Masked Autoencoders As Spatiotemporal Learners

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper studies a conceptually simple extension of Masked Autoencoders (MAE) [31] to spatiotemporal representation learning from videos. We randomly mask out spacetime patches in videos and learn an autoencoder to reconstruct them in pixels. Interestingly, we show that our MAE method can learn strong representations with *almost no inductive bias* on spacetime (only except for patch and positional embeddings), and spacetime-*agnostic* random masking performs the best. We observe that the optimal masking ratio is as high as 90% (*vs*. 75% on images [31]), supporting the hypothesis that this ratio is related to information redundancy of the data. A high masking ratio leads to a large speedup, *e.g.*, $> 4\times$ in wall-clock time or even more. We report competitive results on several challenging video datasets using vanilla Vision Transformers [18]. We observe that MAE can outperform supervised pre-training by large margins. We further report encouraging results of training on real-world, uncurated Instagram data. Our study suggests that the general framework of masked autoencoding (BERT [15], MAE [31], *etc*.) can be a unified methodology for representation learning with minimal domain knowledge.

1 Introduction

The deep learning community is experiencing a trend of unifying methodologies for solving problems in different areas, such as language, vision, speech, and more. For architectures, Transformers [67] have been successfully introduced into computer vision [18] and established as a general building block in both language and vision. For self-supervised representation learning, the *denoising/masked autoencoding* methodology [68] in BERT [15] has been shown effective on learning visual representations from images [31]. Towards unifying methodologies, less domain knowledge ("fewer inductive biases" [18]) is introduced for a specific problem, which urges the models to learn useful knowledge almost purely from data.

Following this philosophy, we study extending Masked Autoencoders (MAE) [31] to the problem of spatiotemporal representation learning. Our method is simple: we randomly mask out spacetime patches in videos and learn an autoencoder to reconstruct them (Fig. 1). Our method has *minimal* domain knowledge: the only spacetime-specific inductive bias is on embedding the patches and their positions; all other components are *agnostic* to the spacetime nature of the problem. In particular, our encoder and decoder are both vanilla Vision Transformers [18] with no factorization or hierarchy, and our random mask sampling is agnostic to the spacetime structures. Our method predicts pixel values and uses no extra problem-specific tokenizer. In a nutshell, our method is simply MAE applied to the set of spacetime patches. Despite minimal inductive biases, our method achieves strong empirical results, suggesting that useful knowledge can be *learned from data*.

It is hypothesized in [31] that the masking ratio (*i.e.*, percentage of removed tokens) in masked autoencoding methods is related to the information redundancy of the problems. For example, natural images are more information-redundant than languages and thus the optimal masking ratio is higher (*e.g.*, than BERT [15]). Our observations on video data support this hypothesis. We find that the optimal masking ratio of MAE is 90% for videos (Fig. 2), higher than the masking ratio of 75% for its image counterpart [31]. This can be understood as a consequence of natural video data being temporally correlated. To the extreme, if a video has T identical static frames, randomly sampling $1/T$

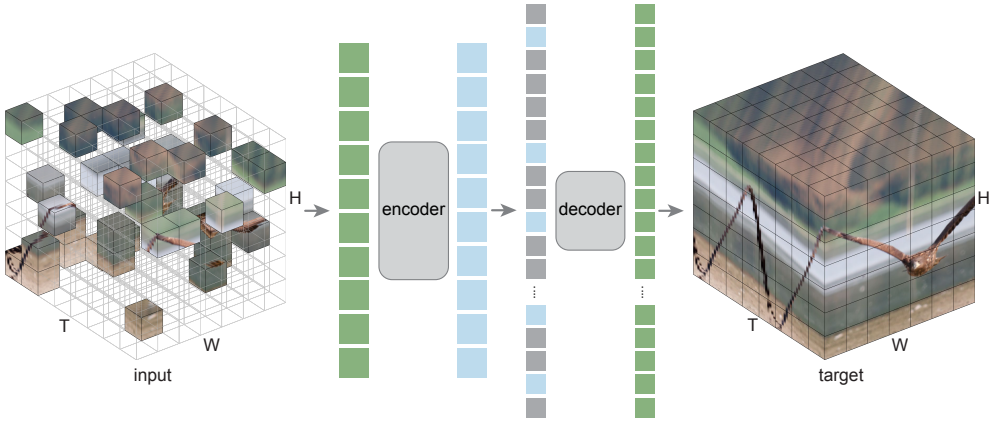


Figure 1: **Masked Autoencoders as spatiotemporal learners**. We mask a large subset (*e.g.*, 90%) of random patches in spacetime. An encoder operates on the set of visible patches. A small decoder then processes the full set of encoded patches and mask tokens to reconstruct the input. Except for patch and positional embeddings, *neither the encoder, the decoder, nor the masking strategy, has any spatiotemporal inductive bias*.

of all spacetime patches would reveal most of the static frame. Because slow motion is more likely than fast motion in natural videos, the masking ratio can be very high as we observe empirically.

The higher masking ratio leads to a more efficient solution in practice. Following the MAE in [31] that applies the encoder only on visible tokens, a masking ratio of 90% reduces the encoder time and memory complexity to $<1/10$. Put together with a small decoder [31], the MAE pre-training can achieve a theoretically $7.7\times$ reduction in computation *vs*. encoding all tokens. In fact, the computation reduction is so large that the data loading time becomes a new bottleneck; even so, we record a $4.1\times$ wall-clock speedup. Such a significant speedup is of great importance for video research that is large-scale and time-consuming.

We report strong results on a variety of video recognition datasets. Our MAE pre-training greatly improves generalization performance: on Kinetics-400 [35], it increases the accuracy of ViT-Large [18] by absolute 13% *vs*. training from scratch, while it takes *less* wall-clock training time overall (pre-training plus fine-tuning). Our MAE pre-training can outperform its supervised pre-training counterpart by big margins. Using vanilla ViT [18], our method achieves competitive results with previous state-of-the-art methods that incorporate more domain knowledge. We also report encouraging results using MAE pre-trained on 1 million random, *uncurated* Instagram videos. These results suggest that self-supervised learning on videos can be tackled in a way similar to its counterparts on language [15] and images [31], under a unified framework.

2 Related Work

Denoising autoencoders. Denoising autoencoders (DAE) [68, 69] present a general methodology for learning representations by reconstructing clean signals from corrupted inputs. Masking as a type of noise dates back to at least a decade ago [69]. One of its most successful developments is BERT [15], which is conceptually masked autoencoding on language tokens.

Denoising/masked autoencoding methods for computer vision have been making continuous progress [50, 9, 18, 31]. A series of recent methods are based on Transformer architectures [67] and are towards a unified solution between vision and language. iGPT [9] pioneers this direction by training Transformers on pixels as tokens. The ViT paper [18] makes a revolutionary step forward by using patches as tokens. It not only establishes strong Transformer architectures for vision tasks, but also explores masked prediction with patches. MAE [31] returns to the basics of the autoencoding concept [68] and draws attention to the decoding aspect. The presence of a meaningful decoder provides more flexibility, *e.g.*, enabling the encoder to operate only on visible patches and leading to a more efficient solution. It empirically shows that a high masking ratio is essential for image tasks [31]. Our study follows this line of research.

