
GPS: Gumbel Permutation Sinkformer for Unified Multimodal Encoder in Multimodal Learning with Similarity-based Tokens Grouping

Junyang Zhang
Department of Electrical Engineering
junyangz@caltech.edu

Tianyi Zhu
Department of Electrical Engineering
tzhu@caltech.edu

Abstract

Current Multimodal Large Language Models (MLLMs) commonly employ separate encoders for text and visual modalities, concatenating their outputs to form an input sequence for the shared LLM transformer backbone. This naive concatenation strategy introduces significant positional noise when performing long range attention, and this error propagate through all deeper transformer layers, thereby undermining the model’s effectiveness in capturing multimodal semantic relationships. To tackle these issues, our project explores the novel approach to reorder image and text tokens into a unified input sequence (like a unified image+text encoder) for large multimodal models (LMMs) like LLaVA, aiming at improve efficiency and task performance by grouping semantically related tokens, addressing limitations in standard concatenation methods. We proposed two novel techniques to solve this problem. First, we proposed a cosine similarity based reordering technique called TokenMixer, which selectively group the most similar text and image tokens based on their cosine similarity score. Second, to ensure proper gradient flow back to the vision encoder, we also proposed another novel trainable Gumbel Permutation Sinkformer (GPS) to adaptively reorder image and text tokens input sequence of the LLM backbone during training. Experiments based on 6 different popular metrics have shown that reordering can improve the vision question answering quality after just a few epochs of LORA finetuning and the proposed GPS architecture can perfectly learn any large permutation matrix, indicating its potential to be integrated into tokens grouping in multimodal learning. Code: <https://github.com/garyz712/GPS>

1 Introduction

Multimodal Large Language Models (MLLMs) like LLaVA Liu et al. [2023] and TinyLLaVA Wang et al. [2023] have shown remarkable capabilities in understanding and reasoning across visual and textual modalities. However, a common architectural limitation in these models lies in their separate encoding of image and text modalities, followed by straightforward token concatenation. Such an approach results in significant computational inefficiency due to long-range attention requirements and severe positional misalignment and bias between related tokens across modalities Alayrac et al. [2022], Li et al. [2023].

The graph Figure 1 below illustrates this limitation using a typical vision-language model pipeline—for example, models like LLaVA or TinyLlava concatenate 576 image tokens (e.g., from a CLIP vision encoder Radford et al. [2021])) with 50 text tokens (e.g., from a user prompt) into a flat sequence: $(I_1, I_2, \dots, I_{576}, T_1, T_2, \dots, T_{50})$. This forces every underlying transformer layers in the LLM backbone to rely on expensive global attention to match semantically related elements—for instance, aligning the word “dog” (T_{10}) with image patches showing a dog (e.g., I_{10}, I_{15}). For

example, from our experiment, for two relevant text and image tokens, if their attention score before softmax is 1.4 without applying rope positional encoding, after placing these two tokens next to each other and applying rope encoding, the consequent attention score will be as similar as 1.2. Unfortunately, in most LMM architecture, the image token will be placed far away from the relevant text token, significantly decreasing the original attention score from 1.4 down to -1.7. As we only care about the content relevance instead of relative token position between different modalities, this gap introduced by their positional noise is extremely costly for the underlying LLM to compensate for without structural guidance.

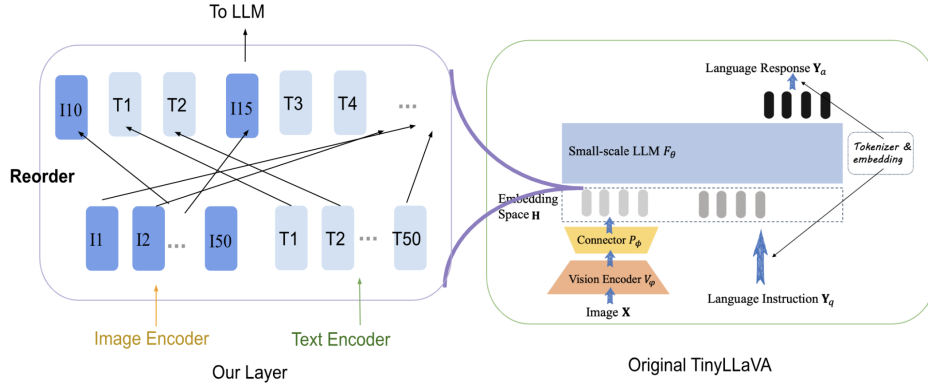


Figure 1: Multimodal Token Reordering in TinyLlava Architecture.

