PuMer: Pruning and Merging Tokens for Efficient Vision Language Models

Qingqing Cao **Bhargavi Paranjape** **Hannaneh Hajishirzi**
{qicao,bparan,hannaneh}@cs.washington.edu
University of Washington

Abstract

Large-scale vision language (VL) models use Transformers to perform cross-modal interactions between the input text and image. These cross-modal interactions are computationally expensive and memory-intensive due to the quadratic complexity of processing the input image and text. We present PuMer[1]: a token reduction framework that uses text-informed **Pru**ning and modality-aware **Mer**ging strategies to progressively reduce the tokens of input image and text, improving model inference speed and reducing memory footprint. PuMer learns to keep salient image tokens related to the input text and merges similar textual and visual tokens by adding lightweight token reducer modules at several cross-modal layers in the VL model. Training PuMer is mostly the same as finetuning the original VL model but faster. Our evaluation for two vision language models on four downstream VL tasks shows PuMer increases inference throughput by up to 2x and reduces memory footprint by over 50% while incurring less than a 1% accuracy drop. [2]

1 Introduction

Large-scale vision language models (Dou et al., 2022; Wang et al., 2022; Zeng et al., 2021; Kim et al., 2021; Wang et al., 2021; Zhang et al., 2021) have shown substantial progress on many vision language tasks such as visual question answering, natural language visual reasoning, and visual entailment. However, state-of-the-art language and vision models are memory intensive and computationally expensive because they use multi-layer self-attention between many language and vision input tokens (small image patches) with quadratic complexity. This inefficiency limits high-throughput cloud deployments and makes it infeasible to run on resource-constrained devices.

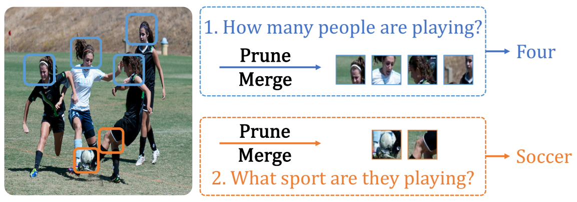
1. How many people are playing?
Prune
Merge
Four
Prune
Merge
2. What sport are they playing?
Soccer

Figure 1: PuMer applies token reduction to VL models via pruning and merging. PuMer makes VL models run faster by text-informed image pruning to remove text-irrelevant image tokens and modality-aware merging to compress similar input tokens.

The key source of inefficiency in deep VL models is that these models need to process the entire input image and text tokens over all the layers. Our intuition is that the input image contains redundant information and only parts of the image (*salient* regions, referred by the text) are required and related to the end task. For example, in Figure 1, most of the image content (the four persons, field) is not needed except for the bottom-center soccer region to answer the visual question "What sport are they playing?". This paper advocates using the correlations between image and text modalities to reduce tokens for VL problems.

In the vision-only or text-only domains, researchers have shown that reducing image or text tokens can improve the model computational complexity through *pruning* (Liang et al., 2021; Rao et al., 2021; Yin et al., 2022; Marin et al., 2021; Goyal et al., 2020) that learns to remove non-salient image or text tokens for a given task; or *merging* (Bolya et al., 2022; Xu et al., 2022; Ryoo et al., 2021) that groups semantically similar tokens. Using either reduction method in isolation is not sufficient for a VL problem setting since *i)* salient image tokens are different given different text inputs, *ii)* pruning alone causes big information loss, hurting the performance, *iii)* merging tokens irrespective of their modality confuses the VL models since text and image token representations cannot

[1]Pronounced as "puma"
[2]Code is available at https://github.com/csarron/PuMer.

be perfectly aligned to the same semantic space. In this paper, we design a lightweight and effective framework that integrates these token reduction strategies into VL models.

We introduce **PuMer**, a token reduction framework that consists of **Pru**ning-and-**Mer**ging operations to gradually reduce image tokens that are not related to text and merge image and text tokens respective to their modality. In particular, we design *(i) text-informed image token pruning* to remove image tokens that are irrelevant to text and are unimportant to the VL task predictions (removing tokens that describe persons and field for the second question in the Figure 1 example); *(ii) modality-aware token merging* to merge semantically redundant tokens for text and image tokens modality independently (combining the image tokens describing each person for the first question in Figure 1). We keep the remaining tokens that are neither pruned nor merged. At the core of PuMer is a set of lightweight non-parametric token reducers that decide which image tokens are pruned and merged as the VL model forward computation proceeds. To reduce abrupt image information loss and improve computational efficiency, we scatter the token reducers at different cross-modal layers in the VL model and reduce the tokens in a cascaded fashion. Fewer tokens are pruned and merged in earlier layers.

PuMer is easy to train since the token reducers contain no parameters and add little overhead. The training procedure is almost the same as finetuning the original VL models, except that we add a knowledge distillation loss that further reduces the accuracy gap compared to finetuned models. Though we focus on inference efficiency, PuMer makes VL models run faster for both training and inference because text and image tokens are reduced in the forward computation.

We evaluate PuMer over two recent VL models ViLT (Kim et al., 2021) and METER (Dou et al., 2022) across five vision language tasks: text-image retrieval tasks (including image-to-text and text-to-image retrieval) (Plummer et al., 2015), visual question answering (VQAv2; Goyal et al. 2017), natural language visual reasoning (NLVR2; Suhr et al. 2019), and visual entailment (SNLI-VE; Xie et al. 2019). Compared to baselines, PuMer improves the model inference throughput by **1.7x**$\sim$**2.1x** and reduces memory footprint by **38%**$\sim$**50%** with minimal (less than 1%) accuracy loss. Our analysis validates that both text-informed image pruning and modality-aware token merging contribute to the token reduction effectiveness of PuMer.

