TokenLearner: Adaptive Space-Time Tokenization for Videos

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we present an approach for representation learning from videos. Instead of relying on hand-designed splitting strategies to obtain space-time tokens from videos, our approach learns to mine important tokens in video frames. This results in efficiently and effectively finding a few important visual tokens and enables modeling of pairwise interactions between such tokens over a longer temporal horizon. We introduce a vector transformer to capture such pairwise space-time relations, and a technique to fuse the transformed tokens while learning their spatio-temporal patterns. The proposed approach is designed with the intention to allow the tokenizer to adaptively react to input video frames containing diverse visual content, and then to have the vector transformer and subsequent modules learn the underlying spatio-temporal interactions and long-range dependencies in video inputs. We show the effectiveness of the method over challenging video classification datasets, outperforming the state-of-the-art, despite using much less compute. We further conduct extensive ablation experiments to study the method.

1 Introduction

Videos provide an abundance of visual information, with an ever-increasing rates of video content generation. Videos themselves are high dimensional data with rich semantic content. Video understanding typically targets action classification or localization, tasks guided by labeled datasets [19, 21, 9, 28]. Videos are also a very valuable source of self-supervised learning, where the underlying feature representations can be learned without explicit labeling. Several unsupervised approaches have demonstrated great advances recently, starting to reduce the gap to supervised ones [20, 37, 22, 1]. For both supervised and self-supervised learning, it is very important to be able to represent video information effectively. But what basic processing mechanism can do that successfully for videos?

Videos, in contrast to images, require employing effective spatial-temporal processing of the both RGB and time streams to capture long-range interactions. An important aspect of this understanding is how to quickly learn which parts of the input video stream are the most important, both spatially and temporally, and to focus computational resources there. Standard approaches take videos as stacked images (i.e., a space-time volume) and tend to extend 2D neural architectures to 3D (e.g., 3D-ResNets [16, 5, 33, 10]). Recent work created 3D 'cubelet' video tokens on a regular 3D-grid which are further processed by a Transformer [2, 3], resulting in computationally heavy models.

The main question addressed in this work is how to adaptively learn information from video inputs to most effectively capture the spatio-temporal interactions. Here are our **main ideas**:

The first key observation is that we can learn to compute the important spatio-temporal tokens from the input stream. More specifically, we compute multiple spatial weight maps per frame with spatial attention. The goal of these maps is to learn which areas are of importance. Here, each spatial weight map is multiplied with the input to form a token, to be processed by the subsequent learning modules.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Further, with these adaptively learned tokens, we apply a vector attention between key-query pairs. This can be thought as a version of Transformer in which the number of heads is the same as channels, allowing us to learn a different attention matrix for each channel. It captures in an efficient way pairwise space-time relations per channel, particularly benefiting tokens with rich channel information. Finally, the transformed tokens are explicitly fused with a linear layer to capture the spatio-temporal patterns formed by the tokens. This results in a lightweight convolution-free approach, which brings forth an efficient video representation, which is able to model long-range dependencies of visual patterns.

The approach is simple, efficient, and, as shown by the results, outperforms methods including both convolutional methods and previous space-time transformer ones from prior art.

2 Adaptive Tokenization in Videos

In recent Video transformer architectures, such as ViViT [2] and TimeSformer [3], the video is first tokenized by cutting the video into 2d spatial or 3d spatio-temporal cubes on a regular grid, which are used as input to the model.

Instead of processing fixed, tokenized inputs, our attention module learns the tokens that are to be used for the recognition task. We gain several important properties by doing so: (1) We make each subsequent layer learn to rely on different space-time tokenizations, potentially allowing different layers to capture different aspects of the video. (2) We enable the adaptive tokenization so that the tokens could be dynamically selected conditioned on the input. (3) We also reduce the total number of tokens for the transformer. This is particularly beneficial considering that the computation is quadratic to the number of tokens.

These dynamically and adaptively generated tokens (Section 2.1) are provided to a vector-attention module to model space-time interactions between them (Sections 2.2), and then they are explicitly fused (Section 2.3). The idea is to capture the spatio-temporal patterns formed by the transformed tokens, and provide better representations for the further token learning in the next layer. Figure 1 provides an overview.

We name our attention module as "TokenLeaner".

2.1 Token learning

Let X be an input tensor with a space-time shape: $X \in \mathbb{R}^{H \times W \times T \times C}$ where $H \times W$ corresponds to the spatial dimension of the input, T is the temporal dimension (i.e., number of frames), and C is the number of channels. Let $\mathbf{x}^t$ be a temporal slice of it, corresponding to the frame t: $x^t \in \mathbb{R}^{H \times W \times C}$.

