AdaVid: Adaptive Video-Language Pretraining

Chaitanya Patel Juan Carlos Niebles Ehsan Adeli
Stanford University
https://chaitanya100100htbp olgithubhtbp olios. pn.lib a y.n nu. du.cn/AdaVid

Abstract

Contrastive video-language pretraining has demonstrated great success in learning rich and robust video representations. However, deploying such video encoders on compute-constrained edge devices remains challenging due to their high computational demands. Additionally, existing models are typically trained to process only short video clips, often limited to 4 to 64 frames. In this paper, we introduce AdaVid, a flexible architectural framework designed to learn efficient video encoders that can dynamically adapt their computational footprint based on available resources. At the heart of AdaVid is an adaptive transformer block, inspired by Matryoshka Representation Learning, which allows the model to adjust its hidden embedding dimension at inference time. We show that AdaVid-EgoVLP, trained on video-narration pairs from the large-scale Ego4D dataset, matches the performance of the standard EgoVLP on short video-language benchmarks using only half the compute, and even outperforms EgoVLP when given equal computational resources. We further explore the trade-off between frame count and compute on the challenging Diving48 classification benchmark, showing that AdaVid enables the use of more frames without exceeding computational limits. To handle longer videos, we also propose a lightweight hierarchical network that aggregates short clip features, achieving a strong balance between compute efficiency and accuracy across several long video benchmarks.

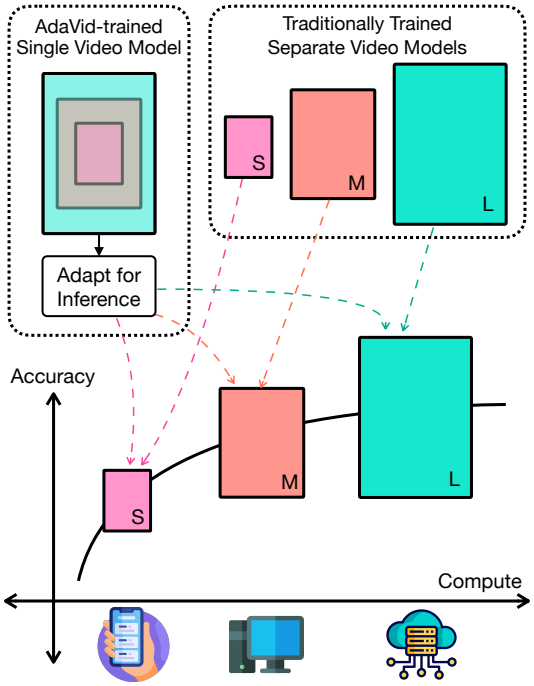

Figure 1. A single AdaVid-trained video model facilitate inference with controllable computational footprint without any postprocessing. It allows one model to adjust its computational demands dynamically according to the requirements, thereby eliminating the need to train multiple distinct models.

1. Introduction

Image-language pretraining [35] has shown remarkable success in learning rich image representations that are robust and transferable to multiple downstream tasks. Inspired by this success, video-language models [2, 5, 26, 45] have emerged as a promising direction to learn rich video representations that are transferable to downstream tasks such as video-text retrieval, video question answering, action recognition etc. Typically, both video and text encoders are transformer-based architectures [2, 37, 45] where compute and memory requirement increases quadratically with the input number of tokens. Video encoders are especially compute-inefficient because even a small number of frames results in a very high number of tokens. Consequently, these models are trained by sampling a small number of video frames (typically 4 to 16). This computational and data inefficiency becomes prohibitive when attempting to train long-form video encoders, especially under contrastive learning frameworks that require larger batch sizes to learn better features. This limitation also restricts their deployment on edge devices with constrained computational resources.

Several prior works have focused on developing efficient transformer architecture, particularly aiming to address the quadratic complexity of self-attention [4, 17, 22]. Many

works leverage the redundancy and structure of video input through space-time attention [5], hierarchical modeling [1], or memory-based architectures [3]. In this work, we draw inspiration from the fact that the computational complexity of transformer is also quadratic with respect to the token dimension and propose AdaVid, an architectural framework to encode long videos in an efficient and adaptive manner. The key component of AdaVid is an adaptive transformer block, inspired by Matryoshka Representation Learning (MRL) [23], that can process input tokens of varying dimensions by sampling appropriate parameters. This design offers the flexibility to dynamically adjust the embedding size of each transformer layer during inference. As shown in Figure 1, one AdaVid-trained video encoder encompasses multiple models of different capacities, enabling the encoding of both long and short videos while accommodating a flexible compute footprint.

To show the effectiveness of AdaVid, we train an adaptive version of EgoVLP [26] within our proposed framework (referred to as AdaVid-EgoVLP) on the large-scale Ego4D [14] video-language dataset. We show that AdaVid-EgoVLP performs equal or better than vanilla EgoVLP and other baselines on several benchmarks while using the full embedding dimension. Additionally, we also show that AdaVid-EgoVLP retains comparable performance on those benchmarks while operating with low embedding dimension (and hence low compute resources). We carry out compute vs. accuracy analysis based on varying dimension sizes of different layers, and provide key insights into these design choices.

