Turbo Training with Token Dropout

Tengda Han[1]
htd@robots.ox.ac.uk

Weidi Xie[1,2]
weidi@robots.ox.ac.uk

Andrew Zisserman[1]
az@robots.ox.ac.uk

[1] Visual Geometry Group
Department of Engineering Science
University of Oxford

[2] Coop. Medianet Innovation Center
Shanghai Jiao Tong University
Shanghai, China

Abstract

The objective of this paper is an efficient training method for video tasks. We make three contributions: (1) We propose Turbo training, a simple and versatile training paradigm for Transformers on multiple video tasks. (2) We illustrate the advantages of Turbo training on action classification, video-language representation learning, and long-video activity classification, showing that Turbo training can largely maintain competitive performance while achieving almost $4\times$ speed-up and significantly less memory consumption. (3) Turbo training enables long-schedule video-language training and end-to-end long-video training, delivering competitive or superior performance than previous works, which were infeasible to train under limited resources.

1 Introduction

The extra temporal axis of video, compared to a static image, can capture rich information such as long term activities and stories. However, it also brings a few orders of magnitude of more data with the concomitant costs in processing and memory requirements. This, together with the long training schedules required to train networks on video data, has tremendously slowed research progress in video understanding and has raised the bar for researchers to make contributions in this area, particularly with limited resources. Furthermore, training consumes a significant amount of energy and hardware, with detrimental consequences for the natural environment.

In this paper, we propose a computation-efficient paradigm for training Transformers on video tasks, termed as **Turbo training**, which only needs to process sparsely sampled visual tokens. It is based on a multi-task training that jointly optimises two objectives: a masked region autoencoder loss, and the standard training loss for the downstream task of interest, for example, cross-entropy for action recognition, or contrastive loss for visual-language representation learning.

Turbo training is possible because the visual data contains redundancy. As observed in the recent self-supervised pre-training work, such as MAE [13] for images and [9, 46] for videos, it is sufficient to use only a small fraction of the input visual tokens to reconstruct the visual signal. For example, in images, 75% of the visual tokens can be dropped, whilst in videos, the dropout can be even more aggressive, and 90% of the spatio-temporal tokens can

be dropped while still being able to learn a strong representation using self-supervision. In Turbo training, we adopt a similar intuition and target efficient supervised training for video understanding. In short: **given the high redundancy in videos, training Transformers with sparsely sampled visual tokens is sufficient for the downstream tasks of interest**. This is beneficial for video understanding tasks in two ways: *first, training efficiency*, it dramatically reduces the computational cost for training transformers; *second, memory requirements*, videos of longer length can be loaded and directly trained on, leading to improved performance and new abilities, *i.e.*, end-to-end training the transformer models.

To summarise, we make the following contributions: (1) We propose Turbo training, a new paradigm for training Transformers on video understanding tasks, that jointly optimises a masked autoencoding and standard training loss for the downstream task of interest. (2) We illustrate the advantages of Turbo training on three video understanding tasks, including action classification, video-language training, and long-video activity classification, showing that training on sparsely sampled video patches can largely maintain competitive performance on most of the video tasks of interest, while achieving almost $4\times$ speed-up and significant reduction in memory consumption. As a consequence, (3) Turbo training enables long-schedule video-language representation learning and end-to-end long-video activity classification, delivering competitive or superior performance than previous methods.

2 Related Works

Masked Autoencoding has been used in the natural language processing community with great success to pre-train language models with a fill-in-the-blank task; specifically, the model is trained to use the context words surrounding a masked token to try to predict what the masked word should be, a typical example would be BERT [6]. In fact, a similar idea has also been investigated in computer vision for self-supervised learning in the seminal work of context autoencoder [32], where the authors proposed to learn a visual representation by training a ConvNet to inpaint the missing region of an image. However, due to the large effective spatial receptive fields of ConvNets, severe boundary artefacts weakened the learnt representation. Recently, with the advent of the Vision Transformers [7], that do not suffer from the same spatial receptive field problem, masked autoencoding has become popular again as an effective method to pre-train large-scale Vision Transformers. Examples include MAE [13], SimMiM [52], and BEiT [2] for images, and [9, 10, 46] for videos.

