Multimodal Graph Learning for Generative Tasks

Minji Yoon
Carnegie Mellon University

Jing Yu Koh
Carnegie Mellon University

Bryan Hooi
National University of Singapore

Ruslan Salakhutdinov
Carnegie Mellon University

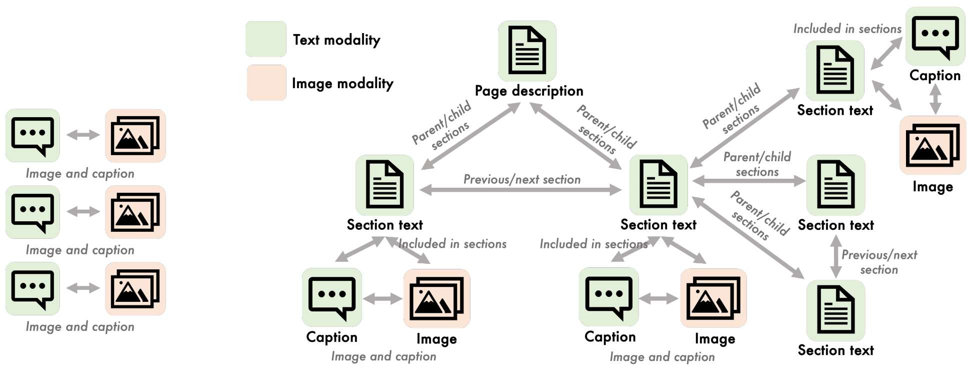
Abstract

Multimodal learning combines multiple data modalities, broadening the types and complexity of data our models can utilize: for example, from plain text to image-caption pairs. Most multimodal learning algorithms focus on modeling simple one-to-one pairs of data from two modalities, such as image-caption pairs, or audio-text pairs. However, in most real-world settings, entities of different modalities interact with each other in more complex and multifaceted ways, going beyond one-to-one mappings. We propose to represent these complex relationships as graphs, allowing us to capture data with any number of modalities, and with complex relationships between modalities that can flexibly vary from one sample to another. Toward this goal, we propose Multimodal Graph Learning (MMGL), a general and systematic framework for capturing information from multiple multimodal neighbors with relational structures among them. In particular, we focus on MMGL for *generative* tasks, building upon pretrained Language Models (LMs), aiming to augment their text generation with multimodal neighbor contexts. We study three research questions raised by MMGL: (1) how can we infuse multiple neighbor information into the pretrained LMs, while avoiding scalability issues? (2) how can we infuse the graph structure information among multimodal neighbors into the LMs? and (3) how can we finetune the pretrained LMs to learn from the neighbor context in a parameter-efficient manner? We conduct extensive experiments to answer these three questions on MMGL and analyze the empirical results to pave the way for future MMGL research.

1 Introduction

There are diverse data modalities in real-world applications, from commonly observed texts, images, and videos to time series data or domain-specific modalities like protein sequences. These various modalities are not collected individually but together with multifaceted relations among them. Wikipedia [2] is one of the most popular sources of multimodal web content, providing multimodal data such as texts, images, and captions. TimeBuilder [29], recently released by Meta, builds personal timelines using each user's multimodal data, including their photos, maps, shopping, and music history. In addition to these examples, important industrial and medical decisions are also made by considering diverse multimodal data such as images, tables, or audio [13, 26]. These multimodal data have complicated $many$-to-$many$ relations among their multimodal entities — which can be represented as graphs — providing open research space on how to understand them holistically.

With the rise of multimodal datasets, various ground-breaking research has been done in multimodal learning. Previously, multimodal learning focused on novel architectures, extending transformers [9, 19, 30] or graph neural networks [12, 25], and training them from scratch using large-scaled multimodal datasets. Fueled by the strong generative power of pretrained Language Models (LMs), recent multimodal approaches [1, 17, 16] are built upon pretrained LMs and focus on the generation



(a) 1-to-1 Multimodal Learning (b) $many$-to-$many$ Multimodal Graph Learning

Figure 1: **Multimodal datasets extracted from Wikipedia**: (a) Most multimodal models target multimodal datasets with clear 1-to-1 mappings between modalities. (b) Multimodal Graph Learning (MMGL) handles multimodal datasets with complicated relations among multiple multimodal neighbors.

of multimodal content. For instance, [16] generates images/text grounded on given text/images using pretrained image encoders and LMs. However, all existing models assume that a pair of modalities with a clear 1-to-1 mapping is provided as input (e.g., image-caption pairs in Figure 1(a)). As a result, they cannot be directly applied on multimodal datasets with more general $many$-to-$many$ mappings among modalities (e.g., multimodal Wikipedia webpage in Figure 1(b)).

