Visual Question Answering (VQA). The VQA task, as introduced in [1], consists of $\langle$Question,Image$\rangle$ pairs, where the questions are about the images. The model is tasked to predict the answer to the question, conditioned on both image and question.

3.1 Embeddings

Since our approach leverages the token and patch embeddings as the initial features for our graphs, we aim to get meaningful token/patch-level embeddings. To test various aspects of our approach, we experiment with different types of embeddings:

Unaligned embeddings. Uni-modal encoders are used to embed each modality *'in a vacuum'*. By using unaligned embeddings as the initial node features of our input graphs, we essentially force our GNN to learn multi-modal fusion as it optimizes for the task at hand. For the text modality we choose the celebrated BERT encoder [5], providing token embeddings that are contextualized within their sentence. To achieve a similar contextualization effect for our image patches, we use the BEiT encoder [3]. This encoder is trained using Masked-Image-Modeling, analogously to the way BERT is trained, and achieves contextualized patch embeddings.

Aligned embeddings A multi-modal encoder is used to embed all modalities together, into a shared latent space, where the resulting embeddings are aligned across modalities. Using aligned embeddings, the model already has access to connections between the two modalities, and needs to learn a simpler mapping from initial features to predictions. We experiment with two multi-modal encoders: the first is the celebrated CLIP encoder [18], which is trained on contrastive pairs, and provides high quality aligned embeddings that were trained to capture global semantic features. The second is the BLIP encoder [12], which is fine-tuned on VQA, and provides aligned embeddings that are tuned specifically for question answering, while capturing fine-grained as well as global semantic features.

3.2 Graph Construction

To represent and introduce connections between the different modalities, we leverage the graph as an intermediate representation of both modalities as a single entity. The constructed graph carries the embeddings of the aforementioned *atomic units* as it's node features. We experiment with two methods of graph construction, each emphasizing different aspects of our desired properties for the graph structure.

Cayley Graphs. The authors of [27] suggest incorporating Cayley graphs for message passing, to prevent over-squashing and bottlenecks induced by general graph structures – leveraging the mathematical properties of Cayley graphs as expander graphs. We adopt a similar approach, and experiment with a Cayley graph structure as a naïve baseline for our approach. Given token embeddings $T \in \mathbb{R}^{N_{\text{tokens}} \times d_{\text{embed}}}$ and patch embeddings $P \in \mathbb{R}^{N_{\text{patches}} \times d_{\text{embed}}}$, we stack them across the first dimension, resulting in the stacked embedding matrix $\texttt{Embeds} \in \mathbb{R}^{(N_{\text{tokens}}+N_{\text{patches}}) \times d_{\text{embed}}}$. We then

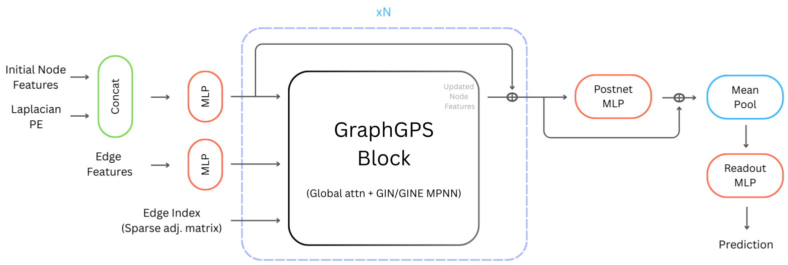


Figure 2: An Illustration of our GNN architecture. PE stands for Positional Encoding. Laplacian PE and edge features are optional and may not be passed. The $\oplus$ symbol represents a residual connection. The mean pool receives the final node embeddings $X \in \mathbb{R}^{|V| \times d_{\text{hidden}}}$ and computes their mean over the first dimension, resulting in a single d_{hidden}-dimensional vector, which is then used for graph-level prediction. Each GraphGPS block uses either GINE – when edge features are present, or GIN otherwise.

proceed to construct a Cayley graph with the minimal number of nodes greater than (or equal to) $N_{\text{tokens}} + N_{\text{patches}}$. We extend our Embeds matrix with zero-vector embeddings to match the number of nodes in the constructed Cayley graph (i.e. the additional *virtual nodes'* features are initialized to zero-vectors). We adopt the publicly available code[2] by the authors of [27] for Cayley graph construction.

