UNIVERSITY OF
OXFORD

On efficient Transformers for large-scale video recognition

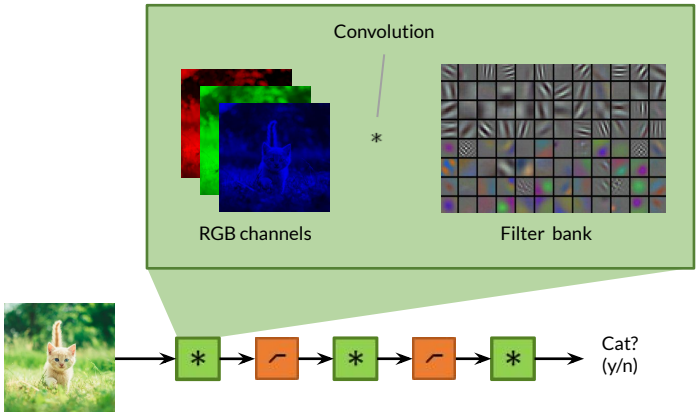
João F. Henriques
Visual Geometry Group

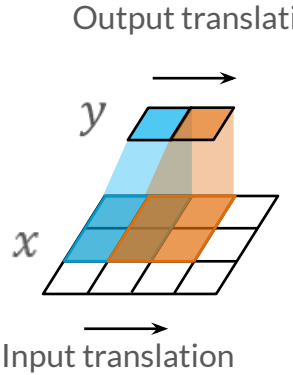
Motivation

Action recognition in videos

- **Proxy**/testbed for other video recognition tasks. (≈ to classification in images)
- Often requires **fine-grained** distinctions to be made based on subtle motions.
- Often requires **long-range** associations.
- *E.g.: swing dancing vs. salsa dancing;*
  dribbling basketball vs. dunking basketball;
  catching ball vs. throwing ball;
  ...

Background



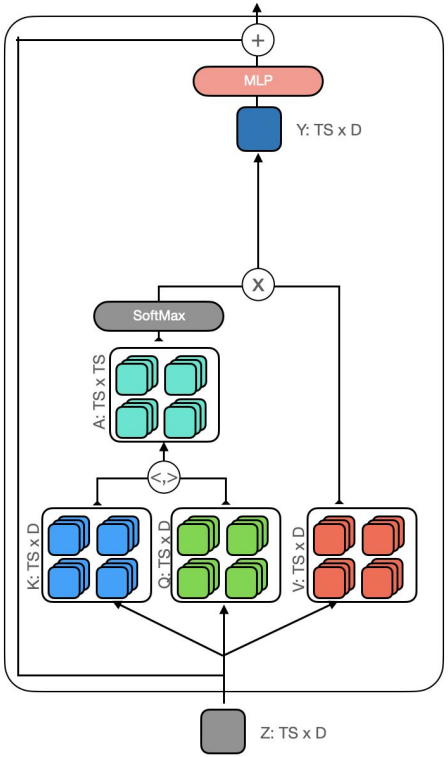


Convolutions limit the receptive field, both spatially and temporally.

- Alleviated with *atrous* convolution.
- Receptive field varies with resolution and framerate; can be difficult to tune.

Tran et al., Learning spatiotemporal features with 3D convolutional networks. In ICCV, 2015.
Carreira & Zisserman, Quo vadis, action recognition? A new model and the Kinetics dataset. In CVPR, 2017.
Tranet et al., A closer look at spatiotemporal convolutions for action recognition. In CVPR, 2018.
Wang et et al., Non-local neural networks. In CVPR, 2018.

Background



Transformer

- Current dominant paradigm in NLP.
- Very little inductive bias compared to CNNs.
  ⇒ Often harder to train, but more flexible.
- Long-range associations / receptive field covers the **full input** at all stages.
- Computation grows quadratically with input ($\mathcal{O}(S^2T^2)$ for input with T frames and S pixels).

Patrick et al., Support-set bottlenecks for video-text representation learning. In ICLR, 2021.
Dosovitskiy et al., An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021.
Touvron et al., Training data-efficient image transformers & distillation through attention. In ICML, 2021.
Doersch et al., Crosstransformers: spatially-aware few-shot transfer. In NeurIPS, 2020.
Torresani et al., Is space-time attention all you need for video understanding? In ICML, 2021.

Recovering lost inductive bias?

UNIVERSITY OF OXFORD

Some inductive bias is beneficial.