Here, we expand the scope of multimodal learning beyond 1-to-1 mappings into multimodal graph learning (MMGL) while preserving generative abilities by integrating them into pretrained LMs. We introduce a systematic framework on how MMGL processes multimodal neighbor information with graph structures among them and generate free-form texts using pretrained LMs (Figure 2). Our MMGL framework extracts *neighbor encodings*, combines them with *graph structure information*, and optimizes the model using *parameter-efficient fine-tuning*. Accordingly, we define three design spaces to study three research questions for MMGL as follows:

- **Research Question 1.** How can we provide multiple multimodal neighbor information to LMs while avoiding scalability issues?
- **Research Question 2.** How can we infuse graph structure information among multimodal neighbors into LMs?
- **Research Question 3.** How can we finetune pretrained LMs to learn through multimodal neighbor information in parameter-efficient ways?

In conventional multimodal learning with the 1-to-1 mapping assumption, typically only one neighbor is provided (e.g., an image for a text caption) [1, 16, 17]. On the contrary, MMGL requires the processing of several neighbors with various data sizes (e.g., image resolution and text sequences of various lengths), which leads to the scalability issue. For *Research Question 1*, we study three neighbor encoding models: (1) *Self-Attention with Text + Embeddings* (SA-Text+Embeddings) precomputes image embeddings using frozen encoders, then concatenates them to the input text sequences with any raw text from neighbors (originally proposed from [31]), (2) *Self-Attention with Embeddings* (SA-Embeddings) precomputes embeddings for both text and image modalities using frozen encoders and concatenates to the input text, and (3) *Cross-Attention with Embeddings* (CA-Embeddings) feeds precomputed text or image embeddings into cross-attention layers of LMs.

In *Research Question 2*, we study how to infuse graph structure information among multimodal neighbors into LMs (e.g., section hierarchy and image orders in Figure 1(b)). We compare the sequential position encoding with two graph position encodings widely used in graph transformers [24, 34]: *Laplacian eigenvector position encoding* (LPE) [6] and *graph neural networks encoding* (GNN) [15] that runs GNNs on precomputed neighbor embeddings using graphs structures before feeding them into LMs.

Research Question 3 seeks to improve the cost and memory efficiency compared to full fine-tuning of LMs. In this work, we explore three parameter-efficient fine-tuning (PEFT) methods [10]: *Prefix tuning* [18], *LoRA* [11], and *Flamingo tuning* [1]. Which PEFT methods to use depends on the neighbor encoding model: when neighbor information is concatenated into the input sequences (*SA-Text+Embeddings* or *SA-Embeddings* neighbor encodings), we can apply *Prefix tuning* or *LoRA* for fine-tuning. When neighbor information is fed into cross-attention layers (*CA-Embeddings* neighbor encoding), we apply *Flamingo tuning* that finetunes only cross-attention layers with gating modules for stable finetuning [1].

Based on our MMGL framework, we run extensive experiments on the recently released multimodal dataset, WikiWeb2M [2]. WikiWeb2M unifies each Wikipedia webpage content to include all text, images, and their structures in a single example. This makes it useful for studying multimodal content understanding with many-to-many text and image relationships, in the context of generative tasks. Here, we focus on the section summarization task that aims to generate a sentence that captures information about the contents of one section by understanding the multimodal content on each Wikipedia page. Through rigorous testing on WikiWeb2M, we provide intuitive empirical answers to research questions raised in MMGL.

In summary, our contributions are:

- **Multimodal Graph Learning (MMGL):** We introduce a systematic MMGL framework for processing multimodal neighbor information with graph structures among them, and generating free-form texts using pretrained LMs.
- **Principled Research Questions:** We introduce three research problems MMGL is required to answer: (1) how to provide multiple neighbor information to the pretrained LMs, (2) how to infuse graph structure information into LMs, and (3) how to fine-tune the LMs parameter-efficiently. This paves research directions for future MMGL research.
- **Extensive Empirical Results:** We show empirically that (1) neighbor context improves generation performance, (2) *SA-Text+Embeddings* neighbor encoding shows the highest performance while sacrificing the scalability, (3) *GNN* embeddings are the most effective graph position encodings, and (4) *SA-Text+Embeddings* neighbor encoding with *LoRA* and *CA-Embeddings* neighbor encoding with *Flamingo tuning* show the highest performance among different PEFT models.