MMGs. We design a graph structure, specifically tailored to represent multi-modal data and introduce connections intra and inter-modally. Our Multi-Modal Graph (MMG) design is motivated by three key insights: 1) visual information benefits from spatial locality and hierarchical processing, 2) cross-modal connections should be asymmetric and meaningful rather than uniform, and 3) fusion should occur at multiple levels of abstraction to enable both fine-grained and global reasoning.
To implement these principles, we employ a hierarchy of virtual nodes, on top of the base representations of each modality, when represented with the respective atomic units: the text token embeddings are represented as a line-graph, and the image patch embeddings are represented as a grid-graph. The image patch nodes connect to virtual *quadrant nodes*, each aggregating spatial information of the nodes in the respective quadrant. These virtual quadrant nodes are then connected in a full graph (K_4), so that information can propagate between different regions of the image. The text token nodes all connect to t virtual *text-global* nodes, each aggregating semantic information about the entire text sequence. Finally, for multi-modal fusion we add f virtual *fusion nodes*, each representing a different global feature of the text-image multi-modal input. All virtual nodes are initialized with zero-vector embeddings. We also augment the initial node features of our graph by concatenating a one-hot representation of the node types we presented. We also add one-hot edge features for each edge type (e.g. intra-image, image-to-quadrant, etc.). An illustration of our MMG structure can be found in appendix A.

3.3 GNN Architecture

The nodes (alongside their features) of our graph, being tokens and image patches, are connected in ways that require attention to both global and local phenomena, both intra and inter-modally. To achieve these properties, as-well as considering the graph as a strong source of relevant inductive biases, we choose to implement a model using the GraphGPS architecture [20], which utilizes global attention alongside MPNNs. Our base architecture is detailed in figure 2.

Graph Pooling. While the base GraphGPS template offers impressive expressive power [15, 16], using it with a mean-pool for graph classification essentially forces all final node representations to participate in the graph-level readout. Following the *'fusion nodes'* motivation from our MMGs, we experiment with Self-Attention Graph Pooling (SAGPool) [10]. The SAGPool layer uses a MPNN as a node-level classifier, computing a score for each node, and filtering out a fraction of the total nodes with the lowest scores. For the MPNN used by the SAGPool layers we use the celebrated GAT architecture [25]. Inspired by [10], we experiment with two ways to incorporate SAGPool layers into our base architecture: 1) Global SAGPool – A single SAGPool layer added right after the postnet MLP and before the mean pool. This method acts as a *'smarter pooling'*, allowing the model to choose which nodes will be used in classification for the graph-level task, and allowing message passing and global attention to *'concentrate'* on specific nodes for classification.
2) Hierarchical SAGPool – Analogous to the way max-pooling is used in CNNs. Multiple SAGPool layers, each added after the GraphGPS block on

[2] https://github.com/josephjwilson/cayley_graph_propagation

Experiment No.	Graph Structure		SAGPool Method			VQA Subset	
	MMG	Cayley	N	G	H	test-dev	validation
1	✓	-	✓	-	-	$44.43_{\pm 0.02}$	$44.015_{\pm 0.025}$
2	✓	-	-	✓	-	$\mathbf{45.305}_{\pm 0.015}$	$\mathbf{44.825}_{\pm 0.035}$
3	✓	-	-	-	✓	$43.85_{\pm 0.01}$	$43.255_{\pm 0.015}$
4	-	✓	✓	-	-	$\mathbf{45.29}_{\pm 0.01}$	$\mathbf{44.605}_{\pm 0.005}$
5	-	✓	-	✓	-	$44.205_{\pm 0.035}$	$43.475_{\pm 0.005}$
6	-	✓	-	-	✓	$44.365_{\pm 0.005}$	$43.655_{\pm 0.035}$

Table 1: The results for BERT-BEiT embeddings. For the validation split we manually calculate the VQA evaluation metric and average over the validation split. In the SAGPool method column, N stands for None (the base model as in figure 2), G is Global SAGPool, and H is Hierarchical SAGPool (As described in section 3.3). The reported result is the mean of the best (w.r.t the VQA accuracy metric) and last checkpoints of the relevant experiment, where ± denotes the standard deviation. The best result for each graph structure is written with a **bold font**, the best result per split (across both graph structures) is highlighted with ▨.

a chosen subset of layers. This method reduces the graph iteratively, with MPNN and global attention steps in-between, allowing the model to essentially *'throw out'* un-important nodes early on in the forward-pass.