- Allows **extrapolating** beyond the training set / generalizing
  (e.g. CNNs maintain a consistent output in translations of the training set, a.k.a. translation equivariance).
  - *Physical motivation:*
    The motion of the camera does not affect the properties of the scene.
  - We want to disentangle an object's motion path from its appearance; translation equivariance is a **subset** of this behaviour.

Example	FC/MLP	CNN
Translated training image	✗	✓
Sparse connectivity	✗	✓

Recovering lost inductive bias?

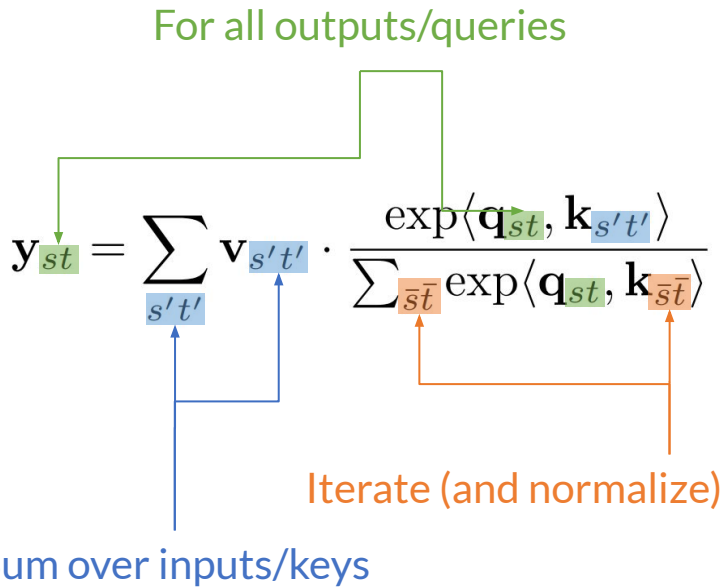
Some inductive bias is beneficial.

- Allows **extrapolating** beyond the training set / generalizing.
- Can be exploited for computational **efficiency** (e.g. reuse computations, introduce sparsity/locality).
- Sometimes allows reducing the #parameters (reduce **overfitting**).

Example	FC/MLP	CNN
Translated training image	✗	✓
Sparse connectivity	✗	✓

⇒ Can Transformers enjoy the same benefits as CNNs?

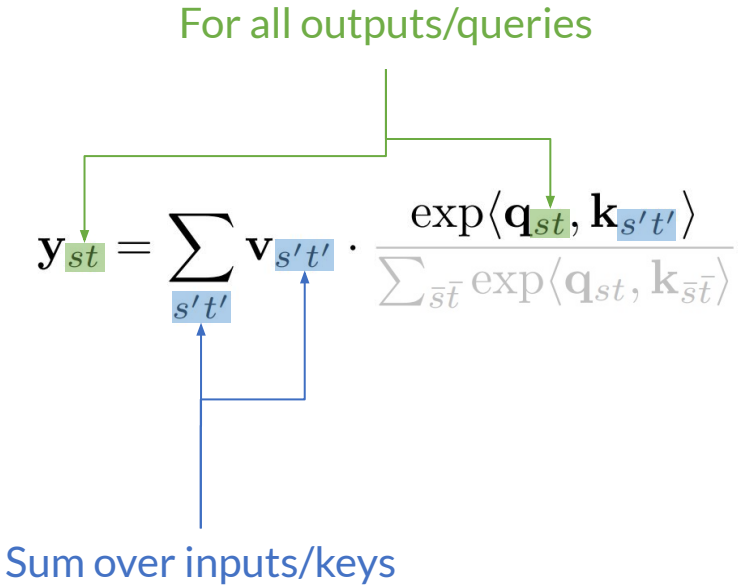
Joint space-time attention



Normalization factor is independent of the summed variable, so it can be computed separately / we can ignore it for now.

Bertasius et al., Is space-time attention all you need for video understanding? In ICML, 2021.
Arnab et al., Vivit: A video vision transformer, 2021.

Joint space-time attention



- Computational complexity: $\mathcal{O}(S^2T^2)$.
- Infeasible for long and high-res. videos.
- Can we get closer to $\mathcal{O}(ST)$?

Bertasius et al., Is space-time attention all you need for video understanding? In ICML, 2021.
Arnab et al., Vivit: A video vision transformer, 2021.

