Multiscale Vision Transformers

Haoqi Fan [*,1] Bo Xiong [*,1] Karttikeya Mangalam [*,1,2]
Yanghao Li [*,1] Zhicheng Yan [1] Jitendra Malik [1,2] Christoph Feichtenhofer [*,1]

[1]Facebook AI Research [2]UC Berkeley

Abstract

We present Multiscale Vision Transformers (MViT) for video and image recognition, by connecting the seminal idea of multiscale feature hierarchies with transformer models. Multiscale Transformers have several channel-resolution scale stages. Starting from the input resolution and a small channel dimension, the stages hierarchically expand the channel capacity while reducing the spatial resolution. This creates a multiscale pyramid of features with early layers operating at high spatial resolution to model simple low-level visual information, and deeper layers at spatially coarse, but complex, high-dimensional features. We evaluate this fundamental architectural prior for modeling the dense nature of visual signals for a variety of video recognition tasks where it outperforms concurrent vision transformers that rely on large scale external pre-training and are 5-10× more costly in computation and parameters. We further remove the temporal dimension and apply our model for image classification where it outperforms prior work on vision transformers. Code is available at: https://github.com/facebookresearch/SlowFast.

1. Introduction

We begin with the intellectual history of neural network models for computer vision. Based on their studies of cat and monkey visual cortex, Hubel and Wiesel [60] developed a *hierarchical* model of the visual pathway with neurons in lower areas such as V1 responding to features such as oriented edges and bars, and in higher areas to more specific stimuli. Fukushima proposed the Neocognitron [37], a neural network architecture for pattern recognition explicitly motivated by Hubel and Wiesel's hierarchy. His model had alternating layers of simple cells and complex cells, thus incorporating downsampling, and shift invariance, thus incorporating convolutional structure. LeCun *et al*. [70] took the additional step of using backpropagation to train the weights of this network. But already the main aspects of hierarchy of visual processing had been established: (*i*) Reduction in spatial resolution as one goes up the processing hierarchy and (*ii*) Increase in the number of different "channels", with each channel corresponding to ever more specialized features.

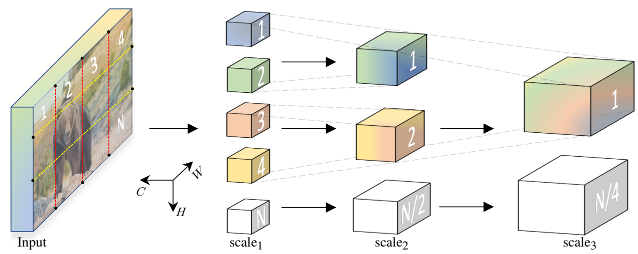

Figure 1. **Multiscale Vision Transformers** learn a hierarchy from *dense* (in space) and *simple* (in channels) to *coarse* and *complex* features. Several resolution-channel *scale* stages progressively *increase* the channel capacity of the intermediate latent sequence while *reducing* its length and thereby spatial resolution.

In a parallel development, the computer vision community developed *multiscale* processing, sometimes called "pyramid" strategies, with Rosenfeld and Thurston [91], Burt and Adelson [10], Koenderink [66], among the key papers. There were two motivations (*i*) To decrease the computing requirements by working at lower resolutions and (*ii*) A better sense of "context" at the lower resolutions, which could then guide the processing at higher resolutions (this is a precursor to the benefit of "depth" in today's neural networks.)

The Transformer [104] architecture allows learning arbitrary functions defined over sets and has been scalably successful in sequence tasks such as language comprehension [29] and machine translation [9]. Fundamentally, a transformer uses blocks with two basic operations. First, is an attention operation [4] for modeling inter-element relations. Second, is a multi-layer perceptron (MLP), which models relations within an element. Intertwining these operations with normalization [2] and residual connections [49] allows transformers to generalize to a wide variety of tasks.

Recently, transformers have been applied to key computer vision tasks such as image classification. In the spirit of architectural universalism, vision transformers [28, 101] approach performance of convolutional models across a variety of data and compute regimes. By only having a first layer that 'patchifies' the input in spirit of a 2D convolution, followed by a stack of transformer blocks, the vision transformer aims to showcase the power of the transformer architecture using little inductive bias.

[*]Equal technical contribution.

In this paper, our intention is to connect the seminal idea of *multiscale feature hierarchies* with the transformer model. We posit that the fundamental vision principle of resolution and channel scaling, can be beneficial for transformer models across a variety of visual recognition tasks.

We present Multiscale Vision Transformers (MViT), a transformer architecture for modeling visual data such as images and videos. Consider an input image as shown in Fig. 1. Unlike conventional transformers, which maintain a constant channel capacity and resolution throughout the network, Multiscale Transformers have several channel-resolution 'scale' stages. Starting from the image resolution and a small channel dimension, the stages *hierarchically expand* the *channel* capacity while *reducing* the *spatial* resolution. This creates a multiscale pyramid of feature activations inside the transformer network, effectively connecting the principles of transformers with multi scale feature hierarchies.

Our conceptual idea provides an effective design advantage for vision transformer models. The early layers of our architecture can operate at high spatial resolution to model *simple* low-level visual information, due to the lightweight channel capacity. In turn, the deeper layers can effectively focus on spatially coarse but *complex* high-level features to model visual semantics. The fundamental advantage of our multiscale transformer arises from the extremely dense nature of visual signals, a phenomenon that is even more pronounced for space-time visual signals captured in *video*.

A noteworthy benefit of our design is the presence of strong implicit temporal bias. We show that vision transformer models [28] trained on natural video suffer no performance decay when tested on videos with *shuffled* frames. This indicates that these models are not effectively using the temporal information and instead rely heavily on appearance. In contrast, when testing our MViT models on shuffled frames, we observe significant accuracy decay, suggesting reliance on temporal information.

Our focus in this paper is video recognition, and we design and evaluate MViT for video tasks (Kinetics [64, 12], Charades [92], SSv2 [43] and AVA [44]). MViT provides a significant performance gain over concurrent video transformers [84, 8, 1], *without* any external pre-training data.

In Fig. A.4 we show the computation/accuracy trade-off for video-level inference, when varying the number of temporal clips used in MViT. The vertical axis shows accuracy on Kinetics-400 and the horizontal axis the overall inference cost in FLOPs for different models, **MViT** and concurrent ViT [28] video variants: VTN [84], TimeSformer [8], ViViT [1]. To achieve similar accuracy level as MViT, these models require significant more computation and parameters (*e.g.* ViViT-L [1] has 6.8× higher FLOPs and 8.5× more parameters at equal accuracy, more analysis in §A.2) and need large-scale external pre-training on ImageNet-**21K** (which contains around 60× more labels than Kinetics-400).

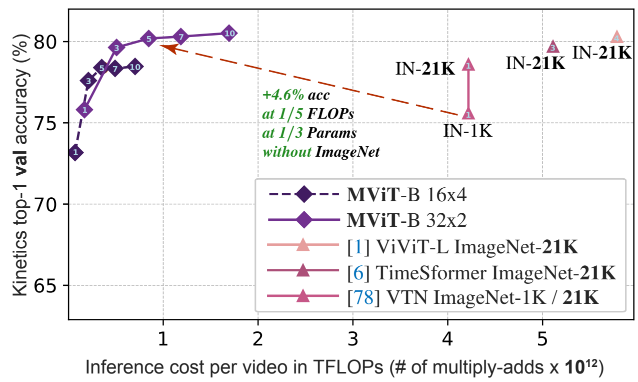

Figure 2. **Accuracy/complexity trade-off** on Kinetics-400 for varying # of inference clips per video shown in MViT curves. Concurrent vision-transformer based methods [84, 8, 1] require over 5× more computation *and large-scale external pre-training* on ImageNet-**21K** (IN-**21K**), to achieve equivalent **MViT** accuracy.