Efficient Deep Neural Networks. Since the resurgence of deep neural networks, reducing the computational cost for training or inference has always considered as an important topic. For instance: in image classification, novel architectures are regularly developed, Binarised Neural Networks [4, 37], MobileNet [14], ShuffleNet [51], EfficientNet [43]; and in object detection, single-stage detectors have drastically improved the efficiency of object detection, *e.g.*, YOLO [38], SSD [29], RetinaNet [27], CornerNet [26], *etc*. When processing videos, the efficiency becomes even more critical. In the literature, approaches have been proposed to intelligently sample the video data, by exploiting the relatively cheap audio modality in videos [22], to exploit variable mini-batch shapes for fast training [49], or to directly train on compressed videos without decoding [50].

Efficient Transformers. Transformer model architectures have gained immense interest due to their effectiveness on a range of tasks, including computer vision, natural language processing, reinforcement learning, *etc*. The self-attention mechanism is a critical design in Transformer that enables each input token to explicitly build relationships with others. In
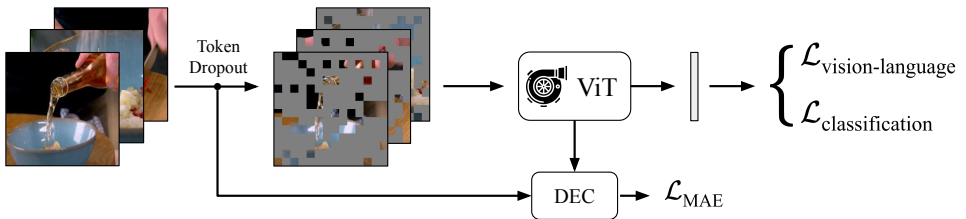
Figure 1. Illustration of Turbo training. We train a Video Transformer model by only using sparsely sampled visual tokens, for faster training and reduced computational cost. To achieve this goal, we exploit a multi-task training scheme, with one task being partial masked autoencoding, and the other being the standard supervised loss for the downstream task of interest, for example, contrastive learning for visual-language pre-training, or cross-entropy for action classification.

order to improve the general efficiency, a rich set of ideas have been explored to reduce the complexity for computing the self-attention affinity matrix, for example, to develop recurrence in the Transformer (Transformer-XL [5]), to reduce the computation consumption by only exploiting Transformer decoders with a controllable number of queries (Perceiver [17]), to use learnable attention pooling (Routing Transformer [39], Sinkhorn Transformer [45], Reformer [21]) to rely on low rank matrix decomposition of the affinity matrix from self-attention (Linformer [48], MotionFormer [34], Performer [3], Linear Transformer [19]), to exploit sparse representation (Product Key Memory [25]) for the affinity matrix. In this paper, our proposal is orthogonal to the above mentioned techniques, specifically, motivated by the high redundancy in visual signals, we aim to train the Transformer on videos with only sparsely sampled visual patches.

3 Method

We start by presenting an overview of Vision Transformers in Section 3.1, and discuss their pros and cons for video understanding tasks. In Section 3.2, we introduce the **Turbo** training, and describe its application for efficiently training the Vision Transformer on videos.

3.1 Vision Transformers for Videos

Generally speaking, given a video with T frames, i.e., $\mathcal{V} = \{I_1, I_2, \ldots, I_T\}$, with $I_k \in \mathbb{R}^{H \times W \times 3}$, as input to a Vision Transformer (ViT), the video is first split into non-overlapping spatial-temporal patches, and projected with linear layers to get a sequence of token vectors. After appending a learnable 'CLS' token (v_{CLS}), the resulting vector sequence is then processed with series of transformer encoder layers, i.e., $\mathbf{z} = \Phi_{\text{ViT}}([v_{\text{CLS}}, v_1, \ldots, v_n] + \text{PE})$, where v_i denotes one of the $n = n_t \cdot n_h \cdot n_w$ spatio-temporal patches of size $t \times h \times w$, with $n_t = \lfloor T/t \rfloor$, $n_h = \lfloor H/h \rfloor$, $n_w = \lfloor W/w \rfloor$, and PE refers to the spatio-temporal positional encodings.

   With the self-attention operations along *all* the spatial-temporal patches, the benefit of applying ViT on videos is prominent – it can capture the long-term dependency in the videos. However, the self-attention also incurs the quadratic complexity, i.e., $\mathcal{O}((n_t \cdot n_h \cdot n_w)^2)$, for scenarios when the input video is long (more than a few seconds), the memory consumption becomes infeasible. As a consequence, the majority of ViT-based architectures are still limited to only processing short video clips.