We also conduct a compute vs. frame count analysis on the Diving48 [25] long video classification benchmark and show that AdaVid enables the model to process more frames within limited compute while maintaining strong classification accuracy. Additionally, we train AdaVid-Agg – a lightweight aggregator network that can distill the sequence of AdaVid-EgoVLP embeddings extracted from consecutive video clips, into a single feature vector for the entire video. Such hierarchical design allows us to train long video-language models using relatively smaller datasets with long video annotations. We show that AdaVid-Agg retains surprisingly high accuracy on several long video benchmarks, even while operating with a fraction of computational resources chosen adaptively at test time. Such ability to adaptively change the embedding dimension (and consequently the FLOPs) is highly desirable for video understanding in compute-constrained edge devices and wearable devices where the compute load may also vary from time to time. In short, our contributions are as follows:

- We propose an adaptive transformer layer capable of processing input tokens with varying dimensions. This design enables a video encoder, composed of these adaptive layers, to perform inference while accommodating different computational requirements.
- We introduce AdaVid-EgoVLP – an adaptive variant of EgoVLP and AdaVid-Agg for short and long video understanding, respectively. We demonstrate improved compute vs. accuracy trade-offs across multiple benchmarks and also show favorable frame-count vs. compute trade-off on the Diving48 long video classification benchmark.
- We investigate different training and evaluation configurations for choosing dimensions of transformer layers. We show that gradually decreasing embedding dimension sizes across layers yields better performance compared to the alternative approach.

2. Related Work

Efficient Deep Learning Models: Several prior works have focused on creating compute-efficient deep learning architectures. Knowledge distillation [18, 37, 39] requires a two stage process to distill the performance of the bigger teacher model into a smaller student model. Methods like pruning [24, 28] and quantization [38] have been proposed to improve the inference efficiency of deep learning models, which may potentially lead to a decrease in performance [10]. Recently, transformers [40] have become the predominant architecture for many pretraining tasks (including language [11, 40], image-language [35], and video-language [2, 26, 45]) due to their ability to leverage larger datasets and learn richer embeddings from tokenized inputs. However, the computational complexity of the attention mechanism scales quadratically with the number of input token, potentially rendering current efficiency techniques insufficient. This presents a significant obstacle when deploying these models in compute-limited environments. Several prior works have explored methods to improve transformer efficiency, including sub-quadratic attention [4], token dropping [17], and token resampling [22]. These approaches, however, often sacrifice accuracy for *permanent* compute efficiency. In contrast, AdaVid maintains compute efficiency when necessary while preserving the accuracy of a standard transformer when operating at full capacity.

Adaptive Models: Adaptive training to obtain multiple models from a single trained model have been explored in the context of CNNs [7, 16, 46] and transformers [9, 19]. Matryoshka Representation Learning [23] proposed to use adaptive dimensions at the final feature vector, while FlexViT [6] introduced vision transformer with flexible input space. Although these approaches simplify training, they require the network backbone to operate at full capacity, offering no computational benefits during inference. MatFormer [12] proposed to use adaptive computation only for FFN layers and projected tokens back to full dimension size for self-attention layers. Since quadratic self-attention is carried out with full token dimension, the computational benefit of this design remains limited, particularly in video understanding tasks where the number of tokens is high. For language

modeling, SHARCS [36] proposed to use a separate router network to predict the difficulty of a sample, requiring a separate heuristic-based computation of sample hardness during every training epoch. More recently, [48] proposed to use a third elastic student network in DINO [8] image pretraining and used multiple cross-view distillation losses between student and teacher models. In contrast, we simplify the design by incorporating an adaptive embedding dimension at every transformer layer, without the need for distillation and heuristic calculations of sample difficulty.

Video-Language Models: Video-language pretraining [2, 26, 27, 30, 32, 33, 45] has become a key method for developing rich video embeddings for various downstream tasks. Since the video encoders have high memory and compute footprint, these models are typically trained on short videos by sampling few (4 to 64) frames. For example, EgoVLP [26] trains on video-narration pairs of Ego4D [14] dataset by sampling 4 frames per video sample. This limits the applicability of such methods on long-form video understanding and compute-constrained environments. Many prior works have proposed video specific solutions to reduce the compute requirements. Slow-fast networks [13] uses two separate networks to process video at different frame-rate whereas sampling-based methods [28, 42, 49] samples few informative frames. To avoid quadratic global attention, some works propose efficient attention mechanism [5, 20, 41]. Memory-based architectures [3, 44] processes videos in a streaming manner with a memory mechanism to store key information of the past. HierVL [1] employs a hierarchical model where a video is broken into small video clips and encoded with a standard video encoder [26]. A separate small aggregator network is employed to aggregate segment features into a long video feature. Although this formulation allows HierVL to train with long videos (up to 64 frames), it compromises its accuracy on short videos in favor of long videos.

Our AdaVid framework for efficient video understanding is orthogonal to these works. AdaVid focuses on training a model with adaptive compute for compute-contrained edge devices and wearable devices. AdaVid can leverage the redundancy in video inputs and learn rich video embeddings with a fraction of compute compared to baselines. We show applicability of AdaVid on a video encoder with space-time attention [26] as well as hierarchical modeling [1].

3. Method

3.1. Preliminary

Video input is denoted as a tensor of size $T \times 3 \times H \times W$ where T is the input number of frames. Each frame is divided into patches of size $P \times P$, giving $N = HW/P^2$ patches per frame. A common practice is to use $H = W = 224$ and $P = 16$ which gives $N = 196$ patches per frame. Each patch of each frame is projected into a D-dimensional space using a linear projector, giving a total TN number of tokens of dimension D. At every transformer layer, these tokens go through self-attention and FFN layers. The complexity of every FFN layer is usually $16ND^2$. See the Appendix for a detailed description of FLOPs computation.

Dense Attention: If each transformer layer were to use vanilla full attention, its compute complexity would be $24TND^2 + 4T^2N^2D$ (See Appendix). Typically, a `[cls]` token is appended to represent the video embedding, but since $TN >> 1$, we ignore it here for brevity. Note that this formulation is quadratic in N, T, and D.

