ARVideo: Autoregressive Pretraining for Self-Supervised Video Representation Learning

Anonymous authors
Paper under double-blind review

Abstract

This paper presents a new self-supervised video representation learning framework **ARVideo**, which *autoregressively* predict the next video token in a tailored sequence order. Two key designs are included. First, we organize autoregressive video tokens into clusters that span both *spatially* and *temporally*, thereby enabling a richer aggregation of contextual information compared to the standard spatial-only or temporal-only clusters. Second, we adopt a randomized spatiotemporal prediction order to facilitate learning from multi-dimensional data, addressing the limitations of a handcrafted spatial-first or temporal-first sequence order. Extensive experiments establish ARVideo as an effective paradigm for self-supervised video representation learning. For example, when trained with the ViT-B backbone, ARVideo competitively attains 81.2% on Kinetics-400 and 70.9% on Something-Something V2, which are on par with the strong benchmark set by VideoMAE. Importantly, ARVideo also demonstrates higher training efficiency, *i.e.*, it trains 14% faster and requires 58% less GPU memory compared to VideoMAE.

1 Introduction

The transformer architecture, as introduced in Vaswani *et al.* (Vaswani et al., 2017), has fundamentally transformed the field of natural language processing (NLP) through its ability to model long-range dependencies with minimal inductive bias. A crucial catalyst for its success lies in self-supervised learning of robust and transferable representations from large volumes of unlabeled data. Within this paradigm, masked language modeling (MLM) (Devlin et al., 2019) and autoregressive modeling (AR) (Radford et al., 2018; Brown et al., 2020; OpenAI, 2023) stand out as two leading approaches. Specifically, MLM masks random portions of input tokens and trains models to predict masked elements; whereas AR predicts subsequent words in a sequence based on all preceding words. These methods have propelled state-of-the-art performance in various NLP tasks.

In the video domain, however, the landscape is different. Previous studies have predominantly relied on supervised pretraining using image datasets, often overlooking the critical aspect of temporal dynamics (Liu et al., 2022b; Bertasius et al., 2021). Recently, there has been a shift towards leveraging NLP-inspired mask language modeling (Devlin et al., 2019) or image-inspired mask image modeling (He et al., 2022; Bao et al., 2022) to directly exploit unlabeled video datasets for pretraining. For instance, VideoMAE (Tong et al., 2022; Feichtenhofer et al., 2022) introduces mask autoencoder (He et al., 2022) for self-supervised video video representation learning; BEVT (Wang et al., 2022a) learns spatial representations from image data and joint-masked image and video modeling. Despite these advancements, autoregressive modeling—another powerful self-supervised learning approach in NLP—has yet to be extensively explored within the context of video data analysis. Specifically, considering the strong temporal structure in videos where each frame depends on previous frames, AR is able to naturally capture these dependencies by modeling the next frame (or features of the next frame) given past frames and to learn how causal changes occur in videos, expecting to induce more robust and nuanced feature representations for facilitating downstream tasks such as action recognition, event classification, or anomaly detection.

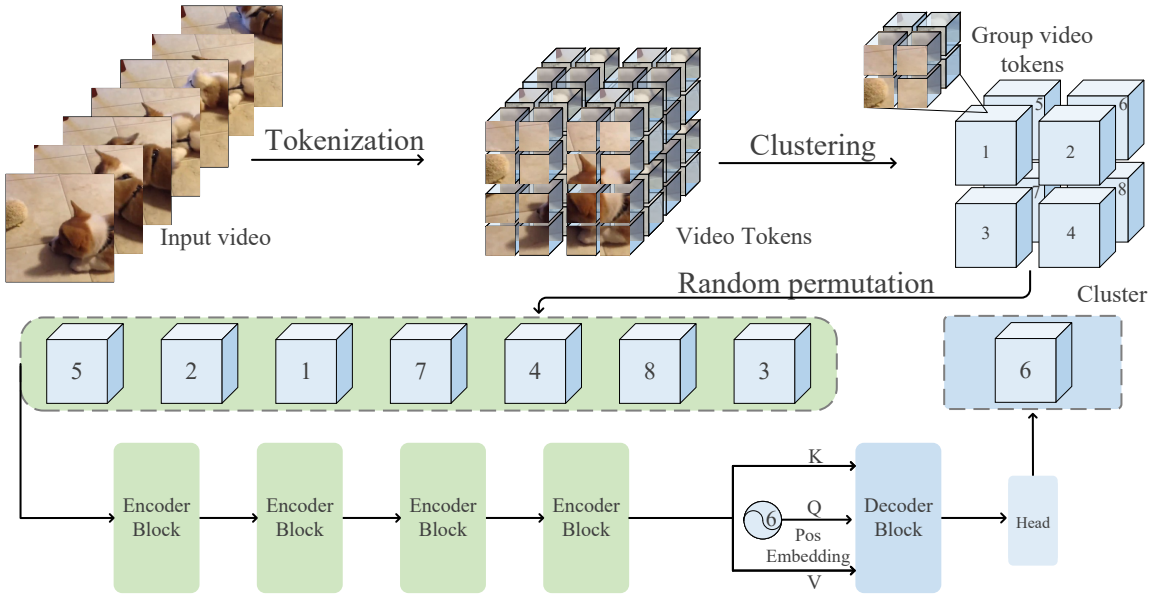

Figure 1: ARVideo autoregressive predicts spatiotemporal cluster from grouping tokens span spatial and temporal dimension. Following (Tong et al., 2022), we first tokenize the video into discrete tokens. These tokens are grouped into clusters, each spanning both spatial and temporal dimensions. Finally, we randomly permute the order of these clusters and adopt an autoregressive strategy to predict subsequent clusters.