We further apply MViT to ImageNet [24] classification, by *simply removing the temporal dimension* of the video architecture, and show significant gains over single-scale vision transformers for image recognition. Our code and models are available in PySlowFast [31] and PyTorchVideo [32].

2. Related Work

Convolutional networks (ConvNets). Incorporating downsampling, shift invariance, and shared weights, ConvNets are de-facto standard backbones for computer vision tasks for image [70, 67, 94, 96, 51, 15, 18, 39, 99, 87, 46] and video [93, 36, 14, 85, 74, 112, 102, 34, 111, 40, 33, 123, 62].

Self-attention in ConvNets. Self-attention mechanisms has been used for image understanding [88, 120, 57, 7], unsupervised object recognition [80] as well as vision and language [83, 71]. Hybrids of self-attention operations and convolutional networks have also been applied to image understanding [56] and video recognition [107].

Vision Transformers. Much of current enthusiasm in application of Transformers [104] to vision tasks commences with the Vision Transformer (ViT) [28] and Detection Transformer [11]. We build directly upon [28] with a staged model allowing channel expansion and resolution downsampling. DeiT [101] proposes a data efficient approach to training ViT. Our training recipe builds on, and we compare our image classification models to, DeiT under identical settings.

An emerging thread of work aims at applying transformers to vision tasks such as object detection [5], semantic segmentation [121, 105], 3D reconstruction [77], pose estimation [113], generative modeling [17], image retrieval [30], medical image segmentation [16, 103, 117], point clouds [45], video instance segmentation [109], object re-identification [52], video retrieval [38], video dialogue [69], video object detection [116] and multi-modal tasks [78, 26, 86, 58, 114]. A separate line of works attempts at modeling visual data with learnt discretized token sequences [110, 89, 17, 115, 21].

Efficient Transformers. Recent works [106, 65, 20, 100, 23, 19, 72, 6] reduce the quadratic attention complexity to make transformers more efficient for natural language processing applications, which is complementary to our approach.

Three concurrent works propose a ViT-based architecture for video [84, 8, 1]. However, these methods rely on pretraining on vast amount of external data such as ImageNet-21K [24], and thus use the vanilla ViT [28] with minimal adaptations. In contrast, our MViT introduces multiscale feature hierarchies for transformers, allowing effective modeling of dense visual input without large-scale external data.

3. Multiscale Vision Transformer (MViT)

Our generic Multiscale Transformer architecture builds on the core concept of *stages*. Each stage consists of multiple transformer blocks with specific space-time resolution and channel dimension. The main idea of Multiscale Transformers is to progressively *expand* the channel capacity, while *pooling* the resolution from input to output of the network.

3.1. Multi Head Pooling Attention

We first describe Multi Head Pooling Attention (MHPA), a self attention operator that enables flexible resolution modeling in a transformer block allowing Multiscale Transformers to operate at progressively changing spatiotemporal resolution. In contrast to original Multi Head Attention (MHA) operators [104], where the channel dimension and the spatiotemporal resolution remains fixed, MHPA *pools* the sequence of latent tensors to reduce the sequence length (resolution) of the attended input. Fig. 3 shows the concept.

Concretely, consider a D dimensional input tensor X of sequence length L, $X \in \mathbb{R}^{L\times D}$. Following MHA [28], MHPA projects the input X into intermediate query tensor $\hat{Q} \in \mathbb{R}^{L\times D}$, key tensor $\hat{K} \in \mathbb{R}^{L\times D}$ and value tensor $\hat{V} \in \mathbb{R}^{L\times D}$ with linear operations

$$\hat{Q} = XW_Q \quad \hat{K} = XW_K \quad \hat{V} = XW_V$$

/ with weights W_Q, W_K, W_V of dimensions $D \times D$. The obtained intermediate tensors are then pooled in sequence length, with a pooling operator $\mathcal{P}$ as described below.

Pooling Operator. Before attending the input, the intermediate tensors $\hat{Q}, \hat{K}, \hat{V}$ are pooled with the pooling operator $\mathcal{P}(\cdot;\mathbf{\Theta})$ which is the cornerstone of our MHPA and, by extension, of our Multiscale Transformer architecture.

The operator $\mathcal{P}(\cdot;\mathbf{\Theta})$ performs a pooling kernel computation on the input tensor along each of the dimensions. Unpacking $\mathbf{\Theta}$ as $\mathbf{\Theta} := (\mathbf{k}, \mathbf{s}, \mathbf{p})$, the operator employs a pooling kernel $\mathbf{k}$ of dimensions $k_T \times k_H \times k_W$, a stride $\mathbf{s}$ of corresponding dimensions $s_T \times s_H \times s_W$ and a padding $\mathbf{p}$ of corresponding dimensions $p_T \times p_H \times p_W$ to reduce an

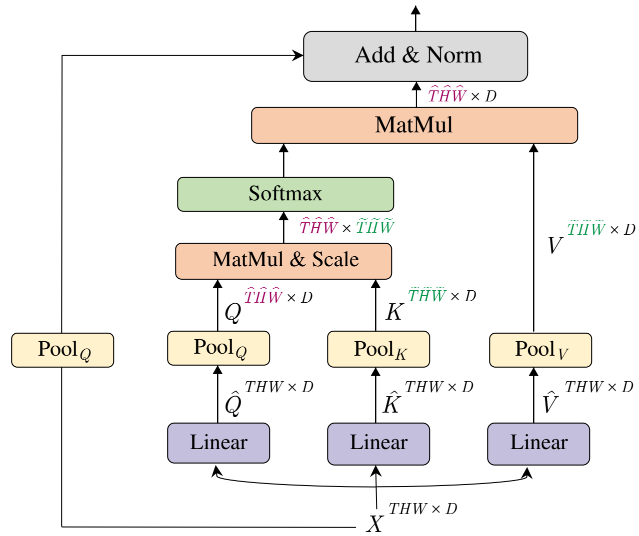

Figure 3. **Pooling Attention** is a flexible attention mechanism that (i) allows obtaining the reduced space-time resolution ($\hat{T}\hat{H}\hat{W}$) of the input (THW) by pooling the query, $Q = \mathcal{P}(\hat{Q};\mathbf{\Theta}_Q)$, and/or (ii) computes attention on a reduced length ($\tilde{T}\tilde{H}\tilde{W}$) by pooling the key, $K = \mathcal{P}(\hat{K};\mathbf{\Theta}_K)$, and value, $V = \mathcal{P}(\hat{V};\mathbf{\Theta}_V)$, sequences.

input tensor of dimensions $\mathbf{L} = T \times H \times W$ to $\tilde{\mathbf{L}}$ given by,

$$\tilde{\mathbf{L}} = \left\lfloor \frac{\mathbf{L} + 2\mathbf{p} - \mathbf{k}}{\mathbf{s}} \right\rfloor + 1$$

with the equation applying coordinate-wise. The pooled tensor is flattened again yielding the output of $\mathcal{P}(Y;\mathbf{\Theta}) \in \mathbb{R}^{\tilde{L}\times D}$ with reduced sequence length, $\tilde{L} = \tilde{T} \times \tilde{H} \times \tilde{W}$.