Space-Time Attention: To avoid calculating dense attention, TimeSformer [5] introduced divided space-time attention where each token first attends to other tokens of the same frame and then attends to the same-positioned tokens of other frames. The compute complexity of each space-time transformer layer is $32TND^2 + 4TND(N + T)$ (See Appendix). Since its complexity scales linearly with the number of frames T, it has been a preferred choice for many subsequent video encoders like Frozen [2], EgoVLP [26], etc. However, because of the high constant of the first term, its relative computational benefits are realized only at a high (> 32) number of frames which is usually not the case for most video understanding tasks.

Hierarchical Modeling: To train with longer videos, some works [1] propose to use hierarchical modeling where the video is divided into S segments of T/S frames each. Each segment is encoded independently by a standard video encoder (like EgoVLP) to give S segment features. This sequence of S segment features is then aggregated by another network to output a single feature embedding for the long video. Such hierarchical modeling makes sense from the perspective of data-efficiency because video modality has a lot of redundancy in pixel values and distant frames relate to each other only through high-level semantic concepts (Dense attention or even space-time attention would be excessive in such cases). However, it does not provide notable computational benefit over space-time attention because the effective complexity of each layer of the feature extractor is $32TND^2 + 4TND(N + T/S)$ (See Appendix). The complexity of the hierarchical aggregator is relatively negligible and can be ignored.

3.2. Adaptive Transformer Layer

Note that the complexity of every transformer component is also quadratic with D, often with a high constant. Building on this insight, we propose to train video encoders that can adaptively use smaller embedding dimension at every transformer layer during inference.

Consider a transformer layer that processes a sequence of K tokens $(x_1, \cdots, x_K)$, producing K output tokens $(y_1, \cdots, y_K)$, where each $x_i, y_i \in \mathbb{R}^D$. As shown in Fig-

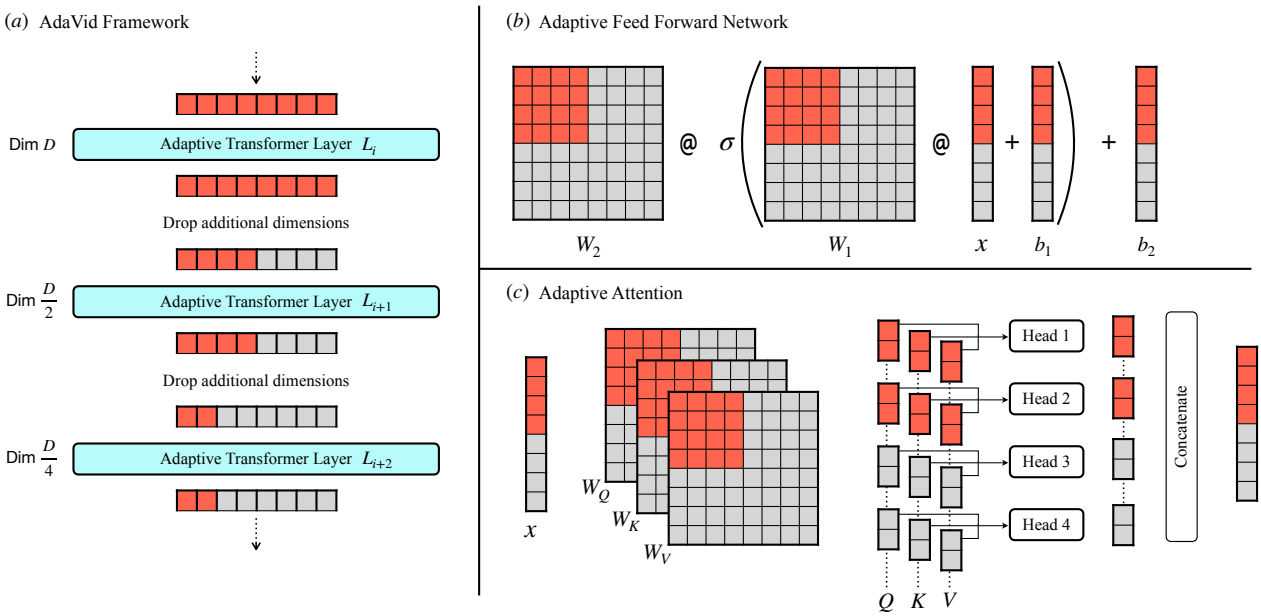

Figure 2. **AdaVid Framework** is designed to train video encoders that facilitate adaptive compute-efficient inference. (a) Key component of AdaVid is the Adaptive Transformer Layer, which is designed to handle input tokens of varying dimension sizes up to D. During each training iteration, each layer processes the input tokens with a randomly selected dimension size, enforcing a coarse-to-fine structure in the model's weights and activations. This allows an AdaVid-trained model to perform inference with a controllable compute footprint. (b) The feedforward layer $W_2\ \sigma(W_1 x + b_1) + b_2$ of the transformer can be modified to accommodate input tokens of size $D/2$ by appropriately slicing the weight and bias parameters. This approach is also applicable to the affine transformation of layer normalization. (c) In multi-head attention, input tokens of size $D/2$ are processed using half the number of heads, rather than reducing the dimension of each head.

ure 2, we adapt its components to handle tokens of smaller dimension $d \leq D$ so that a trained adaptive transformer layer can process a sequence of smaller tokens $(\hat{x}_1, \cdots, \hat{x}_K)$, where each $\hat{x}_i \in \mathbb{R}^d$ consists of the first d values of x_i. It outputs $(\hat{y}_1, \cdots, \hat{y}_K)$, where $\hat{y}_i \in \mathbb{R}^d$ retains as much semantic information as $y_{i[1:d]} \in \mathbb{R}^d$ where $y_{i[1:d]}$ is a vector of first d values of y_i.

