ViViT: A Video Vision Transformer

Anurag Arnab* Mostafa Dehghani* Georg Heigold Chen Sun Mario Lučić† Cordelia Schmid†
Google Research
{aarnab, dehghani, heigold, chensun, lucic, cordelias}@google.com

Abstract

We present pure-transformer based models for video classification, drawing upon the recent success of such models in image classification. Our model extracts spatio-temporal tokens from the input video, which are then encoded by a series of transformer layers. In order to handle the long sequences of tokens encountered in video, we propose several, efficient variants of our model which factorise the spatial- and temporal-dimensions of the input. Although transformer-based models are known to only be effective when large training datasets are available, we show how we can effectively regularise the model during training and leverage pretrained image models to be able to train on comparatively small datasets. We conduct thorough ablation studies, and achieve state-of-the-art results on multiple video classification benchmarks including Kinetics 400 and 600, Epic Kitchens, Something-Something v2 and Moments in Time, outperforming prior methods based on deep 3D convolutional networks. To facilitate further research, we will release code and models.

1. Introduction

Approaches based on deep convolutional neural networks have advanced the state-of-the-art across many standard datasets for vision problems since AlexNet [35]. At the same time, the most prominent architecture of choice in sequence-to-sequence modelling (e.g. in natural language processing) is the transformer [65], which does not use convolutions, but is based on multi-headed self-attention. This operation is particularly effective at modelling long-range dependencies and allows the model to attend over all elements in the input sequence. This is in stark contrast to convolutions where the corresponding "receptive field" is limited, and grows linearly with the depth of the network.

The success of attention-based models in NLP has recently inspired approaches in computer vision to integrate transformers into CNNs [72, 5], as well as some attempts to replace convolutions completely [46, 50]. However, it is only very recently with the Vision Transformer (ViT) [15], that a pure-transformer based architecture has outperformed its convolutional counterparts in image classification. Dosovitskiy *et al*. [15] closely followed the original transformer architecture of [65], and noticed that its main benefits were observed at large scale – as transformers lack some of the inductive biases of convolutions (such as translational equivariance), they seem to require more data [15] or stronger regularisation [61].

Inspired by ViT, and the fact that attention-based architectures are an intuitive choice for modelling long-range contextual relationships in video, we develop several transformer-based models for video classification. Currently, the most performant models are based on deep 3D convolutional architectures [6, 17, 18] which were a natural extension of image classification CNNs [24, 57]. Recently, these models were augmented by incorporating self-attention into their later layers to better capture long-range dependencies [72, 20, 76].

As shown in Fig. 1, we propose pure-transformer models for video classification. The main operation performed in this architecture is self-attention, and it is computed on a sequence of spatio-temporal tokens that we extract from the input video. To effectively process the large number of spatio-temporal tokens that may be encountered in video, we present several methods of factorising our model along spatial and temporal dimensions to increase efficiency and scalability. Furthermore, to train our model effectively on smaller datasets, we show how to reguliarise our model during training and leverage pretrained image models.

We also note that convolutional models have been developed by the community for several years, and there are thus many "best practices" associated with such models. As pure-transformer models present different characteristics, we need to determine the best design choices for such architectures. We conduct a thorough ablation analysis of tokenisation strategies, model architecture and regularisation methods. Informed by this analysis, we achieve state-of-the-art results on multiple standard video classification benchmarks, including Kinetics 400 and 600 [32], Epic Kitchens 100 [11], Something-Something v2 [23] and Moments in Time [42]. We will release code and models.

*Equal contribution
†Equal advising

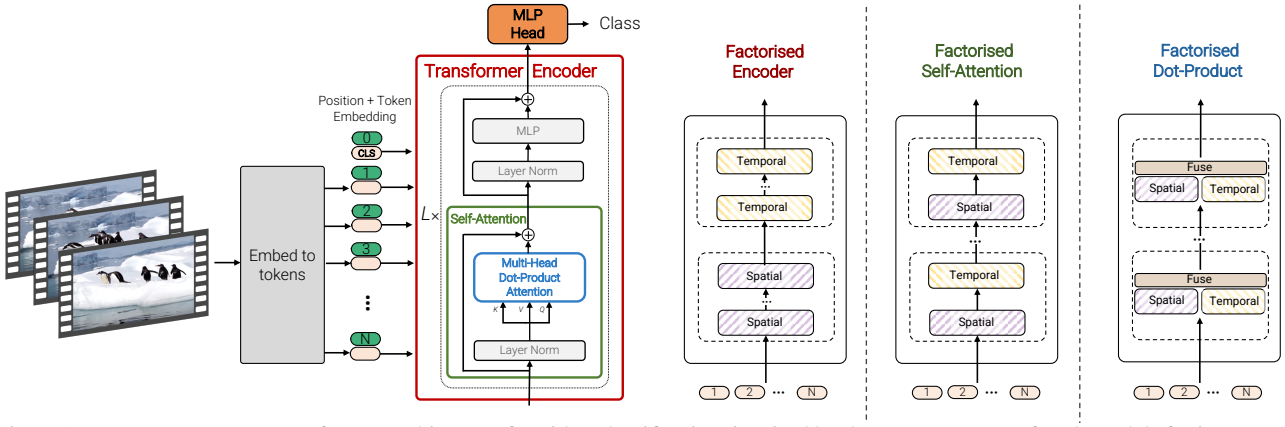


Figure 1: We propose a pure-transformer architecture for video classification, inspired by the recent success of such models for images [15]. To effectively process a large number of spatio-temporal tokens, we develop several model variants which factorise different components of the transformer encoder over the spatial- and temporal-dimensions. As shown on the right, these factorisations correspond to different attention patterns over space and time.

2. Related Work

Architectures for video understanding have mirrored advances in image recognition. Early video research used hand-crafted features to encode appearance and motion information [38, 66]. The success of AlexNet on ImageNet [35, 13] initially led to the repurposing of 2D image convolutional networks (CNNs) for video as "two-stream" networks [31, 53, 44]. These models processed RGB frames and optical flow images independently before fusing them at the end. Availability of larger video classification datasets such as Kinetics [32] subsequently facilitated the training of spatio-temporal 3D CNNs [6, 19, 62] which have significantly more parameters and thus require larger training datasets. As 3D convolutional networks require significantly more computation than their image counterparts, many architectures factorise convolutions across spatial and temporal dimensions and/or use grouped convolutions [56, 63, 64, 78, 17]. We also leverage factorisation of the spatial and temporal dimensions of videos to increase efficiency, but in the context of transformer-based models.