Critically, in the practical instantiation, applying autoregressive pretraining to video data entails the same principle of autoregressively predicting the next *element* in a sequential order based on its predecessors. In natural language, these elements—words—are clearly defined and inherently follow a chronological order. For images, elements could be conceptualized as pixels or patches arranged in a flattened sequence (Chen et al., 2020; El-Nouby et al., 2024; Ren et al., 2024). The further transition to video data, however, introduces additional complexity due to its inherently multidimensional nature (*i.e.*, including both spatial and temporal dimensions). This raises a crucial inquiry: *how should we define an autoregressive 'video element' and establish a visual sequence order for self-supervised video representation learning?*

We note traditional methods, such as converting video into a sequence of cubes (Tong et al., 2022; Bertasius et al., 2021; Wang et al., 2022a; Liu et al., 2022b) and subsequently linearly mapping these cubes into video tokens, generally reveal critical limitations in addressing this query. Specifically, the granularity of these video tokens often fails to encapsulate the rich semantics typically represented by words in text-based models—primarily because 1) these video tokens are too dimensionally limited, and 2) video inherently lacks a sequential order in its spatial dimensions, although it retains this feature in its temporal aspects.

To address these challenges, we hereby present **ARVideo**, a novel autoregressive-based video representation learning paradigm with two key designs (see Figure 1). Firstly, we redefine 'video elements' by grouping video tokens into spatiotemporal video clusters, differentiating from conventional single-dimensional strategies like spatial video clusters or temporal video clusters. This approach improves semantic representation by aggregating more contextually relevant multidimensional information. Secondly, we find that, compared to well-defined yet single-dimensional spatial-first or temporal-first sequence orders, a sequence order that randomly integrates both spatial and temporal dimensions empirically yields significantly stronger results. This suggests that effectively capturing the inherent multidimensionality of video data is crucial for autoregressive modeling. Extensive experiments establish our ARVideo as an effective paradigm for video representation learning. For example, while the autoregressive video representation learning baseline only attains 74.2% on Kinetics-400 and 66.4% on Something-Something V2, ARVideo significantly boosts the results to 81.2% (+7%) and 70.9% (+4.5%), respectively. Notably, these results not only match but, in some aspects, surpass the strong benchmark set by VideoMAE, particularly with respect to training efficiency—ARVideo achieves faster training speeds by 14% and reduces GPU memory consumption by 58%.

2 Related Work

2.1 Video Representation Learning

Video representation learning has witnessed significant exploration, historically driven by supervised learning methods (Tran et al., 2018; Wang et al., 2019; Simonyan & Zisserman, 2014; Bertasius et al., 2021; Liu et al., 2022b) that pretrain backbone networks on labeled image or video data before fine-tuning. However, such methods face challenges due to inherent discrepancy between image and video data, compounded by the scarcity of comprehensively labeled video datasets.

In the era of self-supervised learning, recent work have designed pre-tasks incorporating temporal information for self-supervised video representation learning (Xu et al., 2019; Benaim et al., 2020; Huang et al., 2021; Qian et al., 2021; Ranasinghe et al., 2022) and leveraging contrastive learning for effective visual representations (Qian et al., 2021; Kuang et al., 2021; Li et al., 2021; Diba et al., 2021; Han et al., 2020a;b). Additional, mask reconstruction-based methods inspired by masked language modeling (Devlin et al., 2019) are introduced into self-supervised image and video representation learning. For example, MAE (He et al., 2022) presents a scalable self-supervised learning method to reconstruct masked image patches while VideoMAE (Tong et al., 2022) extends this approach to video data and reconstructs masked spacetime patches. BEVT (Wang et al., 2022b) separates spatial learning from temporal dynamics, training on masked images initially before jointly on masked images and videos. Christoph *et al.* (Feichtenhofer et al., 2022) introduce an efficient video-based MAE extension with minimal biases and significant speedups. In contrast to prior works, our ARVideo proposes a new path for self-supervised video representation learning via autoregressive pretraining.

2.2 Autoregressive Pretraining

As a representative approach for autoregressive pretraining, Generative Pretrained Transformer (GPT) trains language models by autoregressively predicting the next word based on all preceding words in a sentence. Inspired by the success of autoregressive modeling in NLP, researchers start to apply autoregressive pretraining in computer vision. ImageGPT (Chen et al., 2020) learns effective image representations by training a Transformer to autoregressively predict image pixels without any prior knowledge of their 2D structure. SAIM (Qi et al., 2023) adopts an encoder to autoregressively learn contextual information like a standard vision transformer (ViT) and a decoder to predict the current content, mutually reinforcing each other's functions. RandSAC (Hua et al., 2022) arranges image tokens into segments for parallel intra-segment and sequential inter-segment autoregressive prediction. However, applying autoregressive pretraining on video data faces notable challenges due to the extra temporal dimension. ARVideo explores the design of autoregressive video elements and visual sequence orders for video representation learning.

Recently, AR modeling has been adopted for video generation works (Yu et al., 2024; Kondratyuk et al., 2024). In contrast, as a video representation learning framework, ARVideo incorporates some key design differences, including (1) a simpler tokenizer as opposed to a specialized tokenizer (Yu et al., 2024), (2) cluster-wise AR modeling—which operates on groups of tokens—as opposed to token-wise AR modeling using Large Language Models (LLMs) for video generation (Kondratyuk et al., 2024), and (3) random rasterization for spatialtemporal prediction order as opposed to predefined orders optimized for video generation tasks. These distinctions advance ARVideo upon existing works for improved self-supervised learning outcomes.

3 Method

In this section, we first revisit GPT (Radford et al., 2018) and ImageGPT (Chen et al., 2020) to establish the foundation for the proposed ARVideo, as illustrated in Figure 1. We then analyze the inherent difference between image and video data, followed by the design of *elements* and the optimal prediction *order* as the key ingredients in ARVideo for autoregressive prediction with videos. Notably, throughout this paper, we use the term 'clustering' to refer to the predefined grouping of tokens along spatial and temporal dimensions, rather than the conventional meaning of unsupervised grouping based on similarity.