By default we use *overlapping* kernels $\mathbf{k}$ with *shape-preserving* padding $\mathbf{p}$ in our pooling attention operators, so that $\tilde{L}$, the sequence length of the output tensor $\mathcal{P}(Y;\mathbf{\Theta})$, experiences an overall reduction by a factor of $s_T s_H s_W$.

Pooling Attention. The pooling operator $P(\cdot;\Theta)$ is applied to all the intermediate tensors $\hat{Q}$, $\hat{K}$ and $\hat{V}$ independently with chosen pooling kernels $\mathbf{k}$, stride $\mathbf{s}$ and padding $\mathbf{p}$. Denoting θ yielding the pre-attention vectors $Q = \mathcal{P}(\hat{Q};\mathbf{\Theta}_Q)$, $K = \mathcal{P}(\hat{K};\mathbf{\Theta}_K)$ and $V = \mathcal{P}(\hat{V};\mathbf{\Theta}_V)$ with reduced sequence lengths. Attention is now computed on these shortened vectors, with the operation,

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^T/\sqrt{D})V.$$

Naturally, the operation induces the constraints $\mathbf{s}_K \equiv \mathbf{s}_V$ on the pooling operators. In summary, pooling attention is computed as,

$$\text{PA}(\cdot) = \text{Softmax}(\mathcal{P}(Q;\mathbf{\Theta}_Q)\mathcal{P}(K;\mathbf{\Theta}_K)^T/\sqrt{d})\mathcal{P}(V;\mathbf{\Theta}_V),$$

where $\sqrt{d}$ is normalizing the inner product matrix row-wise. The output of the Pooling attention operation thus has its sequence length reduced by a *stride* factor of $s^Q_T s^Q_H s^Q_W$ following the shortening of the query vector Q in $\mathcal{P}(\cdot)$.

stage	operators	output sizes
data layer	stride $\tau{\times}1{\times}1$	$T{\times}H{\times}W$
patch_1	$1{\times}16{\times}16$, D stride $1{\times}16{\times}16$	$D{\times}T{\times}\frac{H}{16}{\times}\frac{W}{16}$
scale_2	$\begin{bmatrix}\text{MHA}(D)\\ \text{MLP}(4D)\end{bmatrix}{\times}N$	$D{\times}T{\times}\frac{H}{16}{\times}\frac{W}{16}$

Table 1. **Vision Transformers (ViT)** base model starts from a data layer that samples visual input with rate $\tau{\times}1{\times}1$ to $T{\times}H{\times}W$ resolution, where T is the number of frames H height and W width. The first layer, patch_1 projects patches (of shape $1{\times}16{\times}16$) to form a sequence, processed by a stack of N transformer blocks (stage_2) at uniform channel dimension (D) and resolution ($T{\times}\frac{H}{16}{\times}\frac{W}{16}$).

Multiple heads. As in [104] the computation can be parallelized by considering h heads where each head is performing the pooling attention on a non overlapping subset of D/h channels of the D dimensional input tensor X.

Computational Analysis. Since attention computation scales quadratically w.r.t. the sequence length, pooling the key, query and value tensors has dramatic benefits on the fundamental compute and memory requirements of the Multiscale Transformer model. Denoting the sequence length reduction factors by f_Q, f_K and f_V we have,

$$f_j = s_T^j \cdot s_H^j \cdot s_W^j,\ \forall\ j \in \{Q, K, V\}.$$

Considering the input tensor to $\mathcal{P}(;\Theta)$ to have dimensions $D \times T \times H \times W$, the run-time complexity of MHPA is $O(THWD/h(D + THW/f_Q f_K))$ per head and the memory complexity is $O(THWh(D/h + THW/f_Q f_K))$.

This trade-off between the number of channels D and sequence length term $THW/f_Q f_K$ informs our design choices about architectural parameters such as number of heads and width of layers. We refer the reader to §C for a detailed analysis and discussions on the runtime-memory complexity trade-off.

3.2. Multiscale Transformer Networks

Building upon Multi Head Pooling Attention (Sec. 3.1), we describe the Multiscale Transformer model for visual representation learning using exclusively MHPA and MLP layers. First, we present a brief review of the Vision Transformer Model that informs our design.

Preliminaries: Vision Transformer (ViT). The Vision Transformer (ViT) architecture [28] starts by dicing the input video of resolution $T{\times}H{\times}W$, where T is the number of frames H the height and W the width, into non-overlapping patches of size $1{\times}16{\times}16$ each, followed by point-wise application of linear layer on the flattened image patches to to project them into the latent dimension, D, of the transformer. This is *equivalent* to a convolution with equal kernel size and stride of $1{\times}16{\times}16$ and is shown as patch_1 stage in the model definition in Table 1.

Next, a positional embedding $\mathbf{E} \in \mathbb{R}^{L \times D}$ is added to each element of the projected sequence of length L with dimension D to encode the positional information and break permutation invariance. A learnable class embedding is appended to the projected image patches.

The resulting sequence of length of $L + 1$ is then processed sequentially by a stack of N transformer blocks, each one performing attention (MHA [104]), multi-layer perceptron (MLP) and layer normalization (LN [3]) operations. Considering X to be the input of the block, the output of a single transformer block, $\text{Block}(X)$ is computed by

$$\begin{aligned} X_1 &= \text{MHA}(\text{LN}(X)) + X \\ \text{Block}(X) &= \text{MLP}(\text{LN}(X_1)) + X_1. \end{aligned}$$

The resulting sequence after N consecutive blocks is layer-normalized and the class embedding is extracted and passed through a linear layer to predict the desired output (*e.g.* class). By default, the hidden dimension of the MLP is $4D$. We refer the reader to [28, 104] for details.

In context of the present paper, it is noteworthy that ViT maintains a constant channel capacity and spatial resolution throughout all the blocks (see Table 1).

stages	operators	output sizes
data layer	stride $\tau{\times}1{\times}1$	$D{\times}T{\times}H{\times}W$
cube_1	$c_T{\times}c_H{\times}c_W$, D stride $s_T{\times}4{\times}4$	$D{\times}\frac{T}{s_T}{\times}\frac{H}{4}{\times}\frac{W}{4}$
scale_2	$\begin{bmatrix}\text{MHPA}(D)\\ \text{MLP}(4D)\end{bmatrix}{\times}N_2$	$D{\times}\frac{T}{s_T}{\times}\frac{H}{4}{\times}\frac{W}{4}$
scale_3	$\begin{bmatrix}\text{MHPA}(2D)\\ \text{MLP}(8D)\end{bmatrix}{\times}N_3$	$2D{\times}\frac{T}{s_T}{\times}\frac{H}{8}{\times}\frac{W}{8}$
scale_4	$\begin{bmatrix}\text{MHPA}(4D)\\ \text{MLP}(16D)\end{bmatrix}{\times}N_4$	$4D{\times}\frac{T}{s_T}{\times}\frac{H}{16}{\times}\frac{W}{16}$
scale_5	$\begin{bmatrix}\text{MHPA}(8D)\\ \text{MLP}(32D)\end{bmatrix}{\times}N_5$	$8D{\times}\frac{T}{s_T}{\times}\frac{H}{32}{\times}\frac{W}{32}$

Table 2. **Multiscale Vision Transformers (MViT)** base model. Layer cube_1, projects *dense* space-time cubes (of shape $c_t{\times}c_y{\times}c_w$) to D channels to reduce spatiotemporal resolution to $\frac{T}{s_T}{\times}\frac{H}{4}{\times}\frac{W}{4}$. The subsequent stages progressively down-sample this resolution (at beginning of a stage) with MHPA while simultaneously increasing the channel dimension, in MLP layers, (at the end of a stage). Each stage consists of N_* transformer blocks, denoted in [brackets].