Concurrently, in natural language processing (NLP), Vaswani *et al*. [65] achieved state-of-the-art results by replacing convolutions and recurrent networks with the transformer network that consisted only of self-attention, layer normalisation and multilayer perceptron (MLP) operations. Current state-of-the-art architectures in NLP [14, 49] remain transformer-based, and have been scaled to web-scale datasets [3]. Many variants of the transformer have also been proposed to reduce the computational cost of self-attention when processing longer sequences [8, 9, 34, 59, 60, 70] and to improve parameter efficiency [37, 12]. Although self-attention has been employed extensively in computer vision, it has, in contrast, been typically incorporated as a layer at the end or in the later stages of the network [72, 5, 29, 74, 80] or to augment residual blocks [27, 4, 7, 54] within a ResNet architecture [24].

Although previous works attempted to replace convolutions in vision architectures [46, 50, 52], it is only very recently that Dosovitisky *et al*. [15] showed with their ViT architecture that pure-transformer networks, similar to those employed in NLP, can achieve state-of-the-art results for image classification too. The authors showed that such models are only effective at large scale, as transformers lack some of inductive biases of convolutional networks (such as translational equivariance), and thus require datasets larger than the common ImageNet ILSRVC dataset [13] to train. ViT has inspired a large amount of follow-up work in the community, and we note that there are a number of concurrent approaches on extending it to other tasks in computer vision [68, 71, 81, 82] and improving its data-efficiency [61, 45]. In particular, [2, 43] have also proposed transformer-based models for video.

In this paper, we develop pure-transformer architectures for video classification. We propose several variants of our model, including those that are more efficient by factorising the spatial and temporal dimensions of the input video. We also show how additional regularisation and pretrained models can be used to combat the fact that video datasets are not as large as their image counterparts that ViT was originally trained on. Furthermore, we outperform the state-of-the-art across five popular datasets.

3. Video Vision Transformers

We start by summarising the recently proposed Vision Transformer [15] in Sec. 3.1, and then discuss two approaches for extracting tokens from video in Sec. 3.2. Finally, we develop several transformer-based architectures for video classification in Sec. 3.3 and 3.4.

3.1. Overview of Vision Transformers (ViT)

Vision Transformer (ViT) [15] adapts the transformer architecture of [65] to process 2D images with minimal changes. In particular, ViT extracts N non-overlapping image patches, $x_i \in \mathbb{R}^{h \times w}$, performs a linear projection and then rasterises them into 1D tokens $z_i \in \mathbb{R}^d$. The sequence of tokens input to the following transformer encoder is

$$\mathbf{z} = [z_{cls}, \mathbf{E}x_1, \mathbf{E}x_2, \ldots, \mathbf{E}x_N] + \mathbf{p}, \quad (1)$$

where the projection by $\mathbf{E}$ is equivalent to a 2D convolution. As shown in Fig. 1, an optional learned classification token z_{cls} is prepended to this sequence, and its representation at the final layer of the encoder serves as the final representation used by the classification layer [14]. In addition, a learned positional embedding, $\mathbf{p} \in \mathbb{R}^{N \times d}$, is added to the tokens to retain positional information, as the subsequent self-attention operations in the transformer are permutation invariant. The tokens are then passed through an encoder consisting of a sequence of L transformer layers. Each layer ℓ comprises of Multi-Headed Self-Attention [65], layer normalisation (LN) [1], and MLP blocks as follows:

$$\mathbf{y}^\ell = \text{MSA}(\text{LN}(\mathbf{z}^\ell)) + \mathbf{z}^\ell \quad (2)$$

$$\mathbf{z}^{\ell+1} = \text{MLP}(\text{LN}(\mathbf{y}^\ell)) + \mathbf{y}^\ell. \quad (3)$$

The MLP consists of two linear projections separated by a GELU non-linearity [25] and the token-dimensionality, d, remains fixed throughout all layers. Finally, a linear classifier is used to classify the encoded input based on $z^L_{cls} \in \mathbb{R}^d$, if it was prepended to the input, or a global average pooling of all the tokens, $\mathbf{z}^L$, otherwise.

As the transformer [65], which forms the basis of ViT [15], is a flexible architecture that can operate on any sequence of input tokens $\mathbf{z} \in \mathbb{R}^{N \times d}$, we describe strategies for tokenising videos next.

3.2. Embedding video clips

We consider two simple methods for mapping a video $\mathbf{V} \in \mathbb{R}^{T \times H \times W \times C}$ to a sequence of tokens $\tilde{\mathbf{z}} \in \mathbb{R}^{n_t \times n_h \times n_w \times d}$. We then add the positional embedding and reshape into $\mathbb{R}^{N \times d}$ to obtain $\mathbf{z}$, the input to the transformer.

Uniform frame sampling As illustrated in Fig. 2, a straightforward method of tokenising the input video is to uniformly sample n_t frames from the input video clip, embed each 2D frame independently using the same method as ViT [15], and concatenate all these tokens together. Concretely, if $n_h \cdot n_w$ non-overlapping image patches are extracted from each frame, as in [15], then a total of $n_t \cdot n_h \cdot n_w$ tokens will be forwarded through the transformer encoder. Intuitively, this process may be seen as simply constructing a large 2D image to be tokenised following ViT. We note that this is the input embedding method employed by the concurrent work of [2].

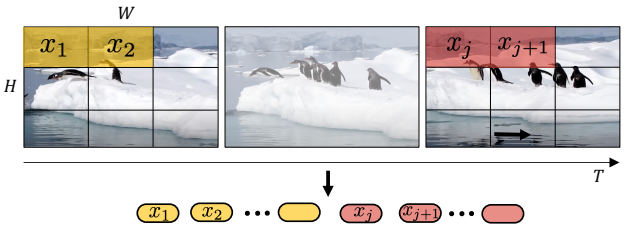


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [15].

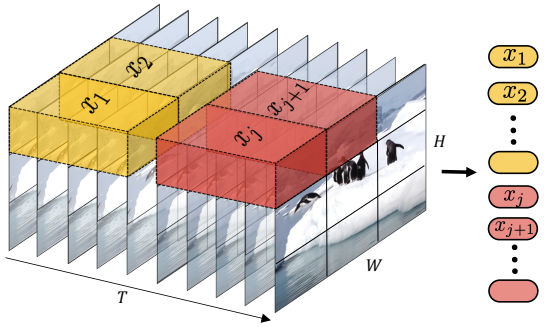


Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

Tubelet embedding An alternate method, as shown in Fig. 3, is to extract non-overlapping, spatio-temporal "tubes" from the input volume, and to linearly project this to $\mathbb{R}^d$. This method is an extension of ViT's embedding to 3D, and corresponds to a 3D convolution. For a tubelet of dimension $t \times h \times w$, $n_t = \lfloor \frac{T}{t} \rfloor$, $n_h = \lfloor \frac{H}{h} \rfloor$ and $n_w = \lfloor \frac{W}{w} \rfloor$, tokens are extracted from the temporal, height, and width dimensions respectively. Smaller tubelet dimensions thus result in more tokens which increases the computation. Intuitively, this method fuses spatio-temporal information during tokenisation, in contrast to "Uniform frame sampling" where temporal information from different frames is fused by the transformer.