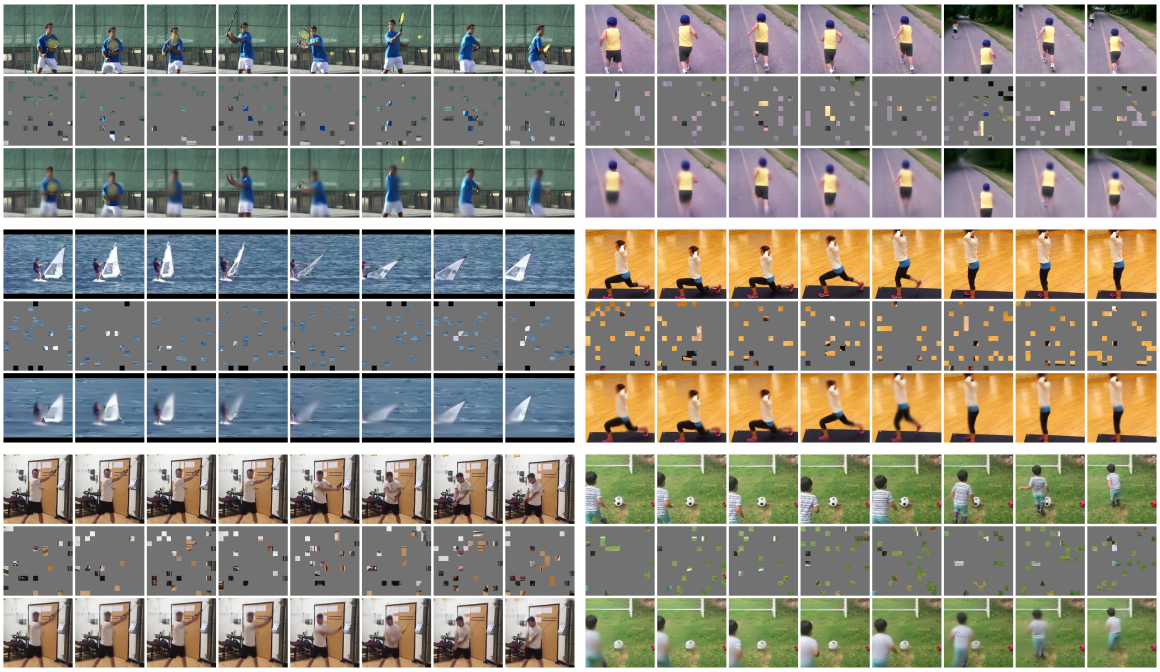


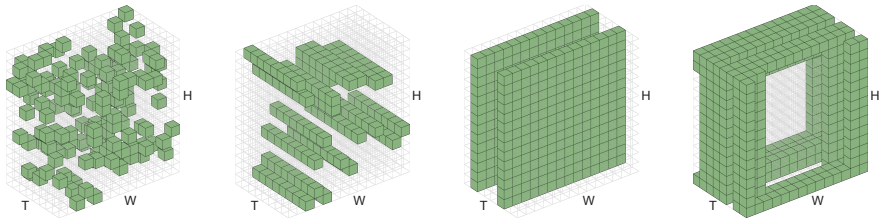
Figure 2: Visualizations on the Kinetics-400 [35] validation set (masking ratio **90%**). We show the original video (top), masked video (middle), and MAE output (bottom) for each sample. This model reconstructs the original pixels. The video size is $16\times224\times224$ and the spacetime patch size is $2\times16\times16$ (the temporal patch size of 2 is not visualized here). Each sample has $8\times14\times14{=}1568$ tokens with 156 being visible. For better visualizations, the known patches in the output are from the original input. Fig. 7 shows more examples.

Figure 3: Visualizations of the same pre-trained model in Fig. 2 but with a masking ratio of **95%**.

Instead of predicting pixels [9, 18, 31, 80], another line of research focuses on the tokenization of the prediction targets [3, 17, 77]. BEiT [3] proposes to use pre-trained dVAE [47, 55] as the reconstruction target. The dVAE tokenizer can be improved by perceptual or adversarial losses [17]. MaskFeat [77] shows that HoG [13] as prediction targets performs strongly.

Self-supervised learning on videos. The presence of the temporal dimension is a focus of self-supervised learning on video data. Related topics include temporal coherence ('slowness') [79, 25], future prediction [61, 72, 70, 45, 44, 71, 16], object motion [1, 75, 49, 76], temporal ordering [46, 23, 38, 78, 81], spatiotemporal contrast [58, 62, 30, 22, 51, 56], *etc*.

Our method also relies on the temporal coherence of videos, but it approaches this goal implicitly. In fact, as our method is largely agnostic to spacetime, the main opportunity for it to make use of



(a) agnostic, 90% (b) space-only, 90% (c) time-only, 75% (d) block, 75%

Figure 4: **Mask sampling**. **(a)**: Random sampling that is spacetime-*agnostic*. **(b)**: Space-only random sampling, broadcasted to all time steps ("tube" masking [77]). **(c)**: Time-only random sampling, broadcasted to all spatial locations ("frame" masking [77]). **(d)**: Block-wise sampling [3] in spacetime, removing large regions ("cube" masking [77]). In this illustration, $T \times H \times W$ is $8 \times 14 \times 14$; green tokens are kept and others are masked out.

the temporal coherence is a *higher* masking ratio (*e.g.*, 90%), which assumes that videos are more information-redundant than images.

There has been growing interest in masking-based methods for self-supervised learning on videos. Previous works focus on tokenizing the prediction targets for the use of videos [65, 73, 77]. Our autoencoding method operates on pixels, which is simpler and requires no extra data or domain knowledge on the tokenizer. Importantly, our method greatly improves the *efficiency* of learning. The practical speedup is of central importance for video-related research, which is in general larger-scale and more time-consuming.

Our work is done independently and concurrently with [66] on a related method.

3 Method

Our method is a simple extension of MAE [31] to spacetime data (Fig. 1). Our goal is to develop the method under a general and unified framework, with as little domain knowledge as possible.

Patch embedding. Following the original ViT [18], given a video clip, we divide it into a regular grid of non-overlapping patches in spacetime [4, 2, 19, 77]. The patches are flattened and embedded by linear projection [18]. Positional embeddings [67] are added to the embedded patches. The patch and positional embedding process is the only process that is spacetime-aware.

Masking. We sample random patches without replacement from the set of embedded patches. This random sampling is *agnostic* to the spacetime structure (Fig. 4 (a)). This structure-agnostic sampling strategy is analogous to that of BERT in 1D [15] and MAE in 2D [31].

It is hypothesized in [31] that the optimal masking ratio is related to the information redundancy of the data. With unstructured random masking, BERT [15] uses a masking ratio of 15% for language and MAE [31] uses a ratio of 75% for images, suggesting that images are more information-redundant than language. Our empirical results on videos support this hypothesis. The optimal masking ratio we observe is 90%. This is in line with the common assumption that natural videos are more information-redundant than images because of temporal coherence. Fig. 2 and 3 present our MAE reconstruction results on unseen validation data with a masking ratio of 90% and 95%.