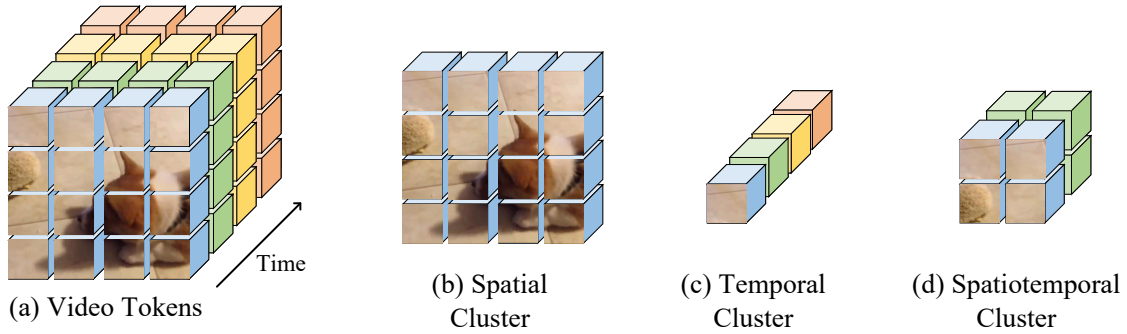

Figure 2: Comparison between video token and different cluster. ARVideo groups multiple tokens into (b) Spatial cluster across the spatial domain (c) Temporal cluster across the spatial domain (d) Spatialtemporal cluster across both the spatial and temporal domain.

3.1 Generative Pretrained Transformer

We first outline the Generative Pretrained Transformer (GPT) framework. Consider an unlabeled language dataset $\mathcal{U}$ comprising sentences $[u^1, ..., u^N]$, where each sentence u^j consists of words $u^j = \{u_1^j, ..., u_n^j\}$. GPT (Radford et al., 2018) autoregressively predicts the next word given all preceding words, minimizing the negative log-likelihood with model parameter θ:

$$L = -log \prod_{i=1}^{n} p(u_i^j | u_1^j, ..., u_{i-1}^j, \theta). \tag{1}$$

This modeling strategy has fundamentally changed the landscape of natural language processing, leading to the development of tremendously successful models like ChatGPT (Radford et al., 2018) and GPT-4 (OpenAI, 2023).

3.2 ImageGPT

Transitioning from natural language processing to image processing necessitates the design of image elements for autoregressive prediction. In ImageGPT, it treats individual pixels as elements. Specifically, given an image $x \in R^{H \times W \times C}$, ImageGPT flattens it into a 1D pixel sequence of length $N = H \times W$, and autoregressively predicts the next pixel given all preceding pixels:

$$L = -log \prod_{i=1}^{N} p(x_i | x_1, ..., x_{i-1}, \theta) \tag{2}$$

This approach incurs significant computational overhead due to the quadratic complexity of self-attention *w.r.t.* the input sequence length. ImageGPT thereby uses smaller image sizes (e.g., 32×32) in pretraining, yielding suboptimal performance. This limitation is pertinent in our development of ARVideo and becomes more pronounced due to the added complexity of video data.

3.3 ARVideo

Illustrated in Figure 1, ARVideo autoregressively pretrains on video data $x \in \mathcal{R}^{T \times H \times W \times C}$. Note that directly extending ImageGPT to videos faces significant challenges, primarily due to the added temporal dimension, which would significantly escalate computational demands, even with low-resolution videos like $4 \times 32 \times 32$. Moreover, pixels as autoregressive elements lack semantic richness compared to words in the language, further necessitating pixel grouping strategies to enhance representation learning. To better facilitate learning from multi-dimensional video data, we also explore prediction orders across spatial and temporal dimensions.

3.3.1 Pixel grouping

From Pixels to Video Tokens. With patch embeddings in ViT, videos can be patchified into non-overlapping cubes (Tong et al., 2022; Bertasius et al., 2021; Wang et al., 2022a; Liu et al., 2022b) of size $P_T \times P_W \times P_H$. Then, each cube is transformed into a video token through a linear projection layer, resulting in $N = \frac{T}{P_T} \times \frac{H}{P_H} \times \frac{W}{P_W}$ video tokens. This tokenization significantly reduces operational elements, thus alleviating computational demands while ensuring that each video token encapsulates richer semantics compared to individual pixels. For example, as reported in Table 1, using video tokens as autoregressive elements for pretraining significantly outperforms approaches without tokenization by 3.3% while keeping pretraining resolution consistent with previous work (Tong et al., 2022; Wang et al., 2022a).

Element	Resolution	Something- Something V2
Pixel	$8 \times 14 \times 14$	60.7
Token	$16 \times 224 \times 224$	64.0

Table 1: Grouping pixels into video tokens facilitates autoregressive pretraining on higher-resolution videos and improves performance by 3.3%.

This promising transition from pixels to video tokens introduces a compelling query: *Can further performance gains be realized by aggregating more tokens?* In pursuit of this, we examine three options: grouping video tokens into spatial, temporal, or spatiotemporal clusters. It is important to note that within each cluster, video tokens are always fully attended to each other. This full-attention configuration helps to enable a more effective consolidation of semantic content within each autoregressive element.

From Tokens to Spatial Clusters. As shown in Figure 2(b), we strategically group spatially neighbored tokens—those sharing the same temporal positions but varying spatially—into spatial clusters. Following the patch embedding step, video tokens within the spatial domain $\frac{H}{P_H} \times \frac{W}{P_W}$ are grouped into one element, resulting in $\frac{T}{P_T}$ autoregressive elements. For example, a video of size $16 \times 224 \times 224$ with a cube embedding size of $2 \times 16 \times 16$ (Tong et al., 2022) here will be transformed into 8 autoregressive elements, with each element comprising 14×14 tokens.

From Tokens to Temporal Clusters. As illustrated in Figure 2(c), our method integrates temporal information by grouping tokens that are temporally adjacent into temporal clusters. Specifically, tokens within the temporal domain $\frac{T}{P_T}$ are grouped into one element, resulting in $\frac{H}{P_H} \times \frac{W}{P_W}$ autoregressive elements. For instance, a video of size $16 \times 224 \times 224$ with a cube embedding size of $2 \times 16 \times 16$ (Tong et al., 2022) here will transformed into 14×14 autoregressive elements, with each element comprising 8 tokens.