3.3. Transformer Models for Video

As illustrated in Fig. 1, we propose multiple transformer-based architectures. We begin with a straightforward extension of ViT [15] that models pairwise interactions between all spatio-temporal tokens, and then develop more efficient variants which factorise the spatial and temporal dimensions of the input video at various levels of the transformer architecture.

Model 1: Spatio-temporal attention This model simply forwards all spatio-temporal tokens extracted from the video, $\mathbf{z}^0$, through the transformer encoder. We note that this has also been explored concurrently by [2] in their "Joint Space-Time" model. In contrast to CNN architectures, where the receptive field grows linearly with the number of layers, each transformer layer models all pair-

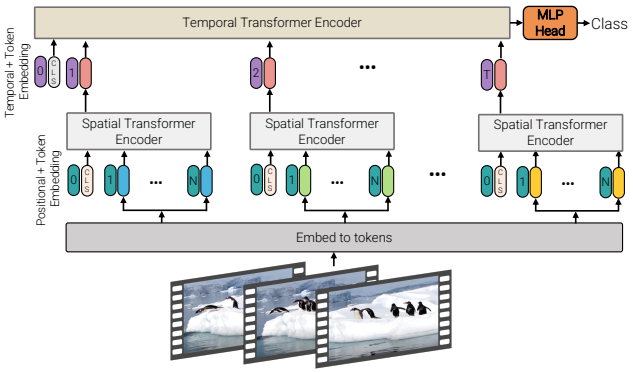


Figure 4: Factorised encoder (Model 2). This model consists of two transformer encoders in series: the first models interactions between tokens extracted from the same temporal index to produce a latent representation per time-index. The second transformer models interactions between time steps. It thus corresponds to a "late fusion" of spatial- and temporal information.

wise interactions between all spatio-temporal tokens, and it thus models long-range interactions across the video from the first layer. However, as it models all pairwise interactions, Multi-Headed Self Attention (MSA) [65] has quadratic complexity with respect to the number of tokens. This complexity is pertinent for video, as the number of tokens increases linearly with the number of input frames, and motivates the development of more efficient architectures next.

Model 2: Factorised encoder As shown in Fig. 4, this model consists of two separate transformer encoders. The first, spatial encoder, only models interactions between tokens extracted from the same temporal index. A representation for each temporal index, $h_i \in \mathbb{R}^d$, is obtained after L_s layers: This is the encoded classification token, $z_{cls}^{L_s}$ if it was prepended to the input (Eq. 1), or a global average pooling from the tokens output by the spatial encoder, $\mathbf{z}^{L_s}$, otherwise. The frame-level representations, h_i, are concatenated into $\mathbf{H} \in \mathbb{R}^{n_t \times d}$, and then forwarded through a temporal encoder consisting of L_t transformer layers to model interactions between tokens from different temporal indices. The output token of this encoder is then finally classified.

This architecture corresponds to a "late fusion" [31, 53, 69, 43] of temporal information, and the initial spatial encoder is identical to the one used for image classification. It is thus analogous to CNN architectures such as [21, 31, 69, 83] which first extract per-frame features, and then aggregate them into a final representation before classifying them. Although this model has more transformer layers than Model 1 (and thus more parameters), it requires fewer floating point operations (FLOPs), as the two separate transformer blocks have a complexity of $\mathcal{O}((n_h \cdot n_w)^2 + n_t^2)$ compared to $\mathcal{O}((n_t \cdot n_h \cdot n_w)^2)$ of Model 1.

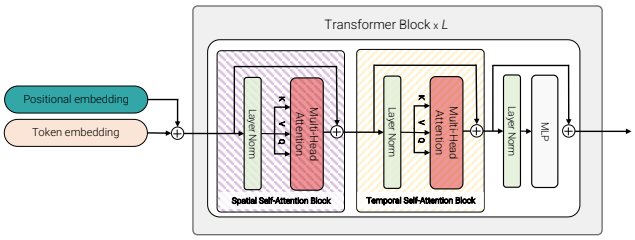


Figure 5: Factorised self-attention (Model 3). Within each transformer block, the multi-headed self-attention operation is factorised into two operations (indicated by striped boxes) that first only compute self-attention spatially, and then temporally.

Model 3: Factorised self-attention This model, in contrast, contains the same number of transformer layers as Model 1. However, instead of computing multi-headed self-attention across all pairs of tokens, $\mathbf{z}^\ell$, at layer l, we factorise the operation to first only compute self-attention spatially (among all tokens extracted from the same temporal index), and then temporally (among all tokens extracted from the same spatial index) as shown in Fig. 5. Each self-attention block in the transformer thus models spatio-temporal interactions, but does so more efficiently than Model 1 by factorising the operation over two smaller sets of elements, thus achieving the same computational complexity as Model 2. We note that factorising attention over input dimensions has also been explored in [26, 75], and concurrently in the context of video by [2] in their "Divided Space-Time" model.

This operation can be performed efficiently by reshaping the tokens $\mathbf{z}$ from $\mathbb{R}^{1 \times n_t \cdot n_h \cdot n_w \cdot d}$ to $\mathbb{R}^{n_t \times n_h \cdot n_w \cdot d}$ (denoted by $\mathbf{z}_s$) to compute spatial self-attention. Similarly, the input to temporal self-attention, $\mathbf{z}_t$ is reshaped to $\mathbb{R}^{n_h \cdot n_w \times n_t \cdot d}$. Here we assume the leading dimension is the "batch dimension". Our factorised self-attention is defined as

$$\mathbf{y}_s^\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_s^\ell)) + \mathbf{z}_s^\ell \quad (4)$$

$$\mathbf{y}_t^\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{y}_s^\ell)) + \mathbf{y}_s^\ell \quad (5)$$

$$\mathbf{z}^{\ell+1} = \mathrm{MLP}(\mathrm{LN}(\mathbf{y}_t^\ell)) + \mathbf{y}_t^\ell. \quad (6)$$

We observed that the order of spatial-then-temporal self-attention or temporal-then-spatial self-attention does not make a difference, provided that the model parameters are initialised as described in Sec. 3.4. Note that the number of parameters, however, increases compared to Model 1, as there is an additional self-attention layer (cf. Eq. 7). We do not use a classification token in this model, to avoid ambiguities when reshaping the input tokens between spatial and temporal dimensions.