2 Related work

Token Reduction in NLP and Vision. Prior work in data pruning (Rao et al., 2021; Yin et al., 2022; Liang et al., 2021; Goyal et al., 2020) focus on single-modality models by either pruning input text or image alone. DynamicViT (Rao et al., 2021) and A-ViT (Yin et al., 2022) both progressively remove the uninformative content and keep salient regions in the input image. This type of pruning does not apply to language and vision tasks where the salient regions depend on the input text. Our work shows different input texts lead to pruning different image regions even for the same input image. PoWER-BERT (Goyal et al., 2020) speeds up the inference of text-based Transformers like BERT (Devlin et al., 2019) by removing the input text tokens, which are not the main computation bottlenecks for most vision and language tasks.

Another line of work seeks to reduce input tokens by merging tokens. SPViT (Kong et al., 2022) and EViT (Liang et al., 2021) select uninformative image tokens and combine them into one token. And EViT also requires expensive pretraining. GroupViT (Xu et al., 2022) combines image tokens via cross-attention to find similar objects for semantic segmentation. Recently, ToMe (Bolya et al., 2022), TokenLearner (Ryoo et al., 2021) and TokenPooling (Marin et al., 2021) combine tokens without pruning and achieves better speedup versus accuracy trade-offs.

Our method is inspired by token pruning and merging works but integrates them into a token reduction framework suitable for VL models. Our key difference is to leverage the relationships between textual and visual tokens to remove and combine tokens. Our experiments (Section 5) show improvements over these lines of work.

Efficient Vision Language Models. Many techniques have focused on model pruning (Lagunas et al., 2021; Yu and Wu, 2021; Yu et al., 2022; TPr), dynamic computation by early exiting (Xin et al., 2020; Zhou et al., 2020; Schwartz et al., 2020; Liu et al., 2020; Cao et al., 2022) or designing small and efficient VL models (Fang et al., 2021; Wang et al., 2020). Combining these orthogonal optimizations with our token reduction method could further accelerate the inference in VL models.

3 Background and Overview

Vision Language Models. Figure 2 shows the backbone of a VL model consisting of a text encoder, an image encoder, and a cross-modal encoder. The input sentence (e.g. a question or a statement) is first tokenized as text tokens and fed to the text encoder to create contextualized text representations. Similarly, the input image is projected into many small image patches, referred to as "image tokens", that are further contextualized by the image encoder. Finally, the cross-modal encoder takes the concatenated text and image tokens and fuses information between image and text modalities via Transformer-style (Vaswani et al., 2017) cross-attention interactions.

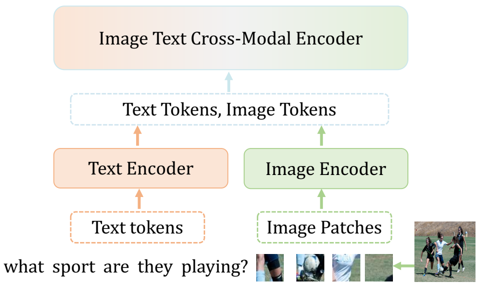
Image Text Cross-Modal Encoder
Text Tokens, Image Tokens
Text Encoder
Image Encoder
Text tokens
Image Patches
what sport are they playing?

Figure 2: General architecture of vision language models. The input image is projected into many small image patches ("tokens") that are processed by the image encoder. The cross-modal attention between text and image tokens has quadratic time complexity, which is computationally expensive. Both ViLT and METER models follow this pattern.

For many VL tasks, the number of tokens of the input image is an order of magnitude more than that of the input text — a visual question can have at most a dozen tokens but the associated image consists of a hundred image tokens. For example, for an image with a resolution of 384x384 and a patch size of 16, the number of tokens is $(384/16)^2 = 576$.

Token Reduction for Efficiency. In this paper, we focus on *reducing* image tokens to improve computational efficiency of the model through *pruning* and *merging*. However, naively removing a large percentage of the image tokens inside the cross-modal layers may cause abrupt image information loss, as the VL model is trained to build representations of the full image for the downstream task. For example, if the soccer region in Figure 1 gets pruned, the VL model is unlikely to output the answer "soccer" for the question "what sport are they playing?". On the other hand, simply merging image tokens without text guidance can lead to suboptimal performance. For example, merging the image regions of the background field and soccer in Figure 1 does not contribute to answering the visual question "how many people are playing?".

The next section describes our text-informed token reduction approach. The basic building blocks of PuMer are lightweight non-parametric *token reducers* that reduce image and text tokens in a cascaded manner to mitigate the information loss and improve the computational efficiency of a VL model.

4 PuMer: Text-Informed Token Reduction Framework

Given a VL cross-modal encoder, PuMer progressively reduces image tokens going through the cross-modal encoder (depicted in Figure 3). PuMer uses lightweight token reducers with no learnable parameters, adding them in different layers of the cross-modal encoder to predict which image tokens are removed or merged.

Token Reducers. For an n-layer cross-modal encoder, after the first f ($f < n$) layers, a token reducer first removes $k\%$ of the image tokens at any layer ℓ between f and n guided by the text information. The tokens removed in layer ℓ are not used in subsequent layers. Then the token reducer merges $r\%$ and $t\%$ of the image and text tokens respectively in layer ℓ. We scatter the token reducers across the cross-modal layers to achieve a better accuracy and efficiency trade-off. Intuitively, reducing at early layers in the cross-modal encoder will have higher inference efficiency but may have bigger performance loss and vice versa. We study this trade-off in more detail in Section 6.2.