3.2 Turbo Training

In this paper, we introduce a **Turbo training** regime, that aims to exploit the redundancy in video data, and optimise a video Transformer for multi-task training with only sparsely

sampled visual tokens. It requires optimizing for two objectives: (i) partial masked autoencoding, that acts as a proxy task to force the model to capture spatio-temporal information to a full extent, with only the sparsely sampled video patches; and (ii) the default objective for the considered downstream task, that may refer to cross-entropy or contrastive learning.

In the following, we start by describing the partial masked autoencoder, and then detailing the use of Turbo training on various video understanding tasks, including: action classification, video-language training and long-video activity classification.

Partial Masked Autoencoder (PMAE).   We use a ViT as the visual encoder ($\Phi_{\text{ViT-enc}}(.)$), that consists of a number of Transformer encoder blocks, operating on the input visual tokens. Inspired by MAE [13], we also use a shallow decoder with much fewer Transformer blocks than the encoder ($\Phi_{\text{dec}}(.)$). After preparing the spatial-temporal patches, we define three operations for the embedded tokens: (1) masking the input visual tokens with a ratio m, which means only $N_i = n_t \cdot n_h \cdot n_w \cdot (1-m)$ tokens are fed into the visual encoder ($\Phi_{\text{ViT-enc}}(.)$); (2) reconstructing at a reconstruction ratio r, that means $N_r = n_t \cdot n_h \cdot n_w \cdot r$ tokens are the targets of the reconstruction tasks; (3) the unused $N_{\text{ignore}} = n_t \cdot n_h \cdot n_w \cdot (m-r)$ tokens are ignored in the training iteration. Note that a natural constraint $r \leq m$ holds, and the original MAE setting [13] can be regarded as the special case when $r = m$. This training regime significantly cuts downs the memory consumption, for example, the complexity for self-attention in the encoder layer is only $\mathcal{O}(N_i^2)$, compared to operating on the full token sequence ($\mathcal{O}((n_t \cdot n_h \cdot n_w)^2)$), while the decoder layer can be quite lightweight, with the complexity of self-attention being $\mathcal{O}((N_i + N_r)^2)$.

Action Classification.   Here, we apply the Turbo training for Transformers on the video action classification task. In addition to the above-mentioned partial MAE loss ($\mathcal{L}_{\text{PMAE}}$), we adopt the standard cross-entropy loss on the short clips. Specifically, when given a labelled video sample $\{\mathcal{V}, y\}$, *i.e.*, a short video clip with an action label, we take the 'CLS' feature from the final layer of the visual encoder, then we pass the visual feature to a classifier, that is parametrised with a single MLP layer (denoted as $g(.)$), and trained with cross entropy loss. The overall objective for classification training is

$$\mathcal{L} = \lambda_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}(g(z_{\text{CLS}}), y) + \mathcal{L}_{\text{PMAE}} \tag{1}$$

In practice we set the weight $\lambda_{\text{CE}} = 1/\log(\texttt{num\_classes})$ to balance two losses.

Visual-language Pre-training.   Recently, visual-language representation learning has attracted growing interest from the community, due to its convenience of data curation procedure and remarkable performance on "zero-shot" generalisation on various image classification tasks [13, 36]. However, the challenge for training the model lies in its requirement for a significant amount of compute and long training schedule. Turbo training can be applied to this scenario, to significantly speed up and reduce the memory cost. In addition to the sparse MAE loss ($\mathcal{L}_{\text{PMAE}}$), here we also adopt a standard InfoNCE loss (noise contrastive learning [47]) to learn the joint embedding for the visual and language streams.

Specifically, given a paired video-language sample $\{\mathcal{V}, y\}$, *e.g.*, a video clip and a sentence that describes the visual scene, we use a Transformer-based language model $\Psi_{\text{text}}(\cdot)$ to encode the sentence, and take the language 'CLS' token as the representation for the entire sentence:

$$z_t = \Psi_{\text{text}}([w_{\text{CLS}}, w_1, \ldots, w_n] + \text{PE})$$

where w_i denotes word embedding after tokenisation, and PE denotes the positional embedding for the language. As for the visual stream, we use the same Transformer-based visual

encoder ($\Phi_{\text{ViT-ENC}}(\cdot)$), as introduced above, and take the final visual 'CLS' feature as the visual feature. The overall visual-language training loss is

$$\mathcal{L} = \lambda_{\text{NCE}} \cdot \mathcal{L}_{\text{NCE}} + \mathcal{L}_{\text{PMAE}} \tag{2}$$

with $\mathcal{L}_{\text{NCE}}$ denoting a bi-directional (visual-to-language & language-to-visual) InfoNCE loss.