The spacetime-agnostic sampling can be more effective than structure-aware sampling strategies, *e.g.*, *space-only*, *time-only*, or *block-wise* sampling (Fig. 4 (b-d)). As neighboring patches in space or in time (Fig. 4(b, c)) are coherent, with a very high masking ratio, space-only or time-only sampling may retain less information and yield an overly difficult pre-training task. For example, time-only sampling from 8 frames with a masking ratio of 87.5% means keeping only a single frame, which presents an overly challenging task of predicting the future and past given only one frame. We observe that optimal masking ratios for structure-aware sampling are in general lower. In contrast, the spacetime-agnostic sampling better utilizes the limited number of visible patches and thus allows to use a higher masking ratio.

Autoencoding. Our encoder is a vanilla ViT [18] applied only on the visible set of embedded patches, following [31]. This design greatly reduces time and memory complexity and leads to a

more practical solution. A masking ratio of 90% reduces the encoder complexity to $<1/10$ (noting that self-attention is quadratically-complex w.r.t. the token set size).

Our decoder is another vanilla ViT on the union of the encoded patch set and a set of mask tokens [31]. Decoder-specific positional embeddings are added to this set [31]. The decoder is designed to be smaller than the encoder [31]. Although the decoder processes the full set, its complexity is smaller than the encoder (*e.g.*, $\sim 1/20$ per token). In our default setting, the overall autoencoder has a complexity reduction of $7.7\times$ *vs.* full encoding (more discussions are in Sec. 5.1 and Table 1).

The decoder predicts the patches in the *pixel* space. In principle we can simply predict a full spacetime patch (*e.g.*, $t\times16\times16$); in practice, we find it sufficient to predict a single time slice of the patch (16×16), which keeps the prediction layer's size manageable. We predict the original pixels or their per-patch normalized values [31] (compared in Table 2b). The training loss function is the mean squared error (MSE) between the prediction and its target, averaged over unknown patches [15].

The encoder and decoder are agnostic to the spacetime structure of the problem. There is *no* hierarchy or spacetime factorization, in contrast to the leading architectures [4, 2, 19]. Our method relies on the global self-attention to learn useful knowledge from data, following the spirit of [18].

4 Implementation

Data pre-processing. For MAE pre-training, our default input size is 16 frames each with 224×224 pixels (*i.e.*, $16\times224\times224$). The 16 frames are sampled from the raw video with a temporal stride of 4 (*i.e.*, 16×4 sampling in the literature [21]), and the starting frame is randomly sampled. In the spatial domain, we perform random resized cropping [63] with a scale range of $[0.5, 1]$, and random horizontal flipping. We do *not* apply other data augmentations unless noted.

Our MAE pre-training is so fast in computation that data loading becomes a new bottleneck that dominates running time in our setup. We adopt *repeated sampling* [33][1] to alleviate this problem. Each time a raw video is loaded and decompressed, we take multiple (4 by default) samples from it. This reduces the data loading and decompressing time per sample. We note that repeated sampling does *not* change the number of samples seen; it only influences the *orders* of the samples seen during training. We always count epochs as "effective epochs", *i.e.*, how many times each raw video is sampled throughout training.

Architecture. Our encoder and decoder are the *vanilla* ViT architectures [18]. We use a temporal patch size of 2 [2, 19, 77] and a spatial patch size of 16×16 [18], denoted as $2\times16\times16$. We use the same patch size for ViT-B/L/H [18] for simplicity. For a $16\times224\times224$ input, this patch size produces $8\times14\times14$ tokens.

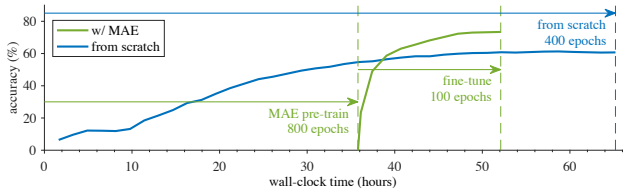
We adopt separable positional embeddings for the encoder. We have two positional embeddings, one for space and the other for time. The spacetime positional embeddings are the sum of them. This separable implementation prevents the size of positional embeddings growing too large in 3D. We use learnable positional embeddings; the sin-cos variant [67] works similarly.

Settings. Our MAE pre-training configuration mostly follows [31]. We use the AdamW optimizer [43] with a batch size of 512. We evaluate the pre-training quality by end-to-end fine-tuning. The choice of evaluating by fine-tuning (instead of linear probing) follows [3, 31]. Our inference process follows the common practice of multi-view testing [74, 21]: it takes K temporal clips (by default $K{=}7$ on Kinetics) to cover the video length, and for each clip it takes 3 spatial views to cover the longer spatial axis (denoted as $K\times3$). The final prediction is the average of all views. The implementation details and hyper-parameters are in the appendix.

5 Experiments

In Sec. 5.1 and Sec. 5.2 we perform ablation experiments on Kinetics-400 (K400) [35]. We do MAE self-supervised pre-training and then fine-tune the encoder with supervision for evaluation. We report top-1 classification accuracy (%) on the K400 validation set. In Sec. 5.3 we study more pre-training datasets and downstream tasks.

[1] In our use case, repeated sampling involves data augmentation and mask sampling.



	scratch	MAE
1-view	60.7	**73.4** (+12.7)
multi-view	71.4	**84.4** (+13.0)

Figure 5: MAE pre-training plus fine-tuning is *much more accurate* and *faster* than training from scratch. Here the x-axis is the wall-clock training time (128 A100 GPUs), and the y-axis is the 1-view accuracy on Kinetics-400 validation. The table shows the final accuracy. The model is ViT-L.

MAE w/	acc.	FLOPs	compute	load+compute
encoder w/ `[M]`	84.3	627.5 G	141.1 hr	147.5 hr
encoder w/o `[M]`	84.4	81.0 G	24.5 hr	35.8 hr
gain		7.7×	5.8×	4.1×

Table 1: **Training time comparison** between a dense encoder (w/ `[M]`) and a sparse encoder (w/o `[M]`) in MAE. The encoder is ViT-L (1024-d, 24-block); the decoder is our default (512-d, 4-block). With a masking ratio of 90%, the sparse variant reduces FLOPs by 7.7×. This reduces computation time by 5.8×. In our infra, computation is so fast that data loading becomes a bottleneck, which leads to an actual speedup of 4.1×. Profiling is with synchronized SGD over 16 nodes, each with 8 A100 GPUs and 80 CPU cores. The training length is 800 epochs.

5.1 Performance

Fig. 5 compares MAE pre-training *vs*. no pre-training (*i.e*., training from scratch), using vanilla ViT-L [18]. The from-scratch recipe follows [77] and has 71.4% accuracy.[2] As a comparison, using MAE pre-training for 800 epochs, the same vanilla ViT-L achieves 84.4% accuracy, which has a large increase of **13.0%** absolute *vs*. training from scratch. This gap is much larger than that on image recognition tasks (~3% [31]), suggesting that MAE pre-training is more helpful for video recognition.

In addition to the accuracy gain, MAE pre-training can *reduce* the overall training cost, as plotted in Fig. 5. The 800-epoch MAE pre-training only takes 35.8 hours. A short fine-tuning (100 epochs here), which takes 16.3 hours, achieves good accuracy thanks to pre-training. The overall training time can be *shorter* than training from scratch (*e.g*., 400 epochs, 65.2 hours), which converges more slowly without pre-training. This shows that MAE is a practical solution to video recognition.