In response to this, we propose two novel approaches to minimize the positional noise in multimodal learning. First, we propose to group relevant text and image tokens based on their cosine similarity score in the embedding space as shown in Figure 1. In this approach we primarily reorder the image tokens and keep the original relative order of the text tokens, since the vision encoder has already encode positional information through Contrastive Language-Image Pre-Training (CLIP) Vision Transformer’s positional embedding layer. In this way, the image token is prepended to the most similar text token if their similarity score is above a predefined reordering threshold. Our experiments have confirmed that cosine similarity-based reordering with 40 minutes of LORA finetuning could reduce overall hallucination and improve vision question answering quality among 6 different popular metrics.

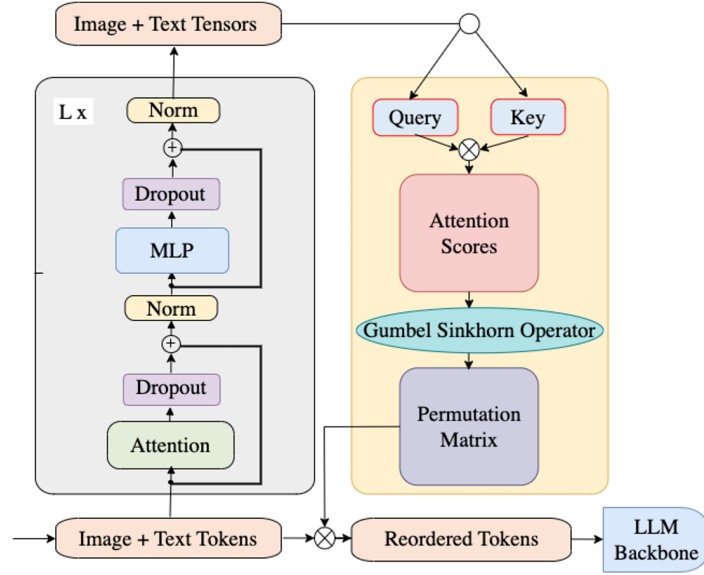


Figure 2: Gumbel Permutation Sinkformer Architecture

However, this reordering operation is inherently non-differentiable, prohibiting further training or finetuning of the vision encoder and the multimodal projector. Therefore, as seen in Figure 2 we propose the second approach: Gumbel Permutation Sinkformer (GPS), which apply the Gumbel-Softmax and Sinkhorn normalization to learn an adaptive permutation matrix and reorder all text and image tokens end-to-end. Inspired by ResNet He et al. [2016], the GPS layers are instructed to learn a residual permutation matrix instead of directly learning a full permutation matrix. From our experiments, with proper parameters, a simple single-headed 4-layer GPS encoder can learn any arbitrary permutation matrix perfectly within seconds on a A100 GPU. GPS can therefore be seamlessly integrated with any Large Multimodal Models for any downstream tasks, such as vision question answering.

2 Related Work

Multimodal Token Content Misalignment Multimodal Large Language Models (MLLMs) aim to seamlessly integrate visual and textual information, typically utilizing an adapter module positioned between the vision encoder and the language model. Adapters can range from simple linear transformations to more complex multi-layer perceptrons, converting visual features (visual patches) from the vision encoder into compatible representations (visual tokens) for the language model’s embedding space. During pre-training, the adapter is tasked with aligning these visual tokens closely to textual embeddings. However, existing methods primarily employ implicit supervision through auto-regressive losses, resulting in significant misalignment between visual and textual embeddings. As seen in Figure 3, the most similar word in the LLM embedding space is strongly misaligned with the image token by nature. Such misalignment substantially impacts the model’s comprehension capabilities, leading to incorrect responses and hallucinations. Additionally, this misalignment complicates further fine-tuning stages, exacerbating the divergence between visual and textual representations and thereby weakening the overall performance and stability of the language model. More importantly, without first aligning tokens content, similarity-based positional alignment cannot be properly applied.