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{2} \left(\log \frac{\exp(z_v \cdot z_t)}{\sum_t \exp(z_v \cdot z_t)} + \log \frac{\exp(z_v \cdot z_t)}{\sum_v \exp(z_v \cdot z_t)} \right)$$

In practice we set the weight $\lambda_{\text{NCE}} = 1/\log(\texttt{batch\_size})$ to balance two losses.

Long-video Activity Classification. Turbo training enables to load longer video sequences for end-to-end training, which was a challenging factor for long-video tasks. By applying Turbo training, the method for long-video activity classification is largely simplified and is similar to the method of short-video action classification. Specifically, given a labelled long video sample $\{\mathcal{V}, y\}$, we still optimise Eq. 1 as the main objective, whilst applying a larger masking ratio on the input video $\mathcal{V}$ is necessary to reduce computational cost.

4 Experiments

In this section, we validate the effectiveness of Turbo training by experimenting on three different tasks: video action classification, video-language representation learning, and long-video activity classification.

4.1 Action Classification

Dataset. We conduct experiments on two datasets, **UCF101** [42], containing 13k short video clips for 101 human actions, and **HMDB51** [23], containing 7k short video clips for 51 human actions. We report the Top1 classification accuracy on this task.

Implementation. In accordance with the experiment setting as used in [46], for each short video clip we decode the video with 5 fps and randomly sample 16 video frames at 224×224 resolution as input, which approximately spans about 3.2 seconds in time. The video frames are passed to the $\Phi_{\text{ViT}}(\cdot)$ with a token dropout controlled by the mask ratio m, and we take the final-layer 'CLS' token as the video feature, which is further passed to a linear classifier and trained with cross-entropy (CE) loss. Also, we keep a light-weight MAE objective controlled by a reconstruction ratio r, that is to reconstruct some randomly selected ($r \times 100\%$) space-time patches with a light weight ViT decoder. The overall objective function is in equation 1. We experiment with different combinations of the mask and reconstruction ratios $\{m, r\}$, to study their effects on turbo training. For more training details, we refer the reader to Supplementary Material.

4.2 Video-Language Representation Learning

Dataset. For the experiments on video-language pre-training, we adopt the recent **HTM-AA** dataset, where the video and language description have been temporally auto-aligned with self-supervision [12]. HTM-AA contains 250K videos sourced from the HowTo100M [30] dataset, with over 3.3M clip-sentence pairs. For evaluation of the learnt video-language representation, we benchmark on a temporal action alignment task on the **HTM-Align** dataset [12]. Specifically, HTM-Align contains 80 long videos from YouTube with a total

of 5K ASR sentences, and each visually alignable sentence has a **manually** labelled temporal segment. Note that, we only use this HTM-Align dataset for evaluation purposes, and all training is done on the automatically curated HTM-AA dataset, thus it can be seen as a 'zero-shot' evaluation on the HTM-Align.

Implementation.   In accordance with [28], we choose MPNet [41] as the language encoder. MPNet takes as input a sentence and outputs a textual feature vector. On the visual side, we use $\Phi_{\mathrm{ViT}}(\cdot)$ and take the final-layer 'CLS' token as the video feature. To demonstrate the effectiveness of Turbo training for the Vision Transformer, we fix the weights of the language model, and only finetune the weights of the ViT architecture, initialized from VideoMAE [46] (pretrained on Kinetics-400 [20]). As in Sect. 4.1, the visual encoder takes as input a 16-frame short video clip extracted with 5 fps, and the language encoder takes the corresponding sentence associated with the video clip. Overall, we take a batch size of at most 32 clip-sentence pairs (depending on different masking ratios) for video-language training, with an InfoNCE loss. We refer the reader to Supplementary Material for more training and evaluation details.

4.3   Long-video Activity Classification

Dataset.   To demonstrate the effectiveness of exploiting Turbo training for end-to-end Transformer training on long video sequences, here we evaluate long-video activity classification on the Breakfast [24] and COIN [44] datasets. In detail, **Breakfast** contains 1712 long videos for 10 cooking activities including 'making coffee' and 'frying egg'. **COIN** contains 11k long videos covering 180 general activities including 'car polishing' and 'assembling sofa'. The videos in these two datasets are often minute-long and untrimmed, containing procedural actions to complete the particular activity.