Multiscale Vision Transformers (MViT). Our key concept is to progressively *grow* the *channel* resolution (*i.e.* dimension), while simultaneously *reducing* the *spatiotemporal* resolution (*i.e.* sequence length) throughout the network. By design, our MViT architecture has *fine* spacetime (and *coarse* channel) resolution in early layers that is up-/downsampled to a coarse spacetime (and *fine* channel) resolution in late layers. MViT is shown in Table 2.

Scale stages. A *scale stage* is defined as a set of N transformer blocks that operate on the same *scale* with identical resolution across channels and space-time dimensions $D{\times}T{\times}H{\times}W$. At the input ($\text{cube}_1$ in Table 2), we project the patches (or cubes if they have a temporal extent) to a smaller channel dimension (*e.g.* $8\times$ smaller than a typical ViT model), but long sequence (*e.g.* $4{\times}4 = 16\times$ denser than a typical ViT model; *cf.* Table 1).

stage	operators	output sizes
data	stride $8\times1\times1$	$8\times224\times224$
patch$_1$	$1\times16\times16$, 768 stride $1\times16\times16$	$768\times8\times14\times14$
scale$_2$	$\begin{bmatrix}\text{MHA}(768)\\ \text{MLP}(3072)\end{bmatrix}\times12$	$768\times8\times14\times14$

(a) **ViT**-B with **179.6**G FLOPs, **87.2**M param, **16.8**G memory, and **68.5**% top-1 accuracy.

stage	operators	output sizes
data	stride $4\times1\times1$	$16\times224\times224$
cube$_1$	$3\times7\times7$, 96 stride $2\times4\times4$	$96\times8\times56\times56$
scale$_2$	$\begin{bmatrix}\text{MHPA}(96)\\ \text{MLP}(384)\end{bmatrix}\times1$	$96\times8\times56\times56$
scale$_3$	$\begin{bmatrix}\text{MHPA}(192)\\ \text{MLP}(768)\end{bmatrix}\times2$	$192\times8\times28\times28$
scale$_4$	$\begin{bmatrix}\text{MHPA}(384)\\ \text{MLP}(1536)\end{bmatrix}\times11$	$384\times8\times14\times14$
scale$_5$	$\begin{bmatrix}\text{MHPA}(768)\\ \text{MLP}(3072)\end{bmatrix}\times2$	$768\times8\times7\times7$

(b) **MViT**-B with **70.5**G FLOPs, **36.5**M param, **6.8**G memory, and **77.2**% top-1 accuracy.

stage	operators	output sizes
data	stride $4\times1\times1$	$16\times224\times224$
cube$_1$	$3\times8\times8$, 128 stride $2\times8\times8$	$128\times8\times28\times28$
scale$_2$	$\begin{bmatrix}\text{MHPA}(128)\\ \text{MLP}(512)\end{bmatrix}\times3$	$128\times8\times28\times28$
scale$_3$	$\begin{bmatrix}\text{MHPA}(256)\\ \text{MLP}(1024)\end{bmatrix}\times7$	$256\times8\times14\times14$
scale$_4$	$\begin{bmatrix}\text{MHPA}(512)\\ \text{MLP}(2048)\end{bmatrix}\times6$	$512\times8\times7\times7$

(c) **MViT**-S with **32.9**G FLOPs, **26.1**M param, **4.3**G memory, and **74.3**% top-1 accuracy.

Table 3. Comparing **ViT**-B to two instantiations of **MViT** with varying complexity, **MViT**-S in (c) and **MViT**-B in (b). **MViT**-S operates at a lower spatial resolution and lacks a first high-resolution stage. The top-1 accuracy corresponds to *5-Center* view testing on K400. FLOPs correspond to a single inference clip, and memory is for a training batch of 4 clips. See Table 2 for the general **MViT**-B structure.

At a stage *transition* (*e.g*. scale$_1$ to scale$_2$ to in Table 2), the channel dimension of the processed sequence is up-sampled while the length of the sequence is down-sampled. This effectively reduces the spatiotemporal resolution of the underlying visual data while allowing the network to assimilate the processed information in more complex features.

Channel expansion. When transitioning from one stage to the next, we expand the channel dimension by increasing the output of the final MLP layer in the previous stage by a factor that is relative to the resolution change introduced at the stage. Concretely, if we down-sample the space-time resolution by $4\times$, we increase the channel dimension by $2\times$. For example, scale$_3$ to scale$_4$ changes resolution from $2D\times\frac{T}{s_T}\times\frac{H}{8}\times\frac{T}{8}$ to $4D\times\frac{T}{s_T}\times\frac{H}{16}\times\frac{T}{16}$ in Table 2. This roughly preserves the computational complexity across stages, and is similar to ConvNet design principles [93, 50].

Query pooling. The pooling attention operation affords flexibility not only in the length of key and value vectors but also in the length of the query, and thereby output, sequence. Pooling the query vector $\mathcal{P}(Q;\mathbf{k};\mathbf{p};\mathbf{s})$ with a kernel $\mathbf{s}\equiv(s^Q_T, s^Q_H, s^Q_W)$ leads to sequence reduction by a factor of $s^Q_T\cdot s^Q_H\cdot s^Q_W$. Since, our intention is to decrease resolution at the beginning of a stage and then preserve this resolution throughout the stage, only the first pooling attention operator of each stage operates at non-degenerate query stride $\mathbf{s}^Q>1$, with all other operators constrained to $\mathbf{s}^Q\equiv(1,1,1)$.

Key-Value pooling. Unlike Query pooling, changing the sequence length of key K and value V tensors, does not change the output sequence length and, hence, the space-time resolution. However, they play a key role in overall computational requirements of the pooling attention operator.

We decouple the usage of K, V and Q pooling, with Q pooling being used in the first layer of each stage and K, V pooling being employed in all other layers. Since the sequence length of key and value tensors need to be identical to allow attention weight calculation, the pooling stride used on K and value V tensors needs to be identical. In our default setting, we constrain *all* pooling parameters $(\mathbf{k};\mathbf{p};\mathbf{s})$ to be identical *i.e.* $\Theta_K\equiv\Theta_V$ within a stage, but vary $\mathbf{s}$ *adaptively* w.r.t. to the scale across stages.

Skip connections. Since the channel dimension and sequence length change inside a residual block, we pool the skip connection to adapt to the dimension mismatch between its two ends. MHPA handles this mismatch by adding the query pooling operator $\mathcal{P}(\cdot;\mathbf{\Theta}_Q)$ to the residual path. As shown in Fig. 3, instead of directly adding the input X of MHPA to the output, we add the pooled input X to the output, thereby matching the resolution to attended query Q.