From Tokens to Spatiotemporal Clusters. Moving beyond the single-dimensional grouping strategies discussed above, we now consider the inherently multidimensional nature of video data by grouping neighboring $K_T \times K_H \times K_W$ tokens into spatiotemporal clusters c with no overlaps, as illustrated in Figure 2(d). This strategy results in a total number of $\hat{N} = \frac{T}{P_T K_T} \times \frac{H}{P_H K_H} \times \frac{W}{P_W K_W}$ clusters, with each containing both spatial and temporal information as an autoregressive element. ARVideo autoregressively predicts the next cluster given all preceding clusters:

$$L = -log \prod_{i=1}^{\hat{N}} p(c_i | c_1, ..., c_{i-1}, \theta) \tag{3}$$

Cluster captures the amount of high-level, semantically meaningful information, which is greater when multiple tokens are grouped together, as opposed to isolated low-level descriptors such as individual pixels or tokens.

3.3.2 SpatialTemporal Prediction Order

For the spatiotemporal cluster, we further explore its prediction order. Specifically, this strategy is expected to yield $\frac{T}{P_T K_T}$ clusters at each spatial position, and $\frac{H}{P_H K_H} \times \frac{W}{P_W K_W}$ clusters at each temporal position.

Pre-defined order. We implement two systematic strategies: a spatial-first order and a temporal-first order. The spatial-first approach prioritizes autoregressive pretraining within the $\frac{H}{P_H K_H} \times \frac{W}{P_W K_W}$ spatiotemporal clusters along the spatial dimension, before transitioning to clusters in subsequent temporal positions. Conversely, the temporal-first approach prioritizes within the $\frac{T}{P_T K_T}$ spatiotemporal clusters along the temporal dimension, then proceeds to clusters in subsequent spatial positions.

Random Rasteration. Inspired by the random sentence permutation technique used in XLNet (Yang et al., 2019) for enhancing autoregressive pretraining, our random rasterization approach scrambles the order of clusters randomly during autoregressive pretraining after adding positional embedding. By randomly reordering or sampling frame sequences, the model encounters more diverse temporal variations and avoids overfitting to a fixed frame order. Besides, random permutation makes model learn short- and long-term dependencies. Specifically, given a current frame t, the model is required to predict t+n for various n values (ranging from 1 to the final frame), thereby training it to handle both immediate and extended temporal relationships in the video. This method avoids the constraints of fixed sequential patterns, such as spatial-first or temporal-first, and allows ARVideo to adaptively model both long- and short-range spatial-temporal information. Such flexibility in autoregressive prediction orders not only captures the inherent multidimensionality of video data more effectively but also fosters a richer, more comprehensive video representation.

3.3.3 Model Architecture

We adopt the ViT (Dosovitskiy et al., 2021; Tong et al., 2022) as the encoder. For the decoder, we take the Transformer decoder with cross attention but without self-attention. This design choice aims to simplify the decoding process, emphasizing interaction between the encoded inputs while reducing training costs. The query of the decoder is randomly initialized but includes position information to facilitate sequence generation. Our model utilizes a strategically designed attention mask as in previous work (Chen et al., 2020; Radford et al., 2018) to enable efficient autoregressive prediction in a parallel computation framework. When transferring to downstream tasks, we remove the decoder and only finetune the encoder. We aggregate the encoder's output tokens using average pooling into a single, representative token. This pooled token then serves as the input for the final classification layer.

4 Experiment

4.1 Dataset and Implementation Details

We primarily evaluate ARVideo on Kinetics-400 (Kay et al., 2017) and Something-Something V2 (Goyal et al., 2017). Specifically, Kinetics-400 contains 400 classes and 260k videos of 10s, with 240k for training and 20k for validation; Something-Something V2 contains 174 classes with 169k videos for training and 25k for validation. While Kinetics-400 provides a broad spectrum of actions with minimal context, Something-Something V2 focuses more on the interaction of actions with objects.

For our experiments, we first pretrain a vanilla video Transformer (Tong et al., 2022) with ARVideo, and then fine-tune the pretrained model on the target action recognition datasets. Additionally, we assess the feature transferability on AvA v2.2 (Gu et al., 2018) and HMDB (Kuehne et al., 2011). AvA v2.2 is a human action localization dataset with 211k videos for training and 57k for validation; HMDB is a small video dataset with 3.5k videos for training and 1.5k videos for validation.

We follow the established protocol in prior work (Tong et al., 2022) to train our models. Instead of using negative log-likelihood as in GPT (Radford et al., 2018), we employ mean square error (MSE) loss to measure the discrepancy between the predicted and target cubes, as utilized in MAE (He et al., 2022). We randomly mask 80% tokens in each element in encoder to reduce the overall training costs; note that, unlike MAE or VideoMAE, we do not reconstruct those masked regions. Please refer to Appendix A for model architecture and training implementation details.

Method	Backbone	pretrain	Epoch	Frames	GFLOPs	Param	Top-1
Supervised pretraining							
TANet (Liu et al., 2021)	ResNet152	IN-1K	100	16	242×4×3	59	79.3
TDN_{En} (Wang et al., 2021)	ResNet101	IN-1K	100	8+16	198×10×3	88	79.4
TimeSformer (Bertasius et al., 2021)	ViT-B	IN-21K	15	8	196×1×3	121	78.3
Motionformer (Patrick et al., 2021)	ViT-B	IN-21K+K400	35	16	370×1×3	109	81.1
Video Swin (Liu et al., 2022a)	Swin-B	IN-21K+K400	30	32	321×1×3	88	82.7
Mask video modeling							
VIMPAC (Tan et al., 2021)	ViT-L	HowTo100M	100	10	N/A×10×3	307	77.4
BEVT (Wang et al., 2022a)	Swin-B	K400	150	32	282×1×3	88	76.2
VideoMAE (Tong et al., 2022)	ViT-B	K400	800	16	180×2×3	87	80.0
VideoMAE (Tong et al., 2022)	ViT-B	K400	1600	16	180×2×3	87	81.5
Autoregressive pretraining							
iGPT (Chen et al., 2020)	ViT-B	IN-1K	300	16	180×2×3	87	61.2
Randsac (Hua et al., 2022)	ViT-B	IN-1K	1600	16	180×2×3	87	70.3
TokenGPT†	ViT-B	IN-1K	300	16	180×2×3	87	68.5
TokenGPT†	ViT-B	K400	800	16	180×2×3	87	74.2
ARVideo	ViT-B	K400	800	16	180×2×3	87	80.1
ARVideo	ViT-B	K400	1600	16	180×2×3	87	81.2

Table 2: **Comparison with the state-of-the-art methods on Kinetics-400**. "N/A" indicates the numbers are not available for us. † indicates the implementation by us with the token replacing pixel in iGPT.