This is achieved by making every basic component of the transformer adaptive. For FFN, every linear projection of form $y = W \cdot x + b$, can be adjusted to $y_{[1:d]} = W_{[1:d,1:d]} \cdot x_{[1:d]} + b_{[1:d]}$, using the upper-left $d \times d$ submatrix of weight matrix W and first d values of bias vector b. Similarly, layer normalization $\text{LN}(x; \gamma, \beta)$ can be adjusted to $\text{LN}(x_{[1:d]}; \gamma_{[1:d]}, \beta_{[1:d]})$. For multi-head attention, instead of reducing the dimension for each head, we reduce the number of heads [36]. In particular, if the vanilla transformer layer has D/H heads each with dimension H, the adaptive transformer layer uses d/H heads. To incorporate this, we only use d which is a multiple of H in our experiments.

3.3. AdaVid Video-Language Pretraining

We present AdaVid as a general architectural framework where every transformer layer of any standard video encoder can be replaced with our adaptive transformer layer. In this paper, we show the effectiveness of AdaVid on contrastive video-language representation learning for short and long videos. Specifically, we introduce AdaVid-EgoVLP and AdaVid-Agg for encoding short and long videos, respectively. These models leverage our adaptive transformer block to encode videos in a compute-adaptive manner.

AdaVid-EgoVLP: For short videos, we follow the exact setup of EgoVLP [26] and train its adaptive counterpart AdaVid-EgoVLP. Its video encoder uses $T = 4$ frames, image size $H = W = 224$, and patch size $P = 16$ to tokenize the video clip input. These tokens are processed by 12 adaptive transformer layers with a maximum dimension size of $D = 768$, each consisting of an adaptive space-time attention [5] module followed by an adaptive feedforward network. We use DistilBERT [37] as our text encoder and finetune it during our experiments. Following EgoVLP, we also use EgoNCE [26] loss to train AdaVid-EgoVLP. We compare AdaVid-EgoVLP with vanilla EgoVLP and other strong baselines on short video benchmarks and carry out compute vs. accuracy analysis.

AdaVid-Agg: For long videos, we follow hierarchical late fusion modeling and train a lightweight AdaVid-Agg model to aggregate a sequence of consecutive video clip features extracted from AdaVid-EgoVLP. In particular, we sample $T = 64$ frames from the input long video (unless mentioned otherwise) and encode $S = 16$ segments

Table 1. AdaVid-EgoVLP evaluation configurations. We evaluate the trained AdaVid-EgoVLP model with different configurations for embedding dimensions. $[768 \times 12]$ indicates that all 12 layers use 768-d tokens. $[768 \times 4, 576 \times 4, 384 \times 4]$ means that first four layers use 768-d, followed by four layers of 576-d, followed by final four layers of 384-d. The FLOPs are computed using the complexity equations provided in Section 3.1 with $T = 4$.

Config.	Layer-wise hidden dimension	FLOPs ($\times 10^{10}$)
d-768	$[768 \times 12]$	18.3
d-576	$[576 \times 12]$	10.4
d-384	$[384 \times 12]$	4.7
d-192	$[192 \times 12]$	1.2
d-dec	$[768 \times 3, 576 \times 3, 384 \times 3, 192 \times 3]$	8.7
d-dec-high	$[768 \times 4, 576 \times 4, 384 \times 4]$	11.2
d-dec-low	$[576 \times 4, 384 \times 4, 192 \times 4]$	5.5
d-inc	$[192 \times 3, 384 \times 3, 576 \times 3, 768 \times 3]$	8.7
d-inc-high	$[384 \times 4, 576 \times 4, 768 \times 4]$	11.2
d-inc-low	$[192 \times 4, 384 \times 4, 576 \times 4]$	5.5

(each containing 4 consecutive frames) independently using AdaVid-EgoVLP. AdaVid-Agg aggregates S segment features into a single feature for the long video. AdaVid-Agg is implemented as a transformer [40] consisting of 12 transformer layers. Since it operates over a short sequence of segment features instead of patch tokens, its compute footprint is negligible compared to the compute required for AdaVid-EgoVLP feature extraction. Note that our setup is simpler than HierVL [1] which requires large-scale multi-node training to jointly learn aggregator and EgoVLP feature extractor and uses specific losses based on the hierarchical annotations. In contrast, we do not require hierarchical annotations and train aggregator independently from the feature extractor using standard InfoNCE [35] loss, and show comparable performance to HierVL. We compare AdaVid-Agg with strong baselines on multiple long video benchmarks.

AdaVid Training: During each iteration of training, an embedding dimension can be chosen for every adaptive transformer layer randomly or based on some strategy. For our experiments, we fix the set of allowed embedding dimensions to be $\{D, 3D/4, D/2, D/4\}$ where D is the full embedding size. If the chosen embedding size is higher than the previous embedding size, we pad with zeros; if it is lower, then we simply drop additional dimensions. AdaVid training can also be viewed as a version of dropout which can provide additional regularization benefits. It forces a coarse-to-fine structure on the manifold of the latents and model weights which can be stripped short during inference based on need. Note that our focus is not on performance improvement, and MRL [23] also reported no performance gains over vanilla training. However, we find that models trained with AdaVid often outperform their vanilla counterparts despite identical training setups, likely due to this regularization effect.

