
Zineng Tang*  Jaemin Cho*  Jie Lei  Mohit Bansal
UNC Chapel Hill
{terran, jmincho, jielei, mbansal}@cs.unc.edu

Abstract

We present PERCEIVER-VL, a vision-and-language framework that efficiently handles high-dimensional multi-modal inputs such as long videos and text. Powered by the iterative latent-cross-attention of Perceiver, our framework scales with linear complexity, in contrast to the quadratic complexity of self-attention used in many state-of-the-art transformer-based models. To further improve the efficiency of our framework, we also study applying LayerDrop on cross-attention layers and introduce a mixed-stream architecture for cross-modal retrieval. We evaluate PERCEIVER-VL on diverse video-text and image-text benchmarks, where PERCEIVER-VL achieves the lowest GFLOPs and latency, while maintaining competitive performance. In addition, we also provide comprehensive analyses over various aspects of our framework, including pretraining data, scalability of latent size and input size, dropping cross-attention layers at inference to reduce latency, modality aggregation strategy, positional encoding, and weight initialization strategy.

1. Introduction

During the past several years, there has been increased interest in vision-and-language learning. Many recent models [69, 52, 9, 68, 45, 72, 41] adopt the transformer [74] architecture to encode vision-and-language inputs. These methods have improved the performance of various tasks, such as text-based image/video retrieval [8, 57, 80, 62, 7, 2, 43, 40] and visual question answering [3, 29, 79, 42, 82]. However, the transformer is based on the self-attention module [74] with a quadratic computational cost in relation to its input length. This makes it difficult for models to process high-dimensional data, such as long videos.

To this end, we propose PERCEIVER-VL, an end-to-end vision-and-language architecture that efficiently handles high-dimensional multi-modal inputs. PERCEIVER-VL is built on the iterative latent cross-attention of the recently proposed PERCEIVER [28, 27]. Concretely, we map a multi-modal input array of size $M$ to a latent array of size $N$ with cross-attention. This changes the computational complexity of the attention modules from $O(M^2)$ to $O(NM)$. Since vision-and-language models often handle very long input arrays (e.g., $M > 1000$), this greatly improves the efficiency for vision-and-language tasks. To further enhance the efficiency of our framework, we also study reducing the number of cross-attention layers based on LayerDrop [17] and using a mixed-stream architecture for cross-modal retrieval tasks. By varying the number of cross-layer attention layers that take the most computation, we allow users to flexibly control the latency at inference. The mixed-stream architecture combines the widely used single-stream and multi-stream architectures and improves the retrieval performance of the multi-stream architecture with minimum increase in latency.

We evaluate PERCEIVER-VL on various video-text (MSRVTT, DiDeMo, LSMDC, ActivityNet, TGIF-QA, MSRVTT-QA) and image-text (Flickr30k, VQAv2, NLVR2) tasks. Overall, PERCEIVER-VL achieves performance competitive to recent vision-and-language models, while maintaining significantly higher efficiency with the lowest GFLOPs and latency. We demonstrate that PERCEIVER-VL scales more efficiently than the transformer-based architecture with respect to video length and frame size. In addition, we show that our method also allows for flexible adoptions to further improve its efficiency: (1) Decreasing the size of latent array during finetuning reduces the computation significantly, with only minimal accuracy drop; (2) Mixed-stream architecture achieves a reasonable accuracy-latency trade-off: higher accuracy than multi-stream and lower latency than...
single-stream for text-to-video retrieval. (3) We apply LayerDrop during training. This allows users to control the latency by reducing the number of cross-attention layers at inference, again with only minimal accuracy drop. Moreover, LayerDrop also acts as a regularizer; training with LayerDrop improves model performance. Lastly, we conduct a comprehensive ablation study, including weight initialization, pretraining dataset, and comparing modality aggregation methods. We find it helpful to initialize the parameters from ViT and CLIP and pretrain jointly on video-text and image-text pairs. We do not find a meaningful difference in whether to combine the two input modalities in joint or separate attention modules, and whether to use learned or Fourier positional encoding for the latent array.