Method	Backbone	Pretrain	Epoch	Frames	GFLOPs	Param	Top-1
Supervised pretraining							
TEINet_{En} (Liu et al., 2020)	$\text{ResNet50}_{\times 2}$	IN-1K	50	8+16	99×10×3	50	66.5
TANet_{En} (Liu et al., 2021)	$\text{ResNet50}_{\times 2}$	IN-1K	50	8+16	99×2×3	51	66.0
TDN_{En} (Wang et al., 2021)	$\text{ResNet101}_{\times 2}$	IN-1K	60	8+16	198×1×3	88	69.6
SlowFast (Feichtenhofer et al., 2019)	ResNet101	K400	196	8+32	106×1×3	53	63.1
MViTv1 (Fan et al., 2021)	MViTv1-B	K400	100	64	455×1×3	37	67.7
TimeSformer (Bertasius et al., 2021)	ViT-B	IN-21K	15	8	196×1×3	121	59.5
TimeSformer (Bertasius et al., 2021)	ViT-L	IN-21K	15	64	5549×1×3	430	62.4
ViViT FE (Arnab et al., 2021)	ViT-L	IN-21K+K400	35	32	995×4×3	N/A	65.9
Motionformer (Patrick et al., 2021)	ViT-B	IN-21K+K400	35	16	370×1×3	109	66.5
Video Swin (Liu et al., 2022a)	Swin-B	IN-21K+K400	30	32	321×1×3	88	69.6
Mask video modeling							
VIMPAC (Tan et al., 2021)	ViT-L	HowTo100M	100	10	N/A×10×3	307	68.1
BEVT (Wang et al., 2022a)	Swin-B	IN-1K+K400	150	32	321×1×3	88	70.6
$\text{MaskFeat}_{\uparrow 312}$ (Wei et al., 2022)	MViT-L	K600	1600	40	2828×1×3	218	75.0
VideoMAE (Tong et al., 2022)	ViT-B	SSv2	800	16	180×2×3	87	69.6
VideoMAE (Tong et al., 2022)	ViT-B	SSv2	2400	16	180×2×3	87	70.8
Autoregressive pretraining							
iGPT (Chen et al., 2020)	ViT-B	IN-1K	300	16	180×2×3	87	54.3
Randsac (Hua et al., 2022)	ViT-B	IN-1K	1600	16	180×2×3	87	59.6
TokenGPT†	ViT-B	IN-1K	300	16	180×2×3	87	59.2
TokenGPT†	ViT-B	SSv2	800	16	180×2×3	87	66.4
ARVideo	ViT-B	SSv2	800	16	180×2×3	87	69.8
ARVideo	ViT-B	SSv2	2400	16	180×2×3	87	70.9

Table 3: **Comparison with the state-of-the-art methods on Something-Something V2**. "N/A" indicates the numbers are not available for us. † indicates the implementation by us with the token replacing pixel in iGPT.

Additionally, we would like to stress that, although AR modeling is leveraged here, the video generation quality of our pretrained model is not the focus of this study—instead, we focus on evaluating the finetuning performance of these pretrained models on a range of downstream video benchmarks.

Method	K400 → AVA v2.2	K400 → HMDB
Contrastive Learning		
MoCo	-	67.9
Mask video modeling		
VideoMAE	26.7	73.3
Autoregressive pretraining		
ARVideo	26.9	74.1

Table 4: Comparison of model transferability. We first pretrain models on Kinetics-400, and then transfer them to AVA v2.2 and HMDB.

4.2 Main results

Kinetics-400. We pretrain the ViT-B backbone for both 800 and 1600 epochs on Kinetics-400, and report the corresponding results in Table 2. Spatiotemporal cluster and random rasteration order are adopted. Notably, ARVideo attains 80.1% top-1 accuracy under 800 epochs and 81.2% top-1 accuracy under 1600 epochs, exhibiting significant improvements over previous autoregressive methods. Specifically, taking 1600-epoch-pretrained ARVideo for comparison, it outperforms iGPT, the baseline model, by a striking **+20.0%**, and Randsac, the previous state-of-the-art autoregressive model on images, by **+10.9%**. Additionally, compared to TokenGPT, which performs token-level autoregressive prediction, ARVideo showed advancements of **+12.7%** when TokenGPT was pretrained on an image dataset, and **+7.0%** when it was pretrained on the Kinetics-400 dataset.

Moreover, we note that ARVideo performs competitively against the strong benchmark—the mask video modeling method, VideoMAE. For example, the performance difference between ARVideo and VideoMAE is only 0.1% with 800 epochs of pretraining; this margin remains minimal at 0.3% with 1600 epoch pretraining. These results validate the effectiveness of ARVideo as a pioneering autoregressive pretraining method in self-supervised video representation learning, equalling—and in some aspects surpassing—the performance of established mask modeling methods.