Model 4: Factorised dot-product attention Finally, we develop a model which has the same computational complexity as Models 2 and 3, while retaining the same number of parameters as the unfactorised Model 1. The factorisation of spatial- and temporal dimensions is similar in spirit

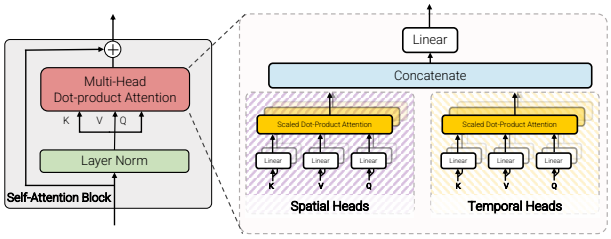


Figure 6: Factorised dot-product attention (Model 4). For half of the heads, we compute dot-product attention over only the spatial axes, and for the other half, over only the temporal axis.

to Model 3, but we factorise the multi-head dot-product attention operation instead (Fig. 6). Concretely, we compute attention weights for each token separately over the spatial- and temporal-dimensions using different heads. First, we note that the attention operation for each head is defined as

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \quad (7)$$

In self-attention, the queries $\mathbf{Q} = \mathbf{X}\mathbf{W}_q$, keys $\mathbf{K} = \mathbf{X}\mathbf{W}_k$, and values $\mathbf{V} = \mathbf{X}\mathbf{W}_v$ are linear projections of the input $\mathbf{X}$ with $\mathbf{X},\mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{R}^{N\times d}$. Note that in the unfactorised case (Model 1), the spatial and temporal dimensions are merged as $N = n_t \cdot n_h \cdot n_w$.

The main idea here is to modify the keys and values for each query to only attend over tokens from the same spatial- and temporal index by constructing $\mathbf{K}_s, \mathbf{V}_s \in \mathbb{R}^{n_h\cdot n_w \times d}$ and $\mathbf{K}_t, \mathbf{V}_t \in \mathbb{R}^{n_t \times d}$, namely the keys and values corresponding to these dimensions. Then, for half of the attention heads, we attend over tokens from the spatial dimension by computing $\mathbf{Y}_s = \text{Attention}(\mathbf{Q},\mathbf{K}_s,\mathbf{V}_s)$, and for the rest we attend over the temporal dimension by computing $\mathbf{Y}_t = \text{Attention}(\mathbf{Q},\mathbf{K}_t,\mathbf{V}_t)$. Given that we are only changing the attention neighbourhood for each query, the attention operation has the same dimension as in the unfactorised case, namely $\mathbf{Y}_s, \mathbf{Y}_t \in \mathbb{R}^{N\times d}$. We then combine the outputs of multiple heads by concatenating them and using a linear projection [65], $\mathbf{Y} = \text{Concat}(\mathbf{Y}_s,\mathbf{Y}_t)\mathbf{W}_O$.

3.4. Initialisation by leveraging pretrained models

ViT [15] has been shown to only be effective when trained on large-scale datasets, as transformers lack some of the inductive biases of convolutional networks [15]. However, even the largest video datasets such as Kinetics [32], have several orders of magnitude less labelled examples when compared to their image counterparts [13, 36, 55]. As a result, training large models from scratch to high accuracy is extremely challenging. To sidestep this issue, and enable more efficient training we initialise our video models from pretrained image models. However, this raises several practical questions, specifically on how to initialise parameters not present or incompatible with image models. We now discuss several effective strategies to initialise these large-scale video classification models.

Positional embeddings A positional embedding $\mathbf{p}$ is added to each input token (Eq. 1). However, our video models have n_t times more tokens than the pretrained image model. As a result, we initialise the positional embeddings by "repeating" them temporally from $\mathbb{R}^{n_w\cdot n_h \times d}$ to $\mathbb{R}^{n_t\cdot n_h\cdot n_w\times d}$. Therefore, at initialisation, all tokens with the same spatial index have the same embedding which is then fine-tuned.

Embedding weights, $\mathbf{E}$ When using the "tubelet embedding" tokenisation method (Sec. 3.2), the embedding filter $\mathbf{E}$ is a 3D tensor, compared to the 2D tensor in the pretrained model, $\mathbf{E}_{\text{image}}$. A common approach for initialising 3D convolutional filters from 2D filters for video classification is to "inflate" them by replicating the filters along the temporal dimension and averaging them [6, 19] as

$$\mathbf{E} = \frac{1}{t}[\mathbf{E}_{\text{image}},\ldots,\mathbf{E}_{\text{image}},\ldots,\mathbf{E}_{\text{image}}]. \quad (8)$$

We consider an additional strategy, which we denote as "central frame initialisation", where $\mathbf{E}$ is initialised with zeroes along all temporal positions, except at the centre $\lfloor\frac{t}{2}\rfloor$,

$$\mathbf{E} = [\mathbf{0},\ldots,\mathbf{E}_{\text{image}},\ldots,\mathbf{0}]. \quad (9)$$

Therefore, the 3D convolutional filter effectively behaves like "Uniform frame sampling" (Sec. 3.2) at initialisation, while also enabling the model to learn to aggregate temporal information from multiple frames as training progresses.

Transformer weights for Model 3 The transformer block in Model 3 (Fig. 5) differs from the pretrained ViT model [15], in that it contains two multi-headed self attention (MSA) modules. In this case, we initialise the spatial MSA module from the pretrained module, and initialise all weights of the temporal MSA with zeroes, such that Eq. 5 behaves as a residual connection [24] at initialisation.

4. Empirical evaluation

We first present our experimental setup and implementation details in Sec. 4.1, before ablating various components of our model in Sec. 4.2. We then present state-of-the-art results on five datasets in Sec. 4.3.

4.1. Experimental Setup

Network architecture and training Our backbone architecture follows that of ViT [15] and BERT [14]. We consider ViT-Base (ViT-B, L=12, N_H=12, d=3072), ViT-Large (ViT-L, L=24, N_H=16, d=4096), and ViT-Huge (ViT-H, L=32, N_H=16, d=5120), where L is the number of transformer layers, each with a self-attention block of N_H heads

Table 1: Comparison of input encoding methods using ViViT-B and spatio-temporal attention on Kinetics. Further details in text.