For every time frame t, we learn to generate a set of S tokens, $Z^t = \{z_i^t\}_{i=1}^S$, from the input frame x^t. Specifically, we formulate a tokenizer function, $z_i = A_i(x)$, which maps the input frame x to a token vector z_i: $\mathbb{R}^{H \times W \times C} \mapsto \mathbb{R}^C$. The idea is to learn our tokenizer function A_i to adaptively select an informative combination of pixels (or spatial locations) in x^t. This way, our tokens will not be fixed splits of the input tensor, but a set of adaptively changing spatial selections. Different tokens will be mined per frame, allowing us to model their space-time relations/interactions. We also set S to be smaller than $H \times W$ (e.g., $S = 8$ and $H \times W = 8 \times 8$ or 16×16), enabling the model to easily consider more frames temporally.

Here, our tokenizer $z_i = A_i(x)$ is implemented with a spatial attention mechanism: i.e., the model learns to compute a weight map (of size $H \times W$) conditioned on the input x. The attention map, which is a function of the input itself, is applied to the input to generate the token:

$$z_i = A_i(x) = W_i \odot x = \alpha_i(x) \odot x, \tag{1}$$

where $\odot$ is the Hadamard product (i.e., element-wise multiplication) and $\alpha_i(\cdot)$ is implemented with a single convolutional layer (or a MLP layer) followed by a sigmoid function. It has a form of an element-wise spatial self-attention. Finally, spatial global average pooling is applied on top of them to reduce the dimensionality to $\mathbb{R}^C$.

The set of tokens are finally collected as: $Z = \bigcup_t \{Z^t\}$, combining a set of S tokens found for each frame. $|Z| = ST$.

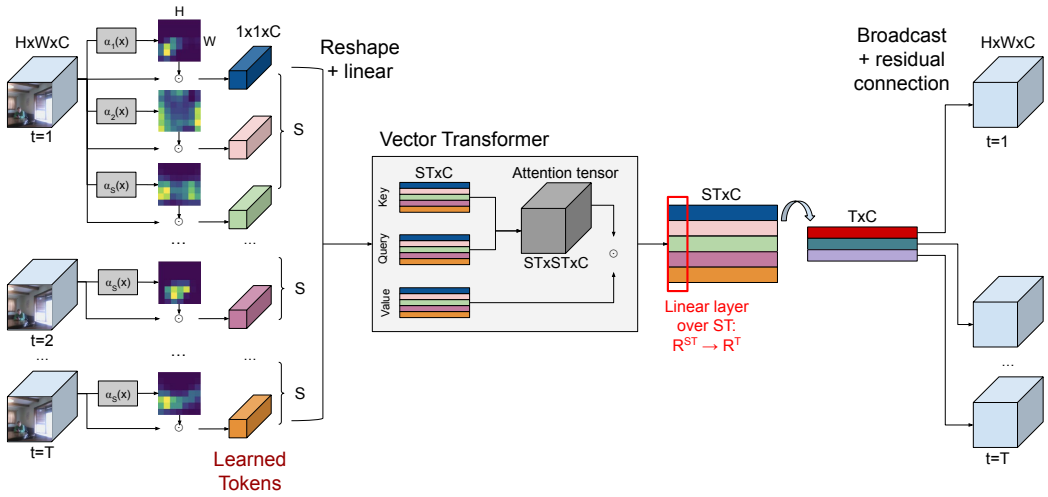

Figure 1: Visual illustration of the TokenLearner. The TokenLearner module first learns to generate a set of token vectors, models their space-time relations with a vector transformer, and fuses them. S is the number of tokens we learn per frame, and T is the number of frames. C is the number of channels, which we made to be identical across the module for efficiency. Note that our tokenization layer could also be combined with any other existing layers (e.g., replacing the vector transformer with a standard multi-head self-attention transformer).

2.2 Pairwise vector attention

Given Z, a set of tokens reflecting different space-time aspects of a video, we model space-time interactions between them with a Transformer mechanism [34]. We follow the formulation of [41], which enables a vector-version of the Transformer, although it is also possible to incorporate other attention mechanisms like multi-head and dot-product attention.