Our code is publicly available at [1].

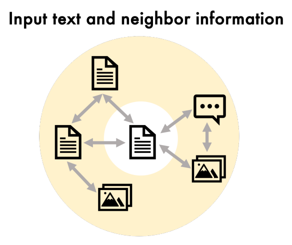
2 Related Work

End-to-End Multimodal Learning: While many discriminative multimodal models [14, 22] have also been developed, we primarily consider related work on generative multimodal models, as this is most closely related with our approach. Several recent approaches tackle multimodal learning by building upon the Transformer [32] architecture. Multimodal extensions typically use either full self-attention over modalities concatenated across the sequence dimension [3, 28] or a cross-modal attention layer [30]. Self-supervised multimodal pretraining methods train these architectures from large-scale unlabeled multimodal data before transferring them to downstream multimodal tasks via fine-tuning [9, 19]. These methods perform end-to-end pre-training, incurring extremely high computation costs, especially as model parameters increase [17]. Moreover, this framework is relatively inflexible for end-to-end pre-trained models to leverage readily available unimodal pre-trained models, such as text-only LMs or pretrained vision models.

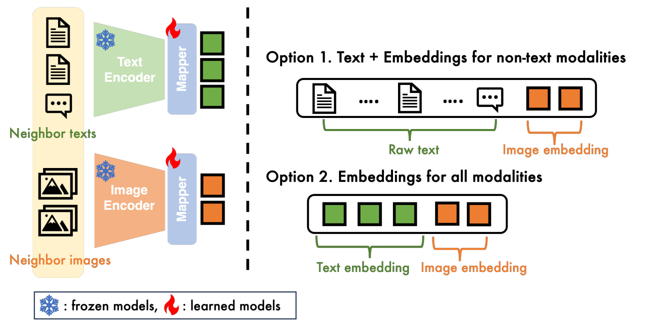
Multimodal Learning with Frozen Image Encoders and Large Language Models: Recently, various vision-language models have been proposed to leverage off-the-shelf pre-trained models and keep them frozen during pretrainig [1, 17, 16]. To input visual information directly to a frozen text-only LLM, a key challenge is to align visual features to the text space. Motivated by Frozen [31], which finetunes a visual encoder to map images into the hidden space of a text-only LLM, Blip-2 [17] and GILL [16] finetune separate image mapping networks whose inputs are precomputed by frozen image encoders and outputs are directly used as soft prompts to LLMs. On the other hand, Flamingo [1] inserts new cross-attention layers into the LLM to inject visual features and pre-trains the new layers on image-text pairs. Note that all these methods primarily focus on processing *interleaved image and text inputs* to generate text outputs.

Graph Neural Networks on Multimodal Graphs Heterogeneous Graph Neural Networks (HGNNs) extend Graph Neural Networks (GNNs) [36] to learn from multimodal heterogeneous

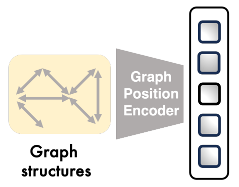
[1] `https://github.com/minjiyoon/MMGL`



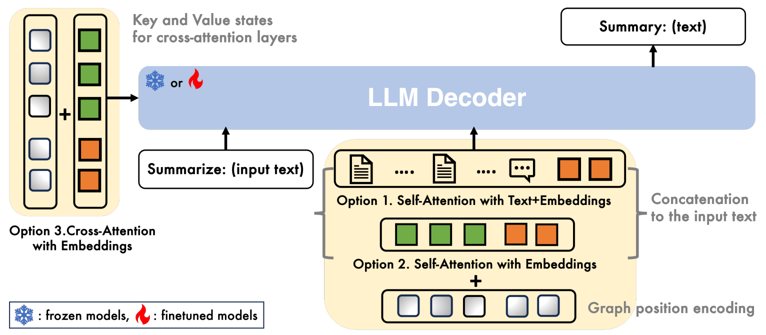
(a) Multimodal neighbors with graph structures



(b) Multimodal neighbor encoding using frozen vision/text encoders



(c) Graph position encodings


Key and Value states for cross-attention layers
+
Option 3.Cross-Attention with Embeddings
or
LLM Decoder
Summary: (text)
Summarize: (input text)
....
....
Option 1. Self-Attention with Text+Embeddings
Option 2. Self-Attention with Embeddings
+
Concatenation to the input text
Graph position encoding
: frozen models,
: finetuned models