For handling the channel dimension mismatch between stage changes, we employ an extra linear layer that operates on the layer-normalized output of our MHPA operation. Note that this differs from the other (resolution-preserving) skip-connections that operate on the un-normalized signal.

3.3. Network instantiation details

Table 3 shows concrete instantiations of the base models for Vision Transformers [28] and our Multiscale Vision Transformers. ViT-Base [28] (Table 3b) initially projects the input to patches of shape $1\times16\times16$ with dimension $D=768$, followed by stacking $N=12$ transformer blocks. With an $8\times224\times224$ input the resolution is fixed to $768\times8\times14\times14$ throughout *all* layers. The sequence length (spacetime resolution + class token) is $8\cdot14\cdot14+1=1569$.

Our MViT-Base (Table 3b) is comprised of 4 scale stages, each having several transformer blocks of consistent channel dimension. **MViT**-B initially projects the input to a channel dimension of $D=96$ with *overlapping* space-time cubes of shape $3\times7\times7$. The resulting sequence of length $8*56*56+1=25089$ is reduced by a factor of 4 for each additional stage, to a final sequence length of $8*7*7+1=393$ at scale$_4$. In tandem, the channel dimension is up-sampled by a factor of 2 at each stage, increasing to 768 at scale$_4$. Note that all pooling operations, and hence the resolution down-sampling, is performed only on the data sequence without involving the processed class token embedding.

We set the number of MHPA heads to $h=1$ in the scale$_1$ stage and increase the number of heads with the channel dimension (channels per-head D/h remain consistent at 96).

At each stage transition, the previous stage output MLP dimension is increased by $2\times$ and MHPA pools on Q tensors with $\mathbf{s}^Q=(1,2,2)$ at the input of the next stage.

model	pre-train	top-1	top-5	FLOPs×views	Param
Two-Stream I3D [14]	-	71.6	90.0	216 × NA	25.0
ip-CSN-152 [102]	-	77.8	92.8	109×3×10	32.8
SlowFast 8×8 +NL [34]	-	78.7	93.5	116×3×10	59.9
SlowFast 16×8 +NL [34]	-	79.8	93.9	234×3×10	59.9
X3D-M [33]	-	76.0	92.3	6.2×3×10	3.8
X3D-XL [33]	-	79.1	93.9	48.4×3×10	11.0
ViT-B-VTN [84]	ImageNet-1K	75.6	92.4	4218×1×1	114.0
ViT-B-VTN [84]	ImageNet-**21K**	78.6	93.7	4218×1×1	114.0
ViT-B-TimeSformer [8]	ImageNet-**21K**	80.7	94.7	2380×3×1	121.4
ViT-L-ViViT [1]	ImageNet-**21K**	81.3	94.7	3992×3×4	310.8
ViT-B (our baseline)	ImageNet-**21K**	79.3	93.9	180×1×5	87.2
ViT-B (our baseline)	-	68.5	86.9	180×1×5	87.2
MViT-S	-	76.0	92.1	32.9×1×5	26.1
MViT-B, 16×4	-	78.4	93.5	70.5×1×5	36.6
MViT-B, 32×3	-	80.2	94.4	170×1×5	36.6
MViT-B, 64×3	-	**81.2**	**95.1**	455×3×3	36.6

Table 4. **Comparison with previous work on Kinetics-400**. We report the inference cost with a single "view" (temporal clip with spatial crop) × the number of views (FLOPs×view$_{\text{space}}$×view$_{\text{time}}$). Magnitudes are Giga (10^9) for FLOPs and Mega (10^6) for Param. Accuracy of models trained with external data is de-emphasized.

We employ K, V pooling in all MHPA blocks, with $\Theta_K \equiv \Theta_V$ and $\mathbf{s}^{K,V} = (1, 8, 8)$ in scale$_1$ and *adaptively* decay this stride w.r.t. to the scale across stages such that the K, V tensors have consistent scale across all blocks.

4. Experiments: Video Recognition

Datasets. We use Kinetics-400 [64] (K400) (~240k training videos in 400 classes) and Kinetics-600 [12]. We further assess transfer learning performance for on Something-Something-v2 [43], Charades [92], and AVA [44].

We report top-1 and top-5 classification accuracy (%) on the validation set, computational cost (in FLOPs) of a single, spatially center-cropped clip and the number of clips used.

Training. By default, all models are trained *from random initialization* ("*from scratch*") on Kinetics, *without* using ImageNet [25] or other pre-training. Our training recipe and augmentations follow [34, 101]. For Kinetics, we train for 200 epochs with 2 repeated augmentation [55] repetitions.

We report ViT baselines that are *fine-tuned* from ImageNet, using a 30-epoch version of the training recipe in [34].

For the temporal domain, we sample a clip from the full-length video, and the input to the network are T frames with a temporal stride of τ; denoted as $T \times \tau$ [34].

Inference. We apply two testing strategies following [34, 33]: *(i)* Temporally, uniformly samples K clips (*e.g.* K=5) from a video, scales the shorter spatial side to 256 pixels and takes a 224×224 center crop, and *(ii)*, the same as *(i)* temporally, but take 3 crops of 224×224 to cover the longer spatial axis. We average the scores for all individual predictions.

All implementation specifics are in §D.

4.1. Main Results

Kinetics-400. Table 4 compares to prior work. From top-to-bottom, it has four sections and we discuss them in turn.

model	pretrain	top-1	top-5	GFLOPs×views	Param
SlowFast 16×8 +NL [34]	-	81.8	95.1	234×3×10	59.9
X3D-M	-	78.8	94.5	6.2×3×10	3.8
X3D-XL	-	81.9	95.5	48.4×3×10	11.0
ViT-B-TimeSformer [8]	IN-**21K**	82.4	96.0	1703×3×1	121.4
ViT-L-ViViT [1]	IN-**21K**	83.0	95.7	3992×3×4	310.8
MViT-B, 16×4	-	82.1	95.7	70.5×1×5	36.8
MViT-B, 32×3	-	83.4	96.3	170×1×5	36.8
MViT-B-24, 32×3	-	**84.1**	**96.5**	236×1×5	52.9

Table 5. **Comparison with previous work on Kinetics-600**.

The first Table 4 section shows prior art using ConvNets.

The second section shows concurrent work using Vision Transformers [28] for video classification [84, 8]. Both approaches rely on ImageNet pre-trained base models. ViT-B-VTN [84] achieves 75.6% top-1 accuracy, which is boosted by 3% to 78.6% by merely changing the pre-training from ImageNet-1K to ImageNet-21K. ViT-B-TimeSformer [8] shows another 2.1% gain on top of VTN, at higher cost of 7140G FLOPs and 121.4M parameters. ViViT improves accuracy further with an even larger ViT-L model.

The third section in Table 4 shows our ViT baselines. We first list our ViT-B, also pre-trained on the ImageNet-21K, which achieves 79.3%, thereby being 1.4% lower than ViT-B-TimeSformer, but is with 4.4× fewer FLOPs and 1.4× fewer parameters. This result shows that *simply fine-tuning an off-the-shelf ViT-B model from ImageNet-**21K*** [28] provides a strong baseline on Kinetics. However, training this model from-scratch with the same fine-tuning recipe will result in 34.3%. Using our "training-from-scratch" recipe will produce 68.5% for this ViT-B model, using the same 1×5, spatial × temporal, views for video-level inference.