Our contributions can be summarized as: (1) We propose PERCEIVER-VL, an efficient vision-and-language framework with linear scalability, on-demand depth reduction, and mixed-stream retrieval architecture. (2) We demonstrate that our framework achieves significantly higher efficiency than recent transformer-based models on various vision-and-language benchmarks, with overall competitive performance. (3) We provide a comprehensive analysis of the efficiency, architectural components, and training strategies of our framework. We hope that our research allows the community to use the highly efficient framework for diverse vision-and-language tasks and inspires future research.

2. Related Work

2.1. Efficient Transformers

Many research works have proposed to reduce the quadratic computation complexity of self-attention in transformers based on different methods, including hashing, sparse attention, kernel trick, low-rank key/value projection, blockwise attention, past memory compression, and inducing point methods. Unlike these methods, PERCEIVER proposes using iterative cross-attention to map an input array to a smaller latent array and apply self-attention to the latent array, which makes the computation scale linearly. PERCEIVER-Io adds a decoder to PERCEIVER to allow the model to tackle various downstream tasks with structured prediction.

To our knowledge, the iterative cross-attention of PERCEIVER for multi-modal inputs has only been studied on audio-video autoencoding task. In this work, we present PERCEIVER-VL, which extends the PERCEIVER framework in vision-and-language domain. We also evaluate PERCEIVER-VL on diverse video-text and image-text benchmarks and conduct extensive experiments to analyze its efficiency. In addition, we introduce new techniques including cross-attention drop and mixed-stream architecture for cross-modal retrieval.

2.2. Vision-and-Language Pretraining

Large-scale pretraining of transformers has achieved huge success in natural language processing. Following this success, image-text multi-modal transformers have achieved improvements on various vision-and-language tasks. Such models take both visual and textual inputs and are pretrained on large image-text/video-text pairs, with multi-modal masked language modeling and vision-text matching objectives under a standard transformer architecture. One prominent issue with such models is that they are hard to scale because of the quadratic computation cost of a standard transformer. In this work, we propose a new vision-and-language pretraining framework that scales more efficiently than the transformer-based frameworks mentioned above.

3. Perceiver-VL

PERCEIVER-VL architecture consists of an input array, a latent array, and an encoder-decoder network. In the following, we explain the details of each component and how PERCEIVER-VL processes high-dimensional vision-and-language data efficiently. In Fig. 1 we illustrate the PERCEIVER-VL architecture.

3.1. Vision and Language Embedding

We extend the single-modal input array of PERCEIVER to the vision-and-language domain, creating the input array \( c \) as a concatenation of visual and text embeddings. The embeddings are created as the sum of (1) modality embedding; (2) temporal embedding; (3) positional embedding; and (4) patch/token embedding. Modality embedding is a learned embedding of a binary modality indicator \( c \in \{V, T\} \). Temporal embedding is a learned embedding of input video frames \( \{1 \cdots L^V\} \), where \( L^V \) is the frame length. Note that temporal embedding is only used for videos; we do not use temporal embedding for images or text. Positional embedding is a learned embedding of 2D patches \( \{1 \cdots L^P\} \) for image/video or token indices for text \( \{1 \cdots L^T\} \), where \( L^P \) is the number of patches for images, and \( L^T \) is the number of text tokens. Patch embedding is learned with a linear projection of non-overlapping image/video input patches (e.g., 32 \( \times \) 32 pixels) \([16]\). We treat an image as a single-frame video so that our model can flexibly process image and video input with a single architecture. Token embedding is a learned embedding of text tokens.