(d) Text generation using frozen/finetuned LMs and encoded neighbor information

Figure 2: **Multimodal Graph Learning (MMGL) framework**: (a) Multiple multimodal neighbors are given with the input text. (b) Multimodal neighbors are first encoded using frozen vision/text encoders and then aligned to the text-only LM space using 1-layer MLP mappers. The mappers are trained during LM fine-tuning. Based on the neighbor encoding scheme, texts could be used without any preprocessing (*Self-Attention with Text+Embeddings*) or encoded into embeddings (*Self-Attention with Embeddings* or *Cross-Attention with Embeddings*). Images are always encoded into embeddings to align to the text-only LM space. (c) Graph structures among neighbors are encoded as graph position encodings. (d) Encoded neighbor information could be infused either by concatenating to the input sequences (*Self-Attention with Text+Embeddings* or *Self-Attention with Embeddings*) or feeding into cross-attention layers (*Cross-Attention with Embeddings*). The graph position encodings are added to the input token/text/image embeddings.

graphs. This is done through precomputing input node embeddings using frozen encoders, and training the GNN to map different modality embeddings either at the input layer [25], intermediate [12], or late layers [35]. However, most HGNN models focus on node classification, and are difficult to adapt for generative tasks. Recently, various approaches have been proposed to fine-tune LLMs with GNNs on text-attributed graphs [4, 8, 38]. These methods specialize in node/edge classification tasks by putting GNN models after LLMs, making them difficult to adapt for use in generative tasks.

3 Multimodal Graph Learning for Generative Tasks

Given multimodal graphs with text or images on each node, we aim to generate text conditioned on each node and its neighbor nodes. More specifically, given text input on a target node, pretrained LMs generate free-form text conditioned on the input text and the multimodal context around the target node. In our multimodal graph learning (MMGL) framework, we first encode each neighbor's information individually using frozen encoders (Figure 2(b)). The frozen encoders could be pretrained ViT [5] or ResNeT [7] for images that map pixels to embeddings, and pretrained LMs [22] for texts that map texts to embeddings (similarly for other modalities). Then, we encode the graph structure around the target node using graph position encodings (Figure 2(c)). Finally, the encoded neighbor information with graph position encodings is fed into the pretrained LMs with the input text to generate text conditioned on the multimodal input content (Figure 2(d)).

The framework leaves us with three design spaces: (1) how can we feed neighbor information to the LMs? (2) how can we infuse graph structure information among multimodal neighbors into LMs? (3) how can we finetune the pretrained LMs to learn from the neighbor context parameter-efficiently? In this section, we investigate each problem and discuss possible methodologies we can apply.

3.1 Research Question 1: Neighbor Encoding

Unlike existing multimodal learning, which assumes a single image (corresponding to the input text) as input, multimodal graph learning considers an arbitrary number of neighbor images/texts as input; thus, scalability is the first problem to solve to learn from multiple multimodal neighbors. In vision-text models, a standard recipe is to first process images with an image encoder (e.g., ViT, ResNet) into image embeddings, then map the embeddings into the text-only LM space, and finally feed them into the LMs. Two popular ways to feed image embeddings into LMs are with full self-attention over modalities concatenated across the sequence dimension [31] or with cross-modal attention layers [30].

Motivated by these two approaches, we propose three neighbor encoding methods as follows:

- **Self-Attention with Text + Embeddings (SA-Text+Embeddings)**: Text neighbors are concatenated as raw texts, while other modalities are first processed by frozen encoders (e.g., ViT for images), and then their embeddings are concatenated to the input sequence. We add a linear mapper that aligns precomputed embeddings into the text space of LLMs.
- **Self-Attention with Embeddings (SA-Embeddings)**: Same as *SA-Text+Embeddings* except text neighbors are also processed by separate frozen encoders, and their embeddings are concatenated to the input sequence. Text encoders could be the same or different from the base LLM model.
- **Cross-Attention with Embeddings (CA-Embeddings)**: All neighbors are processed by separate frozen encoders, mapped into the text space by linear mappers, and then fed into cross-attention layers.