Implementation.   In our experiment, we load $n = \{16, 32, 64\}$ frames as the video input, and train the Transformer end-to-end. Note that, such an end-to-end setting was not easily achievable in previous works due to the large memory footprint. In detail, at the training time, we randomly choose the start and end timestamp among the first and last 20% duration of the video, then uniformly take n frames in between. The rest of the architectural design is similar to that of Sect. 4.1. At inference time, we repeat sampling n frames for 10 times from each video, and average the prediction probability, resembling the multi-crop test that has been widely used in existing work [40]. By default, we use a batch size of 16 videos and train the Transformer for 100 epochs on Breakfast and 50 epochs on COIN.

5   Results

Here, we start by showing the efficiency improvement from Turbo training on action classification (Sect. 5.1.1), then compare with existing works (Sect. 5.1.2). Next, we report the results for video-language representation learning (Sect. 5.2), followed by long-video activity classification (Sect. 5.3).

5.1   Turbo Training for Action Classification

5.1.1   Ablation Study: the speed and performance trade-off

In action classification, we investigate two of the hyperparameters of the PMAE, namely the mask ratio m and reconstruction ratio r. For the training, we use ViT-B initialized with the VideoMAE weights, and then finetune the network on UCF101 for action classification.

Mask%	Recon%	GFLOPs	Acc (%)
0	- (w/o MASK)	180.6	94.8*
50	50 (MAE)	99.3	93.7
75	75 (MAE)	57.6	91.8
75	25 (PMAE)	45.9	91.6
90	90 (MAE)	35.2	89.7
90	10 (PMAE)	18.3	89.6

Train (m%,r%)	Inference m%	Acc (%)
90, 10	90	89.6
90, 10	50	90.1
90, 10	0	**90.3**
75, 25	75	91.6
75, 25	50	91.9
75, 25	0	**92.4**

Table 1. Left: Speed and performance trade-off on the UCF101 classification task. *: our implementation of [46] on UCF101. **Right**: Generalisation to different mask ratios at the inference time.
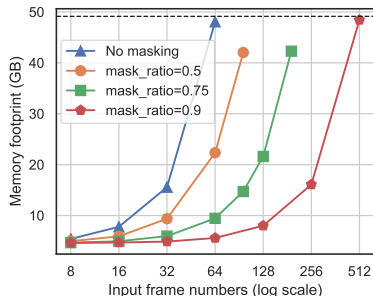

Figure 2. Memory footprint of training one video with different input frames.
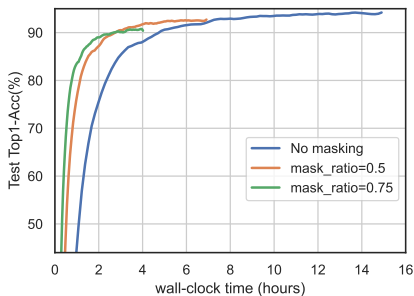

Figure 3. Training progress on UCF101 with a batch size of 16 for 100 epochs.

Both high mask ratio and low reconstruction ratio can reduce the computation. In Table 1 (left), we compute the GFLOPs for the Transformer model during Turbo training. Note that GFLOPs contains both the classifier head for action classification and the decoder for patch reconstruction. It is evident that increasing the masking ratio can reduce the GFLOPs, *e.g.*, by applying a 50% masking ratio for action classification, the GFLOPs can be reduced by half at a cost of only 1% Top1 classification accuracy. Additionally, we find reducing the reconstruction ratio can also reduce the GFLOPs while incurring minimum effect to the final classification accuracy (91.8 to 91.6, and 89.7 to 89.6). In the following experiments, we use settings with $\{m, r\} = \{0.75, 0.25\}$ and $\{0.5, 0.5\}$, that demonstrate a balance between the performance and the compute operations. Figure 2 and 3 show the reduced memory footprints and the significant speed-up during training.

Generalisation to smaller masking ratio during inference. Here, we take the models trained with $\{m, r\} = \{0.9, 0.1\}$ and $\{0.75, 0.25\}$, and test it with different masking ratio at inference time. In Table 1 (right), we show that the model trained with high-rate token dropout can be used with a lower mask ratio at inference time. For example, inference with full patches ($m = 0$) achieves the best performance regardless of the training masking ratio. In the following experiments, we thus use $m = 0$ at inference by default.