4 Experimental Details

Our code is available on our GitHub repository[3]. Convergence plots are available in our WandB report[4] as-well as in appendix A.1. Trained checkpoints for our experiments are available on HuggingFace[5].

Setup. As mentioned in section 3, we only experiment with the VQA task, specifically on the VQA v2 dataset [8]. More specifically, we utilize this[6] VQA v2 distribution on HuggingFace. The VQA task is an open-ended prediction task; but for simplicity, and inspired by [2, 14] – we precompute target classes by taking the $3,000$ most frequent answers from the training set, then treat the task as multi-class classification over these target classes. To obtain labels (for training/validation), we compute a soft-target vector from the annotators' answers by following the VQA metric[7] [1]:

$$\text{Acc}(a) = \min\left(1, \frac{\#\text{Annotators that answered } a}{3}\right)$$

We then train with a binary cross-entropy loss on the soft-target scores. At inference, we take the $\arg\max$ of the model's output.

For evaluation, we compute our inference results on the test set and submit them to the official VQA challenge[8]. The challenge consists of two test splits, being test-dev and test-std; the test-dev split allows unlimited submission and is used for development, while the test-std split limits submission and serves as the official leaderboard. We report test-dev scores for all experiments, and test-std scores only for the second stage of experiments.

Experimental design. To evaluate our approach, we conduct our experiments following a two-stage experimental methodology. In the first stage, we compare all proposed graph structures and GNN pooling methods, while using the unaligned embeddings (as in section 3.1 – BERT for text and BEiT for images), to isolate the architectural contributions of our graph designs and pooling methods from embedding quality effects. This stage serves as a fair comparison baseline, where performance gains can be attributed to our design choices. In the second stage of experiments, we evaluate only the best performing architectures from our baseline first stage using the multi-modally aligned embeddings (as in 3.1 – CLIP and BLIP), aiming to demonstrate the practical performance ceiling of our approach. This two-stage design allows us to test and assess the contributions of each component in our approach to the overall performance, while allowing us to avoid the large computational costs of experimenting with all embedding/graph structure/GNN pooling combinations; all while preserving clear attribution of performance gains to the particular component that influenced them.

Training details. Our hyper-parameters and training setup (optimizer, scheduler, etc.) are detailed in appendix B. An important note is the scale of our models, being 19M-21M parameters.

5 Results & discussion

The results for the first stage of experiments are detailed in table 1. The first thing we can notice is that the general performance of our approach on VQA v2 is quite low – with modern models [22, 26] scoring as high as $70 - 85$ on the VQA metric, and us scoring around $40 - 45$. We explain this due to

[3] https://github.com/D4niel0s/GAMER

[4] https://wandb.ai/the_GAMERs/GAMER/reports/Convergence-plots-for-GAMER-experiments–VmlldzoxNDM3NzQ3Ng

[5] https://huggingface.co/D4niel0s/GAMER_experiments

[6] https://huggingface.co/datasets/pingzhili/vqa_v2

[7] https://visualqa.org/evaluation.html

[8] https://eval.ai/web/challenges/challenge-page/830

Experiment No.	Method Description	Embedding type			VQA Subset	
		CLIP	BLIP	BERT-BEiT	test-dev	test-std
1	Our MMG, Global SAGPool	✓	-	-	38.65±0.36	39.2
2		-	✓	-	44.375±0.005	44.62
3 (Baseline)		-	-	✓	**45.305**±0.015	**45.7**
3	Cayley Graph, No SAGPool	✓	-	-	41.51±0.01	41.78
4		-	✓	-	43.355±0.005	43.45
5 (Baseline)		-	-	✓	45.29±0.01	45.63

Table 2: The results for multi-modal aligned embeddings. The reported results are denoted as in table 1. Experiments No. 3 and 5 are the experiments from table 1, mentioned as a baseline for this stage. For test-std we only report the result of the best performing checkpoint on test-dev, as submissions to test-std are limited. Best results (per split) are highlighted.

our models being much smaller (19M-21M params compared to > 100M baselines), and our initial embeddings being unaligned – essentially forcing our models to perform multimodal fusion on their own. Despite the small scale, we do still see some meaningful multi-modal alignment that our models did learn, since the naïve baselines (guessing most common answers) score around 30%. This is quite impressive given the complex multi-hop reasoning that is required for many questions in VQA v2.

