Theory and Experiments of Multiscale Vision Transformers-

Mammal-Net Dataset...

"In the symphony of pixels and frames, Multiscale Vision Transformers choreograph a visual ballet, their algorithms pirouetting through the dance of images, transforming fleeting moments into an enchanting tapestry of understanding across the cinematic canvas of artificial sight."

Agenda

1. Related work
   - Why Self-Attention
   - Attention Is All You Need (NIPS 2017)
   - Vision Transformer (ViT) (ICLR 2021)
   - ViViT: A Video Vision Transformer (arXiv 2021.03.29)

2. Multiscale Vision Transformers(MViT)
   - Transformer Architecture
   - Multi Head Pooling Attention(MHPA)
   - ViT vs MViT
   - Multiscale Transformer Networks

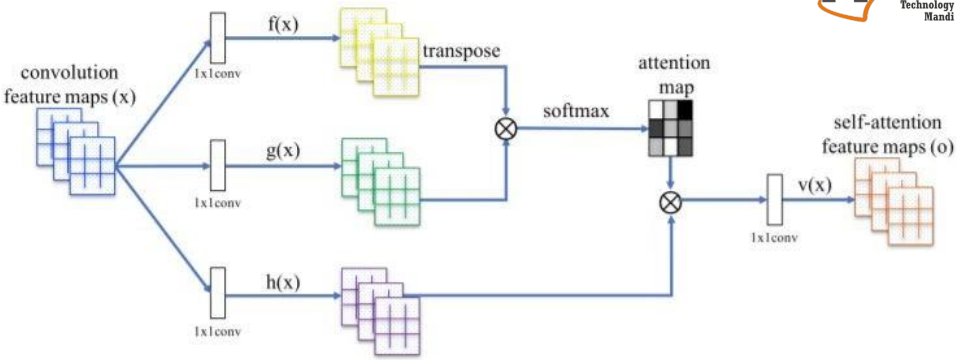
3. Experiments: Video Recognition (Mammal-Net Dataset)
   - Dataset: Mammal-Net
   - Parameters Pretrained on Kinetics 400
   - Parameters Pretrained on Kinetics 400 and Something-Something v2

Related Works

Why Self-Attention

- One is the total computational complexity per layer.
- Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.
- The third is the path length between long-range dependencies in the network.

Learning long-range dependencies is a key challenge in many sequence transduction tasks. As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models.



Attention(Q,K,V) : Self-Attention Generative Adversarial Networks

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Attention Is All You Need

self-attention models for image recognition, object detection, semantic and instance segmentation, video analysis and classification, visual question answering, visual commonsense reasoning, image captioning, vision language navigation, clustering, few-shot learning, and 3D data analysis.

Attention Is All You Need (NIPS 2017)

Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers.

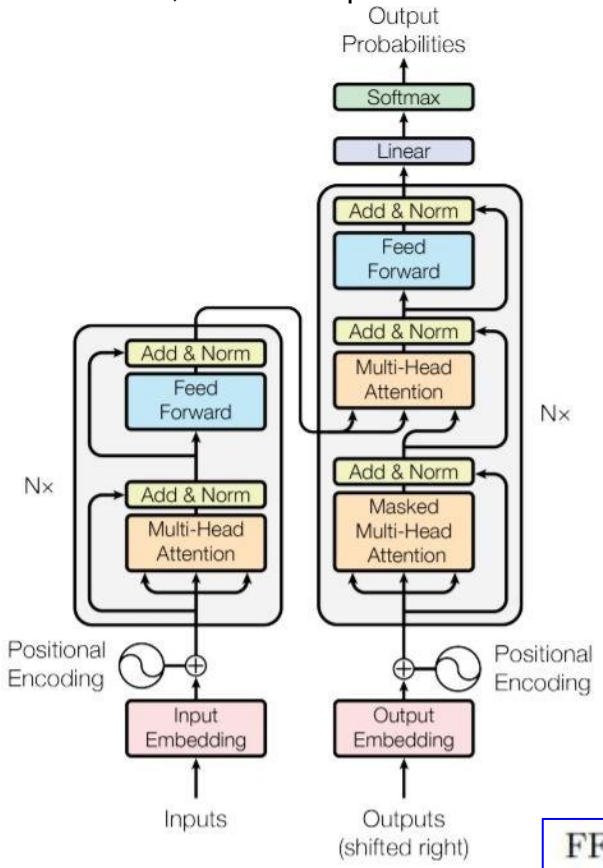
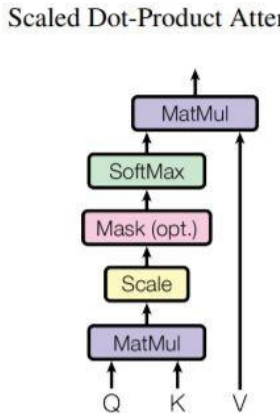
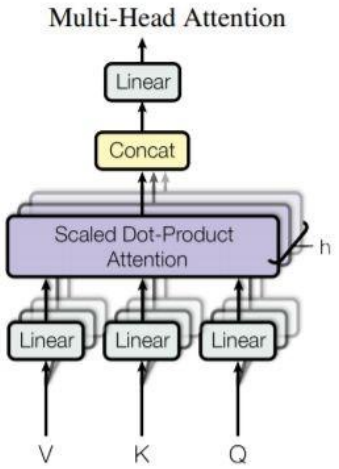