5.1.2 Comparison to state-of-the-art

In Table 2, we compare to the existing approaches, which have been pre-trained on video or multimodal data in a self-supervised manner, and then end-to-end finetuned on the downstream tasks. **Note that**, the comparison here is by no means fair, due to the differences in architecture, training data, and resolutions. However, the key message in this comparison is that, even with only 50% of the visual tokens for finetuning, the model can still achieve competitive performance compared to all other models, while only needing half the training time as the original finetuning with all visual tokens. Additionally, with further self-supervised

Finetune Method	Pretrain Method	Backbone	Pretrain Dataset	Modality	UCF101	HMDB51
Classic	CoCLR [⬚]	S3D	K400	V	87.9	54.6
Classic	CVRL [⬚]	Slow-R152	K600	V	93.6	69.4
Classic	ρ-BYOL [⬚]	Slow-R50	K400	V	94.2	72.1
Classic†	VideoMAE [⬚]	ViT-B	K400	V	94.7	-
Classic	VideoMAE [⬚]	ViT-B	K400+UCF101	V	96.1	73.3
Classic	MIL-NCE [⬚]	S3D-G	HTM	V+T	91.3	61.0
Classic	TAN [⬚]	S3D-G	HTM-AA	V+T	92.0	-
Classic	XDC [⬚]	R(2+1)D-18	IG65M	V+A	94.2	67.1
Classic	GDT [⬚]	R(2+1)D-18	IG65M	V+A	95.2	72.8
Turbo$_{m=0.5,r=0.5}$	VideoMAE [⬚]	ViT-B	K400	V	94.4	71.8
Turbo$_{m=0.5,r=0.5}$	NCE	ViT-B	HTM-AA	V+T	**95.7**	**73.6**

Table 2. Comparing with recent self-supervised methods by finetuning on UCF101 and HMDB51 action classification. We apply Turbo training for finetuning, 'Classic' means traditional finetuning method without any token dropout or not applicable to token dropout. †: our reproduction of fine-tuning the open-sourced model from [⬚] which was trained on K400 only. The grey row: The best result on UCF101 from [⬚] is pretrained on UCF101 for an additional 3200 epochs, therefore it is different to their open-sourced model (only trained on K400). We use $\{m,r\} = \{0.5,0.5\}$ for our Turbo training settings.

Turbo training on video-language data (in Sect. 5.2), the performance can be further boosted.

5.2 Turbo Training for Video-Language Representation Learning

In this section, we experiment with video-language turbo training on the HTM-AA dataset. In detail, we consider three settings: no masking, $\{m,r\} = \{0.5,0.5\}$, and $\{0.75,0.25\}$. For each setting, we use the same implementation and the maximum number of video sequences a single NVidia A40 GPU can handle. We measure the quality of the video-language representation by monitoring the temporal action alignment performance on the HTM-Align dataset [⬚], *i.e.*, training wall-clock time vs. the temporal alignment performance. In detail, we extract the per-second visual features for the long videos in HTM-Align, then use the text embeddings to retrieve the corresponding visual moment in the time axis, recall is 1.0 if the predicted timestamp fall into the ground-truth temporal segment. We report Recall@1 (R@1) only on alignable sentences as in [⬚].

Evaluation Details. We follow the evaluation method of [⬚] that was applied on the CLIP and MIL-NCE features. For each minute-long video, there are two steps to evaluate the backbone feature quality on temporal alignment: (1) use the visual backbone to extract video features for the whole video at 1 feature-per-second, resulting in a sequence of vectors; (2) use the given sentence embedding to compute the similarity score with the sequence of visual features, and determine the most corresponding timestamps. In this paper, we adopt the same evaluation setting, specifically, we extract the video frames at 16 fps and pass every 16 frames into our ViT to compute a single feature vector, resulting in a sequence of visual features at 1 feature-per-second. Therefore, in Table 4 our temporal receptive field (R.F.) is 1 second because there is no long-range temporal modelling in this evaluation.