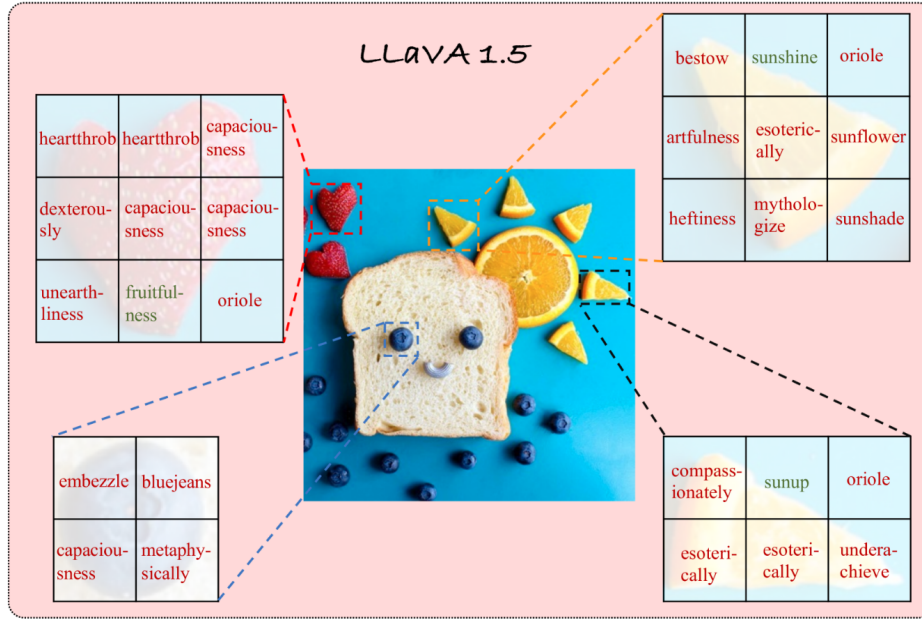


Figure 3: Image Token Misalignment in Llava 1.5

Cross Attention-based Model Flamingo Alayrac et al. [2022] is an influential Vision Language Model designed to address the alignment problem in MLLMs by introducing a cross-attention mechanism between visual and language features. Instead of projecting visual tokens directly into the language embedding space, Flamingo allows the language model to attend to the outputs of a frozen vision encoder using interleaved cross-attention layers. This design enables the model to dynamically condition on visual content when generating text, improving multimodal understanding. However, one major limitation is that cross-attention restricts interactions to only between modalities, without enabling self-attention within each individual modality. This can limit the model’s ability to capture intra-modal dependencies, especially for long-range visual or textual reasoning. Additionally, Flamingo is primarily designed for two modalities—vision and text—which may hinder its flexibility when extending to more complex multimodal scenarios involving audio, video, or structured data.

Supervised Embedding Alignment SEA Yin et al. [2024] is a novel token-level alignment strategy that leverages vision-language pre-trained models such as CLIP to generate fine-grained semantic labels for visual tokens through cosine similarity-based matching from a curated word list Yin et al. [2024]. These labels are then used to explicitly supervise the adapter via a contrastive loss during pre-training, guiding the transformation of visual patches into the LLM’s embedding space. SEA avoids additional annotation costs, maintains inference efficiency, and improves performance across multiple benchmarks and LLM scales. Their results demonstrate that SEA significantly enhances alignment quality and model robustness, particularly in smaller LLMs where misalignment is more severe. However, SEA completely ignore the positional noises mentioned earlier that also exacerbate the misalignment issue.

Circle Rotary Position Embedding Circle-RoPE Wang et al. [2025] address this challenge of positional noise by proposing a novel positional encoding framework that decouples text and image token indices to enhance multimodal understanding. Circle-RoPE projects image token indices onto a 3D circular trajectory orthogonal to the text index direction, ensuring consistent RoPE distances (Per-Token Distance, PTD = 0) between text and image tokens, thus eliminating artificial positional dependencies while preserving intra-image spatial relationships through a mixed-angle circular mapping. However, a significant weakness is its reliance on static positional indices, which does not account for the loss of modality information after the first transformer layer due to token mixing; since deeper layers cannot distinguish text from image tokens without explicit modality tags, the effectiveness of Circle-RoPE’s modality-specific geometric transformations may diminish, limiting its applicability in scenarios requiring persistent modality-aware processing.

Gumbel-Sinkhorn Network Mena et al. [2018] introduces a differentiable approach to learning latent permutations in neural networks by leveraging the Sinkhorn operator—a continuous relaxation of permutation matrices to the Birkhoff polytope of doubly stochastic matrices. The authors define the *Gumbel-Sinkhorn distribution*, a reparameterizable approximation that extends the Gumbel-Softmax trick to permutations. By injecting Gumbel noise into the assignment scores and applying Sinkhorn normalization, the model enables gradient-based learning over permutations. This framework allows for end-to-end training in tasks involving latent combinatorial structures, such as sorting, jigsaw puzzle reconstruction, and neuron identification. Empirically, Gumbel-Sinkhorn networks outperform existing baselines while maintaining simplicity and scalability, and they open the door to principled probabilistic inference in settings with latent matchings.