Something-Something V2. We pretrain the ViT-B backbone for 800 and 2400 epochs on the Something-Something V2 dataset. Spatiotemporal cluster and random rasteration order are adopted. As reported in Table 3, ARVideo achieves top-1 accuracies of 69.8% and 70.9% for 800 and 2400 epochs, respectively, which are significantly stronger than prior autoregressive pretraining methods. For example, under 2400 epochs, ARVideo surpassed the baseline model iGPT by **+16.6%** and outperforms the best-performing image-based autoregressive method, Randsac, by **+11.3%**. It also surpassed TokenGPT pre-trained on image datasets by +11.7% and on the Something-Something V2 dataset by +4.5%. Additionally, when compared to the strong masked video modeling method VideoMAE, ARVideo also performs competitively in both 800 epochs of pretraining (*i.e.*, 0.2% accuracy difference) and 2400 epochs of pretraining (*i.e.*, 0.1% accuracy difference). Together with the observations in Kinetics-400, these results can establish ARVideo as a strong alternative to masked modeling approaches for video analysis.

Transfer Learning. To investigate the feature transferability of ARVideo, we transfer the model trained on Kinetics-400 to AvA v2.2 and HMDB. We can observe that ARVideo demonstrate strong transferability, achieving 26.9 mAP on AvA v2.2 and 74.1% Top-1 accuracy on HMDB—outperforming both VideoMAE and MoCo (see Table 4). For example, compared to VideoMAE, ARVideo shows (slight) improvements of 0.2% on AvA v2.2 and 0.8% on HMDB.

Computation cost. We report the training time and GPU memory usage in Table 5 (with ViT-B trained on Kinetics-400 for 800 epochs, using 8×A6000). Compared to VideoMAE, ARVideo presents significant reductions in both GPU memory usage and training time—ARVideo reduces training cost by 12.4% (from 145 hours to 127 hours) and GPU memory consumption by 36.8% (from 41.3G to 26.1G). This advantage stems from ARVideo's shorter sequence length as we drop the last cluster in the autoregressive modeling.

Attention rank. The self-attention mechanism computes attention scores for a given input sequence, forming what is known as the attention map. The rank of this matrix can serve as a measure of its ability to capture

Method	Encoder		Decoder		Training Time	GPU Memory
	Q	Key/Value	Q	Key/Value		
VideoMAE	160	160	1568	1568	145h	41.3G
ARVideo	300	300	1372	300	**127h** (-12.4%)	**26.1G** (-36.8%)

Table 5: The comparison of pretraining time and GPU memory.

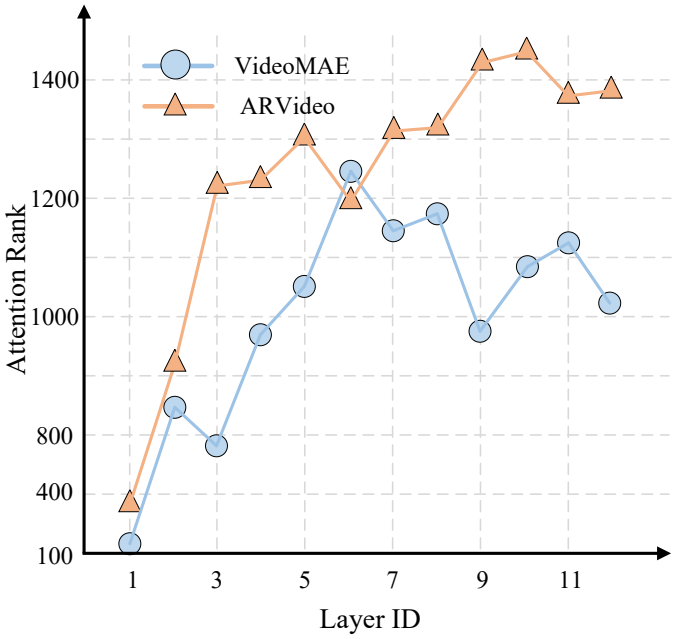

Figure 3: The attention rank comparison between VideoMAE and ARVideo

complex patterns in the data. Typically, high-rank attention matrices suggest a model that can capture a wide variety of patterns and relationships within the data, while low-rank matrices may suggest a model that does not well utilize its full capacity or operates on simpler data (Wang et al., 2020). Following this instruction, we plot the rank of the attention map in each layer of VideoMAE and our ARVideo in Figure 3. We can observe that, across nearly all layers except the 6_{th}, ARVideo maintains higher attention ranks than VideoMAE, indicating a stronger representational ability of our model's self-attention layers.

Compatibility with recent works. We highlight that our proposed ARVideo is compatible with several recent works (Yang et al., 2022; Wang et al., 2023) that incorporate extra motion cues (Yang et al., 2022) or extra knowledge distillation (Wang et al., 2023) to VideoMAE. These methods are orthogonal to both VideoMAE and our ARVideo. As shown in Table 7, ARVideo can be effectively combined with these methods, yielding consistent improvements. Specifically, ARVideo with motion cues achieves a 0.6% improvement over VideoMAE with motion cues (from 71.8% to 72.4%). Similarly, ARVideo with distillation outperforms VideoMAE with distillation by 0.8% (from 73.7% to 74.5%).

4.3 Ablation Study

In this part, we ablate four factors—cluster shape, mask ratio, prediction order, and decoder design. Note that, unless otherwise specified, all ablations are conducted on the ViT-B backbone with 200 epochs of pretraining.

Cluster shape. We group neighboring and non-overlapped $K_T \times K_H \times K_W$ tokens into one cluster and analyze the effect of different cluster shapes. Three situations are considered: 1) $K_T = K_W = K_H = 1$, equivalent

case	K_T	K_H	K_W	Something-Something V2
Token/Cube	1	1	1	64.0
spatial cluster	1	$\frac{H}{P_H}$	$\frac{H}{P_H}$	66.0
spatial cluster	1	7	7	66.2
temporal cluster	$\frac{T}{P_T}$	1	1	65.2
temporal cluster	2	1	1	65.6
spatiotemporal cluster	4	7	7	65.5
spatiotemporal cluster (ARVideo)	2	7	7	**66.8**

Table 6: Ablation study on the cluster shape.