Discussion. As shown in Figure 3, it is clear that a higher mask ratio significantly increases the training efficiency. For example, after being trained for 18h, the 0.75 mask ratio achieves 28% R@1, whereas the baseline setup only gets 11% R@1. Additionally, to get the same performance, the $m = 0.75$ setting gives a $3.5\times$ speed-up comparing with $m = 0.5$ (reaching 27% R@1 requires 11.3 hours and 40 hours respectively). For the baseline setting without Turbo training, it is not feasible to achieve a comparable performance under a single GPU

training budget, showing that Turbo training indeed leads to improved performance and new abilities for existing Transformer models to finetune on video tasks in an end-to-end manner.
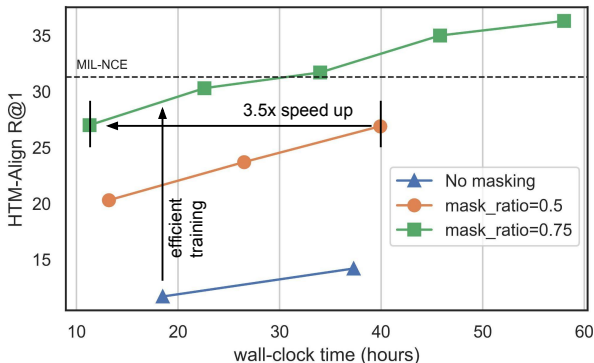

Table 3. Progress of vision-language pretraining monitored by R@1 on HTM-Align dataset.

Method	Arch.	Pretrain Data	Temporal R.F.	R@1
CLIP [36]	ViT-B-32	YFCC	1s	16.8
CLIP [36]	ViT-B-16	YFCC	1s	22.0
MIL-NCE [31]	S3D-G	HTM	1s	31.3
NCE with TAN† [12]	S3D-G	HTM-AA	1s	33.6
Ours (m=0.75, r=0.25)	ViT-B-16	HTM-AA	1s	**36.7**
TAN* [12]	S3D-G & TFM	HTM	64s	49.4

Table 4. Zero-shot text-visual alignment results on the HTM-Align dataset. We adopt the same setting as [12] and report Recall@1 as the metric. ∗: the result from TAN is not directly comparable as their model contains transformers (TFM) on the S3D-G features and has a much longer temporal context (64s) for the temporal alignment task. †: we train the NCE loss on HTM-AA for the same duration (approx. 60h) for a fair comparison.

In Table 4, our video-language representation gets a higher R@1 (36.7) than previous strong methods, including CLIP (22.0) and MIL-NCE (31.3), and the S3D model after end-to-end finetuning on the HTM-AA dataset (33.6), showing the superior quality of the joint video-language embedding learnt from Turbo training. Note that, the best alignment results (49.4) from TAN [12] is not directly comparable with our method, as their model builds on pre-extracted S3D base features and learns long-range temporal Transformers that scan the surrounding 64-second temporal context for the temporal alignment task. Here, we aim to evaluate the quality of *base* features, which is orthogonal to the long-term modelling in TAN [12].

5.3 Turbo Training for Long-video Activity Classification

Turbo training enables the end-to-end training of Transformers on long-video tasks, that used to be a critical bottleneck. We compare three settings: **(1)** F_{16} denotes 16-frame input with $\{m,r\} = \{0.5, 0.5\}$, **(2)** F_{32} denotes 32-frame input with $\{m,r\} = \{0.75, 0.25\}$, and **(3)** F_{64} denotes 64-frame input with $\{m,r\} = \{0.875, 0.125\}$. The number of frames and masking ratios are chosen such that each of the setting has the same number of *visible* patches to the

encoder but with different temporal extent. For frame sampling, we first sample the starting and ending positions from the beginning and the last 20% of the video, then we uniformly sample frames within this segment.

Effect of loading longer sequences.   In Table 5, increasing the frame numbers from 16 to 32 (row B vs. C) shows the longer temporal extent is beneficial for activity classification (e.g. 91.3 vs. 88.2 on Breakfast), we conjecture a denser temporal sampling has a higher chance to sample the characteristic moment in the long video. However, further increasing the frame numbers to 64 does not help the activity classification task. It might be because the high token masking ratio ($m = 0.875$) could frequently miss the key objects or actions in the video frame.