Divided space-time attention

- *Idea:* alternate between **spatial** and **temporal** attention layers

For all outputs/queries

$$\mathbf{y}_{st} = \sum_{s'} \mathbf{v}_{s't} \cdot \frac{\exp\langle \mathbf{q}_{st}, \mathbf{k}_{s't} \rangle}{\sum_{\bar{s}} \exp\langle \mathbf{q}_{st}, \mathbf{k}_{\bar{s}t} \rangle}$$

Sum over inputs/keys **spatially.**

$$\mathcal{O}(S^2T)$$

For all outputs/queries

$$\mathbf{y}_{st} = \sum_{t'} \mathbf{v}_{st'} \cdot \frac{\exp\langle \mathbf{q}_{st}, \mathbf{k}_{st'} \rangle}{\sum_{\bar{t}} \exp\langle \mathbf{q}_{st}, \mathbf{k}_{s\bar{t}} \rangle}$$

Sum over inputs/keys **temporally.**

$$\mathcal{O}(ST^2)$$

Bertasius et al., Is space-time attention all you need for video understanding? In ICML, 2021.
Arnab et al., Vivit: A video vision transformer, 2021.

Divided space-time attention

For all outputs/queries

$$\mathbf{y}_{st} = \sum_{s'} \mathbf{v}_{s't} \cdot \frac{\exp\langle \mathbf{q}_{st}, \mathbf{k}_{s't} \rangle}{\sum_{\bar{s}} \exp\langle \mathbf{q}_{st}, \mathbf{k}_{\bar{s}t} \rangle}$$

Sum over inputs/keys **spatially**

$$\mathcal{O}(S^2T)$$

For all outputs/queries

$$\mathbf{y}_{st} = \sum_{t'} \mathbf{v}_{st'} \cdot \frac{\exp\langle \mathbf{q}_{st}, \mathbf{k}_{st'} \rangle}{\sum_{\bar{t}} \exp\langle \mathbf{q}_{st}, \mathbf{k}_{s\bar{t}} \rangle}$$

Sum over inputs/keys **temporally**

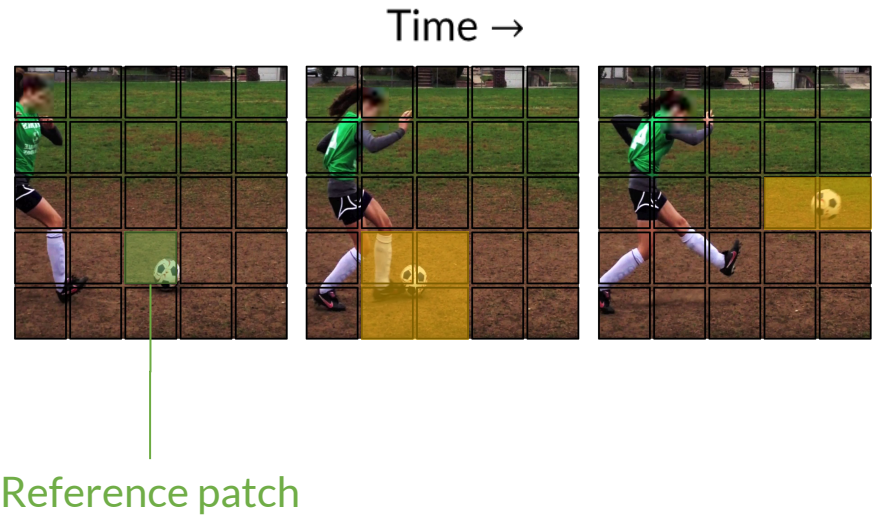
$$\mathcal{O}(ST^2)$$

- Significant computation/memory gains.
- Still has a quadratic bottleneck in each dimension.
- Axis-aligned pooling/attention is artificial.

Moving camera + moving objects

Bertasius et al., Is space-time attention all you need for video understanding? In ICML, 2021.
Arnab et al., Vivit: A video vision transformer, 2021.

Trajectory attention – Motivation



- Consider the reference patch on the left.
- We want to find other patches that contain the ball to compute the output for the ref. patch (similar to a CNN's filter being applied). Why?
  - To leverage **multiple views** of the same obj. to better understand its properties (e.g. overcome occlusion, transient appearance...).
  - To reason about the **motion** of the object.
- **Attention** can achieve these objectives, since it computes feature similarity across space-time.
- After finding the patches, we can aggregate their information into a single output embedding.

Trajectory attention

- *Idea:* perform attention **along trajectories**.