3 Method

3.1 Tokens Reordering by Cosine Similarity

The first token reordering approach we propose is to spatially group each image token with its most semantically relevant text token in the input sequence (measured by their cosine similarity), thereby promoting local attention and alignment. Image token that has a well-matched text token (cosine similarity greater than a threshold) will be inserted to the position right in front of that text token, and the ones without good match will just be placed in original order right before all the text tokens.

Algorithm 1 Cosine Similarity-Based Token Reordering

Require: Image token embeddings $\{\mathbf{I}_1, \dots, \mathbf{I}_M\} \subset \mathbb{R}^d$, outputs of the CLIP vision encoder. Text token embeddings $\{\mathbf{T}_1, \dots, \mathbf{T}_N\} \subset \mathbb{R}^d$, a cosine similarity reordering threshold τ_{align}

Ensure: Reordered multimodal token sequence

- 1: Initialize empty list unmatched_bucket $\leftarrow []$
- 2: Define alignment threshold $\tau_{\text{align}} \in [0, 1]$
- 3: **for** $m = 1$ to M **do**
- 4: Compute cosine similarities:

$$\text{sim}(n) \leftarrow \frac{\mathbf{i}_m \cdot \mathbf{t}_n}{\|\mathbf{i}_m\| \cdot \|\mathbf{t}_n\|}, \quad \forall n \in \{1, \dots, N\}$$

- 5: Find best matching text token:

$$n^*, s^* \leftarrow \arg \max_n \text{sim}(n), \quad \text{sim}(n^*) = s^*$$

- 6: **if** $s^* < \tau_{\text{align}}$ **then**
 - 7: Append image token index m to unmatched_bucket
 - 8: **else**
 - 9: Append image token index m to bucket $_{n^*}$
 - 10: **end if**
 - 11: **end for**
 - 12: Let sequence = unmatched_bucket
 - 13: **for** n in text_sequence **do**
 - 14: Append text token T_n to sequence
 - 15: **for all** $m \in \text{bucket}_n$ **do**
 - 16: Append image token I_m to sequence
 - 17: **end for**
 - 18: **end for**
 - 19: **return** sequence
-

The resulting sequence is structured such that each text token immediately follows its most semantically related image token. Positional embedding is then applied as before to every LLM layer and it is fed into the llama model. It is important to note that this reordering process is non-differentiable, as it involves discrete operations based on cosine similarity. Consequently, gradient flow is interrupted, preventing backpropagation through the CLIP image encoder. As a result, only the later stages of the

model—such as the LLaMA-based language backbone—can be fine-tuned during training, while the image encoder remains frozen.

3.2 Gumbel Permutation Sinkformer

To achieve differentiable token reordering, we propose GPS—a trainable permutation module that learns to align semantically similar multimodal tokens. GPS operates on the outputs of a lightweight transformer stack, which processes the concatenated image and text embeddings. At the final layer, we compute token-wise attention scores $S \in \mathbb{R}^{N \times N}$ and add Gumbel noise to enable differentiable sampling. The noisy scores are scaled by a temperature parameter τ and normalized using the Sinkhorn algorithm to produce a soft permutation matrix $P \in \mathbb{R}^{N \times N}$ that is approximately doubly stochastic, where each row and each column sums to 1, and all entries are non-negative.

During training, we optionally apply a hard permutation by taking the row-wise $\arg\max$ over P to obtain a one-hot permutation matrix P_{hard} . Since this operation is non-differentiable, we adopt the straight-through estimator (STE) to enable gradient flow: we replace P with $P_{\text{hard}} + (P - \text{detach}(P))$. This preserves the non-differentiable hard permutation during the forward pass while using the gradient of the soft permutation during backpropagation, effectively treating $\frac{\partial P}{\partial P} = 1$ and bypassing the non-differentiable $\arg\max$ operation. This design allows GPS to be fully end-to-end trainable and seamlessly integrated into existing multimodal architectures.