Method	Component	Something-Something V2
VideoMAE	VideoMAE	70.8
MotionMAE(Yang et al., 2022)	VideoMAE+Motion	71.8
MVD(Wang et al., 2023)	VideoMAE+Distillation	73.7
ARVideo	ARVideo	70.9
MotionARVideo	ARVideo+Motion(Yang et al., 2022)	72.4
DistillARVideo	ARVideo+Distillation(Wang et al., 2023)	74.5

Table 7: Compatible with recent self-supervised video representation learning framework.

to the TokenGPT, which pertains autoregressively at the token/cube level; 2) $K_T = \frac{T}{P_T}, K_W = K_H = 1$, representing a temporal cluster; and 3) $K_T = 1, K_W = \frac{W}{P_W}, K_H = \frac{H}{P_H}$, representing a spatial cluster.

We report the results in Table 6. Firstly, we can observe that all clustered configurations significantly enhance performance over the TokenGPT baseline. For example, simply grouping tokens into spatial/temporal/spatiotemporal clusters yields 2.0%/2.2%/2.8% improvements, respectively. Then, when comparing different clusters, we note that our spatiotemporal cluster (ARVideo) with $K_T = 2, K_W = K_H = 7$ attains the best performance of 66.8%, outperforming the best-performed spatial cluster ($K_T = 1, K_W = K_H = 7$) by 0.8% and the best-performed temporal clusters ($K_T = 2, K_W = K_H = 1$) by 1.2%. However, it is interesting to note that, if a poorly designed spatiotemporal cluster ($K_T = 4, K_W = K_H = 7$) is used, the performance will drop to 65.5%.

Order	SSv2
Spatial-First	65.6
Temporal-First	66.0
Spatial-temporal random	**66.8**

Table 8: Ablation study on the prediction order.

Mask Ratio	SSv2
75%	66.0
80%	66.8
90%	65.6
95%	64.8

Table 9: Ablation study on the mask ratio from 75% to 95%.

Prediction order. In our evaluation of prediction order, which plays an important role in constructing the video sequence, we first check with the predefined spatial-first and temporal-first orders. As shown in Table 8, temporal-first order achieves 66.0% top-1 accuracy, which is 0.4% higher than spatial-first order. However, our randomized spatial-temporal prediction order, adept at learning both long- and short-range spatial-temporal dynamics, exhibits a superior performance of 66.8%, surpassing the predefined spatial-first approach by 1.2% and the temporal-first approach by 0.8%.

Mask Ratio. To reduce the temporal redundancy, ARVideo randomly masks a portion of tokens as in Flip (Li et al., 2023).This approach differs from VideoMAE, which relies on bidirectional information to reconstruct masked regions. Instead, ARVideo maintains its core AR paradigm by predicting the next cluster in a unidirectional manner. We hereby check how the masking ratio affects the overall performance. As

Method	Decoder		Something-Something V2
	Self-Atten	Cross-Atten	
ARVideo		✓	66.8
ARVideo	✓	✓	66.6

Table 10: Ablation study on the decoder architecture.

Decoder Width	Decoder Depth	Something-Something V2
384	4	66.0
512	4	**66.8**
768	4	66.8
512	2	66.2
512	4	**66.8**
512	8	66.6

Table 11: Ablation study on the decoder depth and width.

shown in Table 9, our study starts from a mask ratio of 75% (*i.e.*, same as the MAE's setup), which achieves 66.0% top-1 accuracy. Increasing the mask ratio to 80% boosted the top-1 accuracy to 66.8%, while further increases to 90% and 95% lower the top-1 accuracies by 1.2% and 2.0%, respectively. We stress that, although ARVideo used a lower mask ratio than VideoMAE, it still enjoys faster training speeds and reduced GPU load (see Section 4.2 and Table 5).

Decoder Architecture. We hereby explore the effects of different decoder architectures. As reported in Table 10, whether or not having self-attention in the decoder has little effect on performance (*i.e.*, 66.6% *vs.* 66.8%), but excluding self-attention significantly reduces computational costs. Therefore, we take the decoder without self-attention by default in ARVideo.

Decoder Width and Depth. Lastly, we systematically ablate two critical aspects in designing decoders: its *width* and *depth*. We start with a four-layer decoder and follow the default setup in VideoMAE. As presented in Table 11, increasing the decoder width shows performance improvement from 66.0% at a width of 384 to 66.8% at a width of 512. Further width increase makes the performance plateau. Meanwhile, in terms of depth, deviations from the four-layer standard negatively impacted performance: *e.g.*, increasing to eight layers decreased performance by 0.2%, while reducing to two layers dropped performance by 0.6% (see the last three rows in Table 11).

Uni-direction vs. Bi-direction. We employ unidirectional attention during the pretraining phase to preserve the autoregressive property, ensuring that each cluster only attends to its preceding clusters. When we replaced this with bidirectional attention during pretraining, we observed a significant performance drop of 16.9% top-1 accuracy as shown in Table 12. This decline is attributed to information leakage, where tokens can access both past and future contexts, thereby disrupting the sequential dependencies essential for effective AR modeling. For downstream tasks, we utilize bidirectional attention to leverage comprehensive video representations for classification. This ablation highlights that unidirectional attention is vital during pretraining to maintain the integrity of AR modeling,

Pretraining	Finetuning	Something-Something V2
Uni-direction Attention	Bi-direction Attention	66.8
Bi-direction Attention	Bi-direction Attention	49.9

Table 12: Ablation study on the Attention.

Case	Patch size	Something-Something V2
Token	16×16 ×2	64.0
Token	32×32 ×4	53.2
Token	114×114 ×4	48.9 (-15.1)
Spatiotemporal cluster	16×16 ×2	71.0(+4.6)

Table 13: Ablation study on tokenization with larger patch size.

Tokenization vs. Clustering. An alternative to clustering is to use tokenization with a larger patch size, thus mapping more pixels into a single token. However, as shown in Table 13, enlarging the patch size to $114 \times 114 \times 4$ results in a notable 15.1% performance drop, whereas our spatiotemporal clustering approach yields a 4.6% improvement.

5 Conclusion

In this paper, we introduce ARVideo for self-supervised video representation learning, inspired by the autoregressive principles of GPT in natural language processing. Diverging from conventional methods, our approach innovatively uses video token clusters as the element for autoregressive prediction, significantly reducing computational demands while still managing to capture essential spatial-temporal dynamics. This advancement improves the efficiency of video data processing and sets a new paradigm for self-supervised video representation learning. The promising results obtained from ARVideo underscore its potential and advocate for further exploration and development of autoregressive pretraining methods within the video domain.