	Top-1 accuracy
Uniform frame sampling	78.5
Tubelet embedding	
Random initialisation [22]	73.2
Filter inflation [6]	77.6
Central frame	79.2

and hidden dimension d. We also apply the same naming scheme to our models (e.g., ViViT-B/16x2 denotes a ViT-Base backbone with a tubelet size of $h \times w \times t = 16 \times 16 \times 2$). In all experiments, the tubelet height and width are equal. Note that smaller tubelet sizes correspond to more tokens at the input, and thus more computation.

We train our models using synchronous SGD and momentum, a cosine learning rate schedule and TPU-v3 accelerators. We initialise our models from a ViT image model trained either on ImageNet-21K [13] (unless otherwise specified) or the larger JFT [55] dataset. Exact experimental hyperparameters are detailed in the appendix.

Datasets We evaluate the performance of our proposed models on a diverse set of video classification datasets:

Kinetics [32] consists of 10-second videos sampled at 25fps from YouTube. We evaluate on both Kinetics 400 and 600, containing 400 and 600 classes respectively. As these are dynamic datasets (videos may be removed from YouTube), we note our dataset sizes are approximately 267 000 and 446 000 respectively.

Epic Kitchens-100 consists of egocentric videos capturing daily kitchen activities spanning 100 hours and 90 000 clips [11]. We report results following the standard "action recognition" protocol. Here, each video is labelled with a "verb" and a "noun" and we therefore predict both categories using a single network with two "heads". The top-scoring verb and action pair predicted by the network form an "action", and action accuracy is the primary metric.

Moments in Time [42] consists of 800 000, 3-second YouTube clips that capture the gist of a dynamic scene involving animals, objects, people, or natural phenomena.

Something-Something v2 (SSv2) [23] contains 220 000 videos, with durations ranging from 2 to 6 seconds. In contrast to the other datasets, the objects and backgrounds in the videos are consistent across different action classes, and this dataset thus places more emphasis on a model's ability to recognise fine-grained motion cues.

Inference The input to our network is a video clip of 32 frames using a stride of 2, unless otherwise mentioned, similar to [18, 17]. Following common practice, at inference time, we process multiple views of a longer video and average per-view logits to obtain the final result. Unless otherwise specified, we use a total of 4 views per video (as this is sufficient to "see" the entire video clip across the various datasets), and ablate these and other design choices next.

Table 2: Comparison of model architectures using ViViT-B as the backbone, and tubelet size of 16×2. We report Top-1 accuracy on Kinetics 400 (K400) and action accuracy on Epic Kitchens (EK). Runtime is during inference on a TPU-v3.

	K400	EK	FLOPs ($\times 10^9$)	Params ($\times 10^6$)	Runtime (ms)
Model 1: Spatio-temporal	80.0	43.1	455.2	88.9	58.9
Model 2: Fact. encoder	78.8	43.7	284.4	100.7	17.4
Model 3: Fact. self-attention	77.4	39.1	372.3	117.3	31.7
Model 4: Fact. dot product	76.3	39.5	277.1	88.9	22.9
Model 2: Ave. pool baseline	75.8	38.8	283.9	86.7	17.3

Table 3: The effect of varying the number of temporal transformers, L_t, in the Factorised encoder model (Model 2). We report the Top-1 accuracy on Kinetics 400. Note that $L_t = 0$ corresponds to the "average pooling baseline".

L_t	0	1	4	8	12
Top-1	75.8	78.6	78.8	78.8	78.9

4.2. Ablation study

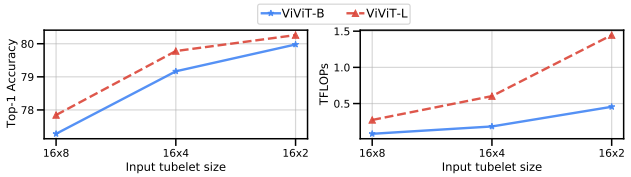
Input encoding We first consider the effect of different input encoding methods (Sec. 3.2) using our unfactorised model (Model 1) and ViViT-B on Kinetics 400. As we pass 32-frame inputs to the network, sampling 8 frames and extracting tubelets of length $t = 4$ correspond to the same number of tokens in both cases. Table 1 shows that tubelet embedding initialised using the "central frame" method (Eq. 9) performs well, outperforming the commonly-used "filter inflation" initialisation method [6, 19] by 1.6%, and "uniform frame sampling" by 0.7%. We therefore use this encoding method for all subsequent experiments.

Model variants We compare our proposed model variants (Sec. 3.3) across the Kinetics 400 and Epic Kitchens datasets, both in terms of accuracy and efficiency, in Tab. 2. In all cases, we use the "Base" backbone and tubelet size of 16×2. Model 2 ("Factorised Encoder") has an additional hyperparameter, the number of temporal transformers, L_t. We set $L_t = 4$ for all experiments and show in Tab. 3 that the model is not sensitive to this choice.

The unfactorised model (Model 1) performs the best on Kinetics 400. However, it can also overfit on smaller datasets such as Epic Kitchens, where we find our "Factorised Encoder" (Model 2) to perform the best. We also consider an additional baseline (last row), based on Model 2, where we do not use any temporal transformer, and simply average pool the frame-level representations from the spatial encoder before classifying. This average pooling baseline performs the worst, and has a larger accuracy drop on Epic Kitchens, suggesting that this dataset requires more detailed modelling of temporal relations.

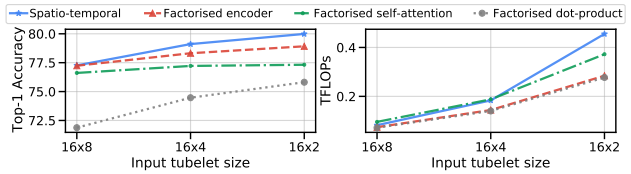
Table 4: The effect of progressively adding regularisation (each row includes all methods above it) on Top-1 action accuracy on Epic Kitchens. We use a Factorised encoder model with tubelet size 16×2.

	Top-1 accuracy
Random crop, flip, colour jitter	38.4
+ Kinetics 400 initialisation	39.6
+ Stochastic depth [28]	40.2
+ Random augment [10]	41.1
+ Label smoothing [58]	43.1
+ Mixup [79]	43.7



(a) Accuracy (b) Compute

Figure 7: The effect of the backbone architecture on (a) accuracy and (b) computation on Kinetics 400, for the spatio-temporal attention model (Model 1).



(a) Accuracy (b) Compute

Figure 8: The effect of varying the number of temporal tokens on (a) accuracy and (b) computation on Kinetics 400, for different variants of our model with a ViViT-B backbone.