To smoothly anneal the temperature τ from an initial value τ_{start} to a final value τ_{end} , we adopt a cosine decay schedule over a fixed number of training steps. At any step $s \in [0, S]$, the temperature is computed as:

$$\tau(s) = \tau_{\text{end}} + \frac{1}{2}(\tau_{\text{start}} - \tau_{\text{end}}) \left(1 + \cos \left(\pi \cdot \frac{s}{S} \right) \right),$$

where S denotes the total number of annealing steps. This schedule begins at τ_{start} when $s = 0$ and gradually decays to τ_{end} as $s \rightarrow S$, allowing the model to start with a smoother (more uniform) distribution and progressively sharpen towards deterministic permutations.

Algorithm 2 Gumbel Permutation Sinkformer (GPS)

Require: Input tokens $X \in \mathbb{R}^{B \times N \times D}$, temperature τ with a cosine decay scheduler, identity matrix scaling factor α , number of blocks L , Sinkhorn iterations T

Ensure: Soft or hard permutation matrix $P \in \mathbb{R}^{B \times N \times N}$

```

// Feature encoding via stacked GPS blocks
1: for  $\ell = 1$  to  $L$  do
2:   Compute  $Q, K, V \leftarrow \text{Linear}(X)$ 
3:    $A_O \leftarrow \frac{QK^\top}{\sqrt{D}}$ 
4:    $S \leftarrow A_O + \alpha \cdot I$  ▷ Default dot-product attention + Scaled identity matrix
5:    $A \leftarrow \text{softmax}(A_O)$ 
6:    $X \leftarrow \text{LayerNorm}(X + AV)$ 
7:    $X \leftarrow \text{LayerNorm}(X + \text{FeedForward}(X))$ 
8: end for
// Gumbel-Sinkhorn permutation sampling
9: Sample Gumbel noise  $G \sim -\log(-\log(U))$ ,  $U \sim \text{Uniform}(0, 1)$ 
10:  $\log \alpha \leftarrow (S + G)/\tau$ 
11:  $P \leftarrow \text{Sinkhorn}(\log \alpha, T)$  ▷ Iteratively softmax normalize rows and columns to get a soft permutation matrix
12: if hard permutation then ▷ Use full permutation matrix in forward pass
13:    $P_{\text{hard}} \leftarrow \text{OneHot}(\arg \max P)$ 
14:    $P \leftarrow P_{\text{hard}} + (P - \text{detach}(P))$  ▷ Straight-through trick:  $\partial P / \partial P = 1$  in backward pass
15: end if
16: return  $P$ 

```

4 Experiments

During this term, we focused on initial exploration of the proposed methods, implementing their prototype, and setting up the training and validation infrastructure. We are able to obtain many valuable data and make some preliminary but meaningful assessments for the results and performance.

4.1 Hardware

The training and evaluation are done on hardwares including:

- Nvidia A100 40GB GPU. For easier coding and prototyping.
- Server with dual H100 80GB set-up, for faster training and larger VRAM.

4.2 Datasets

llava-instruct-mix-vsft-mini This is a small scale, 8000-sample dataset. [UnslothAI, 2024]. Each sample of it includes an image, a question based on this image, and the ground truth answer. We will expand on this and give some samples in later section when analysing the performance of the model on this dataset.

LLaVA-Instruct-150K This is a larger scale dataset with 150k sample, with similar format as the llava-instruct-mix-vsft. This is also one of the dataset used for the original LLaVA model training. Due to the significantly larger size of this dataset, all the training utilizing this are done on the dual H100-80GB set-up.

We implemented dedicated dataloader for both of the dataset, since they are formatted differently. The LLaVA-Instruct-150K also includes downloading images from a separate source, and manually splitting train and test set as it doesn't split by default.

4.3 Training for Method 1: Cosine Similarity

We implemented Algorithm 1 carefully, integrating it to the tinyLLaVA module. We first tried training this on Colab with A100-40GB for the llava-instruct-mix-vsft-mini [UnslothAI, 2024] dataset. It is fine-tuned based on the official pretrained model with LoRA[Microsoft, 2021]. Listed are the training parameters:

Parameter	Value
LoRA Rank	8
LoRA Dropout	0.05
Training Epochs	1
Datatype	fp16
Learning Rate	1×10^{-4} (with scheduler)
Optimizer	paged_adamw_8bit

Table 1: Cosine Similarity Training Configuration

In addition to tracking training and validation loss, we evaluate the BLEU score at each epoch to assess the model's generative quality. BLEU (Bilingual Evaluation Understudy) [Papineni et al., 2002] is a metric that measures the similarity between generated text and reference text based on n-gram overlap. While loss provides a measure of optimization progress, BLEU offers a more interpretable indication of linguistic accuracy and relevance, making it a valuable complement for evaluating model performance, especially in language generation tasks.