3.2. Iterative Mapping to Low-Dim Latent Space

Following \([28]\), PERCEIVER-VL tames the quadratic computation complexity of self-attentions over high-
dimensional inputs, by introducing a latent array $z$ of size $N$ (see ‘Latent Array’ in Fig. 1) that aggregates information from an input array $c$ of size $M$ via iterative cross-attentions (see ‘Cross-Att’ in Fig. 1). PERCEIVER-VL encoder consists of $k$ attention blocks, each of which is a stack of a cross-attention and $l$ self-attentions over a latent array $z$, which results in the computational complexity of $O(kMN + klN^2)$. In comparison, a standard transformer encoder with the same number of self-attention modules has a computational complexity of $O(klM^2)$. Since in vision-and-language tasks where the input size $M$ is larger than the latent array size $N$, the change from quadratic to linear computational complexity w.r.t. $M$ can greatly increase efficiency (see Fig. 4). To disambiguate the latent dimensions, we add a position embedding to the latent array $z$. We add the learned positional embedding $P_{pos}$ for each latent dimension. The choice of learned position encoding is based on simplicity; different from the findings from the single-modality experiments of [28], we did not find the gain from using Fourier feature position encodings [66, 54, 28], as shown in the appendix.

3.3. LayerDrop on Cross-Attention for Reducing Depth on Demand

It is the cross-attention layers that take the highest computation in the attention blocks. Therefore, to further improve the efficiency of PERCEIVER-VL, we apply LayerDrop [17] to cross-attention layers, which allows users to control the latency by changing the number of cross-attention layers during inference. Concretely, we apply dropout [17] to each cross-attention layer with probability $p^{LD}$ during pretraining (see ‘LayerDrop’ in Fig. 1). Note that we do not apply LayerDrop to the first cross-attention layer, to ensure that the model always receives the signal from input. We study the effect of different $p^{LD}$ and the effect of varying the number of cross-attention layers during inference (see Sec. 5.2.4 for details).

3.4. Structured Decoding with Cross-Attention and Query Array

To adapt PERCEIVER-VL to different vision-and-language tasks with structured output space, we give a query array $q$ of arbitrary length $Q$ (see ‘Query Array’ in Fig. 1), to decoder cross-attention and apply a task-specific head (a fully-connected layer) to the cross-attention output. We use a decoder with a single cross-attention [27]. For multi-task learning, we simply concatenate the query array for different tasks. In the following, we describe decoder queries for two vision-and-language pretraining objectives. See appendix for the query constructions for downstream tasks.

3.4.1 Vision-and-Language Pretraining

We use two popular objectives in vision-and-language domain for PERCEIVER-VL pretraining: Vision-Text Matching (VTM) and Masked Language Modeling (MLM). To create the final query for the VTM and MLM tasks, we concatenate the queries for the two tasks, as illustrated in Fig. 1.
Vision-Text Matching (VTM) asks a model to distinguish whether a given pair of visual input (image or video) and text input matches or not. We create an unmatched pair by replacing its visual input with a randomly selected negative one, with 50% probability. We create the VTM query with a learnable embedding ($Q = 1$), illustrated as [CLS] in Fig.1. We apply a linear VTM head to the corresponding decoder output and perform binary classification.

Masked Language Modeling (MLM) asks a model to infer masked text inputs in a given context. Following [14], we randomly mask 15% of the input text tokens. We create the MLM query by adding a positional embedding and a mask embedding ($Q = L^T$). The mask embedding is a learned embedding of a binary indicator variable, where 1 indicates the masked text token. Note that we do not feed the token embeddings to the decoder, i.e., we do not provide the text input. In doing so, we encourage the encoder output $z_e$ to have a compact representation that contains enough information for MLM. We apply a linear MLM head to the corresponding decoder output and use cross-entropy loss.