In general, when we provide text embeddings instead of raw text, the amount of information the LLMs are able to exploit is bottlenecked by the precomputed embeddings. However, raw texts introduce scalability issues as the attention mechanism of LMs uses the $O(T^2)$ compute with the sequence length T. Thus, there is a trade-off between computation cost and scalability. For *SA-Text+Embeddings* and *SA-Embeddings*, we have additional parameters only for mappers that are located outside of the LMs, while *CA-Embeddings* inserts additional cross-attention layers into pretrained LMs and trains them from scratch. This means *CA-Embeddings* could result in an unstable initial state as the pretrained LLM layers are affected by randomly initialized cross-attention layers. In Section 4.4, we explore these three approaches and discuss their empirical results.

3.2 Research Question 2: Graph Structure Encoding

Given neighbor information, we can simply concatenate neighbor information either as raw texts or embeddings and treat them as a sequence. But the neighbors have structures among them. For instance, sections have hierarchical structures, and images are included in certain sections in WikiWeb2M (Figure 1(b)). To encode this graph structure among the neighbor information, we borrow two popular graph position encodings from graph transformers and compare them with sequential position encoding.

- **Laplacian Position Encoding (LPE)**: We exploit Laplacian eigenvectors of neighbors computed from their graph structure as their position encodings.
- **Graph Neural Networks (GNN)**: We first compute neighbor embeddings from frozen encoders and run GNN over the embeddings using the graph structure. Then, we use the output GNN embeddings, which encode graph structure information as position encodings.

LPE has an additional 1-layer MLP mapper to map the Laplacian eigenvectors to the text space of LMs. Parameters used for graph structure encoding (e.g., mappers for *LPE* or *GNN* parameters) are trained with LMs in an end-to-end manner during LM fine-tuning. In Section 4.5, we explore how well these different position encodings bring additional graph structure information among neighbors into LMs and improve performance.

3.3 Research Question 3: Parameter-Efficiency

While we need to fine-tune the pretrained LM model for the specific task and newly added neighbor information, full fine-tuning requires high computation costs and also brings inconvenience in sharing MMGL modules when users decide to use neighbor information. Recently, various parameter-efficient fine-tuning (PEFT) methods have been proposed to fine-tune only a small amount of parameters while preserving the full fine-tuning performance. We choose three different PEFT models proper for the three neighbor encoding approaches we described above.

- **Prefix tuning**: When we choose *SA-Text+Embeddings* or *SA-Embeddings* for neighbor encoding, we do not have any newly added parameters but self-attention layers; thus, we can easily apply Prefix tuning [18], which keeps language model parameters frozen and instead optimizes a sequence of continuous task-specific vectors prepended to the original activation vectors across all layers.
- **LoRA**: Like *Prefix tuning*, low-rank adaptation (LoRA) [11] is suitable for *SA-Text+Embeddings* or *SA-Embeddings* neighbor encodings. LoRA injects trainable rank decomposition matrices into each layer while freezing the original parameters.
- **Flamingo**: For *CA-Embeddings* neighbor encoding, we can directly apply *Flamingo* [1], which fine-tunes only newly added cross-attention layers with *tanh* gating to keep the pretrained LM intact at initialization for improved stability and performance.

In Section 4.6, we explore how well PEFT models preserve the full fine-tuning performance by tuning a small number of parameters.

4 Experiments

4.1 WikiWeb2M dataset

WikiWeb2M dataset [2] is built for the general study of multimodal content understanding with many-to-many text and image relationships. Built upon the WIT dataset [27] which contains only image-caption pairs, WikiWeb2M includes the page title, section titles, section text, images and their captions, and indices for each section, their parent section, their children sections, and many more.

In this work, we focus on the section summarization task to generate a single sentence that highlights a particular section's content. The summary is generated given all images and (non-summary) text present in the target and context sections. We sample 600k Wikipedia pages randomly from WikiWeb2M for the section summarization task. In total, the training/validation/test set sizes for the section summarization task are 680k/170k/170k, respectively.

4.2 Experimental Settings

From WikiWeb2M, we can get four types of information for section summarization: (1) section text, (2) section images, (3) text from page description and other sections, and (4) images from page description and other sections. We provide information incrementally to LMs to study the effectiveness of multimodal neighbor information: (1) *section text*, 2) *section all* (text + image), 3) *page text* (all text from a Wikipedia page the input section belongs to), and 4) *page all* (all text and images from the Wikipedia page).