Epoch	Training Loss	Validation Loss
1	1.0946	1.2527
2	0.9910	1.2241
3	0.8874	1.2226
4	0.7574	1.2147
5	0.6480	1.2241
6	0.5773	1.2059
7	0.4381	1.2084
8	0.3831	1.2603
9	0.3282	1.3199
10	0.2639	1.4761

Table 2: Training Metrics per Epoch

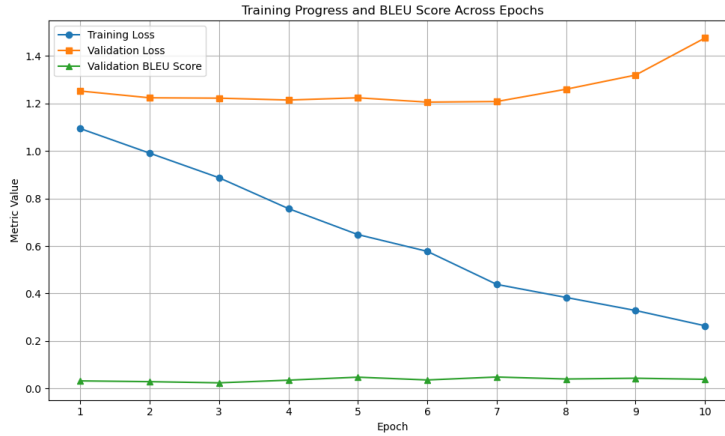


Figure 4: Training Loss for Cosine Similarity

As the training Curve 4 and Table 2 illustrate, the model’s training loss falls consistently from 1.0946 to 0.2639, yet the validation loss oscillates near 1.20 and begins rising after epoch 6. This limited improvement stems mainly from our very small fine-tuning set and a practical issues—imperfect image–text alignment generating meaningless cosine similarity—that we discuss in the next section. Because the gap between training and validation loss widens and BLEU declines after epoch 7, we activated early stopping at the seventh epoch to curb over-fitting and selected that checkpoint for downstream evaluation.

4.4 Analyzing Result for Method 1: Cosine Similarity

For a better examination beyond validation loss, we evaluated the trained Cosine Similarity model on several benchmark metrics and compared the results with those of the original model. For a fair comparison, both models were fine-tuned for 5 epochs using the same training dataset and LoRA parameters, with the best checkpoint selected via early stopping. Below is an overview of the benchmarking metrics used in our evaluation:

BLEU Score Measures n-gram precision between the generated and reference text. It is widely used in machine translation and text generation tasks to evaluate surface-level similarity.

BERTScore (F1) Computes token-level semantic similarity using contextual embeddings from a pretrained BERT model. Unlike BLEU, it captures deeper semantic meaning and is more robust to paraphrasing.

METEOR Evaluates output quality based on exact, stemmed, synonym, and paraphrase matches between candidate and reference texts. It places greater emphasis on recall and is designed to reflect human judgment more accurately.

CIDEr Originally developed for image captioning, CIDEr measures consensus by computing TF-IDF-weighted n-gram similarity across multiple reference sentences. It rewards informative and diverse captions.

ROUGE-L Based on the longest common subsequence (LCS), ROUGE-L captures fluency and informativeness. It is widely used for evaluating text summarization.

Validation Loss Represents the cross-entropy loss on the validation set, indicating how well the model generalizes to unseen data. Lower values suggest better predictive performance.

Table 3: Comparison of Evaluation Metrics Between Models

Metric	Cosine Similarity (Threshold = 0.1)	Original Model
Validation Loss	1.293967	1.280968
Validation BERTScore (F1)	0.8971	0.8952
Validation METEOR	0.3471	0.3183
Validation CIDEr	1.7027	1.7284
Validation ROUGE-L	0.3914	0.3800
Validation BLEU Score	0.3030	0.2839

The results show that the cosine similarity-based model yields improvements across most evaluation metrics compared to the original baseline. While some gains are observed—such as higher BERTScore, METEOR, ROUGE-L and BLEU—the overall impact is limited. This modest improvement is partly attributable to the low LoRA rank used during fine-tuning, which restricts the model’s capacity to adapt. Additionally, other potential sources of error, such as implementation issues or suboptimal projection strategies, may have further constrained performance. Nonetheless, the comparable results across multiple metrics indicate that our model is functioning correctly and improving the performance of the original Vision Language Model. These findings confirm that the modified architecture is at least on par with the original setup and provides a solid foundation for future refinements.