Comparing the two graph structures we proposed, the first thing we observe is that our best experiment used our MMGs, proving that non-trivial graph structures do introduce meaningful inductive biases that help some models perform on VQA. We can also observe different phenomena regarding the interactions between graph structure and graph pooling: With the Cayley graphs, we see that any type of pooling hurts our models' generalization, while with our MMGs the global pooling improves the performance drastically, and hierarchical pooling hurts it. Regarding the hierarchical pooling, this performance drop across the two graph structures can be attributed to the pooling *'throwing out'* too much of the graph, so that it does not contain enough information to answer the questions.

Regarding the global pooling, we can attribute the aforementioned effect to our MMGs forcing information flow from the atomic units (tokens/patches) up to the global fusion nodes, giving the model an ability to decide to not propagate from certain atomic units to the fusion nodes, and later filtering them out, which results in less nodes and as a consequence less noise after the mean-pool. Regarding the effect of global pooling with the Cayley graphs, we can attribute it to the model's inability of distinguishing between node types and having to use the token/patch nodes for classification, which should restrict the model's ability to perform a good classification of relevant nodes (which is exactly what the SAGPool layer is doing).

Our final best results per graph structure are similar, but they both come from different models, each exploiting a specific strength of the specific graph structure – showing a real variation of modeling the multimodal task with the naïve graphs, compared to a designed multi-modal graph.

The results for the second stage of our experiments are shown in table 2. The immediate thing we see is that across our best models, both the generic CLIP and the VQA-pretrained BLIP perform worse than the unaligned BERT-BEiT embeddings. For us this is a shocking result that should be completely inverse from a first glance. We hypothesize that this phenomenon is tied to one of two events: The first being the inability of CLIP and BLIP of capturing fine-grained detail, and mainly capturing global attributes in the [CLS] token which we do not use. The second possibility is that our graph-based approach introduces an inductive bias that makes multi-modal alignment a much easier task, allowing our GNNs to perform multimodal fusion at this small scale, while not being able to compete with modern baselines on the VQA task. Assessing the correctness of out hypothesis is left for future work.

6 Future work

While our results show interesting patterns in our experiments, they do not necessarily point out the causes for these phenomena. An important future effort could be made to explain the roles of the graph structures and graph pooling methods, when paired with certain embeddings; the way the SAGPool layer operates, alongside our designed MMGs, creates a highly interpretable pipeline that could be deeply examined.

Another important future effort is testing our hypothesis from section 5, by incorporating our model (without the prediction head) as a multi-modal alignment mechanism, alongside a bigger, more scalable prediction head (e.g. a large transformer). Our approach could be used as a parameter-efficient way of solving the multi-modal alignment bottleneck, which could be transformative to the way multi-modal tasks are modeled today.

Limitations

Our main limitation was the computational bottleneck we faced. Throughout our experimentation, we only had access to a single `NVIDIA TITAN XP 12GB`, which substantially limited the scale of our experiments.

Ethics Statement

All practices were done in line with Tel Aviv University's ethics guidelines and all data used is free for public use.

References

[1] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2016. Vqa: Visual question answering. *Preprint*, arXiv:1505.00468.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. *Preprint*, arXiv:1707.07998.

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. Beit: Bert pre-training of image transformers. *Preprint*, arXiv:2106.08254.

[4] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Preprint*, arXiv:2006.13009.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *Preprint*, arXiv:2010.11929.

[7] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2020. Learning discrete structures for graph neural networks. *Preprint*, arXiv:1903.11960.

[8] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *Preprint*, arXiv:1612.00837.

[9] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *Preprint*, arXiv:2106.07447.

[10] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. *Preprint*, arXiv:1904.08082.

[11] Jiang Li, Xiaoping Wang, Guoqing Lv, and Zhigang Zeng. 2023. Graphmft: A graph network based multimodal fusion technique for emotion recognition in conversation. *Neurocomputing*, 550:126427.

[12] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pretraining for unified vision-language understanding and generation. *Preprint*, arXiv:2201.12086.

[13] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *Preprint*, arXiv:1908.03557.

[14] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Preprint*, arXiv:1908.02265.

[15] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2025. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. *Preprint*, arXiv:2502.09263.