AdaVid allows us to train a single model 'containing' multiple smaller models of varying capacities and gives us the flexibility of choosing an embedding dimension at test time. It is also possible to create a large set of smaller models by choosing different granularity at each transformer layer [12], even the ones not observed during training [23]. In our experiment, we show that some strategies of choosing embedding dimension fare better than others.

4. Experiments

4.1. Dataset

We use large-scale Ego4D [15] dataset which contains 9645 untrimmed videos of varying lengths from 5 sec to 7 hrs, totaling 3000 hours of video data. These videos contain daily human activities captured from an egocentric perspective using smart glasses. To train AdaVid-EgoVLP, we use a curated set of ∼4M narrations [26] each covering 1-2 seconds of video clip. To train AdaVid-Agg, we additionally use ∼100K summaries [1] each covering 5 minutes of videos.

4.2. Evaluation Benchmarks

We evaluate the quality of video-language embeddings on various zero-shot video-language benchmarks covering short as well as long videos.

EgoMCQ [26]: Given a text description and 5 short video clips (1-2 seconds), classify which video aligns with the text description. It contains ∼40K samples. The metric is classification accuracy. It is divided into two subparts: (1) EgoMCQ (inter) where five candidate clips for each text query are sourced from the whole dataset. This is an easier setting. (2) EgoMCQ (intra), where the candidate clips for each text query are sourced from the same video as the correct video clip. This is a harder setting.

SummaryMCQ [1]: Given a text description and 5 medium-lengthed videos (5 minutes), classify which video aligns best with the text description. The metric is classification accuracy. This is similar to EgoMCQ but for longer 5-minute videos.

Diving-48 [25] is a curated action recognition benchmark designed to evaluate long-term temporal reasoning in action recognition. Each video is categorized into one of 48 classes based on the type of dive, requiring fine-grained spatio-temporal reasoning over extended temporal contexts. Unlike many other action recognition datasets, achieving high classification accuracy on Diving-48 requires processing a larger number of frames.

LongVideoRetrieval: As discussed in [31], creating a long video-language benchmark to test models' long video understanding is a challenging task because many video benchmarks can be solved adequately by observing only a few frames. To evaluate the model on even longer videos, we curate LongVideoRetrieval benchmark by using long video

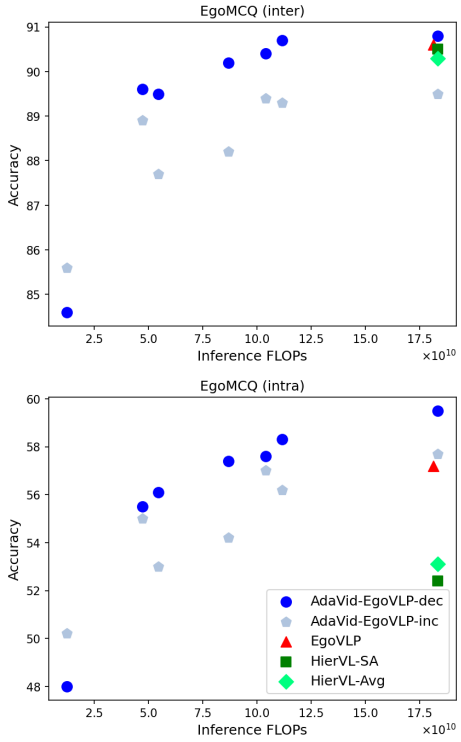

Figure 3. **AdaVid-EgoVLP on two EgoMCQ benchmarks**: AdaVid-EgoVLP-dec, trained with decreasing dimensions for deeper layers, performs better than AdaVid-EgoVLP-inc which was trained with increasing dimensions. AdaVid-EgoVLP-dec performs better than baselines while using maximum compute resources. The same model also retains high accuracy when evaluated with low compute evaluation configurations from Table 1.

captions of Ego4D videos provided by [21]. The objective is to retrieve the corresponding video, based on a given long textual description, from a database of approximately 1500 long videos, which range in length from a few minutes to two hours, with an average duration of 29 minutes. Processing a small number of video frames is not enough for LongVideoRetrieval because the captions are long and cover activities happening throughout the video. The evaluation metric is recall (R@1, R@5 and R@10).

EgoSchema [31]: Given a 3-minute video and a complex question with 5 choices, predict the correct answer. This benchmark was manually curated for long-form video understanding to make sure that the correct answer cannot be derived by observing a short video clip from the entire video. The metric is classification accuracy. Note that this is a VideoQA benchmark with much more complex language than our pretraining dataset.

4.3. AdaVid-EgoVLP

We train AdaVid-EgoVLP for 10 epochs on 8 NVIDIA L40S GPUs with a total batch size of 160, and other hyperparameters the same as EgoVLP. To compensate for our lower batch size compared to EgoVLP (160 vs. 512), and to speed up AdaVid training, we initialize our model with EgoVLP weights and finetune it with adaptive embedding dimensions. We train a single AdaVid-EgoVLP model and evaluate it using different configurations with different compute requirements as mentioned in Table 1. Apart from standard configurations with the same embedding dimension for all layers (d-768, d-576, d-384, d-192), we also evaluate 'd-dec*' configurations where the embedding dimensions decrease for deeper layers and 'd-inc*' where the embedding dimensions increase for deeper layers.