For all outputs/queries

$$\tilde{\mathbf{y}}_{stt'} = \sum_{s'} \mathbf{v}_{s't'} \cdot \frac{\exp\langle \mathbf{q}_{st}, \mathbf{k}_{s't'} \rangle}{\sum_{\bar{s}} \exp\langle \mathbf{q}_{st}, \mathbf{k}_{\bar{s}t'} \rangle}$$

For all steps of a trajectory (**temporally**)

Sum over inputs/keys **spatially**

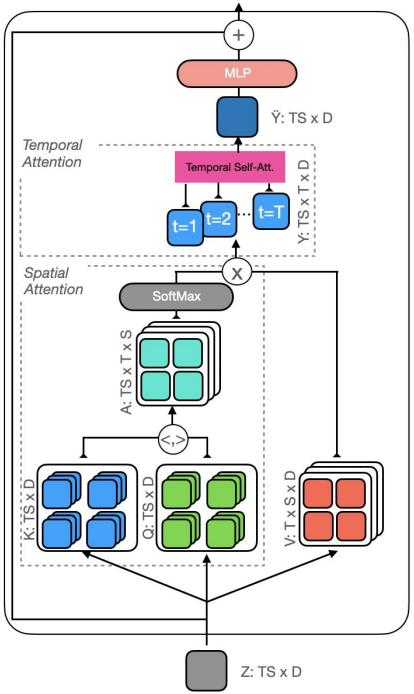
Reference patch

Pool values over all pos. s' for $t' = 0$

Pool values over all pos. s' for $t' = 1$

Yields one embedding per frame, per ref. patch – an object's trajectory over time.

Trajectory attention



- For each ref. patch, this indexes a trajectory over the video.
- Embeddings in the trajectory are then **pooled** via temporal attention.

  ⇒ Summarizes info over the trajectory.
- **Inductive bias:** A scene is composed of objects and a camera that move.
- Overall complexity is $\mathcal{O}(S^2T^2)$, so no better than before. ⇒ Needs to be improved through other means.

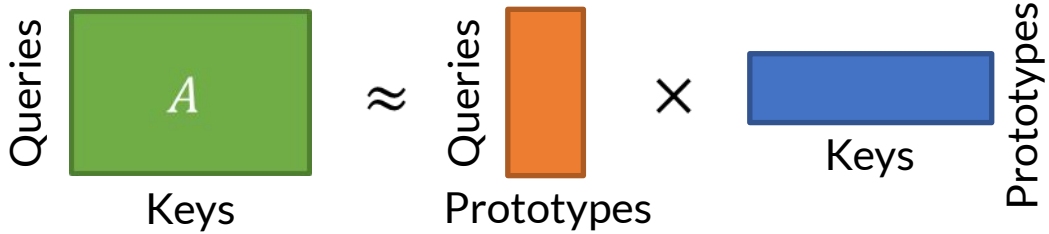
Reference patch

Pool values over all pos. s' for $t' = 0$

Pool values over all pos. s' for $t' = 1$

Computational efficiency?

Idea: Take inspiration from **matrix factorization** methods / low-rank decomposition.



Cost of multiplying this matrix by an arbitrary vector: $\mathcal{O}(S^2T^2) \rightarrow \mathcal{O}(STP)$

- Not just multiplication.
  In general:
  $A \approx f($  $)$
- Due to the softmax, attention matrices usually have high rank.

  ⇒ Poorly approximated by PCA/low-rank decompositions.
- Prototypes must be few, and representative of all keys/queries.

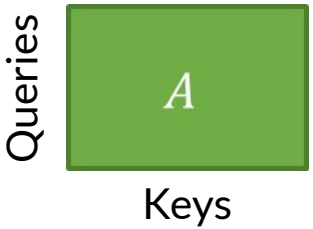
Xiong et al., Nyströmformer: A Nyström-based algorithm for approximating self-attention. In AAAI, 2021.

Beltagy et al., Longformer: The long-document transformer, 2020.

Choromanski et al., Rethinking attention with performers. In ICLR, 2021.

Computational efficiency?

Formulate attention probabilistically.



- Define categorical random variables:
  $A_{ij} \in \{0,1\}$ → If 1, assign key j to query i.
- The attention operator defines a **parametric model** of the probability of event A_{ij}, with a multinomial logistic function:

$$P(A_{i:}) = \mathcal{S}(\mathbf{q}_i^\mathsf{T}\mathbf{K})$$

Softmax — Query vector — Key vectors (concatenated)

- Introduce **latent variables**:
  $U_{jl} \in \{0,1\}$ → If 1, assign key j to prototype l.
- Then: $P(A_{ij}) = \sum_\ell P(A_{ij} \mid U_{\ell j}) P(U_{\ell j})$

⇒

- No approximations made so far (used laws of total/conditional prob.)
- However, $P(A|U)$ is intractable.