[16] Yuankai Luo, Veronika Thost, and Lei Shi. 2023. Transformers over directed acyclic graphs. *Preprint*, arXiv:2210.13148.

[17] Feng Mo, Lin Xiao, Qiya Song, Xieping Gao, and Eryao Liang. 2024. Multimodal graph neural network for recommendation with dynamic de-redundancy and modality-guided feature de-noisy. *Preprint*, arXiv:2411.01561.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

[19] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *Preprint*, arXiv:2212.04356.

[20] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2023. Recipe for a general, powerful, scalable graph transformer. *Preprint*, arXiv:2205.12454.

[21] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[22] Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *Preprint*, arXiv:1908.07490.

[23] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. *Preprint*, arXiv:2204.03162.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. *Preprint*, arXiv:1710.10903.

[26] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. 2022. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *Preprint*, arXiv:2208.10442.

[27] JJ Wilson, Maya Bechler-Speicher, and Petar Veličković. 2025. Cayley graph propagation. *Preprint*, arXiv:2410.03424.

[28] Peng Xu, Xiatian Zhu, and David A. Clifton. 2023. Multimodal learning with transformers: A survey. *Preprint*, arXiv:2206.06488.

[29] Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2025. Thinking in space: How multimodal large language models see, remember, and recall spaces. *Preprint*, arXiv:2412.14171.

[30] Yuhui Zhang, Alyssa Unell, Xiaohan Wang, Dhruba Ghosh, Yuchang Su, Ludwig Schmidt, and Serena Yeung-Levy. 2024. Why are visually-grounded language models bad at image classification? *Preprint*, arXiv:2405.18415.

A Illustrations and plots

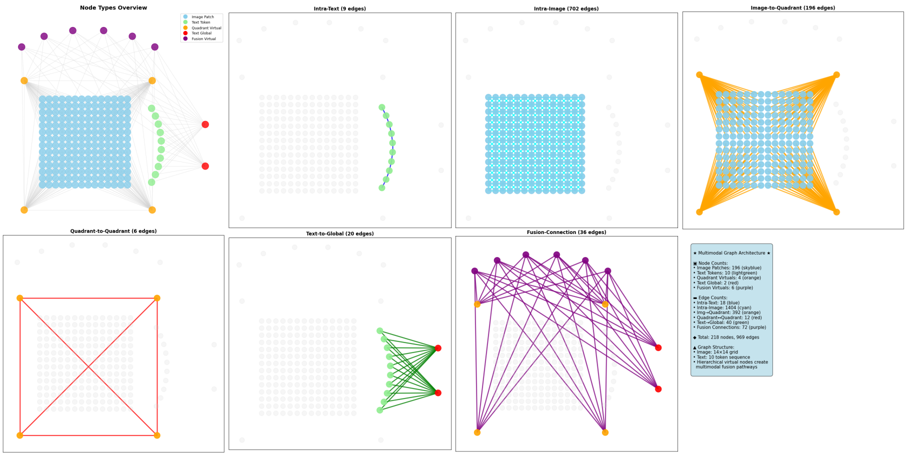
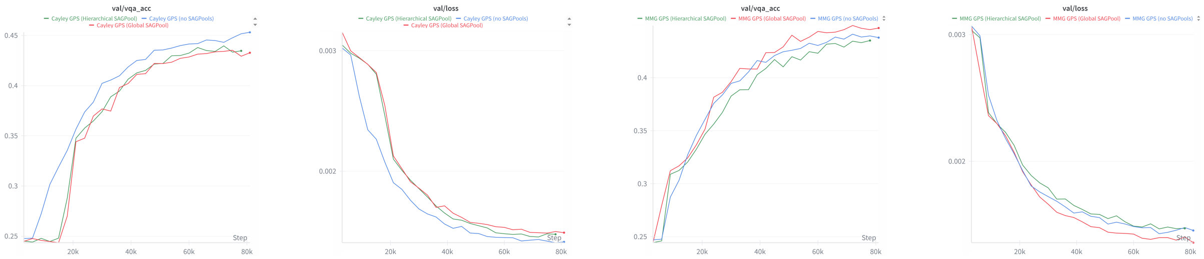