Temporal Method	Clip Model	Pretrain Supervision	Pretrain Data	BF	COIN
Timeception [16]	3D-ResNet	[✓] action labels	K400	71.3	-
VideoGraph [15]	I3D	[✓] action labels	K400	69.5	-
GHRM [53]	I3D	[✓] action labels	K400	75.5	-
Transformer [23]	TimeSformer	[✓] action labels	K400	81.1	83.5
Transformer [23]	TimeSformer	[✗] k-mean on ASR	HTM	81.4	85.3
Transformer [23]	TimeSformer	[✗] Dist. Sup	HTM+WikiHow	89.9	**88.9**
[A] **Turbo** F_{32}	N/A (end-to-end)	[✗] VideoMAE	K400	86.8	82.3
[B] **Turbo** F_{16}	N/A (end-to-end)	[✗] NCE	HTM-AA	88.2	81.2
[C] **Turbo** F_{32}	N/A (end-to-end)	[✗] NCE	HTM-AA	**91.3**	87.5
[D] **Turbo** F_{64}	N/A (end-to-end)	[✗] NCE	HTM-AA	87.3	86.8

Table 5. Procedural activity classification on the Breakfast (BF) and COIN dataset. [✓] denotes supervised training and [✗] denotes training without manual labelling.

Comparison with other works.   When learning from 32 frames on the Breakfast dataset end-to-end, our simple one-stage method outperforms previous methods (91.3 vs. 89.9) which typically require a two-stage setting: extracting short-clip features, then training temporal models on these pre-extracted features. On the COIN dataset, our one-stage method achieves comparable results (87.5 vs. 88.9). We conjecture a longer temporal context with smaller masking ratio could further improve the performance on uncurrated natural videos like COIN, which inevitably demands higher hardware requirements. Comparing row C and row A further shows the effectiveness of our vision-language training from Sect. 5.2.

6   Conclusion

We propose a simple and versatile Turbo training paradigm for Vision Transformers. We show that it is applicable to multiple video tasks including action classification, video-language training, and long-video activity classification, and can achieve superior performance, while significantly reducing the required computation and memory cost. Turbo training has demonstrated the ability to lower the resource requirements and enable end-to-end Transformer training for long videos.

References

[1] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. In *NeurIPS*, 2020.

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: Bert pre-training of image transformers. In *Proc. ICLR*, 2022.

[3] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *Proc. ICLR*, 2021.

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. 2019.

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

[7] 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 *Proc. ICLR*, 2020.

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

[9] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022.

[10] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. MaskViT: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022.

[11] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020.

[12] Tengda Han, Weidi Xie, and Andrew Zisserman. Temporal alignment networks for long-term video. In *Proc. CVPR*, 2022.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proc. CVPR*, 2022.

[14] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[15] Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders. VideoGraph: Recognizing minutes-long human activities in videos. In *ICCV-Workshop*, 2019.

[16] Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders. Timeception for complex action recognition. In *Proc. CVPR*, 2019.

[17] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *Proc. ICML*, 2021.

[18] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. ICML*, 2020.

[20] Will Kay, João 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.

[21] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proc. ICLR*, 2020.

[22] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proc. ICCV*, 2019.

[23] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *Proc. ICCV*, 2011.

[24] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. CVPR*, 2014.

[25] Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. In *NeurIPS*, 2019.

[26] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *Proc. ECCV*, 2018.

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. ICCV*, 2017.

[28] Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and Lorenzo Torresani. Learning to recognize procedural activities with distant supervision. In *Proc. CVPR*, 2022.

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Proc. ECCV*, pages 21–37. Springer, 2016.

[30] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *Proc. ICCV*, 2019.

[31] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proc. CVPR*, 2020.

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

[33] Mandela Patrick, Yuki M. Asano, Polina Kuznetsova, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. On composition of transformations in contrastive self-supervised learning. In *Proc. ICCV*, 2021.

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

[35] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020.

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[37] Vicente Ordonez Joseph Redmon Rastegari, Mohammad and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *Proc. ECCV*, 2016.

[38] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 2016.

[39] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. 2020.

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

[41] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

[42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[43] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. ICML*, 2019.

[44] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proc. CVPR*, 2019.

[45] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *arXiv preprint arXiv:2002.11296*, 2020.

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

[47] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

[49] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krähenbühl. A multigrid method for efficiently training video models. In *Proc. CVPR*, 2020.

[50] Manzil Zaheer Hexiang Hu R. Manmatha Alexander J. Smola Wu, Chao-Yuan and Philipp Krähenbühl. Compressed video action recognition. In *Proc. CVPR*, 2018.

[51] Xinyu Zhou Mengxiao Lin Zhang, Xiangyu and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. CVPR*, 2018.

[52] Zhenda, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: A simple framework for masked image modeling. In *Proc. CVPR*, 2022.

[53] Jiaming Zhou, Kun-Yu Lin, Haoxin Li, and Wei-Shi Zheng. Graph-based high-order relation modeling for long-term action recognition. In *Proc. CVPR*, 2021.