3.5. Mixed-Stream Architecture for Cross-Modal Retrieval

In Fig. 2 we show two widely used architectures used in cross-modal retrieval tasks: (a) single-stream [14] [33] and (b) multi-stream [59] [4]. The single-stream architecture models the multi-modal similarity score $s^{VL}$ with multiple layers of encoder, whereas the multi-stream encoder models the multi-modal similarity simply with a dot product between single-modality encodings $z_e^V$, $z_e^L$ from separate encoders. In many real-world applications, multi-stream architectures are widely used for retrieval for their high efficiency. This is because multi-stream architectures allow us to cache pre-computed visual encodings $z_e^V$ and simply compute dot products with text query $z_e^L$ during inference. In contrast, single-stream architectures tend to achieve higher accuracy but require expensive computation, where a joint input array goes through multiple encoder layers. We propose to use a ‘mixed-stream’ architecture (Fig. 2 (c)) that takes the best of both worlds. Note that a similar idea has been proposed for text retrieval [26]. As shown in Fig. 6 our mixed-stream architecture achieves a good accuracy-latency tradeoff.

4. Experiment Setup

We pretrain PERCEIVER-VL on a combination of video-text and image-text datasets, then finetune it on a set of downstream benchmarks for evaluation. Below, we explain the details of our training and evaluation setup.

4.1. Architecture Details

Model Details. For the PERCEIVER-VL encoder, we use $k = 3$ blocks of 1 cross-attention and $l = 3$ self-attentions, totaling 3 cross-attention layers and 12 self-attention layers. The decoder has 1 cross-attention layer. We follow BERT Base [14] and ViT-B/32 [16] to use a hidden size of 768 and 12 attention heads. We follow ViT-B/32 [16] to use image (and video frame) size 384 and patch size 32. We use PyTorch [56] to implement our model in experiments.

LayerDrop on Cross-Attention. We set a probability of $p^{LD} = 0.5$ to apply dropout to the cross-attention layers during vision-and-language pretraining. Note that we do not apply dropout to the first cross-attention, to ensure that input signal always goes into the latent array. We analyze the effect of using LayerDrop during pretraining, finetuning, and inference, as shown in Table 5.

Modality Aggregation. By default, we map the multi-modal inputs to the latent space by creating an input array based on the concatenation of visual and textual inputs (namely Joint encoding). We also explore two other ways of combining the two modalities: encoding each modality serially with separate cross-attentions, then applying self-attentions (Separate encoding); encoding each modality serially with separate cross-attentions with self-attentions between them (Separate+ encoding). We illustrate these aggregation strategies in Fig. 3. In our ablation study, we found that the three methods perform comparably, where the Joint encoding has the least computation. Therefore, we adopt the Joint encoding as our default modality aggregation method. See appendix for the detailed experiments.

4.2. Weight Initialization from Vision Transformers

To compare with recent methods that use pretrained visual backbone models, we experiment with initializing weights of PERCEIVER-VL with two popular models: ViT-B/32 [16] and CLIP (ViT-B/16) [59]. As these models have 12 self-attention layers, we insert 3 cross-attention layers after every 4 self-attention layers (before 1st/5th/9th).

Two-stage training. Since transformer models do not have cross-attention layers, the cross-attention weights could not warm-start. To stabilize training, after initializing the PERCEIVER-VL weights from CLIP parameters, we first train only the cross-attention layers, while freezing all other modules. After initial convergence (e.g., 1 epoch in MSRVTT [80]), we train the whole model jointly. In our experiment, this two-stage training strategy achieves better weight transfer than single-stage training (see appendix).
4.3. Pretraining Setups

**Video-Text and Image-Text Datasets.** We follow Frozen-in-Time [4] to pretrain PERCEIVER-VL on both video-text and image-text data. For image-text dataset, we use Conceptual Captions (CC) [64], which consists of 3M image-text pairs. For video-text dataset, we use Webvid [4], which consists of 2.5M video-text pairs, with an average video length of 18 seconds.

**Training Details.** We use Adam optimizer [34] with a learning rate 1e-5 and weight decay 0.001. We set 200k training steps for joint Video-Text & Image-Text pretraining. We use the batch size of 4096 with gradient accumulation on 4 RTX 2080Ti GPUs for 14 days.

4.4. Downstream Tasks

After pretraining, we evaluate PERCEIVER-VL on various vision-and-language benchmarks, covering cross-modal retrieval and visual question answering for both video-text and image-text datasets.