Figure 1: The Transformer - model architecture.





$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

This lead to having more stable gradients

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

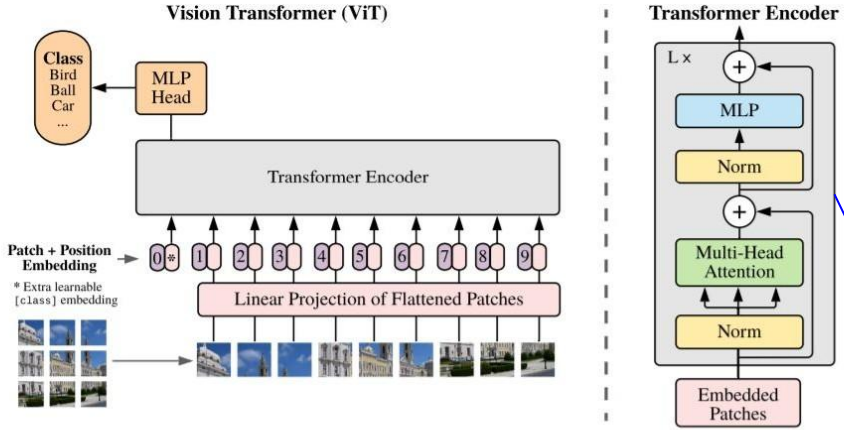
$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Vision Transformer (ViT) (ICLR 2021)

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

CNNs is not necessary and a pure transformer applied directly to sequences of image patches Vision Transformer has much less image-specific inductive bias than CNNs



The sequence of linear embeddings of these patches as an input to a Transformer. The Transformer encoder consists of alternating layers of multi headed self attention (MSA) and MLP blocks. Layernorm (LN) is applied before every block, and residual connections after every block

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$$
$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L$$
$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L$$
$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$

$$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{z}\mathbf{U}_{qkv} \qquad \mathbf{U}_{qkv} \in \mathbb{R}^{D \times 3D_h},$$
$$A = \text{softmax}\left(\mathbf{q}\mathbf{k}^\top / \sqrt{D_h}\right) \qquad A \in \mathbb{R}^{N \times N},$$
$$\text{SA}(\mathbf{z}) = A\mathbf{v}.$$

$$\text{MSA}(\mathbf{z}) = [\text{SA}_1(z); \text{SA}_2(z); \cdots; \text{SA}_k(z)]\,\mathbf{U}_{msa} \qquad \mathbf{U}_{msa} \in \mathbb{R}^{k \cdot D_h \times D}$$

Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

The Vision Transformer performs well when pre-trained on a large JFT-300M dataset. With fewer inductive biases for vision than ResNets.

※ JFT-300M Dataset (300M web images / ~ 19K categories)

ViViT: A Video Vision Transformer (arXiv 2021.03.29)