The token reduction algorithm is described in Algorithm 1. Each token reducer consists of two sequential non-parametric modules: first, a *text-informed pruner* (**TIP**) prunes image tokens that are not related to the accompanying text (Section 4.1); second, a *modality-aware merger* (**MAM**) reduces tokens by merging similar tokens within the image or text modality (Section 4.2). These two steps reduce the image and text tokens to benefit the computational efficiency, while not losing the accuracy. Note that if we only apply text-informed pruning to the images without merging, to achieve similar efficiency gains, we need to set a larger pruning ratio which will

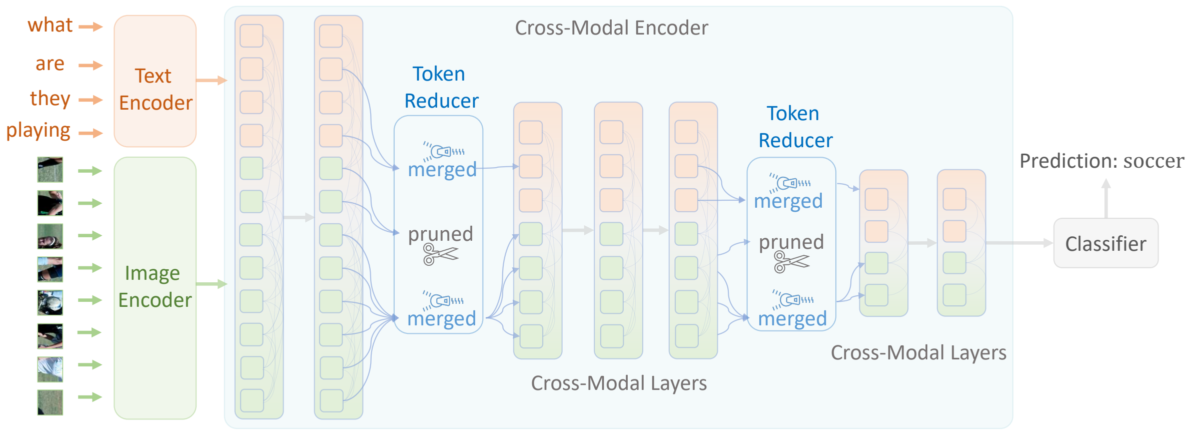

Figure 3: PuMer applies token reducers in the cross-modal layers of a VL model. Each token reducer is non-parametric and uses text-informed pruning and modality-aware merging to reduce image and text tokens.

Algorithm 1 Token Reduction via Text-Informed Image Pruning and Modality-Aware Merging

```
Input: text token vectors $\mathbf{T}$, text-to-image cross attention scores $\mathbf{A}$, image token vectors $\mathbf{V}$,
prune ratio $k$, image merge ratio $r$, text merge ratio $t$
Output: merged text token vectors $\mathbf{T}_m$, pruned and merged image token vectors $\mathbf{V}_m$
 1: for image tokens $\mathbf{V}$, compute text-saliency scores $\mathbf{s}$ using Eq1;        ▷ text-informed image pruning
 2: obtain indices $idx$ of top-$k'$ items in score $\mathbf{s}$, $k' = (1-k)|\mathbf{V}|$;     ▷ $k'$ is the # of kept image tokens
 3: select $k'$ image tokens by the top-$k'$ indices, $\mathbf{V}_p = \mathbf{V}[idx]$;
 4: merge text tokens $\mathbf{T}$ by bipartite soft matching into $\mathbf{T}_m$ = bipartite_merge($\mathbf{T}, t$);
    merge image tokens $\mathbf{V}_p$ into $\mathbf{V}_m$ = bipartite_merge($\mathbf{V}_p, r$)              ▷ modality-aware merging
 5: procedure BIPARTITE_MERGE(input tokens: $\mathbf{X}$, merge ratio: $r$)
 6:     divide tokens $\mathbf{X}$ into two sets of tokens $\mathbf{O}$ and $\mathbf{E}$ based on even and odd order
 7:     for each token $\mathbf{O}_a$ in $\mathbf{O}$, compute its top-1 similar token $\mathbf{E}_b$ in $\mathbf{E}$, save the indices $a$ and $b$ into
        a token edge (an edge between $\mathbf{O}_a$ and $\mathbf{E}_b$), save all token edges as $\mathbf{P}$ and corresponding
        top-1 similarity scores $\mathbf{S}_p$                    ▷ this can be implemented as a fast parallel operation
 8:     $r' = r|\mathbf{X}|$, obtain indices $ind$ of top-$r'$ items in $\mathbf{S}_p$, select top-$r'$ edges: $\mathbf{P}_r = \mathbf{P}[ind]$
 9:     for each token edge ($a$, $b$) in $\mathbf{P}_r$, collect tokens from $\mathbf{O}$ and $\mathbf{E}$, merge tokens in $\mathbf{O}$ and $\mathbf{E}$ that
        are connected via edges (sharing the same token as a vertex node) into $\mathbf{OE}$ by computing
        the average of each token vectors, gather $\mathbf{O}_{rest}$ and $\mathbf{E}_{rest}$ from the rest (unmerged) indices.
10:     output: merged tokens $\mathbf{X}_m = gather(\mathbf{OE}, \mathbf{O}_{rest}, \mathbf{E}_{rest})$
11: end procedure
```

hurt task performance due to substantial information loss. Instead of dropping such information, modality-aware merging helps alleviate information loss by compressing semantically similar content into fewer tokens while still providing efficiency benefits.