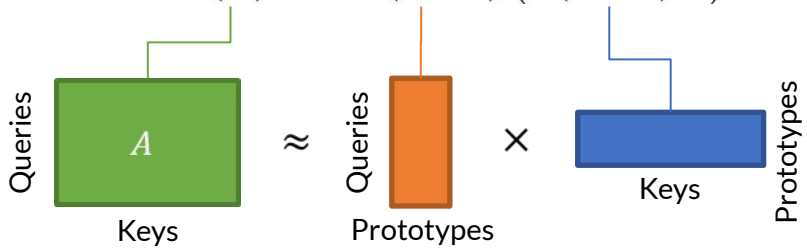
Computational efficiency?

- Exact model with auxiliary "prototype" indicator variables U: $P(A_{ij}) = \sum_{\ell} P(A_{ij} \mid U_{\ell j}) P(U_{\ell j})$

- How to approximate $P(A|U)$? Use a similar parametric model: $\tilde{P}(A \mid U) = \mathcal{S}(\mathbf{Q}^{\mathsf{T}}\mathbf{P})$

- Putting it all together: $\tilde{P}(A)\mathbf{V} = \mathcal{S}(\mathbf{Q}^{\mathsf{T}}\mathbf{P})\left(\mathcal{S}(\mathbf{P}^{\mathsf{T}}\mathbf{K})\mathbf{V}\right)$

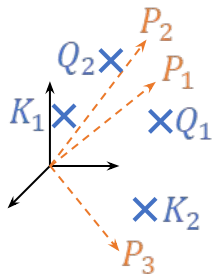
$$\mathcal{O}(S^2T^2) \rightarrow \mathcal{O}(STP)$$

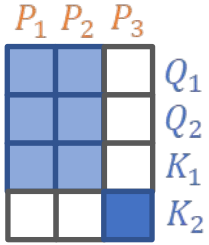


Selecting prototypes

Priorities:

- **Dynamically** adjust to keys/queries to ensure their region is reconstructed well.
- Minimize **redundancy** between prototypes.

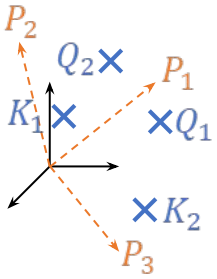


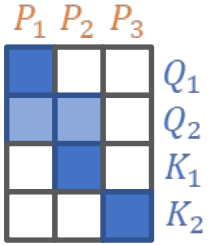


$P_1 \parallel P_2 \perp P_3$

Some suboptimal choices:

- Trainable vectors *(not adaptive)*.
- Random sampling from keys/queries *(often selects collinear vectors)*.
- Clustering keys/queries online *(expensive)*.





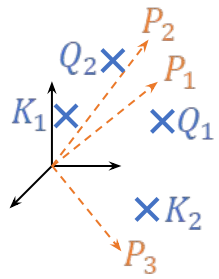
$P_1 \perp P_2 \perp P_3$

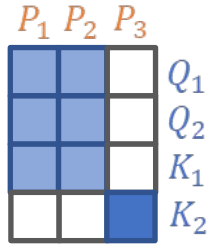
Selecting prototypes

Objective:
Select **most orthogonal** subset of keys/queries.

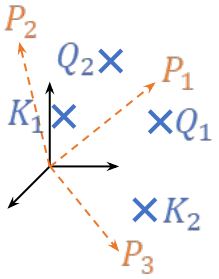
A greedy algorithm:

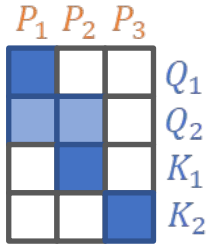
$X \leftarrow$ random subset of $K \cup Q$
For $l \in \{1, \dots, |P|\}$:
  $i^* \leftarrow \text{argmin}_i \sum_{j=1}^{l-1} |\langle X_i, P_j \rangle|$
  $P_l \leftarrow X_{i^*}$





$P_1 \parallel P_2 \perp P_3$





$P_1 \perp P_2 \perp P_3$

Orthoformer

UNIVERSITY OF OXFORD

Algorithm 1 Orthoformer (proposed) attention

$\mathbf{P} \leftarrow \text{MostOrthogonalSubset}(\mathbf{Q}, \mathbf{K}, R)$
$\mathbf{\Omega}_1 = \mathcal{S}(\mathbf{Q}^\mathsf{T}\mathbf{P}/\sqrt{D})$
$\mathbf{\Omega}_2 = \mathcal{S}(\mathbf{P}^\mathsf{T}\mathbf{K}/\sqrt{D})$
$\mathbf{Y} = \mathbf{\Omega}_1(\mathbf{\Omega}_2\mathbf{V})$

- Attention computation grows **linearly** with input, instead of **quadratically**.
- Fewer and less expensive operations than the Nyströmformer.
- Orthogonal subset selection plays a similar role to Nyströmformer's inverse (downweight collinear prototypes).