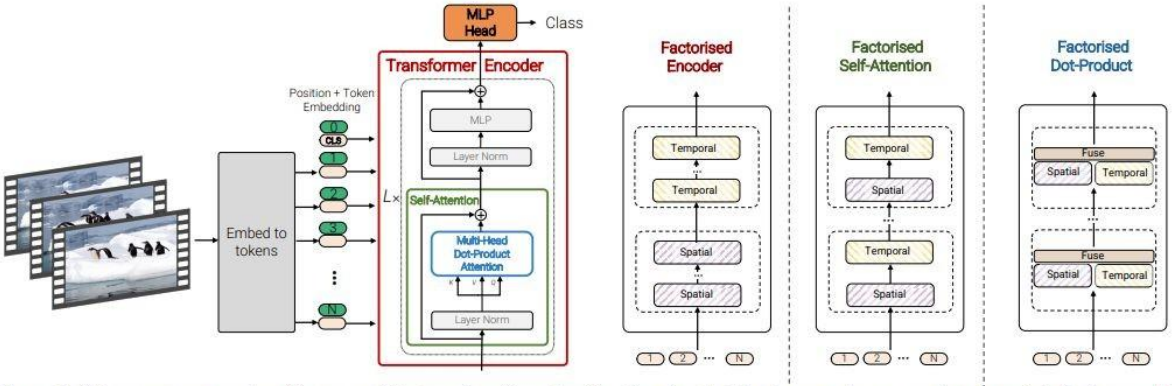


Figure 1: We propose a pure-transformer architecture for video classification, inspired by the recent success of such models for images [15] To effectively process a large number of spatio-temporal tokens, we develop several model variants which factorise different component of the transformer encoder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

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.

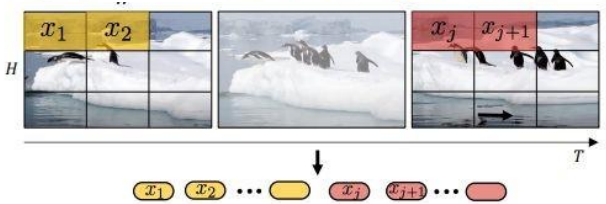


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [15].

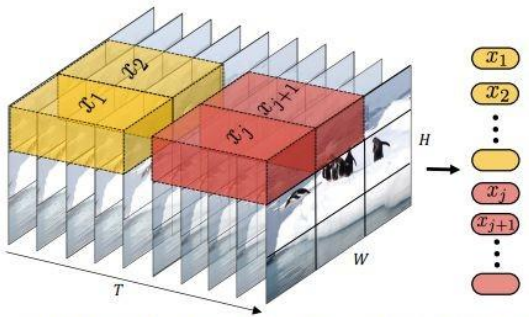


Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

Pure-transformer based models for video classification, we propose several, efficient variants of the model which factorise the spatial- and temporal-dimensions of the input. how we can effectively regularise the model during training and leverage pretrained image models to be able to train on comparatively small datasets.

Model 1: Spatio-temporal attention : all spatio-temporal tokens
Model 2: Factorised encoder : two separate transformer encoders
Model 3: Factorised self-attention : the order of spatial-then-temporal self attention.
Model 4: Factorised dot-product attention : $\mathbf{Y} = \text{Concat}(\mathbf{Y}_s, \mathbf{Y}_t)\mathbf{W}_O$

$t \times h \times w, n_t = \lfloor \frac{T}{t} \rfloor, n_h = \lfloor \frac{H}{h} \rfloor$ and $n_w = \lfloor \frac{W}{w} \rfloor$

Multiscale Vision Transformers(MViT)

Multiscale Vision Transformers(MViT) (arXiv 2021.04.22)

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 [55] 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 spe-

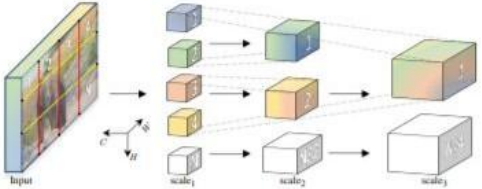


Figure 1. **Multiscale Vision Transformers** learn a hierarchy from *dense* (in space) and *simple* (in channels) to *coarse* and *complex* features. Several resolution-channel *scale* stages progressively *increase* the channel capacity of the intermediate latent sequence while *reducing* its length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

channel corresponding to ever more specialized features.

In a parallel development, the computer vision community developed *multiscale* processing, sometimes called "pyramid" strategies, with Rosenfeld and Thurston [85], Burt and Adelson [8], Koenderink [61], 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 [98] architecture allows learning arbitrary functions defined over sets and has been scalably successful in sequence tasks such as language comprehension [26] and machine translation [7]. Fundamentally, a transformer uses blocks with two basic operations. First, is an attention operation [4] for modeling inter-element re-

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.)

In this paper, the intention is to connect the seminal idea of multiscale feature hierarchies with the transformer model.