Node Types Overview
Image Patch
Text Token
Quadrant Virtual
Text Global
Fusion Virtual
Intra-Text (9 edges)
Intra-Image (702 edges)
Image-to-Quadrant (196 edges)
Quadrant-to-Quadrant (6 edges)
Text-to-Global (20 edges)
Fusion-Connection (36 edges)
★ Multimodal Graph Architecture ★
▣ Node Counts:
• Image Patches: 196 (skyblue)
• Text Tokens: 10 (lightgreen)
• Quadrant Virtuals: 4 (orange)
• Text Global: 2 (red)
• Fusion Virtuals: 6 (purple)
▬ Edge Counts:
• Intra-Text: 18 (blue)
• Intra-Image: 1404 (cyan)
• Img→Quadrant: 392 (orange)
• Quadrant↔Quadrant: 12 (red)
• Text→Global: 40 (green)
• Fusion Connections: 72 (purple)
◆ Total: 218 nodes, 969 edges
▲ Graph Structure:
• Image: 14×14 grid
• Text: 10 token sequence
• Hierarchical virtual nodes create multimodal fusion pathways

Figure 3: An illustration of our proposed MMG structure. The top-left sub-plot showcases the different node types present in our MMGs. All other sub-plots showcase specific nodes and the connections between them: 1) Text token nodes and intra-text edges. 2) Image patch nodes and intra-image edges. 3) Image patch to quadrant conections. 4) Image Quadrant to Quadrant connections. 5) Text token to text global connections. 6) Image quadrants and text global nodes to fusion nodes connections. The diameter of our MMGs is 4. Nodes and edges are colored according to their types.

A.1 Convergence plots

We report our convergence plots for both stages of experimentation.



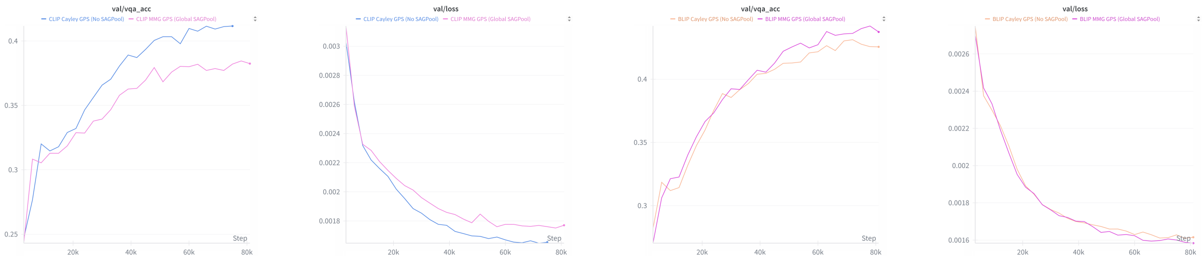
Accuracy plot for Cayley graphs.

Loss plot for Cayley graphs.

Accuracy plot for our MMGs.

Loss plot for our MMGs.

Figure 4: Validation plots for stage 1 of our experimentation. All use BERT-BEiT embeddings.



Accuracy plot for CLIP embeddings.

Loss plot for CLIP embeddings.

Accuracy plot for BLIP embeddings.

Loss plot for BLIP embeddings.

Figure 5: Validation plots for stage 2 of our experimentation.

B Training parameters and details

For all of our experiments, we use 4 GraphGPS layers with 8 attention heads and a hidden dimension of 512. We use deep 4 hidden-layer MLPs (5 total linear operations). For the Global SAGPool we use a pooling ratio of 50%, filtering out half of the graph's nodes for classification. For the hierarchical SAGPool, we use 70% pools after the second and third layers, and an 80% pool after the last layer (An $x\%$ pool leaves $x\%$ of nodes and throws out $(100 - x)\%$ of nodes). All of our models consist of 19M to 21M parameters. When experimenting with our MMGs, we use 6 multi-modal fusion virtual nodes, and 2 text-global virtual nodes, with self-loops added to the graph. We use a 16-dimensional Laplacian PE.

We train with a batch size of 32, with gradient accumulation over 2 batches, resulting in an effective batch-size of 64. We train for 12 epochs, resulting in a total of $\approx$ 80K model updates. Following [20], we use the AdamW optimizer with the default $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ and a linear warm-up,cosine decay learning rate schedule, with a warm-up of 5% of the total training steps. We use a learning rate of $5 \cdot 10^{-5}$ and a weight decay of $5 \cdot 10^{-2}$, after finding that higher learning rates yield unstable or divergent training. Our training took ≈ 2 days per experiment.