The final section of Table 4 lists our **MViT** results. All our models are *trained-from-scratch* using this recipe, *without* any external pre-training. Our small model, **MViT**-S produces 76.0% while being relatively lightweight with 26.1M param and 32.9×5=164.5G FLOPs, outperforming ViT-B by **+7.5**% at **5.5**× less compute in *identical* train/val setting.

Our base model, **MViT**-B provides 78.4%, a **+9.9**% accuracy boost over ViT-B under *identical settings*, while having 2.6×/2.4×fewer FLOPs/parameters. When changing the frame sampling from 16×4 to 32×3 performance increases to 80.2%. Finally, we take this model and fine-tune it for 5 epochs with longer 64 frame input, after interpolating the temporal positional embedding, to reach **81.2**% top-1 using 3 spatial and 3 temporal views for inference (it is sufficient test with fewer temporal views if a clip has more frames). Further quantitative and qualitative results are in §A.

Kinetics-600 [12] is a larger version of Kinetics. Results are in Table 5. We train MViT from-scratch, without any pre-training. **MViT**-B, 16×4 achieves 82.1% top-1 accuracy. We further train a deeper 24-layer model with longer sampling, **MViT**-B-24, 32×3, to investigate model scale on this larger dataset. MViT achieves state-of-the-art of 83.4% with 5-clip center crop testing while having 56.0× fewer FLOPs and 8.4× fewer parameters than ViT-L-ViViT [1] which relies on large-scale ImageNet-21K pre-training.

model	pretrain	top-1	top-5	FLOPs×views	Param
TSM-RGB [76]	IN-1K+K400	63.3	88.2	62.4×3×2	42.9
MSNet [68]	IN-1K	64.7	89.4	67×1×1	24.6
TEA [73]	IN-1K	65.1	89.9	70×3×10	-
ViT-B-TimeSformer [8]	IN-**21K**	62.5	-	1703×3×1	121.4
ViT-B (our baseline)	IN-**21K**	63.5	88.3	180×3×1	87.2
SlowFast R50, 8×8 [34]	K400	61.9	87.0	65.7×3×1	34.1
SlowFast R101, 8×8 [34]		63.1	87.6	106×3×1	53.3
MViT-B, 16×4		64.7	89.2	70.5×3×1	36.6
MViT-B, 32×3		67.1	90.8	170×3×1	36.6
MViT-B, 64×3		**67.7**	**90.9**	455×3×1	36.6
MViT-B, 16×4	K600	66.2	90.2	70.5×3×1	36.6
MViT-B, 32×3		67.8	91.3	170×3×1	36.6
MViT-B-24, 32×3		**68.7**	**91.5**	236×3×1	53.2

Table 6. **Comparison with previous work on SSv2.**

Something-Something-v2 (SSv2) [43] is a dataset with videos containing object interactions, which is known as a 'temporal modeling' task. Table 6 compares our method with the state-of-the-art. We first report a simple ViT-B (our baseline) that uses ImageNet-21K pre-training. Our **MViT**-B with 16 frames has 64.7% top-1 accuracy, which is better than the SlowFast R101 [34] which shares the same setting (K400 pre-training and 3×1 view testing). With more input frames, our **MViT**-B achieves 67.7% and the deeper **MViT**-B-24 achieves 68.7% using our K600 pre-trained model of above. In general, Table 6 verifies the capability of temporal modeling for MViT.

model	pretrain	mAP	FLOPs×views	Param
Nonlocal [107]	IN-1K+K400	37.5	544×3×10	54.3
STRG +NL [108]		39.7	630×3×10	58.3
Timeception [61]	K400	41.1	N/A×N/A	N/A
LFB +NL [111]		42.5	529×3×10	122
SlowFast 50, 8×8 [34]		38.0	65.7×3×10	34.0
SlowFast 101+NL, 16×8 [34]		42.5	234×3×10	59.9
X3D-XL [33]		43.4	48.4×3×10	11.0
MViT-B, 16×4		40.0	70.5×3×10	36.4
MViT-B, 32×3		44.3	170×3×10	36.4
MViT-B, 64×3		**46.3**	455×3×10	36.4
SlowFast R101+NL, 16×8 [34]	K600	45.2	234×3×10	59.9
X3D-XL [33]		47.1	48.4×3×10	11.0
MViT-B, 16×4		43.9	70.5×3×10	36.4
MViT-B, 32×3		47.1	170×3×10	36.4
MViT-B-24, 32×3		**47.7**	236×3×10	53.0

Table 7. **Comparison with previous work on Charades.**

Charades [92] is a dataset with longer range activities. We validate our model in Table 7. With similar FLOPs and parameters, our **MViT**-B 16×4 achieves better results (+2.0 mAP) than SlowFast R50 [34]. As shown in the Table, the performance of **MViT**-B is further improved by increasing the number of input frames and **MViT**-B layers and using K600 pre-trained models.

AVA [44] is a dataset with for spatiotemporal-localization of human actions. We validate our MViT on this detection task. Details about the detection architecture of MViT can be found in §D.2. Table 8 shows the results of our MViT models compared with SlowFast [34] and X3D [33]. We observe that MViT-B can be competitive to SlowFast and X3D using the same pre-training and testing strategy.

model	pretrain	val mAP	FLOPs	Param
SlowFast, 4×16, R50 [34]	K400	21.9	52.6	33.7
SlowFast, 8×8, R101 [34]		23.8	138	53.0
MViT-B, 16×4		24.5	70.5	36.4
MViT-B, 32×3		26.8	170	36.4
MViT-B, 64×3		**27.3**	455	36.4
SlowFast, 8×8 R101+NL [34]	K600	27.1	147	59.2
SlowFast, 16×8 R101+NL [34]		27.5	296	59.2
X3D-XL [33]		27.4	48.4	11.0
MViT-B, 16×4		26.1	70.5	36.3
MViT-B, 32×3		27.5	170	36.4
MViT-B-24, 32×3		**28.7**	236	52.9

Table 8. **Comparison with previvous work on AVA v2.2.** All methods use *single center crop* inference following [33].

4.2. Ablations on Kinetics

We carry out ablations on Kinetics-400 (K400) using 5-clip center 224×224 crop testing. We show top-1 accuracy (Acc), as well as computational complexity measured in GFLOPs for a single clip input of spatial size 224^2. Inference computational cost is proportional as a fixed number of 5 clips is used (to roughly cover the inferred videos with $T\times\tau$=16×4 sampling.) We also report Parameters in M(10^6) and training GPU memory in G(10^9) for a batch size of 4. By default all MViT ablations are with **MViT**-B, $T\times\tau$=16×4 and max-pooling in MHSA.

model	shuffling	FLOPs (G)	Param (M)	Acc
MViT-B		70.5	36.5	**77.2**
MViT-B	✓			70.1 (**−7.1**)
ViT-B		179.6	87.2	68.5
ViT-B	✓			68.4 (**−0.1**)

Table 9. **Shuffling frames in inference. MViT**-B severely drops (−7.1%) for shuffled temporal input, but ViT-B models appear to *ignore* temporal information as accuracy remains similar (−0.1%).