4.1 Text-Informed Image Pruning

The first step is to prune image tokens according to their relevance to the text. The intuition is that only some parts of the image are important for the end language-vision task, hence removing the text-irrelevant parts will not hurt the performance, while it improves the computational efficiency. Unlike previous works (Rao et al., 2021) that use extra learnable parameters to predict which image tokens to prune, we take a different but faster approach without using any parameters. The key idea is to use the text-to-image cross-attention scores[3] that are already available in the VL model to compute how important each image token is to the text. We keep important image tokens and prune the rest. Since this text-informed pruning also removes image tokens during training, it trains faster[4] than parameter-based pruning approaches like Rao et al.

[3]VL models use cross-attention to perform information fusion between different modalities.

[4]We observe 15%∼20% faster training speed in practice.

(2021).

For each cross-modal layer ℓ where the token reducer is applied, we denote the input text token vectors as $\mathbf{T}$, image token vectors as $\mathbf{V}$, and text-to-image cross-attention scores as $\mathbf{A}$ (computed in the cross-attention layer that already exists in a VL model). We first compute the text-saliency scores $\mathbf{s}$ for every image token:

$$s_v = \frac{1}{|T|} \sum_{|T|}^{t=1} \sum_{H}^{h=1} \mathbf{A}_{tv}^h, \tag{1}$$

where $|T|$ is the number of text tokens, H the number of attention heads, t and v are the indices of text and image tokens. This text-saliency score for the image token is text-informed because each value is summed over all text tokens, and an image token with a bigger text-saliency score means it's attended more by the text and hence is more text-relevant. Next, we keep top-k' image tokens[5] $\mathbf{V}_p$ according to their text-saliency score and discard the remaining image tokens.

4.2 Modality-Aware Merging

Once text-irrelevant image tokens are pruned, the remaining image tokens contain more text-salient information but they might still be redundant. For example, multiple image tokens describe the same person in the Figure 1 image and their representations might be similar (their vector distances are close). For the text modality, the token redundancy still exists due to the self-attention contextualization which progressively creates similar information (Goyal et al., 2020). In practice, text tokens are padded to max length for efficient training and inference, these padding tokens also contribute to redundancy.

In this section, we describe our modality-aware merging approach to eliminate such redundancy. In particular, our method merges semantically similar image tokens $\mathbf{V}_p$ into a single image token and similar text tokens $\mathbf{T}$ into a single text token to further reduce the number of tokens. We specifically merge tokens within each modality, i.e., image tokens are merged with similar image tokens, and text tokens are merged with similar text tokens.

To implement modality-aware merging, we need to identify similar tokens and combine their information in a lightweight way. Existing methods such as k-means clustering (Marin et al., 2021), pooling (Pietruszka et al., 2020; Nawrot et al., 2022), grouping (Xu et al., 2022) or learning-based (Ryoo et al., 2021) cause non-negligible overhead and slow down the VL model computation, instead, we use the bipartite soft matching algorithm (Bolya et al., 2022) to find similar tokens and combine them in parallel.

Here, we explain the bipartite matching approach in more detail. Specifically, the inputs are a set of token vectors $\mathbf{X}$ (can be $\mathbf{V}_p$ or $\mathbf{T}$) and a merge ratio r, we form a bipartite graph by dividing the nodes (tokens) into two disjoint sets (say $\mathbf{E}$ and $\mathbf{O}$) of equal size based on their order (even or odd). Then, for each token in $\mathbf{O}$, we find its most similar token in $\mathbf{E}$, and draw an edge between the token pair (lines in the left figure in Figure 4). We select the top-r' edges[6] based on the similarity and merge their corresponding (most similar) token in $\mathbf{E}$ and $\mathbf{O}$. Figure 4 shows an example of bipartite matching. Since the self-attention in a VL model layer already has computed keys and values for each token to measure similarity, following Bolya et al. (2022), we compute the similarity as the dot product $\mathbf{S}_p^{t_1 t_2} = \mathbf{K}_{t_1} \mathbf{K}_{t_2}$ between the keys of each token vector $\mathbf{X}_i$. We keep the rest non-top-r' tokens in $\mathbf{O}_{rest}$ and unmerged tokens in $\mathbf{E}_{rest}$. We also describe this procedure in Algorithm 1.

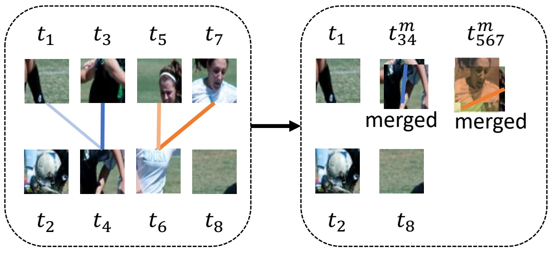

Figure 4: Illustration of merging by bipartite matching. In this example, there are 8 tokens, $\mathbf{E}$ consists of token t_1, t_3, t_5 and t_7, $\mathbf{O}$ has t_2, t_4, t_6, t_8. Assume for t_1, t_3, t_5, t_7 in $\mathbf{E}$, the most similar tokens in $\mathbf{O}$ are t_4, t_4, t_6, t_6 respectively, and $t_3 - t_4, t_7 - t_6, t_5 - t_6$ are the edges (darker and thicker lines mean larger similarity values) with top-r' ($r' = 3$) most similarity, then we merge (t_3, t_4) into one token t_{34}^m, (t_5, t_7, t_6) into one token t_{567}^m, and keep t_1, t_2, t_8, in this case, we reduce three (3/8=37.5%) tokens.