For every token z_i, the output of the Transformer y_i is computed by considering all possible z_j as:

$$y_i = \sum_{z_j \in Z} \gamma(f_q(z_i) \odot f_k(z_j)) \odot f_v(z_j) \tag{2}$$

where i and j are the indexes of the tokens in Z whose size is $|Z| = S$. f_q, f_k, and f_v are the linear layers projecting the tokens. γ is an extra projection layer to match the channel dimensionality followed by a softmax function over j. When the channel sizes of the projections are identical, γ is simplified as a single softmax layer identical to the standard transformer.

In the original transformer notation, the query matrix Q corresponds to our $\{f_q(z_i)\}_i$, and the key matrix K corresponds to our $\{f_k(z_j)\}_j$. Instead of computing the dot product between Q and K as QK^T to generate the attention 'matrix', this vector formulation computes an attention 'tensor' $\{\gamma(f_q(z_i) \odot f_k(z_j))\}_{(i,j)}$ preserving the channel information. It has shape $ST \times ST \times d$ where d is the intermediate channel size. The computed attention tensor is multiplied with the value matrix $\{f_v(z_j)\}_j$ to get the final transformer outputs.

Notice that this vector transformer is a global representation, and the temporal range of the information it is able to capture entirely depends on what tokens we provide to it. With our learnable adaptive tokens, we have the capability to cover a larger number of frames and focus on the temporal structure.

2.3 Token fusion and broadcasting

Next, we fuse information across our space-time tokens. This enables the model to capture spatio-temporal 'patterns' formulated by the tokens. Unlike the vanilla Transformer operation which learns to perform weighted summation of tokens based on pairwise relations, our token fusion component is designed to explicitly consider the patterns formulated by the space-time tokens.

Given a set of output tokens $Y = \{y_i\}_i$, we stack them to form a matrix of size $\mathbb{R}^{ST \times C}$, and apply a linear layer (i.e., a fully connected MLP layer) over the tokens, not channels. That is, we learn a linear

function of $\mathbb{R}^{ST} \mapsto \mathbb{R}^{T}$ where S is the number of our tokens mined per frame and T is temporal size of the input tensor, and apply it to every channel independently. If we denote each channel of y_i as $y_i[c]$, what we learn per frame t is a weighted summation of the tokens per channel:

$$o_t[c] = \sum_{y_i \in Y} w_i[c] \cdot y_i[c] \tag{3}$$

We believe this also has a connection to the observations from the concurrent work, MLPMixer [31].

Once we have the vector representation computed per time step by combining tokens, o_t, we broadcast it to every spatial location (i.e., pixels) of the input tensor and add it with the residual connection from the input.

3 Experiments

3.1 Network Architecture Implementation

We follow the Bottleneck Transformer [30] network style, while taking advantage of X3D [10] as the backbone.

Specifically, we modified X3D to be more computationally efficient by (1) replacing its 3D XYT convolutional layers with a pair of 2D conv. layer and 1D conv. layer, and (2) removing Squeeze-and-Excitation layers [17] and swish activations. Our backbone could be viewed as X(2+1)D. We use the channel sizes and the number of layers identical to X3D-M, which is an efficient model.

Based on such X(2+1)D architecture, and following the Bottleneck Transformer concept, we replace the space-time convolution layers in the last block with our transformers. Figure 2 illustrates the residual module architecture, which is repeated multiple times in the block. We have tried different versions, and our final model is built by replacing 1D temporal convolution with our TokenLearner while keeping the 2D 3×3 convolution layer in the X(2+1)D modules. Compared to the use of the standard 1D conv., the TokenLearner actually has 3.1 less GFLOPS (as also shown in Table 3) in the 64 frame setting.

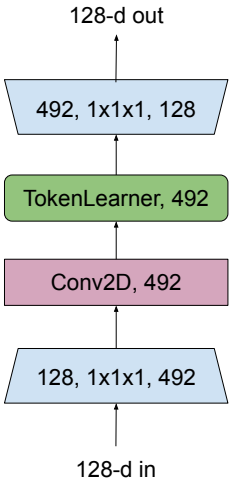

Figure 2: Our network module following the bottleneck transformer, with X(2+1)D backbone. It is an inverted bottleneck.

We provide $224 \times 224 \times 64$ videos for training and $256 \times 256 \times 64$ videos for testing. After the 3rd residual block, the input tensor has shape 8x8x64, and this becomes the input to the TokenLearner. For an efficient implementation the intermediate channel size of TokenLearner was set identical to the output channel size, $d = 432$. Notice that 64 frames were used to best capture longer-term temporal information. $S = 8$ number of tokens were used.