4.4.1 Video-Text Tasks

For video retrieval, we use MSRVTT [80], LSMDC [62], DiDeMo [2], ActivityNet Captions [36]. For video question answering, we use TGIF-QA [29] and MSRVTT-QA.

**Dataset Details.** MSRVTT contains 10K web video clips, with 20 captions for each clip. LSMDC contains 118,081 short clips from 202 movies with each clip containing one caption. DiDeMo contains 10,464 videos with 40,543 temporally localized sentences. ActivityNet Captions have 20k videos with 3.65 temporally localized sentences per video on average, resulting in 100k sentences in total. Videos have an average duration of 180 seconds. For DiDeMo and ActivityNet Captions, we follow previous work [50, 41] to use paragraph-to-video retrieval, where we concatenate all sentences from the same video as a single text query for retrieval. TGIF-QA contains 165K QA pairs from 72K animated GIF videos. We follow [41] to evaluate our model on three TGIF-QA tasks: action, transition, and frame. MSRVTT-QA contains 10K videos with 243K open-ended questions collected from MSRVTT videos.

**Training Details.** We use Adam optimizer [34] with a learning rate 1e-5 and weight decay 0.001. We use 16
frames for ActivityNet Captions and 8 frames for other tasks. We use frame size $384 \times 384$, and maximum text length 40 for all tasks.

### 4.4.2 Image-Text Tasks

For image retrieval, we use Flickr30k [57]. For visual question answering, we use VQAv2 [3] and NLVR2 [67].

**Dataset Details.** VQAv2 contains 204,721 images from COCO [49], with a minimum of 3 questions per image and 10 grounded answers. NLVR2 contains 107,292 examples of sentences grounded in image pairs. Flickr30k dataset has 31,000 images collected from Flickr each with 5 sentences.

**Training Details.** We use Adam optimizer with a learning rate of $1e^{-4}$ and weight decay of 0.001. We use image size $384 \times 384$ and maximum text length 40 for all tasks.

## 5. Results and Analysis

We first compare PERCEIVER-VL with recent methods in video-text / image-text benchmarks, where it achieves the highest efficiency, while maintaining competitive performance (Sec. 5.1). Then we analyze the efficiency of PERCEIVER-VL in detail (Sec. 5.2). In appendix, we also present ablation studies of different architectural components and training strategies for PERCEIVER-VL.

### 5.1. Comparison to State-of-the-Art

In Table 1 we compare PERCEIVER-VL with the state-of-the-art video-text models on 4 text-to-video retrieval (MSRVTT, DiDeMo, LSMDC, ActivityNet) and 2 video question answering (TGIF, MSRVTT-QA) benchmarks. In Table 2 we compare PERCEIVER-VL with state-of-the-art image-text models on text-to-image retrieval (Flickr30k) and 2 visual question answering (VQAv2, NLVR2) benchmarks. The closest baseline of our model is Frozen-in-Time [4] for video inputs of different scales, by varying the number of frames (left) and the frame size (right). All three models have 12 self-attention layers with hidden size 768. Powered by efficient cross-attention-based encoding, PERCEIVER-VL achieves the lowest R@1 (26.0), with the shortest inference time. Our mixed-stream architecture achieves a good accuracy-latency tradeoff, with R@1 (27.2), but also takes the longest inference time. The multi-stream architecture achieves the highest R@1 (28.0), but also takes the longest inference time. The multi-stream architecture achieves the lowest R@1 (26.0), with the shortest inference time. Our mixed-stream architecture achieves a good accuracy-latency tradeoff, with R@1 (26.8) close to single-stream architecture, while running significantly faster.