4.3 Training and Inference

Token reducers in PuMer contain no trainable parameters and can be incorporated into off-the-shelf VL models without changing model architectures for both training and inference. PuMer is easy to

[5] $k' = (1 - k)|\mathbf{V}|$ is the number of kept tokens

[6] $r' = r|\mathbf{X}|$ is the number of merge tokens

train and follows the same setup as finetuning original VL models. To reduce the accuracy drop further, we add a knowledge distillation (Hinton et al., 2015) loss. During training and inference, PuMer has three configurable hyperparameters (keep ratio k, merge ratios r, and t for image and text) to control the efficiency versus accuracy trade-offs.

Implementation Details. We set the pruning and merging ratio in the range of 0.1 to 0.5 in 3 or 4 locations in cross-modal layers. The exact values are in Appendix A.1. In Section 6.2, we study the design choices for different reduction ratios and reduction layer locations. More implementation and training details are in Appendix A.1.

5 Evaluation Setup

5.1 Backbone Vision-Language Models

We evaluate PuMer for two different VL models: ViLT (Kim et al., 2021) with 110 million parameters and a state-of-the-art VL model, METER (Dou et al., 2022) with 330 million parameters. We denote PuMer-ViLT and PuMer-METER as PuMer applied for ViLT and METER respectively.

ViLT is a recent efficient VL model that uses BERT (Devlin et al., 2019) embeddings to encode text and a linear layer to project image patches. ViLT then concatenates the text and image tokens and uses a 12-layer Transformer encoder to perform the cross-modal fusion. ViLT is a relatively lightweight model and has 110 million parameters.

METER is a state-of-the-art VL model that uses RoBERTa (Liu et al., 2019) as the text encoder and CLIP (Radford et al., 2021) as the image encoder, and 12 BERT-like cross-attention layers to fuse the text and image modalities. METER is a large model and has 330 million parameters.

5.2 Evaluation Tasks

We evaluate the models on five vision-language language tasks:

Image-Text Retrieval contains two subtasks: image-to-text retrieval (IR) and text-to-image retrieval (TR). We finetune PuMer and evaluate on the Flickr30K (Plummer et al., 2015).

Visual Question Answering (VQAv2) dataset (Goyal et al., 2017) contains over 1 million diverse open-ended questions about images both from the MSCOCO (Lin et al., 2014) and real-world scenes. Answering these questions requires an understanding of vision, language, and commonsense knowledge.

Visual Entailment (VE) (Xie et al., 2019) is a visual inference task that consists of 570K sentence image pairs constructed from the Stanford Natural Language Inference corpus (Bowman et al., 2015) and Flickr30k (Young et al., 2014). The goal is to predict whether the image premise semantically entails the text.

Natural Language for Visual Reasoning (NLVR2) corpora (Suhr et al., 2019) have over 100K examples of linguistically diverse English sentences written by humans and are grounded in pairs of visually complex images. The goal is to predict whether a sentence is true about two input images.

5.3 Baselines

To compare the benefits of PuMer, we additionally evaluate three baselines:

DynamicViT (Rao et al., 2021) designs several prediction modules parameterized by MLPs to predict which image tokens to prune in vision transformers (Dosovitskiy et al., 2020). For a fair comparison, we use the original DynamicViT configurations (pruning layers and ratios) for the ViLT model.

ToMe (Bolya et al., 2022) uses token merging to reduce the number of tokens in vision transformers. We configure ToMe to make sure similar speedup as PuMer and compare their accuracy.

Note that both DynamicViT and ToMe are designed for vision Transformers and work for image modality, therefore they do not distinguish between the image and text tokens. On the contrary, PuMer is a more general token reduction framework that uses text to guide the image pruning and makes merging modality aware.

Smaller Resolution (SmRes): We downsample the input image to smaller resolutions and finetune the VL models. Using smaller input images directly reduces the computation of VL models.

5.4 Evaluation Metrics

Accuracy Metrics. We measure *VQA accuracy* (Goyal et al., 2017) for the VQAv2 dataset and *accuracy* for both the VE and NLVR2 datasets. For text retrieval (TR) and image retrieval (IR) tasks, the accuracy refers to Top1-recall. Unlike previous works (Kim et al., 2021; Dou et al., 2022), where

Model	Datasets	Original Accuracy	PuMer Accuracy	Throughput Increase	Memory Reduction
METER (SoTA)	Flickr30k TR	94.7	93.8 (-0.9)	1.81x	38%
	Flickr30k IR	82.0	81.2 (-0.8)	1.81x	38%
	VQAv2	77.5	76.8 (-0.7)	1.82x	38%
	SNLI-VE	81.1	80.3 (-0.8)	2.07x	43%
	NLVR2	82.7	82.2 (-0.5)	1.79x	38%
ViLT	Flickr30k TR	78.2	77.6 (-0.6)	1.78x	46%
	Flickr30k IR	60.2	59.6 (-0.7)	1.78x	46%
	VQAv2	69.5	68.9 (-0.6)	1.76x	45%
	SNLI-VE	76.0	75.6 (-0.4)	2.01x	51%
	NLVR2	75.5	74.9 (-0.6)	1.74x	45%

Table 1: Performance and inference efficiency comparison between the original fine-tuned vs PuMer fine-tuned models for the ViLT and METER over four downstream visual reasoning tasks.

their models are trained on the combined training and validation sets, our focus is not to obtain state-of-the-art results, so we train the two VL models on the training set and report the results on the test set. All the accuracy numbers are average values across 3 runs.