We use Open Pre-trained Transformer (OPT-125m) [37] for the base LM to read the input section text and generate a summary. For text and image encoders for neighbor information, we use text/image encoders from CLIP [22]. Following [23], we finetune OPT for 10000 steps of 125 batch size with learning rate 10^{-4}. The text/image encoders are frozen across all experiments. We measure BLEU-4 [21], ROUGE-L [20], and CIDEr [33] scores on the validation set. All experiments are run on 4 Nvidia-RTX 3090 GPUs with 24GB memory.

4.3 Effectiveness of Neighbor Information

We first examine the effectiveness of multimodal neighbor information. As described in Section 4.2, we provide more information incrementally to the base LM: (1) *section text*, (2) *section all* (text + image), 3) *page text*, and 4) *page all* (all texts and images). Here, we use *Self-Attention with Text+Embeddings (SA-Text+Embeddings)* neighbor encoding across different input types. For images, we first compute the image embeddings from the frozen CLIP image encoder and concatenate them

Table 1: **Effectiveness of neighbor information**: As more neighbor information is fed to LMs together with input texts (*section text, section all* => *page text, page all*), generation performance is improved. We increase the input sequence length to 1024 to encode *page text* and *page all* as more information is required to be encoded. The best results are colored in red, while the second-best results are colored in blue.

Input type	Input length	BLEU-4	ROUGE-L	CIDEr
Section text	512	8.31	40.85	79.68
Section all	512	8.03	40.41	77.45
Page text	1024	9.81	42.94	92.71
Page all	1024	9.96	43.32	96.62

Table 2: **Neighbor encodings in MMGL**: We encode multiple multimodal neighbor information using three different neighbor encodings, *Self-Attention with Text+Embeddings* (SA-TE), *Self-Attention with Embeddings* (SA-E), and *Cross-Attenion with Embeddings* (CA-E). While SA-TE shows the best performance, SA-TE requires a longer input length (1024) to encode texts from neighbors in addition to the original text input, leading to scalability issues. The best results are colored in red.

	BLEU-4			ROUGE-L			CIDEr		
Input type	**SA-TE**	**SA-E**	**CA-E**	**SA-TE**	**SA-E**	**CA-E**	**SA-TE**	**SA-E**	**CA-E**
Section all	8.03	7.56	8.35	40.41	39.89	39.98	77.45	74.33	75.12
Page text	9.81	8.37	8.47	42.94	40.92	41.00	92.71	80.14	80.72
Page all	9.96	8.58	8.51	43.32	41.01	41.55	96.01	82.28	80.31
Max input length	1024	512	512	1024	512	512	1024	512	512

right after the text of a section each image belongs to preserve the structure. The results in Table 1 indicate that *more multimodal neighbor information is helpful:* performance significantly improves when going from *section* to *page* content, and further when adding *page all* content, based on their BLEU-4, ROUGE-L, and CIDEr scores.

Discussion: Missing Modalities. Performance of *section all* decreased slightly from *section text*, despite the addition of section images. In Wikipedia, not every section has corresponding images. Thus, in the *section all* case, input to the LMs is inconsistent with some samples having text and images, while other samples only have text. This points to an important unaddressed *missing modality issue* that is common in the real world, which is not typically encountered in the conventional 1-to-1 multimodal setting, emphasizing the importance of developing MMGL approaches that are robust to the presence of missing modalities.

4.4 Neighbor Encoding

We encode multiple multimodal neighbor information using three different neighbor encodings, *Self-Attention with Text+Embeddings* (SA-TE), *Self-Attention with Embeddings* (SA-E), and *Cross-Attenion with Embeddings* (CA-E). While SA-E and CA-E encode all modalities, including text, into embeddings using frozen encoders, SA-TE encodes text neighbors as they are by concatenating them to the input text sequence. Thus SA-TE requires longer input sequence lengths (1024) to encode additional texts, leading to potential scalability issues. On the other hand, SA-E and CA-E require one token length to encode one text neighbor, improving scalability with shorter input lengths (512). The results in Table 2 indicate that *scalability is traded off with performance:* SA-TE consistently performs better than SA-E and CA-E on different input types at the cost of longer input lengths.