The training details are described in the Appendix.

3.2 Datasets

The datasets used in our experiments vary in their properties.

Charades dataset: The Charades dataset [28] is a dataset collected by assigning activity tasks which people in various environments are acting out, for example by performing a sequence of actions which involve interaction with objects. For example, sitting on the couch and reading a book, closing the book, standing up and speaking on the phone. It comprises 8000 training and 1686 validation videos with an average duration of 30 seconds. It has 157 activity classes. This dataset is very challenging as it is a multi-class, multi-label video dataset, that is, more than one activity can occur at the same time, and it includes fine grained motions or interactions with small objects in real-world environments. We follow the standard evaluation protocols, reporting the mean Average Precision (mAP) % (v1 classification setting of the dataset). We used the frame rate of 6 fps to obtain the training/testing videos. The dataset has a Non-Commercial Use license.

AViD dataset: The Anonymized Videos from Diverse countries (AViD) dataset [24] is a unique dataset which is representative of the world's population video content generation. It is collected

Table 1: Performance on the Charades multi-label classification task. Performance is measured using the Mean Average Precision (mAP) since more than one ground truth action is possible. Methods with RGB and optical flow input modalities are listed.

Method	Input	Pre-train	mAP
I3D [5]	RGB	Kinetics	32.9
I3D from [35]	RGB	Kinetics	35.5
I3D + Non-local [35]	RGB	Kinetics	37.5
EvaNet [23]	RGB	Kinetics	38.1
STRG [36]	RGB	Kinetics	39.7
LFB-101 [39]	RGB	Kinetics	42.5
SGFB-101 [18]	RGB	Kinetics	44.3
SlowFast-101 [11]	RGB+RGB	Kinetics	45.2
AssembleNet-50 [27]	RGB+Flow	None	47.0
AssembleNet-101 [27]	RGB+Flow	Kinetics	58.6
AssembleNet++ [26] (w/o object)	RGB+Flow	None	55.0
Backbone (X(2+1)D-M)	RGB	None	56.6
X(2+1)D-M w/ Space+Time transformer from [3]	RGB	None	58.5
Ours	RGB	None	**59.6**

Table 2: Performance on the Anonymized Videos from Diverse countries (AViD) dataset. Performance in terms of mean accuracy is shown in % averaged over 887 classes. Previous approaches results are reported from [24], all based on training from scratch with RGB-only inputs.

Method	Accuracy
I3D [5]	46.5
(2+1)D ResNet-50	46.7
3D ResNet-50	47.9
SlowFast-50 4x4 [11]	48.5
SlowFast-50 8x8 [11]	50.2
SlowFast-101 16x4 [11]	50.8
Backbone (X(2+1)D-M)	48.6
X(2+1)D-M w/ Space+Time transformer from [3]	50.1
Ours	**53.6**

from videos uploaded from multiple countries across six continents and demonstrates higher diversity compared to other video datasets in its concepts, actions and visual representations. For example a 'greeting' in certain countries involves a handshake, in some a kiss, but in others a slight bow. The dataset is explicitly designed to contain less bias, encourage diversity, while respecting privacy and licenses. The AViD dataset contains 887 classes and 450k videos (410k training 40k testing) and is of comparable size to Kinetics-400 and Kinetics-600 datasets with 400 and 600 classes respectively, also containing variable duration videos $3 - 15s$. We report classification accuracy over the 887 classes.

All the videos in the dataset have the Creative Commons License.

3.3 Results

Charades dataset results: In Table 1 we compare the proposed TokenLearner to the state-of-the-art methods. Our approach outperforms these, including recent work of which is most aligned to ours. While reporting the best accuracies of the approaches in Table 1, we also measured mean and variance of our proposed approach: over the 5 runs with random initialization the mean was 59.48 and the standard deviation was 0.15.

Anonymized Videos from Diverse countries (AViD) results: Table 2 shows the results on the AViD dataset. As seen, our approach outperforms prior work on this challenging dataset too. We also compared ours to the reimplementation of TimeSformer module [3] applied to the same backbone as ours. This uses disjoint spatial and temporal transformer modules, which was also introduced in [2].

Table 3: Comparison between different space-time transformer modules. They were all applied to the same backbone architecture (i.e., the Bottleneck Transformer-style with X(2+1)D). The Charades dataset is used in this experiment. FLOPS are estimated with 64-frame settings.