### 5.2. Efficiency Analysis

#### 5.2.1 Scaling Input Array

In Fig. 4 we compare the computations of PERCEIVER-VL, ViLT-B/32 [33], and Frozen-in-Time [4] for video inputs of different scales, by varying the number of frames (left) and the frame size (right). All three models have 12 self-attention layers with hidden size 768. Powered by efficient cross-attention-based encoding, PERCEIVER-VL shows a remarkably better scalability (lower GFLOPs) than ViLT-B/32 and Frozen-in-Time in both plots.

#### 5.2.2 Scaling Latent Array

We study the effect of varying the latent array size $N$ to explore whether we can further improve the efficiency of PERCEIVER-VL. In Fig. 5, we show the effect of varying the latent array sizes during finetuning in terms of computation and downstream performance on MSRVTT. We use $N=128$ during pretraining. When scaling up or down the latent array for a pretrained model, we simply initialize a new latent array where we empirically find that it gives similar performance compared to interpolating the pretrained latent array. We can see that the GFLOPs scales linearly with $N$, while the retrieval performance remains reasonably well in three different pretraining setups (e.g., CC+Webvid PT: 24.0 $\rightarrow$ 24.6 $\rightarrow$ 26.8 $\rightarrow$ 27.1 with latent array size: 32 $\rightarrow$ 64 $\rightarrow$ 128 $\rightarrow$ 256).

#### 5.2.3 Mixed-Stream Architecture for Retrieval

In Fig. 6 we compare different retrieval architecture variants discussed in Sec. 3.5 and Fig. 2 in terms of accuracy and inference time on MSRVTT val split. The single-stream architecture achieves the highest R@1 (27.2), but also takes the longest inference time. The multi-stream architecture achieves the lowest R@1 (26.0), with the shortest inference time. Our mixed-stream architecture achieves a good accuracy-latency tradeoff, with R@1 (26.8) close to single-stream architecture, while running significantly faster.

#### 5.2.4 LayerDrop to Encoder Cross-Attentions

In Table 3 we analyze the effect of applying LayerDrop (LD) [17] to cross-attention layers in encoder, as discussed in Sec. 3.3 on MSRVTT retrieval. We use the mixed-stream architecture as the default setting. First, we observe that LD
Table 1. Finetuning performance on text-to-video retrieval and video question answering benchmarks. We report $R@1$ for text-to-video image or video source datasets for Pretraining Datasets accuracy. For Flickr30k, we show text-to-image retrieval $R@1$ (see appendix for $R@5/R@10$). Note that for brevity, we only show the models that use input modalities other than video and text (i.e., audio) and 2) the models that use CLIP visual encoder [59] (the cross-attention layers of Perceiver-VL cannot be initialized with CLIP parameters and trained from scratch; see the discussion in Sec. 5.1). $MSR=MSRVTT$, $DDM=DiDeMo$, $LSM=LSMDC$, $ACT=ActivityNet$, $TGF=TGIF$-QA. $N=128$ means latent size $N=128$. Multi and mixed mean multi-stream and mixed-stream respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pretraining Datasets</th>
<th>Visual Backbone</th>
<th>Retrieval ↑</th>
<th>QA Accuracy ↑</th>
<th>GFLOPs ↓</th>
<th>Time (ms) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models using additional object tag inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VinVL-Base [55]</td>
<td>COCO/CC/SBU/Flickr/I</td>
<td>Faster-RCNN [55]</td>
<td>-</td>
<td>75.95</td>
<td>83.08</td>
<td>1023.3</td>
</tr>
<tr>
<td>UNITER-Base [41]</td>
<td>COCO/CC/SBU/VG</td>
<td>Faster-RCNN [1]</td>
<td>72.5</td>
<td>72.70</td>
<td>75.80</td>
<td>949.9</td>
</tr>
<tr>
<td>VILT-B/32 [33]</td>
<td>COCO/CC/SBU/VG</td>
<td>VIT-B/32 [16]</td>
<td>64.4</td>
<td>71.26</td>
<td>76.13</td>
<td>55.9</td>
</tr>
<tr>
<td>Ours, $N=128$</td>
<td>COCO/CC/SBU/VG</td>
<td>VIT-B/32 [16]</td>
<td>62.4</td>
<td>71.62</td>
<td>75.53</td>
<td>30.5</td>
</tr>
<tr>
<td>LXMERT [69]</td>
<td>COCO/VG*</td>
<td>Faster-RCNN [1]</td>
<td>-</td>
<td>72.42</td>
<td>74.50</td>
<td>952.0</td>
</tr>
<tr>
<td>VisualBERT [46]</td>
<td>COCO</td>
<td>Faster-RCNN [1]</td>
<td>-</td>
<td>70.80</td>
<td>67.00</td>
<td>425.0</td>
</tr>
<tr>
<td>Ours, $N=128$</td>
<td>COCO/VG</td>
<td>VIT-B/32 [16]</td>
<td>61.7</td>
<td>70.45</td>
<td>74.87</td>
<td>30.5</td>
</tr>
<tr>
<td>Ours, $N=64$</td>
<td>CC/Webvid</td>
<td>VIT-B/32 [16]</td>
<td>61.0</td>
<td>70.12</td>
<td>74.52</td>
<td>17.0</td>
</tr>
<tr>
<td>Ours, $N=128$</td>
<td>CC/Webvid</td>
<td>VIT-B/32 [16]</td>
<td>61.8</td>
<td>70.91</td>
<td>75.44</td>
<td>30.5</td>
</tr>
</tbody>
</table>