Algorithm 2 Nyströmformer [85] attention

$\mathbf{P}_q, \mathbf{P}_k \leftarrow \text{SegmentMeans}(\mathbf{Q}, \mathbf{K}, R)$
$\mathbf{\Omega}_1 = \mathcal{S}(\mathbf{Q}^\mathsf{T}\mathbf{P}_k/\sqrt{D})$
$\mathbf{\Omega}_2^{-1} = \text{IterativeInverse}(\mathcal{S}(\mathbf{P}_q^\mathsf{T}\mathbf{P}_k/\sqrt{D}), N_{\text{iter}})$
$\mathbf{\Omega}_3 = \mathcal{S}(\mathbf{P}_q^\mathsf{T}\mathbf{K}/\sqrt{D})$
$\mathbf{Y} = \mathbf{\Omega}_1\left(\mathbf{\Omega}_2^{-1}\left(\mathbf{\Omega}_3\mathbf{V}\right)\right)$

Nyströmformer:

- Iterative inverse is expensive, and does not converge (more iterations seem to hurt performance).
- Approximations are made that diverge significantly from the Nyström method.

Experiments

UNIVERSITY OF OXFORD

Table 5: **Comparison to the state-of-the-art on Long Range Arena benchmark.** GFLOPS and CUDA maximum Memory (MB) are reported for the ListOps task. Note that our algorithm achieves the best overall results with far fewer prototypes (64) than the other methods.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Avg↑	GFLOPS↓	Mem.↓
Exact [76]	36.69	63.09	78.22	31.47	66.35	55.16	1.21	4579
Performer-256 [14]	36.69	63.22	**78.98**	29.39	**66.55**	54.97	0.49	885
Nyströmformer-128 [85]	**36.90**	64.17	78.67	**36.16**	52.32	53.64	0.62	745
Orthoformer-64	33.87	**64.42**	78.36	33.26	66.41	**55.26**	**0.24**	**344**

- About **half** the memory and GFLOPS cost of the best approximation.
- **No loss** of performance (unlike the other approximations).

Experiments

Application: **action recognition**

- Use ViT [1] as the base model (12 layers / 12 attention heads / embeddings size 768).
- Separate time-space positional encodings (from TimeSformer [2]).
- Cubic image tokenization (ViViT [3]).
- Added our **Trajectory Attention** and **Orthoformer**.

Chosen using ablation experiments (details in the paper).

Datasets:

- Kinetics-400 *(appearance cues are more dominant)*
- Something-Something V2 (*motion cues are more dominant*)
- Epic Kitchens 100

SSv2 keeps **objects consistent** across different action classes.

[1] Dosovitskiy et al., An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021.

[2] Bertasius et al., Is space-time attention all you need for vide understanding? In ICML, 2021.

[3] Arnab et al., Vivit: A video vision transformer, 2021.

Experiments

Table 4: **Attention ablations:** We compare trajectory attention with alternatives and ablate its design choices. We report GFLOPS and top-1 accuracy (%) on K-400 and SSv2. Att_T: temporal attention, Avg_T: temporal averaging, Norm_{ST}: space-time normalization, Norm_S: spatial normalization.

Attention	Att_T	Avg_T	Norm_S	Norm_{ST}	GFLOPS	K-400	SSv2
Joint Space-Time	–	–	–	–	180.6	79.2	64.0
Divided Space-Time	–	–	–	–	185.8	78.5	64.2
	✗	✓	✓	✗	180.6	76.0	60.0
	✓	✗	✗	✓	369.5	77.2	60.9
Trajectory	✓	✗	✓	✗	369.5	**79.7**	**66.5**

- Top-1 accuracy (%) on K-400 and SSv2 **improves with trajectory attention**, especially on SSv2 (**+2.3%**) where **motion cues** are more important.

Experiments

Table 3: **Orthoformer ablations:** We ablate various aspects of our Orthoformer approximation. E denotes exact attention and A denotes approximate attention. We report max CUDA memory consumption (GB) and top-1 accuracy (%) on K-400 and SSv2.