Efficiency Metrics. We measure the actual inference throughput (examples per second) of the VL models on the GPU hardware and compare them to the original finetuned models, and we report the *throughput increase*. We also measure the peak memory consumed during the model inference phase and report *memory reduction* ratio compared to the original finetuned models. These two runtime metrics reflect actual efficiency and are found to be more accurate to compare resource consumption instead of using the FLOPs complexity metric (Graham et al., 2021). For comparison purposes, we include the FLOPs comparison in the appendix Appendix A.2.

For inference throughput measurements, we increase the batch size until the model gets out of GPU memory, and run the inference with the batch size that gives the biggest throughput for 30 seconds on a single GPU. For inference memory footprint, we use the same batch size for the original VL model and PuMer version and report the peak memory difference. For ViLT models, we use GTX 1080 Ti GPU and start the batch size from 32 with a step of 8; for METER models, we use an A40 GPU and start the batch size from 16 with a step of 8.

6 Experimental Results

6.1 Main Results

PuMer is faster and remains accurate. Table 1 shows the main results comparing performance, inference speed, and memory reduction of PuMer versus the original models. Overall, we observe over **1.7x $\sim$ 2x speedup** in inference throughput and over **35% $\sim$ 51% reduction** in memory footprint for both ViLT and METER models on the VL tasks. Importantly, the task performance of PuMer remains competitive compared to the original finetuned VL models with only $<$1% drop in accuracy.

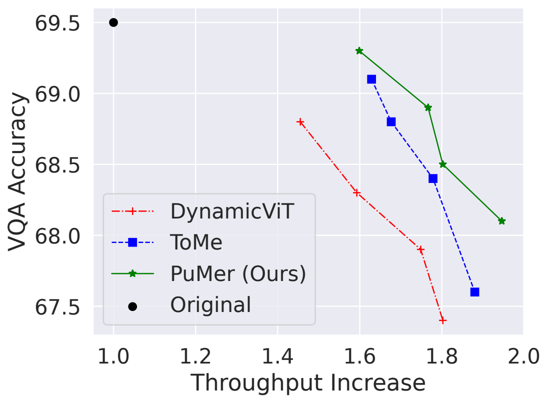

Figure 5: Comparing PuMer with DynamicViT and ToMe for the ViLT model on the VQAv2 dataset. Setting different pruning and merging ratios for DynamicViT and ToMe gives different inference throughput and accuracy numbers. Right and top lines are better trade-offs.

PuMer is more accurate and faster than previous token reduction methods. Figure 5 presents the accuracy versus inference throughput increase trade-offs for PuMer, DynamicViT and ToMe applied to the ViLT model on the VQAv2 dataset. Given a similar throughput increase (like 1.8x), PuMer has the best accuracy compared to DynamicViT and ToMe. Similarly, for a given accuracy drop constraint (like $< 1\%$), PuMer provides a bigger throughput increase.

Model	Image Resolution	VQAv2 Accuracy	Throughput Increase	Memory Reduction
Resolution	192x192	74.3 (-3.2)	4.23x	75%
	224x224	75.2 (-2.3)	3.48x	66%
	256x256	76.1 (-1.4)	2.67x	54%
	320x320	77.0 (-0.5)	1.62x	37%
PuMer	320x320	76.3 (-1.2)	2.86x	59%
PuMer	384x384	76.8 (-0.7)	1.82x	38%
Original	384x384	77.5	1x	0%

Table 2: Performance and inference efficiency comparison between the smaller resolution baselines and PuMer for the METER model on the VQAv2 test set.

PuMer provides larger efficiency gains over smaller resolution baselines. Table 2 shows the results for the METER model on the VQAv2 dataset when comparing PuMer with downsampling the input image to smaller resolutions. Using smaller resolution input images improves the inference throughput and reduces memory footprint but comes with larger accuracy drops. The closest resolution is 320x320 which is slightly more (0.2%) accurate than PuMer, but it has 20% lower inference throughput. Meanwhile, PuMer is orthogonal to downsampling strategies, and applying PuMer to smaller images could provide additional efficiency gains; for input image resolution 320x320, PuMer improves METER throughput by 1.76x with a 0.7% accuracy drop[7] (see the 3rd row numbers in Table 2).

6.2 Ablation Study

Model	VQA Accuracy	Throughput Increase
ViLT	69.5	1x
PuMer-ViLT	68.9 (-0.6)	1.76x
w/o text-informed image pruning	69.2 (-0.3)	1.52x
w/o modality-aware merging	69.1 (-0.4)	1.46x
w/o distillation	68.6 (-0.9)	1.76x

Table 3: Ablation analysis for each component in PuMer on the VQAv2 dataset for ViLT model.

Effectiveness of PuMer Components. To show how each component in PuMer affects the VL task accuracy and model inference efficiency, we ablate the three components — text-informed image pruning, modality-aware merging and distillation — in Table 3. Applying text-informed image pruning or modality-aware merging individually has shown improvements in model inference throughput with smaller accuracy loss. But stacking the two techniques together provides bigger inference efficiency without losing much task performance. Without knowledge distillation, PuMer is still accurate and fast and adding it further reduces the performance gap.