Table 2. Results on EgoMCQ benchmark

Method	EgoMCQ (inter)	EgoMCQ (intra)
EgoVLP [26]	90.6	57.2
HierVL-Avg [1]	90.3	53.1
HierVL-SA [1]	90.5	52.4
AdaVid-EgoVLP	**90.8**	**59.5**

Choosing Dimensions for AdaVid training: To find the optimal strategy for varying layer dimensions during training, we trained two versions: (1) AdaVid-EgoVLP-dec where we randomly sample layer dimensions during training ensuring that each layer has an equal or lower dimension than the previous layer, i.e., gradually decreasing dimension sizes. We evaluate this model on standard and decreasing configurations from Table 1. (2) AdaVid-EgoVLP-inc where we sample gradually increasing dimensions during training. We evaluate this model on standard and increasing configurations from Table 1. The results on two subsets of EgoMCQ benchmark are shown in Figure 3. We can see that AdaVid-EgoVLP-dec performs better than AdaVid-EgoVLP-inc in various evaluation settings despite having the same FLOPs. This result is in line with the results of [36] where the authors used adaptive embedding only at deeper layers. It also aligns with our intuition that deeper layers can afford to strip away low-level details and only store high-level concepts in a smaller subspace. The opposite strategy bottlenecks the information at early layers and hurts model performance in a significant manner. We use AdaVid-EgoVLP-dec to carry out the rest of the analysis, and only use standard and decreasing configurations from Table 1 for evaluation.

AdaVid-EgoVLP is accurate while being efficient. Figure 3 also compares AdaVid-EgoVLP with baselines. EgoVLP is trained using the same architecture and supervision, and with a higher batch size. HierVL additionally uses summary supervision to associate low-level narrations with high-level activities. Despite that, AdaVid-EgoVLP-dec outperforms both baselines by a noticeable margin when using the full embedding size during evaluation as reported in Table 2. We attribute this improvement to the implicit dropout-like regularization provided by AdaVid training.

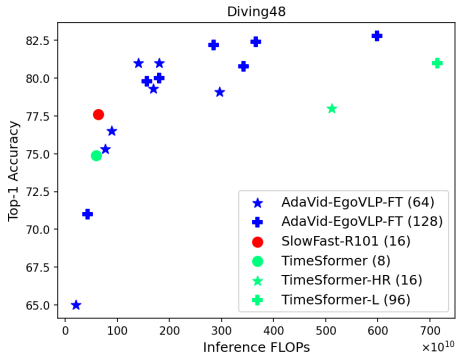

Figure 4. **Results on Diving-48**: We evaluate AdaVid using various evaluation configurations from Table 1 with 64 and 128 frames. With adaptive compute, AdaVid can process more frames efficiently, outperforming vanilla-trained baselines.

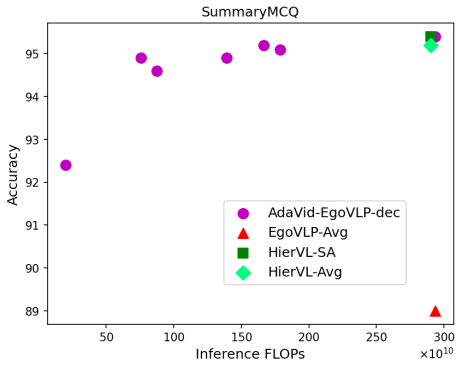

Figure 5. **Results on SummaryMCQ**: AdaVid-Agg achieves comparable performance to HierVL baselines with full embedding dimensions, while also demonstrating robust performance with significantly reduced computational resources as needed.

Table 3. Results on Diving-48. The baselines are pretrained on ImageNet-21K, while the AdaVid models are pretrained on Ego4D.

Method	Num. Frames	FLOPs $\times 10^{10}$	Top-1 Accuracy
SlowFast-R101 [13]	16	64	77.6
TimeSformer [2]	8	59	74.9
TimeSformer-HR [2]	16	511	78.0
TimeSformer-L [2]	96	714	81.0
AdaVid-EgoVLP-FT (d-dec)	64	140	81.0
AdaVid-EgoVLP-FT (d-dec)	128	285	**82.2**

Table 4. Results on SummaryMCQ and LongVideoRetrieval

Method	SummaryMCQ	LongVideoRetrieval		
		R@1	R@5	R@10
EgoVLP [26]	89.0	7.1	21.3	31.8
HierVL-Avg [1]	95.2	-	-	-
HierVL-SA [1]	**95.4**	16.1	48.9	62.8
AdaVid-Agg	**95.4**	**16.6**	**50.3**	**66.5**

AdaVid-EgoVLP-dec also performs equal to or better than baselines despite using approximately 0.5x FLOPs from various configurations as shown in Figure 3. It maintains good performance even with 0.25x FLOPs, but observes a small drop at the lowest configuration with 0.06x FLOPs, especially for EgoMCQ(intra) which requires more fine-grained video understanding. Although different configurations with comparable FLOPs show almost similar performance, some tasks may benefit from an optimized configuration. Overall, AdaVid allows a single model to exhibit varying levels of computational efficiency, enabling it to allocate more compute to challenging tasks while maintaining high efficiency for simpler ones. Such flexibility in a single-trained model is a unique feature of the AdaVid framework.

AdaVid can process more frames with limited compute: We finetune AdaVid-EgoVLP on the Diving-48 dataset (referred to as AdaVid-EgoVLP-FT) and evaluate it using different numbers of uniformly sampled frames during inference. Figure 4 and Table 3 present the results, compared against TimeSformer [5], which shares the same architecture as ours but is trained in a standard (non-adaptive) manner, ensuring a fair comparison. Our results show that AdaVid-EgoVLP-FT can process up to 128 frames using significantly less compute, while outperforming the baseline that relies on much higher computational resources. This shows that when a task demands processing a larger number of frames for temporally fine-grained analysis, AdaVid can effectively scale down the embedding dimension to accommodate more frames within a fixed compute budget.