MAE pre-training is fast because its encoder is only applied on the sparse set of visible patches, without the mask token `[M]`. We profile the pre-training performance in Table 1. With a masking ratio of 90%, the sparse encoder reduces the FLOPs (floating-point operations) by >10×. After counting the decoder, the sparse design of MAE reduces FLOPs by 7.7×. In our implementation, this reduction should produce a 5.8×computational speedup, if the video data *were* already pre-processed and loaded in memory. Our speedup ratio is *so high* that the video pre-processing and loading time becomes a new bottleneck. In our system, the data loading step increases the wall-clock training time from 24.5 hours to 35.8 hours. Nevertheless, this still leads to a significant speedup of 4.1×.[3]

5.2 Ablation experiments

Masking ratio. Fig. 6 shows the influence of the masking ratio jointly with the pre-training length. The ratio of 90% works the best. The ratio of 95% performs surprisingly well, which can catch up if trained long enough (Fig. 6 left). A higher masking ratio leads to *fewer* tokens encoded by the encoder; to have a more comprehensive look, we plot the results w.r.t. the total number of encoded tokens (Fig. 6 right). Under this measure, the ratios of 90% and 95% perform closely.

The lower masking ratios of 75% and 50% perform worse, even though the encoder sees more tokens and has higher computation cost. The ratio of 75% is optimal for its image counterpart [31], but not for videos. This observation can be explained by the assumption that video data is more information-redundant.

[2]The ViT-B result is 68.5% [77] trained from scratch using this recipe.

[3]The speedup is closer to 5.8× if using *slower* GPUs (V100 instead of A100) that can hide the loading time.

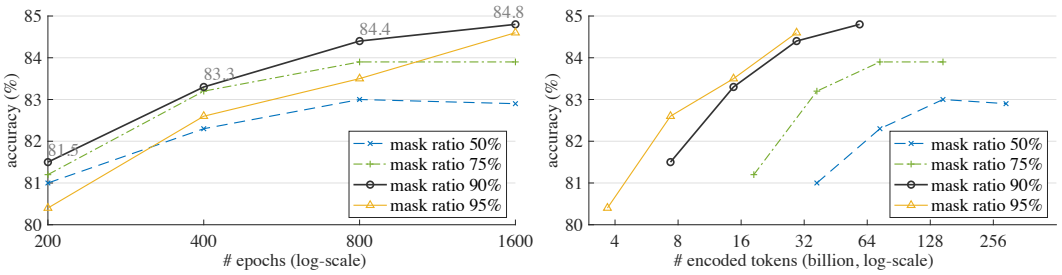


Figure 6: **Masking ratio**. Every point represents a single pre-training and fine-tuning experiment. **Left**: x-axis is the epochs (proportional to the number of *decoded* tokens). **Right**: x-axis is the number of *encoded* tokens.

case	ratio	acc.
agnostic	90	**84.4**
space-only	90	83.5
time-only	75	79.1
block	75	83.2

(a) **Mask sampling**. See also Fig. 4. Random sampling that is spacetime-*agnostic* works the best.

case	acc.
pixel (w/o norm)	83.8
pixel (w/ norm)	**84.4**
HOG	84.0
dVAE token	83.8

(b) **Reconstruction target**. Pixels as reconstruction targets work well with no domain knowledge.

case	acc.
center crop	83.9
rand crop	**84.4**
rand crop (stronger)	83.4
rand crop + color jit	83.8

(c) **Data augmentation**. Strong augmentation is unnecessary.

rep.	acc.	speed
1	83.7	1.0×
2	84.3	1.8×
4	**84.4**	**3.0**×

(d) **Repeated sampling**. All entries see the same # samples. Data loading overhead is reduced.

dim	acc.
128	80.8
256	83.1
512	**84.4**
1024	83.7

(e) **Decoder width**. Unlike the image counterpart [31], an overly narrow decoder degrades accuracy noticeably.

blocks	acc.
1	83.2
2	83.6
4	**84.4**
8	84.3

(f) **Decoder depth**. Unlike the image counterpart [31], an overly shallow decoder degrades accuracy.

Table 2: **Ablation experiments** on Kinetics-400. The model is ViT-L, with an input size of 16×224×224 and a spacetime patch size of 2×16×16. The pre-training length is 800 epochs. The entries marked in gray are the same, which specify the default settings. This table format follows [31].

Mask sampling strategy. Our method follows the structure-agnostic random sampling methodology in BERT [15] and MAE [31]. Table 2a reports that this simple solution works the best in our method.

We compare with other strategies as illustrated in Fig. 4. *Space-only* sampling, which samples on the 2D spatial axes and broadcasts along the temporal axis, works reasonably well (83.5%). *Time-only* sampling, with a masking ratio of 75% (*i.e.*, keep 2 time steps out of 8), performs poorly (79.1%); if we increase its masking ratio to 87.5% (keep 1 out of 8), the accuracy drops further to 75.4%. Time-only sampling is related to future/past frame prediction, which can be an overly difficult task in our scenario. Block-wise sampling [3], in its spacetime variant [77], has 83.2% accuracy with 75% masking ratio (a higher ratio is worse).

Reconstruction target. Our method performs decently by reconstructing the original, unmodified pixels (83.8%, Table 2b). Using per-patch normalized pixels [31] improves by 0.6%. This observation is similar to that of its image counterpart [31]. Using HOG [13] as the target [77] works strongly too.

The autoencoding nature of our method (*i.e.*, predicting pixels) provides a self-contained solution. In contrast, an extra tokenizer (*e.g.*, dVAE [47, 9]), as is used in [3, 73], may require external data to train and additional domain knowledge to design (*e.g.*, the dVAE used is a ConvNet [37]). Applying the extra dVAE tokenizer to each frame is computationally heavy, which slows down training by 1.6× in our implementation. Our pixel-based method is simpler and performs better (Table 2b).

Data augmentation. Temporal data can provide natural augmentation, *e.g.*, on view points, motion, deformation, occlusion. These forms of natural augmentation have been incorporated by random temporal sampling. Table 2c compares additional augmentation on the spatial domain. Even using *no* spatial augmentation (center crop only) works competitively, similar to the observation on images

pre-train set	# pre-train data	pre-train method	K400	AVA	SSv2
-	-	none (from scratch)	71.4	-	-
IN1K	1.28M	supervised	78.6	17.3	50.2
IN1K	1.28M	MAE	82.3	26.3	65.6
K400	240k	supervised	-	21.6	55.7
K400	240k	MAE	84.8	31.1	72.1
K600	387k	MAE	**84.9**	32.5	73.0
K700	537k	MAE	n/a[†]	33.1	**73.6**
IG-uncurated	1M	MAE	84.4	**34.2**	**73.6**

Table 3: **Influence of pre-training data**, evaluated on K400, AVA, and SSv2 as the downstream tasks. The MAE pre-training length is 1600 epochs on K400/600/700 and IG-uncurated. No intermediate fine-tuning is used. The model is ViT-L. [†]: *The K700 training set has 13.9k duplicated videos with the K400 validation set (19.9k), so it is not legitimate to train on K700 to get K400 results.*

[31]. Random cropping with a mild scale range of $[0.5, 1]$ works well, while stronger cropping (range $[0.08, 1]$, [63]) reduces accuracy; adding color jittering reduces accuracy too, similar to [31].

It is practically valuable for self-supervised learning methods to be *less dependent* on data augmentation. There are a variety of applications in which augmentation is not valid or is hard to induce, *e.g.*, medical imaging, hyper-spectral imaging, remote sensing, geometric data (point cloud, key points, *etc.*), and their temporal extensions. Our method could be generalized to these cases.