[7] 1.76=2.86/1.62, 0.7=77.0-76.3

Token Reduction Design Choices. Given a 12-layer VL cross-modal encoder like ViLT, many combinations of reduction locations and ratios achieve similar inference speedups. Reducing tokens at earlier layers with lower ratios has similar computation efficiency to pruning at later layers with higher ratios. For comparing the accuracy with different numbers of reduction layers, we control the inference throughput to be similar to PuMer by selecting the pruning and merging ratios and locations. Table 4 shows cascaded reduction at 4 layers (2th, 4th, 6th, 8th) has higher accuracy and speedups.

The ratios row in Table 4 shows reducing (via pruning or merging) more tokens leads to a bigger throughput increase but has a significant ($>1\%$) accuracy drop while reducing fewer tokens is more accurate but causes lower throughput. As shown in the locations row, we find that reducing tokens in the earlier layers leads to bigger throughput but drops accuracy by 1.8%, while reducing tokens in the later layers is slightly more accurate but provides fewer benefits in throughput. Overall, for ViLT on the SNLI-VE task, we choose a 4-layer cascaded token reduction strategy with a pruning ratio of 0.1 and merging ratio of 0.3 and 0.2 for image and text respectively, and scatter the reduction locations more evenly to balance accuracy and speed trade-offs.

7 Conclusion

Large vision language models have been effective at visual reasoning tasks due to their complex cross-modal interactions between the text and image tokens. These cross-modal interactions are computationally expensive because all image and text tokens are processed in many layers. We introduce a token reduction framework — PuMer that uses text-informed image pruning and modality-aware merging techniques to effectively reduce the image and text tokens inside cross-modal layers. PuMer progressively removes the redundant image and text information and makes VL models run faster with minimal task performance drop. PuMer is

Choice	Reduction Layers	Prune Ratio	Image Merge Ratio	Text Merge Ratio	VE Accuracy	Throughput Increase
ratios	2,5,8	0.1	0.3	0.2	75.8 (-0.2)	1.77x
	2,5,8	0.3	0.3	0.2	74.7 (-1.3)	2.04x
	2,5,8	0.1	0.3	0.5	74.9 (-1.1)	1.89x
	2,5,8	0.1	0.5	0.2	73.8 (-2.1)	2.12x
# of layers	2	0.1	0.3	0.2	75.9 (-0.15)	1.43x
	2,4	0.1	0.3	0.2	75.8 (-0.2)	1.69x
	2,4,6	0.1	0.3	0.2	75.7 (-0.3)	1.80x
locations	2,3,4	0.2	0.2	0.2	74.2 (-1.8)	2.03x
	7,8,9	0.2	0.2	0.2	75.9 (-0.1)	1.31x
PuMer (Ours)	2,4,6,8	0.1	0.3	0.2	75.6 (-0.4)	2.01x
ViLT	-	-	-	-	76.0	1.00x

Table 4: Design choices analysis of prune and merge ratios, # of reduction layers, and reduction locations for the ViLT model on SNLI-VE task.

easy to train and speeds up both training and inference of vision and language models across diverse downstream visual reasoning tasks.

Acknowledgements

This research was supported partly by NSF IIS-2044660, an Allen Investigator Distinguished award. We thank the anonymous reviewers and the members of the UW NLP group for their comments and feedback on this paper.

8 Limitations

Our method does not apply to VL models where the cross-modal encoder layers are relatively lightweight. For example, the vision encoder is much more computationally expensive than the cross-modal encoder for VL models like ALBEF (Li et al., 2021) and X-VLM (Zeng et al., 2021), therefore, the end to end inference speed improvement is marginal. Reducing the image tokens inside the vision encoder could further improve the model efficiency, we leave this exploration to future work.

References

TPrune: Efficient Transformer Pruning for Mobile Devices: ACM Transactions on Cyber-Physical Systems: Vol 5, No 3.

2022. Deepspeed.

2022. huggingface/accelerate.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token Merging: Your ViT But Faster. ArXiv:2210.09461 [cs].

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Qingqing Cao, Prerna Khanna, Nicholas D. Lane, and Aruna Balasubramanian. 2022. MobiVQA: Efficient On-Device Visual Question Answering. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):44:1–44:23.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Zi-Yi Dou, Yichong Xu, Zhe Gan, Jianfeng Wang, Shuohang Wang, Lijuan Wang, Chenguang Zhu, Pengchuan Zhang, Lu Yuan, Nanyun Peng, Zicheng Liu, and Michael Zeng. 2022. An empirical study of training end-to-end vision-and-language transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18166–18176.

Zhiyuan Fang, Jianfeng Wang, Xiaowei Hu, Lijuan Wang, Yezhou Yang, and Zicheng Liu. 2021. Compressing Visual-Linguistic Model via Knowledge Distillation. pages 1428–1438.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3690–3699. PMLR. ISSN: 2640-3498.

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. pages 6904–6913.

Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. 2021. LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference. pages 12259–12269.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. ArXiv:1503.02531 [cs, stat].

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5583–5594. PMLR. ISSN: 2640-3498.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Mengshu Sun, Wei Niu, Xuan Shen, Geng Yuan, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. 2022. SPViT: Enabling Faster Vision Transformers via Soft Token Pruning. ArXiv:2112.13890 [cs].

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block Pruning For Faster Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. In *Advances in neural information processing systems*, volume 34, pages 9694–9705. Curran Associates, Inc.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2021. EViT: Expediting Vision Transformers via Token Reorganizations.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 740–755, Cham. Springer International Publishing.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a Self-distilling BERT with Adaptive Inference Time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. Number: arXiv:1907.11692 arXiv:1907.11692 [cs].

Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. 2021. Token Pooling in Vision Transformers. ArXiv:2110.03860 [cs].