Discussion: Information Loss. In conventional multimodal learning with 1-to-1 mappings, SA-TE is commonly used to infuse text input as it is, and image inputs as embeddings are precomputed by frozen encoders [1, 16, 17]. These methods successfully generate texts grounded on the input images, showing image embeddings' effectiveness as input to the pretrained LMs. However, the performance gap between SA-TE and SA-E in Table 2 indicates that text embeddings likely lead to *information loss* in the LMs. This could be either because the 1-layer MLP mapper that aligns precomputed text embeddings into the text space of the LMs is not expressive enough, or because longer input texts

Table 3: **Graph structure encoding in MMGL**: We encode graph structures among multimodal neighbors using sequential position encodings (*Sequence*), Graph Neural Network embeddings (*GNN*), and Laplacian position encodings (*LPE*). Computed position encodings are added to input token/text/image embeddings and fed into LMs. We use *Self-Attention with Embeddings (SA-E)* neighbor encoding and *Prefix tuning* in this experiment. The best results are colored in red.

Metric	PEFT	Sequence	GNN	LPE
BLEU-4	**Prefix tuning**	6.91	6.98	6.80
	LoRA	7.12	7.30	7.13
ROUGE-L	**Prefix tuning**	38.98	39.13	39.10
	LoRA	39.05	39.48	39.35
CIDEr	**Prefix tuning**	68.20	69.29	68.15
	LoRA	68.86	70.86	69.34

Table 4: **Parameter-efficient finetuning in MMGL**: We apply *Prefix tuning* and *LoRA* for *Self-Attention with Text+Embeddings (SA-TE)* and *Self-Attention with Embeddings (SA-E)* neighbor encodings. For *Cross-Attention with Embeddings (CA-E)* neighbor encoding, we apply *Flamingo*-style finetuning that finetunes only newly added cross-attention layers with gating modules. Note that *SA-E* and *CA-E* neighbor encodings have more parameters than *SA-TE* because (frozen) text encoders are added to encode text neighbors. The best results are colored in red, while the second-best results are colored in blue.

Neighbor encoding (max length)		**SA-TE (1024)**		**SA-E (512)**		**CA-E (512)**
Metric	**Input type**	**Prefix tuning**	**LoRA**	**Prefix tuning**	**LoRA**	**Flamingo**
BLEU-4	**Section all**	6.70	6.65	6.80	7.07	6.96
	Page text	7.84	7.94	6.88	7.09	7.81
	Page all	8.21	8.18	6.91	7.12	8.12
ROUGE-L	**Section all**	38.67	38.84	38.97	39.30	39.43
	Page text	40.61	40.98	38.38	39.69	40.29
	Page all	41.08	41.25	38.98	39.05	40.95
CIDEr	**Section all**	65.84	65.00	67.24	68.61	69.31
	Page text	78.12	78.60	66.55	69.26	76.20
	Page all	81.07	80.75	68.20	68.86	82.37
# Finetuned parameters		20M	82M	20M	84M	90M
# Total parameters		230M	250M	300M	320M	363M
% Finetuned parameters		9%	33%	7%	26%	25%

compared to short texts used in the conventional multimodal learning (e.g., one-sentence captions) makes LMs hard to learn from precomputed text embeddings. From a practical angle, our results illuminate the trade-off between scalability and performance. Meanwhile, our results emphasize the need for more MMGL research to address the challenging issue of information loss when using embeddings to capture text information.

4.5 Graph Structure Encoding

In addition to each modality on neighbors, multimodal graphs contain graph structure information among neighbors. We encode the graph structures among multimodal neighbors using sequential position encodings (*Sequence*), Graph Neural Network embeddings (*GNN*), and Laplacian position encodings (*LPE*). Computed position encodings are first mapped to the text space of LMs by 1-layer MLP, added to input token/text/image embeddings, and fed into LMs. In Table 3, *GNN* embeddings show the best performance. Especially, the improvement over *Sequence* position encoding shows the *importance of graph-aware structure encoding methods* in MMGL.

4.6 Parameter-Efficient Fine-Tuning

Full fine-tuning of pretrained LMs requires high computational costs. For parameter-efficient fine-tuning for MMGL, we study *Prefix tuning* and *LoRA* for *Self-Attention with Text+Embeddings (SA-TE)* and *Self-Attention with Embeddings (SA-E)* neighbor encodings. For *Cross-Attention with Embeddings (CA-E)* neighbor encoding, we apply *Flamingo*-style finetuning that finetunes only newly added cross-attention layers with gating modules.