Repeated sampling. As our method is fast in computation, we adopt repeated sampling [33] to reduce the data loading overhead. Table 2d reports its influence. Using 2 or 4 repetitions increases wall-clock speed by $1.8\times$ or $3.0\times$, as a loaded and decompressed file is reused multiple times.

Decoder capacity. Table 2e and 2f report the influence of the decoder width and depth. Using an overly small decoder degrades accuracy by large margins. This is unlike its image counterpart [31], in which a 128-d or 1-block decoder has no degradation if fine-tuning is applied. We hypothesize that the higher-dimensional video data are more complex and thus require higher decoding capacity. On the other hand, our optimal decoder (512-d, 4-block) is still substantially smaller than the encoder (1024-d, 24-block). This is similar to the observation on its image counterpart [31].

5.3 Influence of Data

Transfer learning ablation. Table 3 studies pre-training on different datasets and transferring to various downstream tasks. The pre-training datasets include ImageNet-1K (IN1K) [14] and Kinetics-400, 600, and 700 [35, 6, 7]. The downstream tasks include K400, AVA [29], and SomethingSomething v2 (SSv2) [27]. We do *not* perform any intermediate fine-tuning (see appendix), so the comparison here is influenced by the data scale/distribution but not by the number of their labels.

First we compare with pre-training on the IN1K images. MAE pre-training on IN1K[4] is 3.7% better than IN1K supervised pre-training (78.6% to 82.3%); this image-based MAE is even better than K400 *supervised* pre-training, on both AVA (21.6% to 26.3%) and SSv2 (55.7% to 65.6%).

MAE pre-training on K400 has *massive* gains over supervised pre-training on K400: it improves by **9.5%** on AVA (21.6% to 31.1%) and **16.4%** on SSv2 (55.7% to 72.1%). MAE pre-training on K400 videos also substantially outperforms MAE pre-training on IN1K images: it increases by **2.5%** on K400 (82.3% to 84.8%), **4.8%** on AVA (26.3% to 31.1%), and **6.5%** on SSv2 (65.6% to 72.1%), suggesting that MAE pre-training on videos is highly beneficial for these video tasks.

With more pre-training data (K600/K700) without labels, we observe noticeable improvements on AVA and SSv2: comparing with K400 pre-training, MAE with K700 has an extra gain of **2.0%** gain on AVA (31.1% to 33.1%) and **1.5%** on SSv2 (72.1% to 73.6%).

Real-world data. We further study MAE pre-training on *real-world* Instagram videos. We study two sets: (i) Instagram videos *curated* (IG-curated) [24] with hashtags similar to K400 classes, and (ii) random, *uncrated* Instagram videos (IG-uncurated). Both sets have 1 million videos.

[4]The IN1K pre-trained model is from https://github.com/facebookresearch/mae.

data	# videos	200-ep.	400-ep.	800-ep.
K400	240k	81.5	83.3	**84.4**
IG-curated	240k	79.0	81.6	83.2
IG-curated	512k	81.9	83.5	83.9
IG-curated	1M	**83.5**	84.1	84.2
IG-uncurated	1M	83.2	**84.5**	**84.4**

Table 4: **Real-world Instagram data** for MAE pre-training. We pre-train MAE on each individual set for 200, 400, and 800 epochs. We compare fine-tuning accuracy on K400. The model is ViT-L.

Table 3 (last row) reports transfer learning results on AVA and SSv2 using IG-*uncurated* pre-training. Notably, on AVA, MAE with IG-uncurated is *better* than MAE with curated Kinetics pre-training (*e.g.*, by **3.1/1.7/1.1**% over K400/600/700 pre-training); on SSv2, MAE with IG-uncurated is among the best, on par with the K700 counterpart.

Table 4 presents more results on the dataset size and training epochs. Pre-training on a 240k subset of IG-curated (the same size as K400) performs worse on K400 classification, which can be caused by the domain shift of data. However, increasing the dataset size of IG-curated to 512k and 1M shows good gains: under the same number of pre-training epochs (200 and 400), it can *outperform* K400 pre-training even when evaluating on K400. IG-uncurated performs similarly well as IG-curated, although the videos are randomly sampled and unrelated to K400 classes. This behavior is *not* observed on contrastive learning methods for videos: *e.g.*, in [22] it is empirically shown that data curation has a major impact on contrastive learning [32, 10, 28] performance.

We believe that our exploration with real-world data has encouraging results. It is a more realistic use case of unsupervised learning at scale. We hope this exploration will shed light on future study.

5.4 System-level Comparisons

We provide system-level comparisons with the leading results on K400, AVA, and SSv2. The detailed tables are in the appendix (Table 7, 8, 9). These results are multifaceted, involving architecture designs, computational complexity, model sizes, input resolution, pre-training data and methods, *etc.*, as we summarize in the tables. Our results are competitive and are close to the leading entries. In particular, our results are based only on *vanilla* ViT architectures, while the leading methods are hierarchical or specialized for videos. Our results demonstrate the potential of using fewer inductive biases and learning more from data, which is a pursuit of self-supervised learning.

5.5 Video Pre-training for Image Recognition

Finally, we report preliminary results on video pre-training for image recognition. The usage of vanilla ViT allows to convert to 2D easily: we only "deflate" patch embeddings by summing in time. Using ViT-L pre-trained by MAE on K400 / IG-uncurated, we obtain 83.7% / 84.1% accuracy on IN1K image classification. This is better than training ViT-L from scratch on IN1K (82.6% [31]), though lower than MAE pre-training on IN1K (85.9% [31]). Considering the large domain gap, we believe this result is decent and its improvement over training from scratch is encouraging. We hope it will motivate the community to explore video pre-training for *general* visual representation learning.

6 Conclusion

We have explored a simple extension of MAE [31] to video data. We have drawn several interesting observations. (i) We find that it is possible to learn strong representations with minimal domain knowledge or inductive biases. This follows the spirit of the ViT paper [18]. Similar to BERT [15] and MAE [31], we show that self-supervised learning on videos can be tackled in a conceptually unified framework. (ii) We empirically show that the masking ratio is an important factor for general masked autoencoding methods [69], and its optimal values may depend on the nature of the data (language, images, videos, *etc.*). (iii) We report encouraging results of pre-training on real-world, uncurated data. It achieves strong performance, close to pre-training on controlled, curated data (*e.g.*, Kinetics). To the best of our knowledge, promising results on uncurated data are rare in the literature.

In spite of these observations, open problems remain. The scale of data we have explored is orders of magnitudes smaller than the language counterparts [52, 15, 53, 5]. While our method has largely improved the efficiency of self-supervised learning, the high-dimensional video data still present a major challenge for scaling up. We hope our study will provide initial signals for future research.

References

[1] Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, 2015.

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

[3] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image Transformers. *arXiv:2106.08254*, 2021.

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

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[6] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv:1808.01340*, 2018.

[7] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the Kinetics-700 human action dataset. *arXiv:1907.06987*, 2019.

[8] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

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

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[11] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[12] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020.

[13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *NAACL*, 2019.

[16] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. DynamoNet: Dynamic Action and Motion Network. In *ICCV*, 2019.