Piotr Nawrot, Jan Chorowski, Adrian Łańcucki, and Edoardo M. Ponti. 2022. Efficient Transformers with Dynamic Token Pooling. ArXiv:2211.09761 [cs] version: 1.

Michał Pietruszka, Łukasz Borchmann, and Filip Graliński. 2020. Sparsifying Transformer Models with Differentiable Representation Pooling. *arXiv:2009.05169 [cs]*. ArXiv: 2009.05169.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649. ISSN: 2380-7504.

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. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR. ISSN: 2640-3498.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *arXiv:2106.02034 [cs]*. ArXiv: 2106.02034.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. TokenLearner: What Can 8 Learned Tokens Do for Images and Videos?

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The Right Tool for the Job: Matching Model and Instance Complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A Corpus for

Reasoning about Natural Language Grounded in Photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jianfeng Wang, Xiaowei Hu, Pengchuan Zhang, Xiujun Li, Lijuan Wang, Lei Zhang, Jianfeng Gao, and Zicheng Liu. 2020. MiniVLM: A Smaller and Faster Vision-Language Model. *arXiv:2012.06946 [cs]*. ArXiv: 2012.06946.

Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. OFA: Unifying Architectures, Tasks, and Modalities Through a Simple Sequence-to-Sequence Learning Framework. Technical Report arXiv:2202.03052, arXiv. ArXiv:2202.03052 [cs] version: 2 type: article.

Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. 2021. SimVLM: Simple Visual Language Model Pretraining with Weak Supervision.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. 2019. Visual Entailment Task for Visually-Grounded Language Learning. Technical Report arXiv:1811.10582, arXiv. ArXiv:1811.10582 [cs] type: article.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early Exiting BERT for Efficient Document Ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88, Online. Association for Computational Linguistics.

Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. 2022. GroupViT: Semantic Segmentation Emerges From Text Supervision. pages 18134–18144.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10818.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78. Place: Cambridge, MA Publisher: MIT Press.

Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. 2022. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2022.

Hao Yu and Jianxin Wu. 2021. A Unified Pruning Framework for Vision Transformers. *arXiv:2111.15127 [cs]*. ArXiv: 2111.15127.

Yan Zeng, Xinsong Zhang, and Hang Li. 2021. Multi-Grained Vision Language Pre-Training: Aligning Texts with Visual Concepts.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. VinVL: Revisiting Visual Representations in Vision-Language Models. pages 5579–5588.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In *Advances in neural information processing systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

A Appendix

A.1 PuMer Details

Implementation. We use the Transformers (Wolf et al., 2020) and Accelerate (Hug, 2022) with DeepSpeed (Dee, 2022) library to implement the training tasks. We conduct training jobs on 4 Nvidia A100 GPUs. For both ViLT and METER model, we first follow the training hyperparameters in their original papers and finetune the pretrained model to obtain task-specific models. These models are used as baselines for measuring accuracy drop and also used as the teacher model for PuMer distillation. For baseline VL models, we finetune both METER and ViLT models on the studied VL tasks for 10 epochs. For PuMer, we finetune 20 epochs using early stopping with a penitence of 5 (the accuracy won't improve after 5 epochs).

We list all training hyperparameters in Table 5.

METER	Retrieval	VQAv2	NLVR2	SNLI-VE
cross-modal lr	2.5e-5	2.5e-5	5e-5	1e-5
classifier lr	2.5e-5	2.5e-4	1e-4	2e-5
batch size per gpu	32	32	16	32
image size	384	384	288	384
patch size	16	16	16	16
ViLT	**Retrieval**	**VQAv2**	**NLVR2**	**SNLI-VE**
cross-modal lr	1e-4	1e-4	1e-4	1e-4
classifier lr	1e-4	1e-3	1e-4	1e-3
batch size per gpu	32	64	32	64
image size	384	384	384	384
patch size	32	32	32	32

Table 5: Hyperparameters for finetuning PuMer and original VL models.

We list the default reduction layers and ratios for different VL tasks in Table 6.

METER	VQAv2	NLVR2	SNLI-VE	Retrieval
Reduction Layers	0,2,4,6	2,4,6	0,2,4,6	2,4,6
Prune Ratio	0.2	0.3	0.3	0.2
Image Merge Ratio	0.2	0.5	0.5	0.5
Text Merge Ratio	0.2	0.2	0.2	0.2
ViLT	**VQAv2**	**NLVR2**	**SNLI-VE**	**Retrieval**
Reduction Layers	2,5,8	2,5,8	2,4,6,8	2,5,8
Prune Ratio	0.1	0.1	0.1	0.1
Image Merge Ratio	0.3	0.3	0.3	0.3
Text Merge Ratio	0.2	0.2	0.2	0.2

Table 6: Reduction layers and ratios for PuMer-METER and PuMer-ViLT on the VL tasks.

Model	Datasets	Original	PuMer	Speedup
METER	VQAv2	92	64.7	1.42x
	SNLI-VE	92	59	1.56x
	NLVR2	184	131	1.40x
ViLT	VQAv2	16	8.7	1.84x
	SNLI-VE	16	7.7	2.08x
	NLVR2	32	17.4	1.84x

Table 7: GFLOPs comparison between PuMer and original VL models for METER and ViLT.

A.2 Model Inference FLOPs Comparison

We measure FLOPs of both PuMer and the original model for METER and ViLT using the fvcore tool[8]. The results are shown in Table 7.

[8]https://github.com/facebookresearch/fvcore/blob/main/docs/flop_count.md

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 8

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

section 5 and 6

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
section 5 and appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
section 5 and appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
appendix

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.