As described in Sec. 3.3, all factorised variants of our model use significantly fewer FLOPs than the unfactorised Model 1, as the attention is computed separately over spatial- and temporal-dimensions. Model 4 adds no additional parameters to the unfactorised Model 1, and uses the least compute. The temporal transformer encoder in Model 2 operates on only n_t tokens, which is why there is a barely a change in compute and runtime over the average pooling baseline, even though it improves the accuracy substantially (3% on Kinetics and 4.9% on Epic Kitchens). Finally, Model 3 requires more compute and parameters than the other factorised models, as its additional self-attention block means that it performs another query-, key-, value- and output-projection in each transformer layer [65].

Model regularisation Pure-transformer architectures such as ViT [15] are known to require large training datasets, and we observed overfitting on smaller datasets like Epic Kitchens and SSv2, even when using an ImageNet pretrained model. In order to effectively train our models on such datasets, we employed several regularisation strategies that we ablate using our "Factorised encoder" model in Tab. 4. We note that these regularisers were originally proposed for training CNNs, and that [61] have recently explored them for training ViT for image classification.

Table 5: The effect of spatial resolution on the performance of ViViT-L/16x2 and spatio-temporal attention on Kinetics 400.

Crop size	224	288	320
Accuracy	80.3	80.7	81.0
GFLOPs	1446	2919	3992
Runtime	58.9	147.6	238.8

Each row of Tab. 4 includes all the methods from the rows above it, and we observe progressive improvements from adding each regulariser. Overall, we obtain a substantial overall improvement of 5.3% on Epic Kitchens. We also achieve a similar improvement of 5%, from 60.4% to 65.4%, on SSv2 by using all the regularisation in Tab. 4. Note that the Kinetics-pretrained models that we initialise from are from Tab. 2, and that all Epic Kitchens models in Tab. 2 were trained with all the regularisers in Tab. 4. For larger datasets like Kinetics and Moments in Time, we do not use these additional regularisers (we use only the first row of Tab. 4), as we obtain state-of-the-art results without them. The appendix contains hyperparameter values and additional details for all regularisers.

Varying the backbone Figure 7 compares the ViViT-B and ViViT-L backbones for the unfactorised spatio-temporal model. We observe consistent improvements in accuracy as the backbone capacity increases. As expected, the compute also grows as a function of the backbone size.

Varying the number of tokens We first analyse the performance as a function of the number of tokens along the temporal dimension in Fig. 8. We observe that using smaller input tubelet sizes (and therefore more tokens) leads to consistent accuracy improvements across all of our model architectures. At the same time, computation in terms of FLOPs increases accordingly, and the unfactorised model (Model 1) is impacted the most.

We then vary the number of tokens fed into the model by increasing the spatial crop-size from the default of 224 to 320 in Tab. 5. As expected, there is a consistent increase in both accuracy and computation. We note that when comparing to prior work we consistently obtain state-of-the-art results (Sec. 4.3) using a spatial resolution of 224, but we also highlight that further improvements can be obtained at higher spatial resolutions.

Varying the number of input frames In our experiments so far, we have kept the number of input frames fixed to 32. We now ablate this choice whilst effectively keeping the amount of computation to process a single clip fixed. This is done by increasing the tubelet length t in proportion to

Table 6: Comparisons to state-of-the-art across multiple datasets. For "views", $x \times y$ denotes x temporal crops and y spatial crops. "320" denotes models trained and tested with a spatial resolution of 320 instead of 224.

(a) Kinetics 400

Method	Top 1	Top 5	Views
blVNet [16]	73.5	91.2	–
STM [30]	73.7	91.6	–
TEA [39]	76.1	92.5	10 × 3
TSM-ResNeXt-101 [40]	76.3	–	–
I3D NL [72]	77.7	93.3	10 × 3
CorrNet-101 [67]	79.2	–	10 × 3
ip-CSN-152 [63]	79.2	93.8	10 × 3
LGD-3D R101 [48]	79.4	94.4	–
SlowFast R101-NL [18]	79.8	93.9	10 × 3
X3D-XXL [17]	80.4	94.6	10 × 3
TimeSformer-L [2]	80.7	94.7	1 × 3
ViViT-L/16x2	80.6	94.7	4 × 3
ViViT-L/16x2 320	**81.3**	**94.7**	4 × 3
Methods with large-scale pretraining			
ip-CSN-152 [63] (IG [41])	82.5	95.3	10 × 3
ViViT-L/16x2 (JFT)	82.8	95.5	4 × 3
ViViT-L/16x2 320 (JFT)	83.5	95.5	4 × 3
ViViT-H/16x2 (JFT)	**84.8**	**95.8**	4 × 3

(b) Kinetics 600

Method	Top 1	Top 5	Views
AttentionNAS [73]	79.8	94.4	–
LGD-3D R101 [48]	81.5	95.6	–
SlowFast R101-NL [18]	81.8	95.1	10 × 3
X3D-XL [17]	81.9	95.5	10 × 3
TimeSformer-HR [2]	82.4	**96.0**	–
ViViT-L/16x2	82.5	95.6	4 × 3
ViViT-L/16x2 320	**83.0**	95.7	4 × 3
ViViT-L/16x2 (JFT)	84.3	96.2	4 × 3
ViViT-H/16x2 (JFT)	**85.8**	**96.5**	4 × 3

(c) Moments in Time

	Top 1	Top 5
TSN [69]	25.3	50.1
TRN [83]	28.3	53.4
I3D [6]	29.5	56.1
blVNet [16]	31.4	59.3
AssembleNet-101 [51]	34.3	62.7
ViViT-L/16x2	**38.0**	**64.9**

(d) Epic Kitchens 100 Top 1 accuracy

Method	Action	Verb	Noun
TSN [69]	33.2	60.2	46.0
TRN [83]	35.3	65.9	45.4
TBN [33]	36.7	66.0	47.2
TSM [40]	38.3	**67.9**	49.0
SlowFast [18]	38.5	65.6	50.0
ViViT-L/16x2 Fact. encoder	**44.0**	66.4	**56.8**

(e) Something-Something v2

Method	Top 1	Top 5
TRN [83]	48.8	77.6
SlowFast [17, 77]	61.7	–
TimeSformer-HR [2]	62.5	–
TSM [40]	63.4	88.5
STM [30]	64.2	89.8
TEA [39]	65.1	–
blVNet [16]	65.2	**90.3**
ViViT-L/16x2 Fact. encoder	**65.4**	89.8

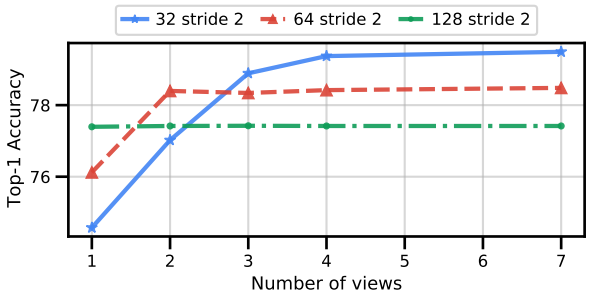


Figure 9: The effect of varying the number of frames input to the network when keeping the number of tokens constant by adjusting the tubelet length t. We use ViViT-B, and spatio-temporal attention on Kinetics 400. A Kinetics video contains 250 frames (10 seconds sampled at 25 fps) and the accuracy for each model saturates once the number of equidistant temporal views is sufficient to "see" the whole video clip.

the number of input frames, such that the number of tokens processed by the network is constant.