Table 5. Results on EgoSchema. Models in gray are pretrained on signficantly larger corpus of video datasets.

Method	FLOPs $\times 10^{10}$	EgoSchema	
		Subset	Fullset
InternVideo [43]	2000+	-	32.1
SeViLA [47]	2000+	25.7	22.7
LongViViT [34]	1000+	56.8	33.3
MC-ViT-B [3]	600+	61.2	42.3
HierVL [1]	293	52.4	41.6
AdaVid-Agg (d-768)	293	56.2	40.9
AdaVid-Agg (d-384)	75	54.2	40.2
AdaVid-Agg (d-192)	20	52.0	38.3

4.4. AdaVid-Agg

We train AdaVid-Agg for 10 epochs on 8 NVIDIA L40S GPUs on Ego4D short narrations (1-2s) as well as long summary (5 minutes) annotations using a total batch size of 256. AdaVid-Agg is trained on the extracted features of a pre-trained AdaVid-EgoVLP model where we use different inference configurations provided in Table 1 to extract features of varying granularity. We use a learning rate of 10^{-5} and decouple weight decay [29] regularization of 0.1 to ac-

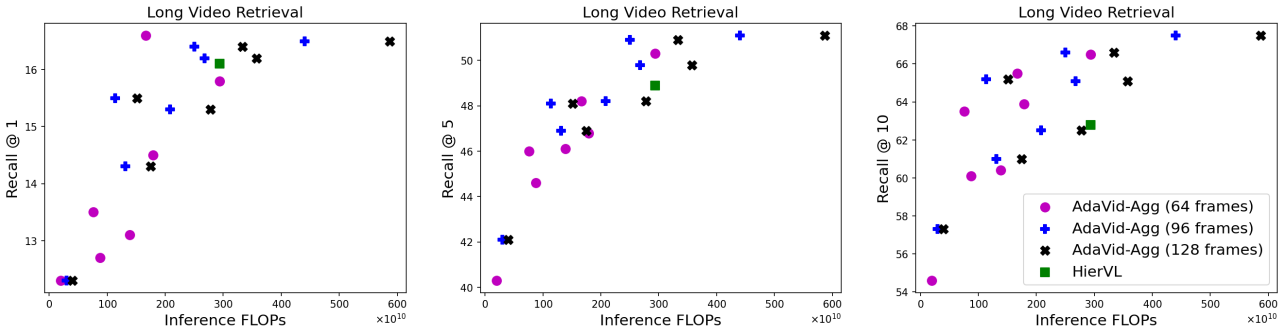

Figure 6. **AdaVid-Agg on LongVideoRetrieval**: We evaluate AdaVid-Agg on text-to-video retrieval from a database of very long videos. We evaluate the dimension configurations mentioned in Table 1 using 64, 96, and 128 frames. X-axis shows the compute required to encode a single long video. AdaVid-Agg outperforms HierVL under various inference settings with less compute.

count for a relatively smaller video summary dataset. Similar to before, we train a single AdaVid-Agg model and evaluate it on multiple long video benchmarks. Since AdaVid-Agg has an extremely small compute footprint relative to the underlying feature extractor AdaVid-EgoVLP, we operate the aggregator at full capacity and carry out compute vs. accuracy analysis by applying different evaluation configurations (Table 1) on the feature extractor.

Results on SummaryMCQ: We first evaluate AdaVid-Agg on SummaryMCQ benchmark, which is the long video counterpart of EgoMCQ benchmark, and compare against HierVL baselines in Figure 5. Similar to AdaVid-EgoVLP, AdaVid-Agg matches the performance of HierVL while using the full embedding dimension as shown in Table 4, and maintains strong performance even with 0.25x compute. EgoVLP-Avg baseline independently encodes $S = 16$ short segments of the input video and averages their features. Its significantly worse performance highlights the importance of temporal aggregator network and long video summary supervision in both HierVL and AdaVid-Agg.

Results on EgoSchema: In Table 5, we show AdaVid-Agg accuracy on two subsets of EgoSchema benchmark [31]. Even though strong HierVL baseline jointly trains feature extractor and aggregator with higher batch size and uses annotation hierarchy to define additional losses, AdaVid-Agg shows similar accuracy as HierVL, and more importantly retains strong performance in low-compute configurations. For example, when operating AdaVid-EgoVLP with 0.25x compute, it shows very small drop in performance. Even with 0.0625x compute, AdaVid-Agg outperforms many large video models, trained on much bigger corpus of video datasets. Note that the questions in EgoSchema benchmark has complex language structure in contrast to the simpler narrations and summary annotations of Ego4D on which our method has been trained. With extensive datasets containing high-quality annotations, AdaVid has the potential to train accurate and dynamically efficient video models suitable for deployment on edge devices.

Results on LongVideoRetrieval: We evaluate AdaVid-Agg on LongVideoRetrieval benchmark using different evaluation configurations. In addition to 64 frames, we also evaluate our model using 96 and 128 frames, and show our results in Figure 6. Overall, our single model shows progressively improving retrieval performance when evaluated with compute resources ranging from 0.2 TFLOPs to 6 TFLOPs. AdaVid-Agg outperforms HierVL while using equal or even less compute as shown in Table 4. It is also possible to do adaptive retrieval [23] where the large set of candidate videos are ranked using an efficient inference configuration to find a smaller set of promising candidates that can be reranked again with more accurate inference with more compute. We leave this exploration for further research.