Future work. Building upon our current framework, we believe it would be both interesting and promising to further scale our training data to a billion-scale dataset and enhance compatibility with LLMs. These advancements aim to improve the scalability and versatility of our approach, potentially leading to more robust performance and broader applications in video analysis and understanding. Additionally, integrating our method with LLMs could open new avenues for multimodal learning and more sophisticated video interpretation tasks, thereby increasing the overall impact of our research in the field.

References

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *ICLR*, 2022.

Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9922–9931, 2020.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Ali Diba, Vivek Sharma, Reza Safdari, Dariush Lotfi, Saquib Sarfraz, Rainer Stiefelhagen, and Luc Van Gool. Vi2clr: Video and image for visual contrastive learning of representation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1502–1512, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. *ICML*, 2024.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *European conference on computer vision*, pp. 312–329. Springer, 2020a.

Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. *Advances in Neural Information Processing Systems*, 33:5679–5690, 2020b.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Tianyu Hua, Yonglong Tian, Sucheng Ren, Michalis Raptis, Hang Zhao, and Leonid Sigal. Self-supervision through random segments with autoregressive coding (randsac). In *The Eleventh International Conference on Learning Representations*, 2022.

Deng Huang, Wenhao Wu, Weiwen Hu, Xu Liu, Dongliang He, Zhihua Wu, Xiangmiao Wu, Mingkui Tan, and Errui Ding. Ascnet: Self-supervised video representation learning with appearance-speed consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8096–8105, 2021.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *ICML*, 2024.

Haofei Kuang, Yi Zhu, Zhi Zhang, Xinyu Li, Joseph Tighe, Sören Schwertfeger, Cyrill Stachniss, and Mu Li. Video contrastive learning with global context. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3195–3204, 2021.

Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

Rui Li, Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Motion-focused contrastive learning of video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2105–2114, 2021.

Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. Scaling language-image pre-training via masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23390–23400, 2023.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *CVPR*, 2022a.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3202–3211, 2022b.

Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. TEINet: Towards an efficient architecture for video recognition. In *AAAI*, 2020.

Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. In *ICCV*, 2021.

OpenAI. Gpt-4 technical report, 2023.

Mandela Patrick, Dylan Campbell, Yuki Asano, Ishan Misra, Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and Joao F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In *NeurIPS*, 2021.

Yu Qi, Fan Yang, Yousong Zhu, Yufei Liu, Liwei Wu, Rui Zhao, and Wei Li. Exploring stochastic autoregressive image modeling for visual representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2074–2081, 2023.

Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge J. Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *CVPR*, 2021.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Kanchana Ranasinghe, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Michael S Ryoo. Self-supervised video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2874–2884, 2022.

Sucheng Ren, Zeyu Wang, Hongru Zhu, Junfei Xiao, Alan Yuille, and Cihang Xie. Rejuvenating i-gpt for scalable visual representation learning. In *ICML*, 2024.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *NeurIPS*, 2014.

Hao Tan, Jie Lei, Thomas Wolf, and Mohit Bansal. Vimpac: Video pre-training via masked token prediction and contrastive learning. *arXiv preprint arXiv:2106.11250*, 2021.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35: 10078–10093, 2022.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE TPAMI*, 2019.

Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. TDN: Temporal difference networks for efficient action recognition. In *CVPR*, 2021.

Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. Bevt: Bert pretraining of video transformers. In *CVPR*, 2022a.

Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. Bevt: Bert pretraining of video transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14733–14743, 2022b.

Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Lu Yuan, and Yu-Gang Jiang. Masked video distillation: Rethinking masked feature modeling for self-supervised video representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6312–6322, 2023.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, 2022.

Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, 2019.

Haosen Yang, Deng Huang, Bin Wen, Jiannan Wu, Hongxun Yao, Yi Jiang, Xiatian Zhu, and Zehuan Yuan. Self-supervised video representation learning with motion-aware masked autoencoders. *arXiv preprint arXiv:2210.04154*, 2022.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *ICLR*, 2024.

A Implementation Details

All experiments except main results follow the same hyperparameters including using the ViT-B backbone with depth of 12 and channel of 768 with 200 epochs of pretraining. TokenGPT indicates we perform token wise autoregressive modeling without any grouping or permutation. TokenGPT is a simple baseline of autoregressive modeling for video self-supervised learning.

Architecture: We utilize ViT-B as the backbone of our model. A patch embedding layer with dimensions $2 \times 16 \times 16$ is employed to map the video input into video tokens. The encoder comprises 12 blocks, each containing an attention layer and an MLP layer, with each block having a channel size of 768. Conversely, the decoder consists of 4 blocks, each featuring a cross-attention layer and an MLP layer, with a channel size of 512.

Training Hyperparameters: We employ the AdamW optimizer with a weight decay of 0.05 and a base learning rate of 6e-4. The training schedule comprises a 40-epoch warmup phase followed by a cosine decay learning rate schedule.

Finetuning Hyperparameters: During finetuning, we remove the decoder and fine-tune the entire encoder. For the Something-Something V2 dataset, we employ the AdamW optimizer with a base learning rate of 5e$-$4 and a weight decay of 0.05. The batch size is set to 512, and we utilize a cosine decay learning rate schedule with 5 warmup epochs over a total of 40 training epochs. Our data augmentation strategies include repeated augmentation (factor of 2) and RandAugment with parameters (9, 0.5), while flip augmentation is disabled. Additionally, we apply label smoothing (0.1), mixup (0.8), and cutmix (1.0). The drop path rate is configured at 0.1, and no dropout is applied for Something-Something V2. The layer-wise learning rate decay factor is set to 0.75.

For the Kinetics-400 dataset, most settings remain unchanged except for the following adjustments: the base learning rate is increased to $1e^{-3}$, flip augmentation is enabled, and the total number of training epochs is extended to 75.