[17] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. PeCo: Perceptual codebook for BERT pre-training of Vision Transformers. *arXiv:2111.12710*, 2021.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[19] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale Vision Transformers. In *ICCV*, 2021.

[20] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, 2020.

[21] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *ICCV*, 2019.

[22] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *CVPR*, 2021.

[23] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *ICCV*, 2017.

[24] Deepti Ghadiyaram, Matt Feiszli, Du Tran, Xueting Yan, Heng Wang, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019.

[25] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *ICCV*, 2015.

[26] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[27] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[28] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020.

[29] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A. Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

[30] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Workshop on Large Scale Holistic Video Understanding, ICCV*, 2019.

[31] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

[32] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[33] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020.

[34] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

[35] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The Kinetics human action video dataset. *arXiv:1705.06950*, 2017.

[36] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. MoviNets: Mobile video networks for efficient video recognition. In *CVPR*, 2021.

[37] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[38] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequence. In *ICCV*, 2017.

[39] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. *arXiv:2112.01526*, 2021.

[40] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer v2: Scaling up capacity and resolution. *arXiv:2111.09883*, 2021.

[41] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video Swin Transformer. *arXiv:2106.13230*, 2021.

[42] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[44] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017.

[45] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[46] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.

[47] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[48] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *CVPR*, 2021.

[49] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[50] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

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

[52] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[53] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[54] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[55] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

[56] Adria Recasens, Pauline Luc, Jean-Baptiste Alayrac, Luyu Wang, Florian Strub, Corentin Tallec, Mateusz Malinowski, Viorica Pătrăucean, Florent Altché, Michal Valko, et al. Broaden your views for self-supervised video learning. In *ICCV*, 2021.

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[58] Pierre Sermanet et al. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

[59] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv:1803.02155*, 2018.

[60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

[61] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[62] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv:1906.05743*, 2019.

[63] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

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

[66] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv:2203.12602*, 2022.

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

[68] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[69] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

[70] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabelled video. In *CVPR*, 2016.

[71] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018.

[72] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016.

[73] 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*, 2022.

[74] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[75] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[76] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019.

[77] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. *arXiv:2112.09133*, 2021.

[78] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *CVPR*, 2018.

[79] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. In *Neural Computation*, 2002.

[80] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: A simple framework for masked image modeling. *arXiv:2111.09886*, 2021.

[81] 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.

[82] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. *arXiv:2201.04288*, 2022.

[83] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *arXiv:2111.11432*, 2021.

[84] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[85] Bowen Zhang, Jiahui Yu, Christopher Fifty, Wei Han, Andrew M Dai, Ruoming Pang, and Fei Sha. Co-training Transformer with videos and images improves action recognition. *arXiv:2112.07175*, 2021.

[86] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See final paragraph of §6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See §C.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We are going to release the code publicly soon.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See §A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Error bars are not reported because it would be too computationally expensive, *e.g*. Kinetics-400 has ~72 more frames than ImageNet.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Table 1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See [35, 6, 7, 14, 29, 27].
   (b) Did you mention the license of the assets? [No] Licenses are available at the dataset websites.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We are using publicly available datasets. More information can be found in the respective dataset documentation [35, 6, 7, 14, 29, 27].
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We are using publicly available datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7 Appendix

A Implementation Details

Kinetics action classification. Our settings mainly follow [31, 77]. Table 5a summarizes our pre-training settings on Kinetics. Table 5b shows the corresponding fine-tuning settings for ViT-B/L/H. For fine-tuning, we add a linear classifier layer to the encoder's averaged tokens [18].

For fine-tuning the intermediately fine-tuned checkpoints from K600 in Table 7, we use the setting in Table 5b with a lower learning rate (8e-4) and shorter duration (40 epochs for ViT-L; 30 for ViT-H) and an increased drop path rate of 0.3 for ViT-H.

AVA action detection. Table 6a summarizes our fine-tuning settings on AVA [29]. The settings mainly follow [39, 77]. We follow the detection architecture in [21, 39, 77] that adapts Faster R-CNN [57] for video action detection. Only for the AVA results in Table 8, we use relative positions [59, 54] (as implemented in [39]) during fine-tuning.

SSv2 action classification. Table 6b summarizes our fine-tuning settings on SSv2 [27]. The settings mainly follow [39, 77]. For the frame sampling, we split each video into segments, and sample one frame from each segment to form a clip following [39, 19].

Fine-tuning from image pre-training. In Table 3 we have compared with ImageNet-based supervised/MAE pre-training. When fine-tuning these variants for videos, we inflate the 2D kernel of the patch embedding layer to 3D [8] and initialize the temporal position embeddings by zero.

config	value
optimizer	AdamW [43]
optimizer momentum	β_1, β_2=0.9, 0.95 [9]
weight decay	0.05
base learning rate†	1.6e-3
learning rate schedule	cosine decay [42]
warmup epochs [26]	120
epochs	default 800
repeated sampling [33]	4
augmentation	hflip, crop [0.5, 1]
batch size	512
gradient clipping	0.02

(a) Kinetics pre-training

config	ViT-B	ViT-L	ViT-H
optimizer	AdamW [43]		
optimizer momentum	β_1, β_2=0.9, 0.999		
weight decay	0.05		
base learning rate†	1.6e-2	4.8e-3	1.6e-3
learning rate schedule	cosine decay [42]		
warmup epochs [26]	5		
epochs	150	100	75
repeated sampling [33]	2	2	1
augmentation	RandAug (9, 0.5) [12]		
batch size	768	256	256
mixup [86]	0.8		
cutmix [84]	1.0		
label smoothing [64]	0.1		
drop path [34]	0.1	0.2	0.2
dropout [60]	0.3	0.3	0.5
layer-wise decay [11]	0.65	0.75	0.8

(b) Kinetics fine-tuning

Table 5: Settings on Kinetics. †: *lr* = *base_lr* × batchsize / 256 per the linear *lr* scaling rule [26].

config	values
optimizer	SGD
weight decay	1e-8
learning rate	4.8
learning rate schedule	cosine decay [42]
warmup epochs [26]	5
epochs	30
batch size	128
drop path [34]	0.2
dropout [60]	0.5
layer-wise decay [11]	0.75 (L) 0.85 (H)

(a) AVA fine-tuning

config	values
optimizer	SGD
weight decay	1e-4
learning rate	0.64 (L) 0.32 (H)
learning rate schedule	cosine decay [42]
warmup epochs [26]	3
epochs	40
augmentation	RandAug (9, 0.5) [12]
batch size	256
mixup [86]	0.8
cutmix [84]	1.0
label smoothing [64]	0.1
drop path [34]	0.2
dropout [60]	0.5
layer-wise decay [11]	0.75 (L) 0.85 (H)

(b) SSv2 fine-tuning

Table 6: Settings on AVA and SSv2.