Table 2. Finetuning performance on text-to-image retrieval and visual question answering benchmarks. For NLVR2, we show Test-P accuracy. For Flickr30k, we show text-to-image retrieval R@1 (see appendix for R@5/R@10). Note that for brevity, we only show the image or video source datasets for Pretraining Datasets; the datasets that added additional text annotations are not included in the column (we use * to highlight them). For example, LXMERT is trained with image-text datasets COCO and VG, as well as the three QA datasets based on COCO and VG images, i.e., VQA2, VGQA and GQA. We also gray out models that use additional object tags in the first block and are not comparable to our model. GFLOPs shows the inference cost on a single sample, Time (ms) indicates the average inference time over all samples in VQA2 minival split; For a fair comparison, we gray out models that are pretrained with more data. $N=128$ means latent size N=128.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pretraining Datasets</th>
<th>Visual Backbone</th>
<th>Retrieval ↑</th>
<th>QA Acc. ↑</th>
<th>GFLOPs ↓</th>
<th>Time (ms) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models using other input modalities (e.g., audio)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HERO [55]</td>
<td>TV/HT100M</td>
<td>ResNet152+Slowfast [12] [13]</td>
<td>20.5</td>
<td>-</td>
<td>-</td>
<td>935.2</td>
</tr>
<tr>
<td>MMT [17]</td>
<td>HT100M</td>
<td>S3D+VGG+DenseNet [18] [39] [24]</td>
<td>26.6</td>
<td>-</td>
<td>12.9</td>
<td>-</td>
</tr>
<tr>
<td>AVLNET [67]</td>
<td>HT100M</td>
<td>ResNet152+ResNetX [68] [24] [25]</td>
<td>27.1</td>
<td>-</td>
<td>17.0</td>
<td>-</td>
</tr>
<tr>
<td>Models with CLIP initialization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hanyuan-DML [35]</td>
<td>-</td>
<td>CLIP (ViT-B/16)</td>
<td>55.0</td>
<td>52.1</td>
<td>29.7</td>
<td>57.3</td>
</tr>
<tr>
<td>CLIP2TV [40]</td>
<td>-</td>
<td>CLIP (ViT-B/16)</td>
<td>49.3</td>
<td>45.3</td>
<td>-</td>
<td>44.1</td>
</tr>
<tr>
<td>DRL [25]</td>
<td>-</td>
<td>CLIP (ViT-B/32)</td>
<td>48.8</td>
<td>49.0</td>
<td>26.5</td>
<td>46.2</td>
</tr>
<tr>
<td>CAMoE-DML [10]</td>
<td>-</td>
<td>CLIP (ViT-B/32)</td>
<td>47.3</td>
<td>-</td>
<td>25.9</td>
<td>-</td>
</tr>
<tr>
<td>MDMMT-2 [42]</td>
<td>-</td>
<td>CLIP (ViT-B/32)</td>
<td>48.5</td>
<td>-</td>
<td>26.9</td>
<td>-</td>
</tr>
<tr>
<td>Ours, $N=128$</td>
<td>CC/Webvid</td>
<td>CLIP (ViT-B/16)</td>
<td>45.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HT100M [53]</td>
<td>HT100M</td>
<td>ResNet152+ResNetX [68] [24] [25]</td>
<td>14.9</td>
<td>-</td>
<td>7.1</td>
<td>-</td>
</tr>
<tr>
<td>ClipBERT [41]</td>
<td>COCO/CC</td>
<td>ResNet50 [24]</td>
<td>22.0</td>
<td>20.4</td>
<td>-</td>
<td>21.3</td>
</tr>
<tr>
<td>Ours, $N=128$</td>
<td>CC/Webvid</td>
<td>ViT-B/32 [16]</td>
<td>32.6</td>
<td>30.5</td>
<td>15.8</td>
<td>33.9</td>
</tr>
</tbody>
</table>