To gain deeper insight into model performance, we visualized the outputs on several test set samples:

Example 1: Prompt What problems might the placement of this stop sign cause?



Ground-truth The unusually large stop sign, covering a significant portion of the lane or entrance, can cause several problems. First, it might obstruct the view of oncoming traffic, pedestrians, or other obstacles for drivers, which increases the risk of accidents. Additionally, the sign’s size and position could confuse drivers or make it difficult for them to navigate around the sign, causing traffic disruptions or delays. Furthermore, the placement might block access to parking lots or entrances to buildings, affecting businesses or residents relying on the area for parking and access. It is important

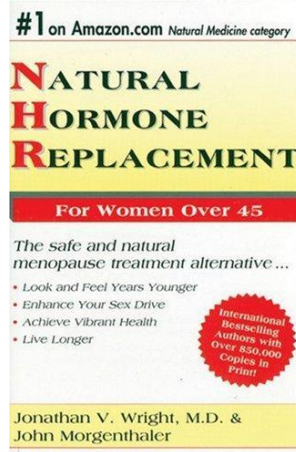
that traffic signs adhere to standard size regulations and are placed in appropriate locations to ensure traffic safety, smooth flow, and clear communication of road rules.

Base Model Answer : The stop sign is placed next to the road, partially obscured by a railing and surrounded by a bridge and a fence. This placement of the stop sign could cause several problems:

1. Visibility: The sign might be partially obscured by the railing or the bridge, making it difficult for drivers and pedestrians to notice and obey the stop sign. 2. Safety: The presence of a fence and the bridge near the sign might lead to distractions or accidents, as drivers might not pay full attention to the sign or fail to notice it at all. 3. [truncated]

Trained Cosine Similarity Model Answer : The placement of the stop sign next to a wooden fence and by a roadside pedestrian bridge could cause several problems. First, it might be difficult for drivers to notice the sign due to the proximity of the fence and the bridge, leading to potential accidents at the intersection. Second, the close proximity of the stop sign to the fence and the bridge could potentially obstruct access to the area for pedestrians or vehicles. This could result in disruptions to the flow of traffic and increase the risk of incidents. Lastly, the combination of the stop sign and adjacent structures may also [truncated]

Example 2: Prompt Who wrote this book?



Ground-truth John Morgenthaler

Base Model Answer : Arthur Benavie

Trained Cosine Similarity Model Answer : Dr. John Report

To understand the underlying cause of the observed marginal performance difference, we inspected the cosine similarity values computed between image and text tokens, which serve as the criteria for reordering. The similarity scores were mostly low—well below the alignment threshold $\tau_{\text{align}} = 0.1$ —effectively resulting in reordering only for some most important tokens:

```
[[ 0.0209, -0.0111, -0.0021, ..., -0.0124, 0.0080, 0.0054],
 [-0.0213, -0.0077, -0.0057, ..., -0.0013, -0.0156, -0.0004],
 [ 0.0280, -0.0002, 0.0015, ..., -0.0173, -0.0138, 0.0076],
 ...,
 [-0.0069, 0.0341, 0.0365, ..., 0.0147, -0.0113, 0.0102],
 [ 0.0109, 0.0391, 0.0433, ..., 0.0247, 0.0027, -0.0037],
 [ 0.0085, 0.0292, 0.0260, ..., 0.0168, 0.0159, -0.0047]]
```

Upon deeper analysis of the LLaVA architecture, we found that it utilizes only the vision encoder from CLIP, while the text tokens remain in their original embedding space. Consequently, **the image and**

text embeddings are not projected into a shared representation space, making cosine similarity between them less convincing.

To enable more meaningful cross-modal comparison, a projection mechanism is required to map both image and text tokens into a common space. This highlights the need for a more effective projection strategy. For future work, we will be drawing ideas from the paper discussed in the background section [Yin et al., 2024] — that have **addressed alignment and comparability in multimodal embeddings**, and incorporate similar ideas to improve the effectiveness of token reordering.

4.5 Method 2: Gumbel Permutation Sinkformer

Another drawback of using cosine similarity is that the upstream CLIP encoder is not trainable. Method 2 was implemented to address this limitation. To validate its effectiveness, we generated a random reordering matrix as ground truth. The model learned as expected—training loss decreased rapidly, and accuracy reached a high value within the first few steps, as shown in the training loss diagram below.