The results in Table 4 show that *LoRA performs better than Prefix tuning* for *SA-TE* and *SA-E* neighbor encodings with more fine-tuned parameters ($7-9\%$ for *Prefix tuning* and $26-33\%$ for *LoRA*). However, *Prefix tuning* still shows comparable performance with *LoRA* using nearly 4 times fewer parameters with *SA-TE* neighbor encoding. *Flamingo* with *CA-E* neighbor encoding shows comparable performance with *LoRA* with *SA-TE* neighbor encoding employing the similar numbers of fine-tuned parameters ($82M$ for *LoRA* and $90M$ for *Flamingo*). Note that *SA-E* and *CA-E* neighbor encodings have more parameters than *SA-TE*, attributed to the inclusion of (frozen) text encoders for text neighbor processing.

In Table 2 (without PEFT), it is evident that *CA-E* neighbor encoding lags in performance compared to *SA-TE* neighbor encoding. However, when infused with Flamingo, gating modules in Flamingo effectively ensure that the pre-trained LMs remain unaffected by randomly set cross-attention layers at initialization, thereby enhancing the performance of *CA-E*, as shown in Table 4 (with PEFT). This underscores the pivotal role of strategic initialization when introducing supplementary modules for neighbor encoding in MMGL and when integrating them into the pre-trained LMs.

5 Conclusion

In this work, we extend the conventional multimodal learning with 1-to-1 mappings between a pair of modalities into multimodal graph learning (MMGL) with $many$-to-$many$ relations among multiple modalities. Our MMGL framework is systematically structured around three critical components: (1) neighbor encodings, (2) graph structure encodings, and (3) parameter-efficient fine-tuning. Through rigorous testing on the WikiWeb2M dataset, we explored different options for each component: (1) three variations of neighbor encodings, *Self-Attention with Text+Embeddings*, *Self-Attention with Embeddings*, and *Cross-Attention with Embeddings*, highlighting the balance between scalability and performance, (2) three different graph position encodings, *sequence*, *LPE*, and *GNN*, and (3) three PEFT models, *prefix tuning*, *LoRA*, and *Flamingo*, and their trade-off between parameter-efficiency and performance. Our in-depth analyses and findings aim to lay the groundwork for future MMGL research, igniting further exploration in this field.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[2] Andrea Burns, Krishna Srinivasan, Joshua Ainslie, Geoff Brown, Bryan A. Plummer, Kate Saenko, Jianmo Ni, and Mandy Guo. A suite of generative tasks for multi-level multimodal webpage understanding, 2023.

[3] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer, 2020.

[4] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*, 2021.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2023.

[9] Lisa Anne Hendricks, John Mellor, Rosalia Schneider, Jean-Baptiste Alayrac, and Aida Nematzadeh. Decoupling the role of data, attention, and losses in multimodal transformers. *Transactions of the Association for Computational Linguistics*, 9:570–585, 2021.

[10] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[12] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[13] Bing Huang, Feng Yang, Mengxiao Yin, Xiaoying Mo, Cheng Zhong, et al. A review of multimodal medical image fusion techniques. *Computational and mathematical methods in medicine*, 2020, 2020.

[14] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. Generating images with multimodal language models. *arXiv preprint arXiv:2305.17216*, 2023.

[17] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[18] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[19] Paul Pu Liang, Yiwei Lyu, Xiang Fan, Jeffrey Tsaw, Yudong Liu, Shentong Mo, Dani Yogatama, Louis-Philippe Morency, and Russ Salakhutdinov. High-modality multimodal transformer: Quantifying modality & interaction heterogeneity for high-modality representation learning. *Transactions on Machine Learning Research*, 2022.

[20] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[24] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[25] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[26] Prabhjot Singh, Yanyan Wu, Robert Kaucic, Jiaqin Chen, and Francis Little. Multimodal industrial inspection and analysis. 2007.

[27] Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2443–2449, 2021.

[28] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[29] Wang-Chiew Tan, Jane Dwivedi-Yu, Yuliang Li, Lambert Mathias, Marzieh Saeidi, Jing Nathan Yan, and Alon Y Halevy. Timelineqa: A benchmark for question answering over timelines. *arXiv preprint arXiv:2306.01069*, 2023.

[30] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 6558. NIH Public Access, 2019.

[31] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

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

[33] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[34] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[35] Minji Yoon, John Palowitch, Dustin Zelle, Ziniu Hu, Ruslan Salakhutdinov, and Bryan Perozzi. Zero-shot transfer learning within a heterogeneous graph via knowledge transfer networks. *Advances in Neural Information Processing Systems*, 35:27347–27359, 2022.

[36] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[37] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[38] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*, 2022.