A multiscale pyramid of features with early layers operating at high spatial resolution to model simple low-level visual information visual pathway with a hierarchical model (oriented edges and bars - lower / more specific stimuli - higher) 5-10× more costly in computation and parameters

Transformer Architecture

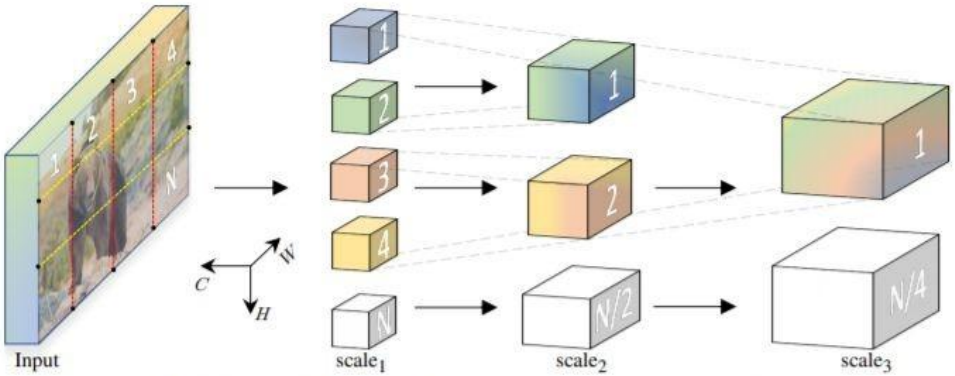


Figure 1. **Multiscale Vision Transformers** learn a hierarchy from *dense* (in space) and *simple* (in channels) to *coarse* and *complex* features. Several resolution-channel *scale* stages progressively *increase* the channel capacity of the intermediate latent sequence while *reducing* its length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

MViT 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 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.

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.

Multi Head Pooling Attention(MHPA)

Multiscale Transformers to operate at progressively changing spatio-temporal resolution.(a self attention operator that enables flexible resolution modeling in a transformer block allowing)

Multi Head Attention MHA) operators, 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.

Concretely, consider a D dimensional input tensor X of sequence length L, $X \in \mathbb{R}^{L \times D}$. Following MHA [25], 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.

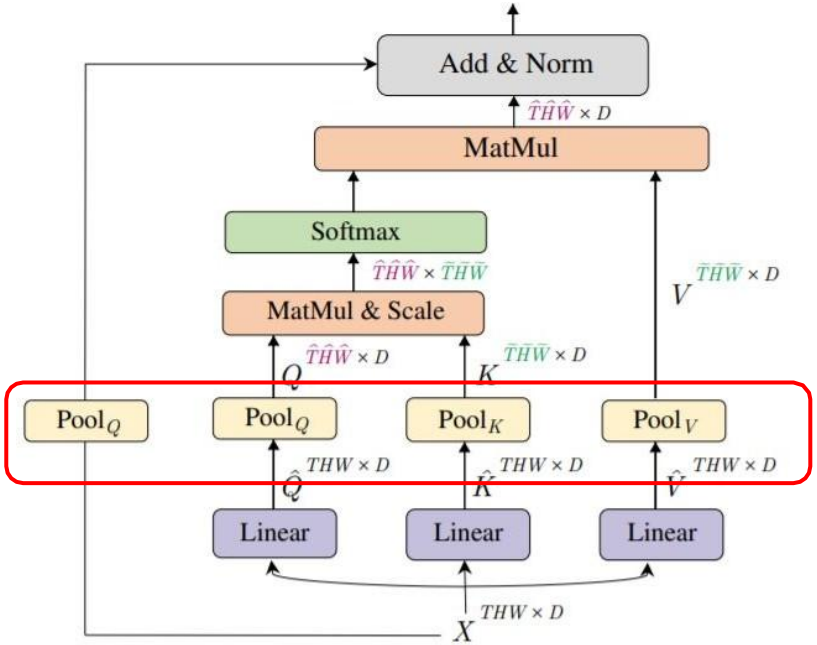


Figure 3. **Pooling Attention** is a flexible attention mechanism that (i) allows obtaining the reduced space-time resolution ($\hat{T}\hat{H}\hat{W}$) of the input (THW) by pooling the query, $Q = \mathcal{P}(\hat{Q}; \Theta_Q)$, and/or (ii) computes attention on a reduced length ($\tilde{T}\tilde{H}\tilde{W}$)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