pre-train	extra data	architecture	input size	top-1	top-5	FLOPs	param.
scratch		SlowFast [21]	64×224^2	79.8	93.9	$234 \times 3 \times 10$	60
scratch		X3D-XL [20]	16×312^2	79.1	93.9	$48 \times 3 \times 10$	11
scratch		MoViNet [36]	120×320^2	81.5	95.3	$386 \times 1 \times 1$	31
scratch		MViT-B [19]	64×224^2	81.2	95.1	$455 \times 3 \times 3$	37
scratch		MViTv2-B [19]	32×224^2	82.9	95.7	$255 \times 1 \times 5$	51
supervised	IN21K	Swin-B [41]	32×224^2	82.7	95.5	$282 \times 3 \times 4$	88
supervised	IN21K	Swin-L [41]	32×224^2	83.1	95.9	$604 \times 3 \times 4$	197
supervised	IN21K	Swin-L [41]	32×384^2	84.9	96.7	$2107 \times 5 \times 10$	200
BEVT [73]	IN1K+DALLE	Swin-B [41]	32×224^2	81.1	n/a	$282 \times 3 \times 4$	88
MaskFeat [77]		MViTv2-L [39]	16×224^2	84.3	96.3	$377 \times 1 \times 10$	218
MaskFeat [77]		MViTv2-L [39]	40×352^2	86.7	97.3	$3790 \times 3 \times 4$	218
MaskFeat [77]	K600	MViTv2-L [39]	40×352^2	87.0	97.4	$3790 \times 3 \times 4$	218
MAE		ViT-B	16×224^2	81.3	94.9	$180 \times 3 \times 7$	87
MAE		ViT-L	16×224^2	84.8	96.2	$598 \times 3 \times 7$	304
MAE		ViT-H	16×224^2	85.1	96.6	$1193 \times 3 \times 7$	632
MAE	K600	ViT-L	16×224^2	86.5	**97.2**	$598 \times 3 \times 7$	304
MAE	K600	ViT-H	16×224^2	**86.8**	**97.2**	$1193 \times 3 \times 7$	632
using in-house data for supervision:							
supervised	JFT-300M	ViViT-L [2]	32×320^2	83.5	94.3	$3980 \times 3 \times 1$	308
supervised	JFT-300M	ViViT-H [2]	32×320^2	84.9	95.8	$3981 \times 3 \times 4$	654
supervised + text	FLD-900M	Florence [83]	n/a $\times 384^2$	86.5	97.3	n/a $\times 3 \times 4$	647
SimMIM [80] + sup.	IN21K+70M	SwinV2-G [40]	8×384^2	86.8	n/a	n/a $\times 5 \times 4$	3000
supervised	JFT-3B+SSv2+MiT+IN	CoVeR [85]	16×448^2	87.2	n/a	n/a $\times 3 \times 1$	n/a
supervised	WTS-60M	MTV-H [82]	32×280^2	89.9	98.3	$6130 \times 3 \times 4$	n/a

Table 7: **System-level comparisons on Kinetics-400 action classification**. We report top-1 and top-5 accuracy on the validation set. The input size is $T \times H \times W$. FLOPs (in 10^9) are presented as "FLOPs per view $\times$ spatial views $\times$ temporal views", following the literature. Parameters are in 10^6. The "extra data" column specifies the data used in addition to K400. Entries using spatial resolution $>224^2$ are noted in gray; entries using in-house data for supervision are in light blue. Our results with K600 are with intermediate fine-tuning.

**This table does not include results using K700, because the K700 training set has 13.9k videos duplicated with the K400 validation set (19.9k). Results with K700 are in Table 8 (AVA) and Table 9 (SSv2).*

B Additional Experimental Results

B.1 System-level Comparisons

Kinetics-400. Table 7 compares on Kinetics-400 (K400). Our results are competitive with the leading ones. Importantly, our method is much *simpler* than many other entries. Our method is the only leading entry based on *vanilla* ViT, while others were based on hierarchical or specialized designs for videos. We use the standard 224×224 spatial resolution, while higher-resolution fine-tuning and testing may improve results at a higher cost. Our model does *not* use relative position embedding, which could have extra gains that are orthogonal to our thesis. Our results can compete with some strong results that were based on in-house data for supervision.

AVA. Table 8 compares on AVA [29] action detection. Using only a resolution of 16×224^2, our results are close to those of MaskFeat on higher-resolution inputs (40×312^2). Importantly, our architectures are plain ViT models without feature hierarchies, yet they perform strongly on this detection task.

SSv2. Table 9 compares on SSv2 [27] action classification. On the resolution of 16×224^2 and using vanilla ViT, our results compare favorably with those of MaskFeat on 40×312^2 inputs.

pre-train	pre-train data	architecture	input size	mAP center	mAP full	FLOPs	param.
supervised	K400	SlowFast [21]	32×224^2	23.8	-	138	53
supervised	K400	MViTv1-B [19]	64×224^2	27.3	-	455	36
supervised	K400	MViTv2-B [39]	32×224^2	28.1	29.0	225	51
MaskFeat [77]	K400	MViTv2-L [39]	40×312^2	**36.3**	**37.5**	2828	218
MAE	K400	ViT-L	16×224^2	34.8	35.7	598	304
MAE	K400	ViT-H	16×224^2	**35.7**	**36.2**	1193	632

(a) AVA results using Kinetics-400 pre-training

pre-train	pre-train data	architecture	input size	mAP center	mAP full	FLOPs	param.
supervised	K600	SlowFast [21]	64×224^2	27.5	-	296	59
supervised	K600	X3D-XL [20]	16×312^2	27.4	-	48	11
supervised	K600	MViT-B [19]	32×224^2	28.7	-	236	53
supervised	K600	MViTv2-B [39]	32×224^2	29.9	30.5	225	51
supervised	K600	ACAR [48]	64×224^2	-	31.4	n/a	n/a
MaskFeat [77]	K600	MViTv2-L [39]	40×312^2	**37.8**	**38.8**	2828	218
MAE	K600	ViT-L	16×224^2	36.5	37.2	598	304
MAE	K600	ViT-H	16×224^2	**38.0**	**39.1**	1193	632

(b) AVA results using Kinetics-600 pre-training

pre-train	pre-train data	architecture	input size	mAP center	mAP full	FLOPs	param.
supervised	K700	MViTv2-B [39]	32×224^2	31.3	32.3	225	51
supervised	K700	ACAR [48]	64×224^2	-	33.3	n/a	n/a
supervised	K700 + IN21K	MViTv2-L [39]	40×312^2	33.5	34.4	2828	213
MAE	K700	ViT-L	16×224^2	37.3	38.3	598	304
MAE	K700	ViT-H	16×224^2	**38.2**	**39.0**	1193	632

(c) AVA results using Kinetics-700 pre-training

Table 8: **System-level comparisons on AVA v2.2 action detection**. We report mAP using center-crop or full-resolution inference, following the literature. FLOPs (in 10^9) are measured with center-crop inference. Parameter numbers are in 10^6. Only in this table, following MaskFeat [77], our results are with intermediate fine-tuning and with relative positions [59, 54] during fine-tuning.

pre-train	pre-train data	architecture	input size	top-1	top-5	FLOPs	param.
supervised	K400	SlowFast [21]	32×224^2	63.1	87.6	$106 \times 3 \times 1$	53
supervised	K400	MViTv1-B [19]	64×224^2	67.7	90.9	$454 \times 3 \times 1$	37
supervised	K400	MViTv2-B [39]	32×224^2	70.5	92.7	$225 \times 3 \times 1$	51
supervised	K400 + IN21K	Swin-B [41]	32×224^2	69.6	92.7	$321 \times 3 \times 1$	89
supervised	K400 + IN21K	MViTv2-B [39]	32×224^2	72.1	93.4	$225 \times 3 \times 1$	51
supervised	K400 + IN21K	MViTv2-L [39]	40×224^2	73.3	94.1	$2828 \times 3 \times 1$	213
BEVT [73]	K400 + IN1K	Swin-B [41]	32×224^2	71.4	n/a	$321 \times 3 \times 1$	88
MaskFeat [77]	K400	MViTv2-L [39]	40×312^2	**74.4**	**94.6**	$2828 \times 3 \times 1$	218
MAE	K400	ViT-L	16×224^2	72.1	93.9	$598 \times 3 \times 1$	304
MAE	K400	ViT-H	16×224^2	**74.1**	**94.5**	$1193 \times 3 \times 1$	632