(a) Orthoformer is competitive with Nyström.

Attention	Approx.	Mem.	K-400	SSv2
Trajectory (E)	N/A	7.4	**79.7**	**66.5**
Trajectory (A)	Performer	5.1	72.9	52.7
	Nyströmformer	3.8	**77.5**	**64.0**
	Orthoformer	3.6	**77.5**	63.8

(b) Selecting orthogonal prototypes is the best strategy.

Attention	Selection	Mem.	K-400	SSv2
Trajectory (E)	N/A	7.4	**79.7**	**66.5**
Trajectory (A)	Seg-Means	3.6	75.8	60.3
	Random	3.6	76.5	62.5
	Orthogonal	3.6	**77.5**	**63.8**

- (left) The proposed **Orthoformer** approximation is competitive with the the Nyströmformer on K-400/SSv2, while being less complex, using less memory and computation.
- (right) Random selection or clustering of prototypes degrades performance.

Experiments

UNIVERSITY OF OXFORD

(a) **Something–Something V2**

Model	Pretrain	Top-1	Top-5	GFLOPs × views
SlowFast [25]	K-400	61.7	-	65.7×3×1
TSM [46]	K-400	63.4	88.5	62.4×3×2
STM [33]	IN-1K	64.2	89.8	66.5×3×10
MSNet [40]	IN-1K	64.7	89.4	67×1×1
TEA [45]	IN-1K	65.1	-	70×3×10
bLVNet [23]	IN-1K	65.2	90.3	128.6×3×10
VidTr-L [44]	IN-21K+K-400	60.2	-	351×3×10
Tformer-L [7]	IN-21K	62.5	-	1703×3×1
ViViT-L [2]	IN-21K+K-400	65.4	89.8	3992×4×3
MViT-B [22]	K-400	67.1	90.8	170×3×1
Mformer	IN-21K+K-400	66.5	90.1	369.5×3×1
Mformer-L	IN-21K+K-400	**68.1**	**91.2**	1185.1×3×1
Mformer-HR	IN-21K+K-400	67.1	90.6	958.8×3×1

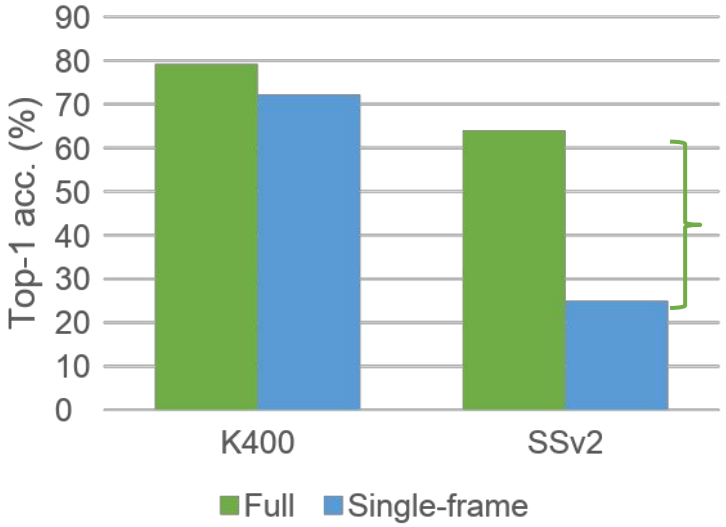
(b) **Kinetics-400**

Method	Pretrain	Top-1	Top-5	GFLOPs × views
I3D [10]	IN-1K	72.1	89.3	108×N/A
R(2+1)D [75]	-	72.0	90.0	152×5×23
S3D-G [84]	IN-1K	74.7	93.4	142.8×N/A
X3D-XL [24]	-	79.1	93.9	48.4×3×10
SlowFast [25]	-	79.8	93.9	234×3×10
VTN [51]	IN-21K	78.6	93.7	4218×1×1
VidTr-L [44]	IN-21K	79.1	93.9	392×3×10
Tformer-L[7]	IN-21K	80.7	94.7	2380×3×1
MViT-B [22]	-	81.2	95.1	455×3×3
ViViT-L [2]	IN-21K	**81.3**	94.7	3992×3×4
Mformer	IN-21K	79.7	94.2	369.5×3×10
Mformer-L	IN-21K	80.2	94.8	1185.1×3×10
Mformer-HR	IN-21K	81.1	**95.2**	958.8×3×10

- **SOTA** on SSv2 (+1%), which is more reliant on motion cues.
- Competitive with the much larger ViViT-L model on K400.