Frame shuffling. Table 9 shows results for randomly shuffling the input frames in time during testing. All models are trained without any shuffling and have temporal embeddings. We notice that our **MViT**-B architecture suffers a significant accuracy drop of **-7.1**% (77.2 → 70.1) for shuffling inference frames. By contrast, ViT-B is surprisingly robust for shuffling the temporal order of the input.

kernel $\mathbf{k}$	pooling func	Param	Acc
$\mathbf{s}$	max	36.5	76.1
$2\mathbf{s}+1$	max	36.5	75.5
$\mathbf{s}+1$	max	36.5	77.2
$\mathbf{s}+1$	average	36.5	75.4
$\mathbf{s}+1$	conv	36.6	78.3
3×3×3	conv	36.6	**78.4**

Table 10. **Pooling function**: Varying the kernel $\mathbf{k}$ as a function of stride $\mathbf{s}$. Functions are average or max pooling and conv which is a learnable, channel-wise convolution.

Pooling function. The ablation in Table 10 looks at the kernel size $\mathbf{k}$ w.r.t. the stride $\mathbf{s}$, and the pooling function (max/average/conv). First, we see that having equivalent kernel and stride $\mathbf{k}$=$\mathbf{s}$ provides 76.1%, increasing the kernel size to $\mathbf{k}{=}2\mathbf{s}{+}1$ decays to 75.5%, but using a kernel $\mathbf{k}{=}\mathbf{s}{+}1$ gives a clear benefit of 77.2%. This indicates that *overlapping pooling is effective*, but a too large overlap ($2\mathbf{s}{+}1$) hurts. Second, we investigate average instead of max-pooling and observe that accuracy decays by from 77.2% to 75.4%.

Third, we use conv-pooling by a learnable, channelwise convolution followed by LN. This variant has +1.2% over max pooling and is used for all experiments in §4.1 and §5.

model	clips/sec	Acc	FLOPs×views	Param
X3D-M [33]	7.9	74.1	4.7×1×5	3.8
SlowFast R50 [34]	5.2	75.7	65.7×1×5	34.6
SlowFast R101 [34]	3.2	77.6	125.9×1×5	62.8
ViT-B [28]	3.6	68.5	179.6×1×5	87.2
MViT-S, max-pool	**12.3**	74.3	32.9×1×5	26.1
MViT-B, max-pool	6.3	77.2	70.5×1×5	36.5
MViT-S, conv-pool	9.4	76.0	32.9×1×5	26.1
MViT-B, conv-pool	4.8	78.4	70.5×1×5	36.6

Table 11. **Speed-Accuracy tradeoff on Kinetics-400.** Training throughput is measured in clips/s. MViT is *fast* and *accurate*.

Speed-Accuracy tradeoff. In Table 11, we analyze the speed/accuracy trade-off of our MViT models, along with their counterparts vision transformer (ViT [28]) and ConvNets (SlowFast 8×8 R50, SlowFast 8×8 R101 [34], & X3D-L [33]). We measure training throughput as the number of video clips per second on a single M40 GPU.

We observe that both MViT-S and MViT-B models are not only significantly more accurate but also much faster than both the ViT-B baseline and convolutional models. Concretely, MViT-S has **3.4**× higher throughput speed (clips/s), is **+5.8**% more accurate (Acc), and has **3.3**× fewer parameters (Param) than ViT-B. Using a conv instead of max-pooling in MHSA, we observe a training speed reduction of ~20% for convolution and additional parameter updates.

5. Experiments: Image Recognition

We apply our video models on static image recognition by using them with single frame, $T = 1$, on ImageNet-1K [25].

Training. Our recipe is identical to DeiT [101] and summarized in §D.5. Training is for 300 epochs and results improve for training longer [101].

5.1. Main Results

For this experiment, we take our models which were designed by ablation studies for video classification on Kinetics and *simply remove the temporal dimension*. Then we train and validate them ("from scratch") on ImageNet.

Table 12 shows the comparison with previous work. From top to bottom, the table contains RegNet [87] and EfficientNet [99] as ConvNet examples, and DeiT [101], with DeiT-B being identical to ViT-B [28] but trained with the improved recipe in [101]. Therefore, this is the vision transformer counterpart we are interested in comparing to.

model	Acc	FLOPs (G)	Param (M)
RegNetZ-4GF [27]	83.1	4.0	28.1
RegNetZ-16GF [27]	84.1	15.9	95.3
EfficientNet-B7 [99]	84.3	37.0	66.0
DeiT-S [101]	79.8	4.6	22.1
DeiT-B [101]	81.8	17.6	86.6
DeiT-B $\uparrow 384^2$ [101]	83.1	55.5	87.0
Swin-B (concurrent) [79]	83.3	15.4	88.0
Swin-B $\uparrow 384^2$ (concurrent) [79]	84.2	47.0	88.0
MViT-B-16, max-pool	82.5	7.8	37.0
MViT-B-16	83.0	7.8	37.0
MViT-B-24	84.0	14.7	72.7
MViT-B-24-320^2	**84.8**	32.7	72.9

Table 12. **Comparison to prior work on ImageNet**. RegNet and EfficientNet are ConvNet examples that use different training recipes. DeiT/MViT are ViT-based and use identical recipes [101].

The bottom section in Table 12 shows results for our Multiscale Vision Transformer (MViT) models.

We show models of different depth, **MViT**-B-Depth, (16 and 24 layers), where **MViT**-B-16 is our base model and the deeper variant is simply created by repeating the number of blocks N_* in each scale stage (*cf.* Table 3b) and using a larger channel dimension of $D = 112$. All our models are trained using the identical 300-epoch recipe as DeiT [101], except repeated augmentation which we found not beneficial.

We make the following observations:

(i) Our lightweight **MViT**-B-16 achieves 82.5% top-1 accuracy, with only 7.8 GFLOPs, which outperforms the DeiT-B counterpart by +0.7% with lower computation cost (2.3×fewer FLOPs and Parameters). If we use conv instead of max-pooling, this number is increased by +0.5% to 83.0%.

(ii) Our deeper model **MViT**-B-24, provides a gain of +1.0% accuracy at slight increase in computation.

(iii) A larger model, **MViT**-B-24-320^2 with input resolution 320^2 reaches 84.8%, corresponding to a +1.7% gain, at 1.7×fewer FLOPs, over DeiT-B$\uparrow 384^2$. These results suggest that Multiscale Vision Transformers have an architectural advantage over Vision Transformers.

Finally, compared to the best model of concurrent Swin [79] (which was designed for image recognition), MViT has +0.6% better accuracy at 1.4×less computation.