Multi Head Pooling Attention(MHPA)

Pooling Operator. The operator P(·; Θ) performs a pooling kernel computation on the input tensor along each of the dimensions. Unpacking Θ as Θ := (k, s, p) (kernel, stride, padding) input tensor of dimensions L = T × H × W to L˜ given by,

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

Pooling Attention. The pooling operator P(·; Θ) is applied to all the intermediate tensors Qˆ, Kˆ and Vˆ independently with chosen pooling kernels k, stride s and padding p. Denoting θ yielding the pre-attention vectors Q = P(Qˆ; ΘQ), K = P(Kˆ ; ΘK) and V = P(Vˆ ; ΘV) with reduced sequence lengths. Attention is now computed on these shortened vectors, with the operation,

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

Multiple heads. 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. 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.

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

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 $O(THWh(D/h + THW/f_Q f_K))$.

ViT vs MViT

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 ($T \times \frac{H}{16} \times \frac{W}{16}$).

ViT maintains a constant channel capacity and spatial resolution throughout all the blocks.This is equivalent to a convolution with equal kernel size and stride of 1×16×16 and is shown as patch1 stage in the model definition in Table 1. sequence of length L with dimension D to encode the positional information and break permutation invariance.

positional embedding $\mathbf{E} \in \mathbb{R}^{L \times D}$

$$X_1 = \text{MHA}(\text{LN}(X)) + X$$

$$\text{Block}(X) = \text{MLP}(\text{LN}(X_1)) + X_1$$

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 spatio-temporal 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].

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.

Multiscale Transformer Networks

1. **Scale stages** : the channel dimension of the processed sequence is upsampled while the length of the sequence is down-sampled. This effectively reduces the spatio-temporal resolution of the underlying visual data while allowing the network to assimilate the processed information in more complex features.

2. **Channel expansion** : 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. down-sample the space-time resolution by 4×, we increase the channel dimension by 2×.

3. **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. our intention is to decrease resolution at the beginning of a stage and then preserve this resolution throughout the stage

4. **Key-Value pooling** : 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 (k; p; s) to be identical i.e. $\Theta K \equiv \Theta V$ within a stage, but vary s adaptively w.r.t. to the scale across stages.

5. **Skip connections** : MHPA handles this mismatch by adding the query pooling operator $P(\cdot; \Theta Q)$ to the residual path. 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.

Experiments: Video Recognition (Mammal-Net Dataset)

Dataset: MammalNet (arXiv 2306.00576)

MammalNet: A Large-scale Video Benchmark for Mammal Recognition and Behavior Understanding

Jun Chen[1*] Ming Hu[1*] Darren J. Coker[1] Michael L. Berumen[1]
Blair Costelloe[2,3] Sara Beery[4] Anna Rohrbach[5] Mohamed Elhoseiny[1]
[1]King Abdullah University of Science and Technology (KAUST)
[2] Max Planck Institute of Animal Behavior , [3] University of Konstanz
[4]Massachusetts Institute of Technology, [5]University of California, Berkeley

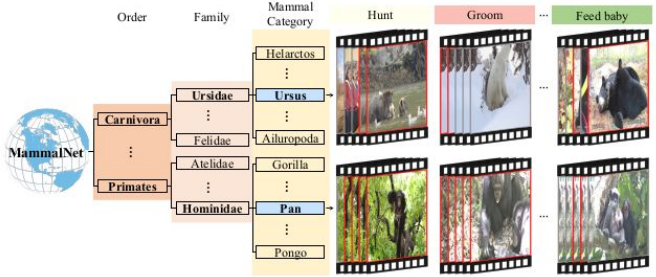
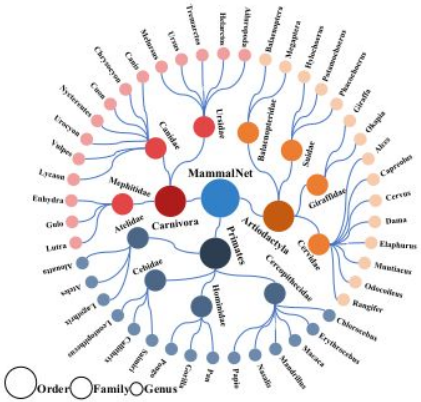
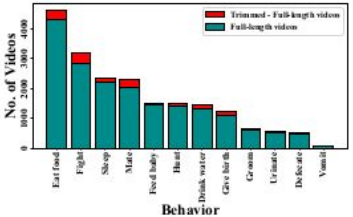