Experiments

Train model on **single frames only** and assess drop in performance.



More dramatic drop (-39%) in SSv2, showing that motion cues are much more important in this dataset.

Experiments

UNIVERSITY OF OXFORD

(c) **Epic-Kitchens**

Method	Pretrain	A	V	N
TSN [78]	IN-1K	33.2	60.2	46.0
TRN [86]	IN-1K	35.3	65.9	45.4
TBN [36]	IN-1K	36.7	66.0	47.2
TSM [46]	IN-1K	38.3	**67.9**	49.0
SlowFast [25]	K-400	38.5	65.6	50.0
ViViT-L [2]	IN-21K+K-400	44.0	66.4	56.8
Mformer	IN-21K+K-400	43.1	66.7	56.5
Mformer-L	IN-21K+K-400	44.1	67.1	57.6
Mformer-HR	IN-21K+K-400	**44.5**	67.0	**58.5**

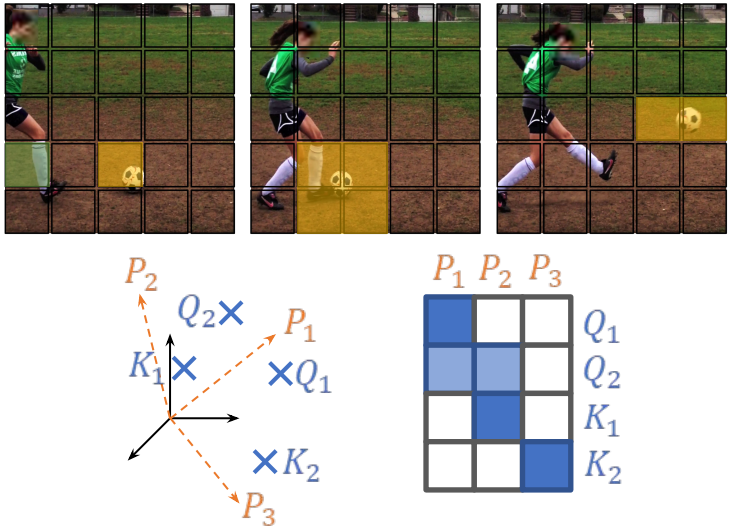
(d) **Kinetics-600**

Model	Pretrain	Top-1	Top-5	GFLOPs × views
AttnNAS [81]	-	79.8	94.4	-
LGD-3D [56]	IN-1K	81.5	95.6	-
SlowFast [25]	-	81.8	95.1	234×3×10
X3D-XL [24]	-	81.9	95.5	48.4×3×10
Tformer-HR [7]	IN-21K	82.4	96.0	1703×3×1
ViViT-L [2]	IN-21K	83.0	95.7	3992×3×4
MViT-B-24 [22]	-	**83.8**	**96.3**	236×1×5
Mformer	IN-21K	81.6	95.6	369.5×3×10
Mformer-L	IN-21K	82.2	96.0	1185.1×3×10
Mformer-HR	IN-21K	82.7	96.1	958.8×3×10

- Improvement of 2.3% on Epic-Kitchens Nouns, which depends more on motion.
- Competitive performance on K600.

Conclusions

- Aggregating information along **implicit motion trajectories** can inject a realistic induction bias into video Transformers.
- The **quadratic dependency** on input size (resolution/video length) can be lowered to **linear** with a simple but well-grounded probabilistic formulation.
- Using **orthogonality** as the criterion for choosing attention prototypes is very effective.
- We observe better results (some SOTA) on **motion-focused** datasets, and the proposed **Orthoformer** can be applied to other domains.



Algorithm 1 Orthoformer (proposed) attention

$\mathbf{P} \leftarrow \text{MostOrthogonalSubset}(\mathbf{Q}, \mathbf{K}, R)$
$\mathbf{\Omega}_1 = \mathcal{S}(\mathbf{Q}^\top \mathbf{P}/\sqrt{D})$
$\mathbf{\Omega}_2 = \mathcal{S}(\mathbf{P}^\top \mathbf{K}/\sqrt{D})$
$\mathbf{Y} = \mathbf{\Omega}_1(\mathbf{\Omega}_2 \mathbf{V})$

Thank you



Mandela Patrick* Dylan Campbell* Yuki M. Asano*

Ishan Misra Florian Metze Christoph Feichtenhofer

Andrea Vedaldi João F. Henriques

University of Oxford / Facebook Research

*joint first authors