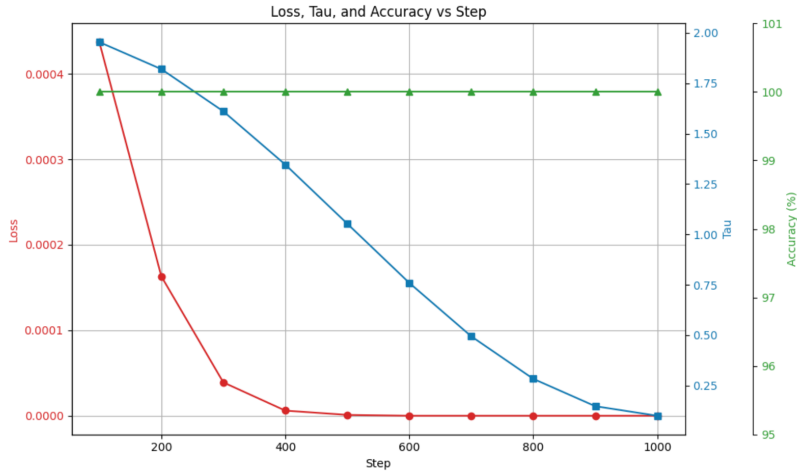


Figure 5: GPS can learn any arbitrary permutation perfectly with cosine temperature tau scheduling (here token length = 800).

However, this method does not address the embedding misalignment problem discussed in section 4.4. In future work, we plan to implement this approach and integrate it with our reordering algorithm to compare its performance against Method 1: Cosine Similarity Reordering.

5 Future Works

Throughout this semester, we tried two novel approaches to address the image-text positional misalignment problem in vision-language models by token reordering. We have laid the foundation for a model training and evaluation framework, implemented prototypes for these methods, examined the result, and identified the issues at the current stage. We believe that by resolving these issues, we can develop an effective approach that directly addresses the core research questions and contributes to improving the performance of vision-language models.

Addressing misalignment problem Fixing the issue discussed in Section 4.4 is our top priority, as it is fundamental to ensuring that our results are meaningful and interpretable.

Bug Find and Fix Throughout the project, we observed that many cases of poor performance or failed tests were caused by small bugs in the code, rather than methodology. It is essential to systematically identify and eliminate them. This will ensure the system is correctly implemented and optimized for validating our proposed method, and will allow for fair comparisons with alternative approaches.

Training on larger scale & Benchmarking Once a reliable and bug-free system is in place, we plan to scale up training using a much larger dataset (e.g., the 150k-sample corpus), conduct additional experiments, and run standard benchmarks such as VQA to quantitatively evaluate and compare performance with existing models.

Acknowledgment

We thank Caltech CS159 team for GPU computing resources and Stanford Tambe Lab for technical guidance.

References

- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, et al. Flamingo: A visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35: 23716–23736, 2022.
- 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.
- Junnan Li, Dongxu Li, et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.
- Haotian Liu, Chunyuan Li, Qingyang Wu, Yong Jae Lee, et al. Llava: Large language-and-vision assistant. *arXiv preprint arXiv:2304.08485*, 2023.
- Gonzalo Mena, David Belanger, et al. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations (ICLR)*, 2018.
- Microsoft. Lora: Low-rank adaptation of large language models. <https://github.com/microsoft/LoRA>, 2021. Accessed: 2025-06-02.
- 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 (ACL)*, pages 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135.
- Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.
- UnslothAI. Llava-instruct-mix-vsft-mini dataset. <https://huggingface.co/datasets/unsloth/llava-instruct-mix-vsft-mini>, 2024. Accessed: 2025-06-02.
- Chengcheng Wang, Jianyuan Guo, Hongguang Li, Yuchuan Tian, Ying Nie, Chang Xu, and Kai Han. Circle-rope: Cone-like decoupled rotary positional embedding for large vision-language models. *arXiv preprint arXiv:2505.16416*, 2025.
- Shengshen Wang, Bowen Dai, et al. Tinyllava: Lightweight language and vision assistant. *arXiv preprint arXiv:2311.15102*, 2023.
- Yuanyang Yin, Yaqi Zhao, Yajie Zhang, Ke Lin, Jiahao Wang, Xin Tao, Pengfei Wan, Di Zhang, Baoqun Yin, and Wentao Zhang. Sea: Supervised embedding alignment for token-level visual-textual integration in mllms. *arXiv preprint arXiv:2408.11813*, 2024.