acts as a regularizer, as we see LD improves the MSRVTT accuracy in the first block, while increasing $p^{LD}$ too high 0.5 → 0.7 does not help the performance (28.8 → 26.6). The last row in the bottom block achieves the best accuracy (27.1), with LD during both pretraining and finetuning. Second, removing cross-attention layers without LD during finetuning hurts performance (see 26.1 → 24.0 in the middle block). Lastly, with LD during finetuning, the latency of the inference time can be reduced by 19.4% (72.0 ms → 58.0 ms), with minimal accuracy drop (see 27.1 → 26.3 in the bottom block). This indicates that, with a LD-finetuned model, we can control its latency on demand at the inference time by varying the number of cross-attention layers, without storing checkpoints of multiple models.
Figure 4. Input scaling comparison between PERCEIVER-VL, ViLT-B/32 [33], and Frozen-in-Time [4] for video inputs with different the number of frames (left) and frame size (right). Note that the GFLOPs are illustrated in log-scale. On both plots, PERCEIVER-VL shows remarkably lower computation compared to the other two models.

Figure 5. The efficiency-accuracy tradeoff of using different latent array size $N$ during finetuning on MSRVTT. During pretraining, we use the latent array size $N = 128$ (blue vertical line). We use mixed-stream architecture by default.

Figure 6. Comparison of retrieval architectures with different number of MSRVTT video-text pairs. The inference time is calculated from the total retrieval time.

### 6. Conclusion

In this work, we present PERCEIVER-VL, a vision-and-language framework that efficiently handles high-dimensional multi-modal inputs such as long videos and text. The efficiency of PERCEIVER-VL comes from linear complexity based on iterative cross-attention, LayerDrop on cross-attention layers, and a mixed-stream architecture for cross-modal retrieval. Experiments on diverse vision-and-language benchmarks show that our framework has a remarkably higher efficiency than state-of-the-art models, while achieving competitive or better performance. Moreover, we comprehensively analyze the efficiency of our framework, including measuring the scalability in terms of input and latent array size, reducing latency by dropping cross-attention layers, comparing architecture variants, and an ablation study on model training details. It would be an interesting future work to further explore efficient vision-and-language modeling with even more diverse tasks.

### Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by ARO Award W911NF2110220, DARPA KAIROS Grant FA8750-19-2-1004, ONR Grant N000141812871, and NSF-AI Engage Institute DRL-211263. The views, opinions, and/or findings contained in this article are those of the authors and not of the funding agency.
References


Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L. Berg, Alexander Kunitsyn, Maksim Kalashnikov, Maksim Dz-Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Sebastian Goodman, Kevin Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg.


Alecs Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learn-