(a) SSv2 results using Kinetics-400 pre-training

pre-train	pre-train data	architecture	input size	top-1	top-5	FLOPs	param.
supervised	K600	MViTv1-B [19]	32×224^2	67.7	90.9	$454 \times 3 \times 1$	37
MaskFeat [77]	K600	MViTv2-L [39]	40×312^2	**75.0**	**95.0**	$2828 \times 3 \times 1$	218
MAE	K600	ViT-L	16×224^2	73.0	94.2	$598 \times 3 \times 1$	304
MAE	K600	ViT-H	16×224^2	**75.2**	**94.9**	$1193 \times 3 \times 1$	632

(b) SSv2 results using Kinetics-600 pre-training

pre-train	pre-train data	architecture	input size	top-1	top-5	FLOPs	param.
MAE	K700	ViT-L	16×224^2	73.6	94.4	$598 \times 3 \times 1$	304
MAE	K700	ViT-H	16×224^2	**75.5**	**95.0**	$1193 \times 3 \times 1$	632

(c) SSv2 results using Kinetics-700 pre-training

Table 9: **System-level comparisons on SSv2 action classification**. Notations of FLOPs (10^9) and parameters (10^6) follow Table 7. We do not use intermediate fine-tuning here (see Table 10).

pre-train data	#	intermediate FT	K400	AVA	SSv2
K400	240k		84.8	31.1	72.1
K400	240k	✓	-	35.6	72.6
K600	387k		84.9	32.5	73.0
K600	387k	✓	86.5	36.8	73.1
K700	537k		n/a	33.1	73.6
K700	537k	✓	n/a	38.2	73.7

Table 10: **Influence of intermediate fine-tuning**, evaluated on AVA and SSv2. The model is ViT-L. The MAE pre-training length is 1600 epochs on K400/600/700. Using K700 training set for K400 validation is not legitimate due to the duplications in these training and validation sets.

B.2 Ablation on Intermediate Fine-tuning

In Table 3 we have shown results of self-supervised pre-training directly transferred to downstream datasets. Following the literature, we also investigate an another scenario: after self-supervised pre-training, we perform *intermediate fine-tuning* on the pre-training set using labels, before transferring. Table 10 studies its influence. Intermediate fine-tuning has substantial improvements on AVA, while on SSV2 its effect is marginal.

C Potential negative societal impacts

The potential negative societal impact of our approach is related to other machine learning methods that are learning from training data and therefore reflect statistics and biases of the datasets used. Our method could also be used to generate content that may or may not reflect biases of the training data.

Video understanding research requires compute-intensive experiments because of the large input dimensionality and the large training datasets which can have negative environmental impact. To mitigate this effect, our approach greatly reduces the computational cost for performing research. Further, there is potential of misuse for video classification methods, *e.g.* for surveillance purpose.

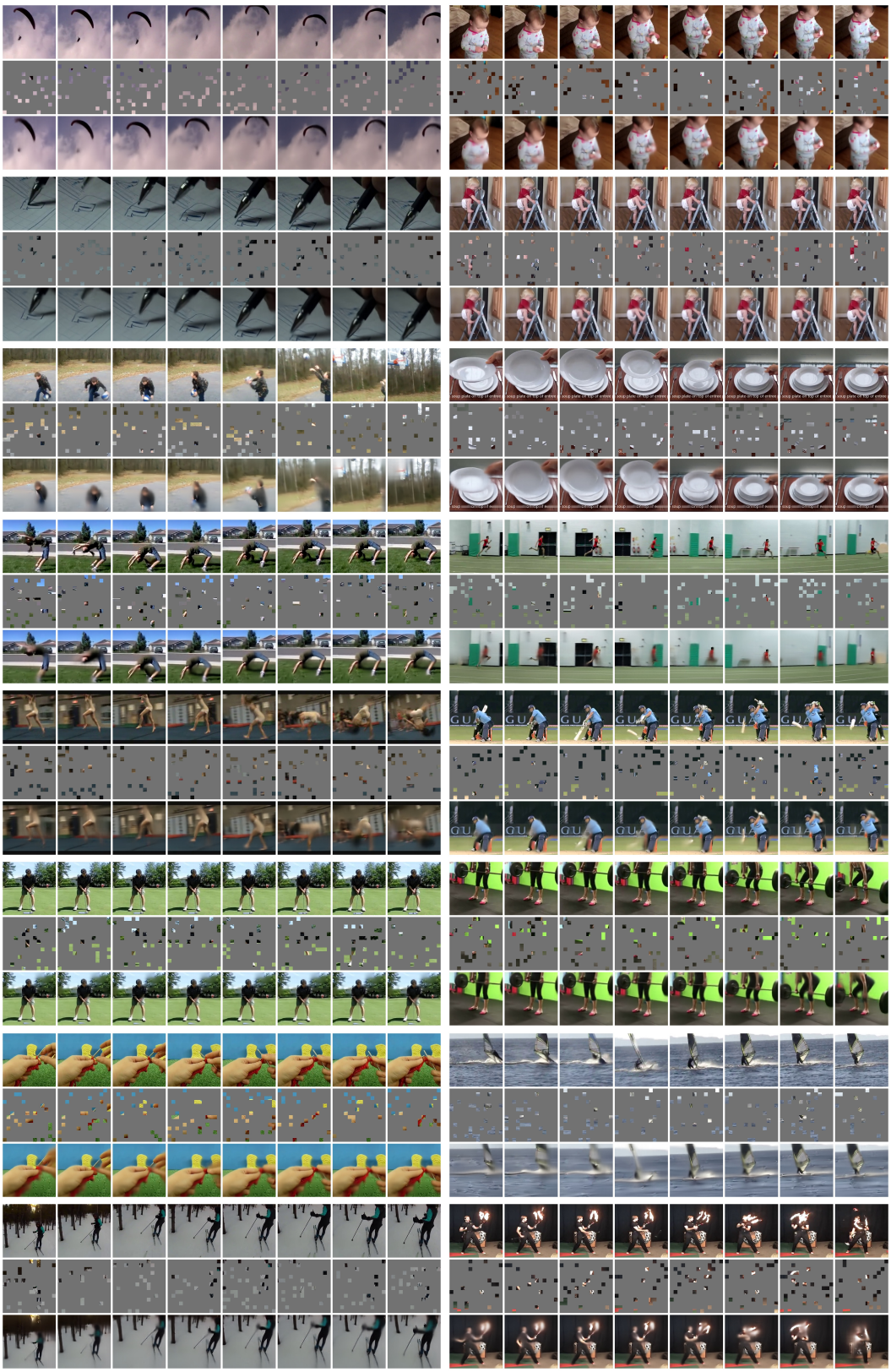


Figure 7: More visualizations on Kinetics-400 following Fig. 2 (masking ratio 90%).