6. Conclusion

We have presented Multiscale Vision Transformers that aim to connect the fundamental concept of multiscale feature hierarchies with the transformer model. MViT hierarchically expands the feature complexity while reducing visual resolution. In empirical evaluation, MViT shows a fundamental advantage over single-scale vision transformers for video and image recognition. We hope that our approach will foster further research in visual recognition.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 2, 3, 6, 14

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1, 17

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 4

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1

[5] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection. *arXiv preprint arXiv:2012.09958*, 2020. 2

[6] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *International Conference on Learning Representations*, 2021. 3

[7] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. *International Conference on Computer Vision*, 2019. 2

[8] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 2, 3, 6, 7, 14

[9] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[10] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987. 1

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. ECCV*, pages 213–229. Springer, 2020. 2

[12] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv:1808.01340*, 2018. 2, 6

[13] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 13

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

[15] Chun-Fu Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogerio Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *arXiv preprint arXiv:1807.03848*, 2018. 2

[16] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021. 2

[17] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proc. ICML*, pages 1691–1703. PMLR, 2020. 2

[18] Yunpeng Chen, Haoqi Fang, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*, 2019. 2

[19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 3

[20] 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. *arXiv preprint arXiv:2009.14794*, 2020. 3

[21] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *arXiv preprint arXiv:2102.10882*, 2021. 2

[22] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proc. CVPR*, 2020. 17, 19

[23] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*, 2020. 3

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255. Ieee, 2009. 2, 3

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 6, 8, 18

[26] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. *arXiv preprint arXiv:2006.06666*, 2020. 2

[27] Piotr Dollár, Mannat Singh, and Ross Girshick. Fast and accurate model scaling. In *Proc. CVPR*, 2021. 8

[28] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3, 4, 5, 6, 8, 14, 17, 18

[29] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018. 1

[30] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021. 2

[31] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. PySlowFast. `https://github.com/facebookresearch/slowfast`, 2020. 2, 17

[32] Haoqi Fan, Tullie Murrell, Heng Wang, Kalyan Vasudev Alwala, Yanghao Li, Yilei Li, Bo Xiong, Nikhila Ravi, Meng Li, Haichuan Yang, Jitendra Malik, Ross Girshick, Matt

Feiszli, Aaron Adcock, Wan-Yen Lo, and Christoph Feichtenhofer. PyTorchVideo: A deep learning library for video understanding. In *Proceedings of the 29th ACM International Conference on Multimedia*, 2021. https://pytorchvideo.org/. 2

[33] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proc. CVPR*, pages 203–213, 2020. 2, 6, 7, 8, 14

[34] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *Proc. ICCV*, 2019. 2, 6, 7, 8, 14, 15, 17, 18

[35] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition in ActivityNet challenge 2019. http://static.googleusercontent.com/media/research.google.com/en//ava/2019/fair_slowfast.pdf, 2019. 13

[36] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016. 2

[37] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 1

[38] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval. In *Proc. ECCV*, volume 5. Springer, 2020. 2

[39] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE PAMI*, 2019. 2

[40] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proc. CVPR*, 2019. 2

[41] Ross Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 18

[42] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 17, 18

[43] 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. 2, 6, 7, 18

[44] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *Proc. CVPR*, 2018. 2, 6, 7, 18

[45] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020. 2

[46] Zhang Hang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, and Yue Sun. Resnest: Split-attention networks. 2020. 2

[47] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *arXiv preprint arXiv:1803.01719*, 2018. 17, 18

[48] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. ICCV*, 2017. 18

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. CVPR*, 2015. 1

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5, 17

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016. 2

[52] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. *arXiv preprint arXiv:2102.04378*, 2021. 2

[53] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. 17

[54] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 17

[55] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proc. CVPR*, pages 8129–8138, 2020. 6, 17, 18

[56] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proc. CVPR*, pages 3588–3597, 2018. 2

[57] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proc. ICCV*, pages 3464–3473, 2019. 2

[58] Ronghang Hu and Amanpreet Singh. Transformer is all you need: Multimodal multitask learning with a unified transformer. *arXiv preprint arXiv:2102.10772*, 2021. 2

[59] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proc. ECCV*, 2016. 17, 18

[60] DH Hubel and TN Wiesel. Receptive fields of optic nerve fibres in the spider monkey. *The Journal of physiology*, 154(3):572–580, 1960. 1

[61] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *Proc. CVPR*, 2019. 7

[62] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proc. CVPR*, pages 2000–2009, 2019. 2

[63] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018. 18

[64] 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. *arXiv:1705.06950*, 2017. 2, 6

[65] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 3

[66] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984. 1

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[68] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *Proc. ECCV*, pages 345–362. Springer, 2020. 7

[69] Hung Le, Doyen Sahoo, Nancy F Chen, and Steven CH Hoi. Multimodal transformer networks for end-to-end video-grounded dialogue systems. *arXiv preprint arXiv:1907.01166*, 2019. 2

[70] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1, 2

[71] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019. 2

[72] Rui Li, Chenxi Duan, and Shunyi Zheng. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv preprint arXiv:2007.14902*, 2020. 3

[73] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proc. CVPR*, pages 909–918, 2020. 7

[74] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018. 2

[75] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. In *Proc. ICCV*, 2019. 18

[76] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proc. CVPR*, pages 7083–7093, 2019. 7

[77] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. *arXiv preprint arXiv:2012.09760*, 2020. 2

[78] Wei Liu, Sihan Chen, Longteng Guo, Xinxin Zhu, and Jing Liu. Cptr: Full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*, 2021. 2

[79] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 8

[80] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020. 2

[81] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 17, 18

[82] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 17, 18

[83] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019. 2

[84] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021. 2, 3, 6, 14

[85] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proc. ICCV*, 2017. 2

[86] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 2

[87] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proc. CVPR*, June 2020. 2, 8

[88] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 2

[89] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 2

[90] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 18

[91] Azriel Rosenfeld and Mark Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, 100(5):562–569, 1971. 1

[92] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2, 6, 7, 18

[93] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 5

[94] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. 2

[95] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018. 18

[96] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 2

[97] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 17

[98] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the in-

ception architecture for computer vision. *arXiv:1512.00567*, 2015. 17, 18

[99] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2, 8

[100] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *Proc. ICML*, pages 9438–9447. PMLR, 2020. 3

[101] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 6, 8, 17, 18

[102] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 2019. 2, 6

[103] Jeya Maria Jose Valanarasu, Poojan Oza, Ilker Hacihaliloglu, and Vishal M Patel. Medical transformer: Gated axial-attention for medical image segmentation. *arXiv preprint arXiv:2102.10662*, 2021. 2

[104] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 1, 2, 3, 4

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

[106] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 3

[107] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. CVPR*, 2018. 2, 7

[108] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proc. ECCV*, 2018. 7

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

[110] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020. 2

[111] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proc. CVPR*, 2019. 2, 7

[112] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv:1712.04851*, 2017. 2

[113] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Towards explainable human pose estimation by transformer. *arXiv preprint arXiv:2012.14214*, 2020. 2

[114] Jun Yu, Jing Li, Zhou Yu, and Qingming Huang. Multimodal transformer with multi-view visual representation for image captioning. *IEEE transactions on circuits and systems for video technology*, 30(12):4467–4480, 2019. 2

[115] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 2

[116] Zhenxun Yuan, Xiao Song, Lei Bai, Wengang Zhou, Zhe Wang, and Wanli Ouyang. Temporal-channel transformer for 3d lidar-based video object detection in autonomous driving. *arXiv preprint arXiv:2011.13628*, 2020. 2

[117] Boxiang Yun, Yan Wang, Jieneng Chen, Huiyu Wang, Wei Shen, and Qingli Li. Spectr: Spectral transformer for hyperspectral pathology image segmentation. *arXiv preprint arXiv:2103.03604*, 2021. 2

[118] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. ICCV*, 2019. 17, 18

[119] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *Proc. ICLR*, 2018. 17, 18

[120] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proc. CVPR*, pages 10076–10085, 2020. 2

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

[122] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 17, 19

[123] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 2