5. Conclusion

In this paper, we proposed AdaVid framework as a promising direction to learn flexible models that can encompass multitudes of big and small models into a single one. We showed the effectiveness of AdaVid on video-language pretraining where the video modality has traditionally been compute and data intensive. Our short video feature extractor, AdaVid-EgoVLP, serves as a flexible replacement for EgoVLP, while AdaVid-Agg is an efficient aggregator of short video segment features. AdaVid models outperform strong baselines such as HierVL and EgoVLP on both short and long video benchmarks, while also maintaining strong performance under low-compute settings. We conduct a detailed analysis of accuracy vs. compute and frame-count vs. compute for AdaVid models and baselines, offering valuable insights into leveraging the flexibility of each adaptive layer. We believe that the AdaVid framework enables the deployment of video-language models on edge devices, supporting efficient long video understanding in a compute-efficient manner.

Acknowledgements: This work was partially supported by Panasonic and the NIH Grant R01AG089169. We thank Yuta Kyuragi for his helpful feedback on the manuscript.

References

[1] Kumar Ashutosh, Rohit Girdhar, Lorenzo Torresani, and Kristen Grauman. Hiervl: Learning hierarchical video-language embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23066–23078, 2023. 2, 3, 5, 6, 7

[2] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE International Conference on Computer Vision*, 2021. 1, 2, 3, 7

[3] Ivana Balažević, Yuge Shi, Pinelopi Papalampidi, Rahma Chaabouni, Skanda Koppula, and Olivier J Hénaff. Memory consolidation enables long-context video understanding. *arXiv preprint arXiv:2402.05861*, 2024. 2, 3, 7

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 1, 2

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, page 4, 2021. 1, 2, 3, 4, 7

[6] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14496–14506, 2023. 2

[7] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 3

[9] Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric P Xing. Vision transformer slimming: Multi-dimension searching in continuous optimization space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4931–4941, 2022. 2

[10] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016. 2

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[12] Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M Kakade, Ali Farhadi, et al. Matformer: Nested transformer for elastic inference. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, 2023. 2, 5

[13] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 3, 7

[14] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 2, 3

[15] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 5

[16] Matteo Grimaldi, Luca Mocerino, Antonio Cipolletta, and Andrea Calimera. Dynamic convnets on tiny devices via nested sparsity. *IEEE Internet of Things Journal*, 10(6):5073–5082, 2022. 2

[17] Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. Transkimmer: Transformer learns to layer-wise skim. *arXiv preprint arXiv:2205.07324*, 2022. 1, 2

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[19] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020. 2

[20] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. In *European Conference on Computer Vision*, pages 87–104. Springer, 2022. 3

[21] Md Mohaiminul Islam, Ngan Ho, Xitong Yang, Tushar Nagarajan, Lorenzo Torresani, and Gedas Bertasius. Video recap: Recursive captioning of hour-long videos. *arXiv preprint arXiv:2402.13250*, 2024. 6

[22] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 1, 2

[23] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, 2022. 2, 5, 8

[24] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*, 2021. 2

[25] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018. 2, 5

[26] Kevin Qinghong Lin, Alex Jinpeng Wang, Mattia Soldan, Michael Wray, Rui Yan, Eric Zhongcong Xu, Difei Gao, Rongcheng Tu, Wenzhe Zhao, Weijie Kong, et al. Egocentric video-language pretraining. *arXiv preprint arXiv:2206.01670*, 2022. 1, 2, 3, 4, 5, 6, 7

[27] Kevin Qinghong Lin, Jinpeng Wang, Mattia Soldan, Michael Wray, Rui Yan, Eric Z Xu, Difei Gao, Rong-Cheng Tu, Wenzhe Zhao, Weijie Kong, et al. Egocentric video-language pretraining. *Advances in Neural Information Processing Systems*, 35:7575–7586, 2022. 3

[28] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 2, 3

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7

[30] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020. 3

[31] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36, 2024. 5, 6, 8

[32] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2630–2640, 2019. 3

[33] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9879–9889, 2020. 3

[34] Pinelopi Papalampidi, Skanda Koppula, Shreya Pathak, Justin Chiu, Joe Heyward, Viorica Patraucean, Jiajun Shen, Antoine Miech, Andrew Zisserman, and Aida Nematzdeh. A simple recipe for contrastively pre-training video-first encoders beyond 16 frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14386–14397, 2024. 7

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 2, 5

[36] Mohammadreza Salehi, Sachin Mehta, Aditya Kusupati, Ali Farhadi, and Hannaneh Hajishirzi. Sharcs: Efficient transformers through routing with dynamic width sub-networks. *arXiv preprint arXiv:2310.12126*, 2023. 3, 4, 6

[37] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 1, 2, 4

[38] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8815–8821, 2020. 2

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 2

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 5

[41] Jue Wang and Lorenzo Torresani. Deformable video transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14053–14062, 2022. 3

[42] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. *arXiv preprint arXiv:2403.10517*, 2024. 3

[43] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 7

[44] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022. 3

[45] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*, 2021. 1, 2, 3

[46] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 2

[47] Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. Self-chained image-language model for video localization and question answering. *Advances in Neural Information Processing Systems*, 36, 2024. 7

[48] Yingying Zhang, Xin Guo, Jiangwei Lao, Lei Yu, Lixiang Ru, Jian Wang, Guo Ye, Huimei He, Jingdong Chen, and Ming Yang. Poa: Pre-training once for models of all sizes. In *European Conference on Computer Vision*, pages 131–148. Springer, 2024. 3

[49] Yuan Zhi, Zhan Tong, Limin Wang, and Gangshan Wu. Mgsampler: An explainable sampling strategy for video action recognition. In *Proceedings of the IEEE/CVF International conference on Computer Vision*, pages 1513–1522, 2021. 3