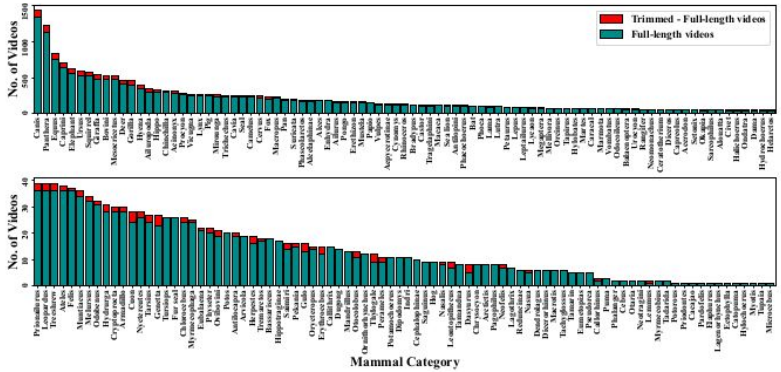
Figure 1. We propose MammalNet, a large-scale video benchmark for recognizing mammals and their behavior. It is built around a biological mammal taxonomy spanning 17 orders, 69 families and 173 mammal categories, and includes 12 common high-level mammal behaviors (e.g., hunt, groom). MammalNet enables the study of animal and behavior recognition, both separately and jointly. It also facilitates investigating challenging compositional scenarios which test models' zero- and low-shot transfer abilities. Moreover, MammalNet includes behavior detection by localizing when a behavior occurs in an untrimmed video. Our dataset is the first to enable animal behavior analysis at scale in an ecologically-grounded manner, and exemplifies multiple challenges for the computer vision community, such as recognition of imbalanced, hierarchical distributions of fine-grained categories and generalization to unseen or seldom seen scenarios.

Abstract

Monitoring animal behavior can facilitate conservation efforts by providing key insights into wildlife health, population status, and ecosystem function. Automatic recognition of animals and their behaviors is critical for capitalizing on the large unlabeled datasets generated by modern video devices and for accelerating monitoring efforts at scale. However, the development of automated recognition systems is currently hindered by a lack of appropriately labeled datasets. Existing video datasets 1) do not classify animals according to established biological taxonomies; 2) are too small to facilitate large-scale behavioral studies and are often limited ized annotations and therefore do not facilitate localization of targeted behaviors within longer video sequences. Thus, we propose MammalNet, *a new large-scale animal behavior dataset with taxonomy-guided annotations of mammals and their common behaviors. MammalNet contains over 18K videos totaling 539 hours, which is ∼10 times larger than the largest existing animal behavior dataset [36]. It covers 17 orders, 69 families, and 173 mammal categories for animal categorization and captures 12 high-level animal behaviors that received focus in previous animal behavior studies. We establish three benchmarks on MammalNet: standard animal and behavior recognition, compositional low-shot animal and behavior recognition, and behavior*







Datasets	Dataset Properties								Tasks	
	Publicly Available?	Taxonomy-guided Animal Annotation?	No. of Videos	No. of Actions	No. of Behaviors	No. of Animal Categories	No. of Mammal Categories	Total Duration	Animal Classification	Action/Behavior Recognition
Wild Felines [20]	×	×	2,700	3	-	3	3	-	✓	✓
Wildlife Actions [29]	×	×	10,600	7	-	32	11	-	✓	✓
Animal Kingdom [36]	✓	×	4,301	140	-	850	-	50 (h)	×	✓
MammalNet (ours)	✓	✓	18,346	-	12	173	173	539 (h)	✓	✓

Experiments Results

Hardware:
We trained our models on Param-Himalaya 4 compute nodes with 8 NVIDIA V100 GPUs of 16GB RAM each for a 4 days long each task.