Figure 9 shows that as we increase the number of frames input to the network, and enlarge the tubelet length, t, accordingly, our accuracy from processing a single clip increases, as the network incorporates longer temporal context. However, common practice on datasets such as Kinetics [18, 72, 39] is to average results over multiple, shorter "views" of the same video clip. Following this multi-view testing protocol, processing shorter clips is actually more advantageous. We also observe from Fig. 9 that the accuracy saturates once the number of views is sufficent to cover the whole video, and that this is consistent across models with different numbers of input frames. Consequently, we use 32 frames, sampled with a stride of 2, as our network-input for all experiments, and use 4 temporal views for multi-view testing. Finally, we note that as our models can process longer videos without increasing the number of tokens, they offer an efficient method for processing longer videos than those considered by existing video classification datasets, and keep it as an avenue for future work.

4.3. Comparison to state-of-the-art

Based on our ablation studies in the previous section, we compare to the current state-of-the-art using two of our model variants. We use the unfactorised spatio-temporal attention model (Model 1) for the larger datasets, Kinetics and Moments in Time. For the smaller datasets, Epic Kitchens and SSv2, we use our Factorised encoder model (Model 2).

Kinetics Tables 6a and 6b show that our spatio-temporal attention models outperform the state-of-the-art on Kinetics 400 and 600 respectively. Following standard practice, we take 3 spatial crops (left, centre and right) [18, 17, 63, 72] for each temporal view, and notably, we require significantly fewer views than previous CNN-based methods.

We surpass the previous CNN-based state-of-the-art using ViViT-L/16x2 pretrained on ImageNet, and achieve comparable accuracy to [2] who concurrently proposed a pure-transformer architecture. We then obtain further improvements by increasing the spatial resolution from 224 to 320 as expected given the ablation in Tab. 5. Moreover, by initialising our backbones from models pretrained on the larger JFT dataset [55], we obtain further improvements. Although these models are not directly comparable to previous work, we do also outperform [63] who pretrained on the large-scale, Instagram dataset [41]. Our best model uses a ViViT-H backbone pretrained on JFT and significantly advances the best reported results on Kinetics 400 and 600 to 84.8% and 85.8%, respectively.

Moments in Time We surpass the state-of-the-art by a significant margin as shown in Tab. 6c. We note that the videos in this dataset are diverse and contain significant label noise, making this task challenging and leading to lower accuracies than on other datasets.

Epic Kitchens 100 Table 6d shows that our Factorised encoder model outperforms previous methods by a significant margin. In addition, our model obtains substantial improvements for Top-1 accuracy of "noun" classes, and the only method which achieves higher "verb" accuracy used optical flow as an additional input modality [40, 47]. Furthermore, all variants of our model presented in Tab. 2 outperformed the existing state-of-the-art on action accuracy. We note that we use the same model to predict verbs and nouns using two separate "heads", and for simplicity, we do not use separate loss weights for each head.

Something-Something v2 (SSv2) Finally, Tab. 6e shows that we achieve state-of-the-art Top-1 accuracy with our Factorised encoder model (Model 2), albeit with a smaller margin compared to previous methods. Notably, our Factorised encoder model significantly outperforms the concurrent TimeSformer [2] method by 2.9%, which also proposes a pure-transformer model, but does not consider our Factorised encoder variant or our additional regularisation.

SSv2 differs from other datasets in that the backgrounds and objects are quite similar across different classes, meaning that recognising fine-grained motion patterns is necessary to distinguish classes from each other. Our results suggest that capturing these fine-grained motions is an area of improvement and future work for our model. We also note an inverse correlation between the relative performance of previous methods on SSv2 (Tab. 6e) and Kinetics (Tab. 6a) suggesting that these two datasets evaluate complementary characteristics of a model.

5. Conclusion and Future Work

We have presented four pure-transformer models for video classification, with different accuracy and efficiency profiles, achieving state-of-the-art results across five popular datasets. Furthermore, we have shown how to effectively regularise such high-capacity models for training on smaller datasets and thoroughly ablated our main design choices. Future work is to remove our dependence on image-pretrained models. Finally, going beyond video classification towards more complex tasks is a clear next step.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. In *arXiv preprint arXiv:1607.06450*, 2016. 3

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *arXiv preprint arXiv:2102.05095*, 2021. 2, 3, 4, 8, 9

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *CVPR Workshops*, 2019. 2

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 5, 6, 8

[7] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. A2-nets: Double attention networks. In *NeurIPS*, 2018. 2

[8] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. In *arXiv preprint arXiv:1904.10509*, 2019. 2

[9] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *arXiv preprint arXiv:2009.14794*, 2020. 2

[10] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *arXiv preprint arXiv:1909.13719*, 2019. 7, 12, 13

[11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision. In *arXiv preprint arXiv:2006.13256*, 2020. 1, 6

[12] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *ICLR*, 2019. 2

[13] 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. 2, 5, 6

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 2, 3, 5

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 3, 5, 7

[16] Quanfu Fan, Chun-Fu Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *NeurIPS*, 2019. 8

[17] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 1, 2, 6, 8

[18] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 1, 6, 8

[19] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NeurIPS*, 2016. 2, 5, 6

[20] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *CVPR*, 2019. 1

[21] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *NeurIPS*, 2017. 4

[22] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[23] 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. 1, 6

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 5

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). In *arXiv preprint arXiv:1606.08415*, 2016. 3

[26] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. In *arXiv preprint arXiv:1912.12180*, 2019. 4

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 2

[28] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 7, 12, 13

[29] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 2

[30] Boyuan Jiang, Mengmeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019. 8

[31] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2, 4

[32] Will Kay, Joao 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. In *arXiv preprint arXiv:1705.06950*, 2017. 1, 2, 5, 6

[33] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019. 8

[34] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 2

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, volume 25, 2012. 1, 2

[36] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020. 5

[37] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020. 2

[38] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3), 2005. 2

[39] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020. 8

[40] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 8, 9

[41] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 8

[42] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *PAMI*, 42(2):502–508, 2019. 1, 6