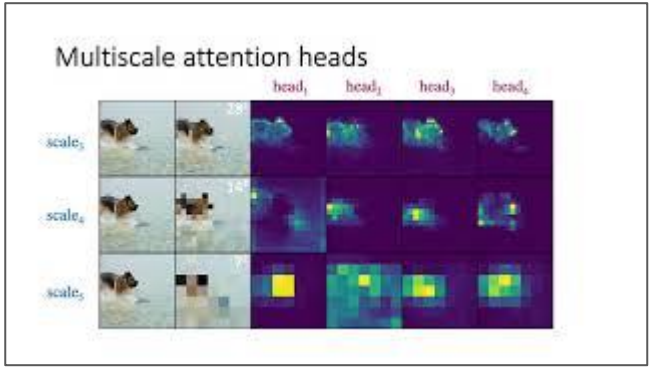
Results:

MammalNet	Animal Classification				Behavior Classification			
Pretrain	**Many**	**Medium**	**Few**	**All**	**Many**	**Medium**	**Few**	**All**
–	48.5	35.5	12.9	19.7	42.3	29.2	11.6	27.7
K400	**66.7**	**56.0**	**23.4**	**32.5**	**50.9**	**42.4**	**20.0**	**37.8**
K400 + Something-Something V2	59.2	51.3	21.7	31.8	39.6	36.1	18.3	35.2

- Initially, we evaluated the MVIT model using parameters trained solely on the Mammal-Net dataset.
- we assessed the MVIT model by loading parameters pre-trained on the Kinetics 400 dataset and subsequently fine-tuning it on the Mammal-Net dataset.
- In an experimental phase, I attempted to utilize pre-trained model parameters trained on Kinetics 400 by first training on the Something-Something V2 dataset before completing the training on Mammal-Net.

Kinetics 400: Extensive dataset capturing diverse global human activities, enabling broad video understanding.

Something-Something V2: Comprehensive video dataset portraying various human actions, facilitating fine-grained analysis.



Codes

```
# MViT options
(/slowfast/config/defaults.py)
_C.MVIT =
CfgNode()
_C.MVIT.PATCH_KERNEL = [3, 7, 7] \ _C.MVIT.PATCH_STRIDE = [2, 4, 4] \ _C.MVIT.PATCH_PADDING = [2, 4, 4]
_C.MVIT.EMBED_DIM = 96 \ _C.MVIT.MLP_RATIO = 4.0 \ _C.MVIT.DEPTH = 16 \ _C.MVIT.NORM =
# MViT options (/slowfast/models/video_model_builder.py)
@MODEL_REGISTRY.register()
class MViT(nn.Module):
        if cfg.MVIT.NORM == "layernorm":
            norm_layer = partial(nn.LayerNorm, eps=1e-6)
        self.patch_dims = [
            self.input_dims[i] // self.patch_stride[i]
            for i in range(len(self.input_dims))
        ]
        num_patches = math.prod(self.patch_dims)
        dim_mul, head_mul = torch.ones(depth + 1), torch.ones(depth + 1)
        for i in range(len(cfg.MVIT.DIM_MUL)):
            dim_mul[cfg.MVIT.DIM_MUL[i][0]] = cfg.MVIT.DIM_MUL[i][1]
        for i in range(len(cfg.MVIT.HEAD_MUL)):
            head_mul[cfg.MVIT.HEAD_MUL[i][0]] = cfg.MVIT.HEAD_MUL[i][1]
```

https://github1s.com/facebookresearch/SlowFast/blob/master/slowfast/config/defaults.py

Codes

```
        for i in range(len(cfg.MVIT.POOL_Q_STRIDE)):

        for i in range(len(cfg.MVIT.POOL_KV_STRIDE)):

        for i in range(depth):
            num_heads = round_width(num_heads, head_mul[i])
            embed_dim = round_width(embed_dim, dim_mul[i], divisor=num_heads)
            dim_out = round_width( embed_dim, dim_mul[i + 1],
                divisor=round_width(num_heads, head_mul[i + 1]),
            )
            self.blocks.append(
                MultiScaleBlock(
                    dim=embed_dim, dim_out=dim_out, num_heads=num_heads, mlp_ratio=mlp_ratio,
                    qkv_bias=qkv_bias, drop_rate=self.drop_rate, drop_path=dpr[i], norm_layer=norm_layer,
                    kernel_q=pool_q[i] if len(pool_q) > i else [],
                    kernel_kv=pool_kv[i] if len(pool_kv) > i else [],
                    stride_q=stride_q[i] if len(stride_q) > i else [],
                    stride_kv=stride_kv[i] if len(stride_kv) > i else [],
                    mode=mode, has_cls_embed=self.cls_embed_on, pool_first=pool_first,
                )
            )
```

https://github1s.com/facebookresearch/SlowFast/blob/master/slowfast/models/video_model_builder.py

Indian Institute of Technology Mandi

Thanks