[43] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. In *arXiv preprint arXiv:2102.00719*, 2021. 2, 4

[44] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[45] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable visual transformers with hierarchical pooling. In *arXiv preprint arXiv:2103.10619*, 2021. 2

[46] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018. 1, 2

[47] Will Price and Dima Damen. An evaluation of action recognition models on epic-kitchens. In *arXiv preprint arXiv:1908.00867*, 2019. 9

[48] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, 2019. 8

[49] 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. 2

[50] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 1, 2

[51] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *ICLR*, 2020. 8

[52] Zhuoran Shen, Irwan Bello, Raviteja Vemulapalli, Xuhui Jia, and Ching-Hui Chen. Global self-attention networks for image recognition. In *arXiv preprint arXiv:2010.03019*, 2021. 2

[53] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014. 2, 4

[54] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *CVPR*, 2021. 2

[55] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 5, 6, 8

[56] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 2

[57] 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. 1

[58] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 7, 12, 13

[59] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *arXiv preprint arXiv:2011.04006*, 2020. 2

[60] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. In *arXiv preprint arXiv:2009.06732*, 2020. 2

[61] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 7

[62] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2

[63] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019. 2, 8

[64] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 2

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 3, 4, 5, 7

[66] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1), 2013. 2

[67] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *CVPR*, 2020. 8

[68] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *arXiv preprint arXiv:2012.00759*, 2020. 2

[69] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 4, 8

[70] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. In *arXiv preprint arXiv:2006.04768*, 2020. 2

[71] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *arXiv preprint arXiv:2102.12122*, 2021. 2

[72] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 1, 2, 8

[73] Xiaofang Wang, Xuehan Xiong, Maxim Neumann, AJ Piergiovanni, Michael S Ryoo, Anelia Angelova, Kris M Kitani, and Wei Hua. Attentionnas: Spatiotemporal attention cell search for video classification. In *ECCV*, 2020. 8

[74] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *arXiv preprint arXiv:2011.14503*, 2020. 2

[75] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *ICLR*, 2020. 4

[76] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019. 1

[77] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *CVPR*, 2020. 8

[78] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 2

[79] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 7, 12, 13

[80] Li Zhang, Dan Xu, Anurag Arnab, and Philip HS Torr. Dynamic graph message passing networks. In *CVPR*, 2020. 2

[81] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *arXiv preprint arXiv:2012.09164*, 2020. 2

[82] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *arXiv preprint arXiv:2012.15840*, 2020. 2

[83] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 4, 8

Appendix

A. Additional experimental details

In this appendix, we provide additional experimental details. Section A.1 provides additional details about the regularisers we used and Sec. A.2 details the training hyperparamters used for our experiments.

A.1. Further details about regularisers

In this section, we provide additional details and list the hyperparameters of the additional regularisers that we employed in Tab. 4. Hyperparameter values for all our experiments are listed in Tab. 7.

Stochastic depth Stochastic depth regularisation was originally proposed for training very deep residual networks [28]. Intuitively, the outputs of a layer, ℓ, are "dropped out" with probability, $p_{\text{drop}}(\ell)$ during training, by setting the output of the layer to be equal to its input.

Following [28], we linearly increase the probability of dropping a layer according to its depth within the network,

$$p_{\text{drop}}(\ell) = \frac{\ell}{L} p_{\text{drop}}, \tag{10}$$

where ℓ is the index of the layer in the network, and L is the total number of layers.

Random augment Random augment [10] randomly applies data augmentation transformations sequentially to an input example. We follow the public implementation[1], but modify the data augmentation operations to be temporally consistent throughout the video (in other words, the same transformation is applied on each frame of the video).

The authors define two hyperparameters for Random augment, "number of layers" , the number of augmentation transformations to apply sequentially to a video and "magnitude", the strength of the transformation that is shared across all augmentation operations. Our values for these parameters are shown in Tab. 7.

Label smoothing Label smoothing was proposed by [58] originally to regularise training Inception-v3. Concretely, the label distribution used during training, $\tilde{y}$, is a mixture of the one-hot ground-truth label, y, and a uniform distribution, u, to encourage the network to produce less confident predictions during training:

$$\tilde{y} = (1 - \lambda)y + \lambda u. \tag{11}$$

There is therefore one scalar hyperparamter, $\lambda \in [0, 1]$.

[1] https://github.com/tensorflow/models/blob/master/official/vision/beta/ops/augment.py

Mixup Mixup [79] constructs virtual training examples which are a convex combination of pairs of training examples and their labels. Concretely, given (x_i, y_i) and (x_j, y_j) where x_i denotes an input vector and y_i a one-hot input label, mixup constructs the virtual training example,

$$\begin{aligned} \tilde{x} &= \alpha x_i + (1 - \alpha) x_j \\ \tilde{y} &= \alpha y_i + (1 - \alpha) y_j. \end{aligned} \tag{12}$$

Our choice of the hyperparamter $\alpha \in [0, 1]$ is detailed in Tab. 7.

A.2. Training hyperparameters

Table 7 details the hyperparamters for all of our experiments. We use synchronous SGD with momentum, a cosine learning rate schedule with linear warmup, and a batch size of 64 for all experiments. As aforementioned, we only employed additional regularisation when training on the smaller Epic Kitchens and Something-Something v2 datasets.

Table 7: Training hyperparamters for experiments in the main paper. "–" indicates that the regularisation method was not used at all. Values which are constant across all columns are listed once. Datasets are denoted as follows: K400: Kinetics 400. K600: Kinetics 600. MiT: Moments in Time. EK: Epic Kitchens. SSv2: Something-Something v2.

	K400	K600	MiT	EK	SSv2
Optimisation					
Optimiser			Synchronous SGD		
Momentum			0.9		
Batch size			64		
Learning rate schedule			cosine with linear warmup		
Linear warmup epochs			2.5		
Base learning rate	0.1	0.1	0.25	0.5	0.5
Epochs	30	30	10	50	35
Data augmentation					
Random crop probability			1.0		
Random flip probability			0.5		
Scale jitter probability			1.0		
Maximum scale			1.33		
Minimum scale			0.9		
Colour jitter probability	0.8	0.8	0.8	–	–
Rand augment number of layers [10]	–	–	–	2	2
Rand augment magnitude [10]	–	–	–	15	20
Other regularisation					
Stochastic droplayer rate, p_{drop} [28]	–	–	–	0.2	0.3
Label smoothing λ [58]	–	–	–	0.2	0.3
Mixup α [79]	–	–	–	0.1	0.3