Module	Accuracy (%)	GFLOPs	# params
Conv2D + Conv1D	56.6	18.3	2.24M
Conv2D + MLPMixer [31]	57.0	13.8	2.06M
Conv2D + Temporal transformer	58.4	20.3	2.43M
Spatial + Temporal transformer	58.5	13.2	1.49M
Temporal + Spatial transformer	54.9	13.2	1.49M
Ours (SpatialT + TokenLearner)	58.7	10.0	1.56M
Ours (Conv2D + TokenLearner)	59.6	17.2	2.49M

Table 4: Comparing different components of our TokenLearner. Charades dataset.

Module	Accuracy (%)
Standard transformer (MHSA)	58.4
Vector transformer (VectT)	58.1
Prior-only-attention + broadcasting	58.6
Vector transformer (VectT) + broadcasting	58.9
Vector transformer (VectT) + token fusion	59.0
Token learning + MHSA + fusion	59.0
Token learning + VectT + fusion	59.6

3.4 Ablations

Comparison between multiple space-time transformers. As also suggested in previous literature, it is a common strategy for video representations to pair a layer focusing on spatial information with a layer focusing on temporal information (e.g., R(2+1)D [33] and TimeSformer [3]). Table 3 shows the results of this ablation. For spatial and temporal transformer implementations, we followed the multi-head self-attention (MHSA) of the original transformer [34], as was done in [2, 3]. As seen the proposed TokenLearner is more accurate than other popular combinations. The TokenLearner also effectively only uses a fraction of the Tokens per frame (i.e., 8) as opposed to other methods which use 16×16 or 32×32 tokens.

We also report the amount of computation and the number of parameters of these models. This depends on the input size and the hyper parameter setting, and our measurement is based on the input size (i.e., $H \times W \times T \times C$) of $8 \times 8 \times 64 \times 492$.

One of the main benefits of the TokenLearner (in addition to the adaptive tokenization of the input and that we explicitly fuse the tokens to capture their spatio-temporal patterns) is that, unlike the disjoint space/time transformers used in this ablation study, it is a joint space-time transformer. Simultaneously, it still manages its computation to be much more tractable (as shown in Table 3): A naive full version of the space-time transformer would require consideration of $8 \times 8 \times 64 = 4096$ tokens in our case, building and multiply the attention tensor of size $4096 \times 4096 \times 492$. On the other hand, the TokenLearner learns to consider $8 \times 64 = 512$ tokens jointly.

Different components. We did an ablation to evaluate components of our adaptive token transformer and their combinations. We conducted ablations removing/adding different components of our model. In addition to the vector transformer described in Section 2, we also tried an ablation of replacing it with the multi-head self-attention. Table 4 shows the results, demonstrating the benefits each of the elements bring to the approach. For this experiment, we used the module composed of Conv2D + transformer, which we found to perform the best from the above ablation.

TokenLearner alternatives. We also compared our spatial attention-based token learning with alternative approaches: (1) using a fixed grid to split each frame into the same number of tokens (i.e., 8 tokens) and (2) the approach of directly generating tokens using a fully connected layer. In the second approach, we directly model $z_i = A_i(x)$ as a dense layer, producing $H \times S \times C$ tensor based on the $H \times W \times T \times C$ input. The fixed split tokenization method provided us the accuracy of 58.8

on Charades, as opposed to 59.6 of ours. The direct token generation method provided the accuracy of 56.6 on Charades, failing to obtain better tokens. These results suggest the importance of spatial attention for the token learning. The same vector transformer and token fusion (from Section 2) were used for this ablation.

3.5 Visualizations

Figure 3 shows visualizations of the tokens being learned with our approach. We show the spatial attention maps (i.e., $\alpha_i(x)$) from the first TokenLearner module, as the inputs to the higher-level TokenLearner becomes more mixed spatially and temporally. We are able to observe that they tend to focus more on human regions, and that they change over time responding to the changes in the visual input. Among the $S = 8$ tokens per frame we learn, we visualize 4 of them.

4 Related work

Video understanding relies on both the spatial and the temporal information in the video. In order to adequately capture both motion and appearance information in videos, full 3D space-time convolutional layers as well as (2+1)D convolutional layers have been used [32, 5, 33, 40]. More advanced network designs have also been extremely popular in video CNNs particularly two-stream ones [29, 12, 13, 14, 7, 11] and, recently, architecture searched ones [10, 27, 23].

Attention-based architectures, e.g., the Transformer [34] have shown remarkable success in both Natural Language processing (NLP) and computer vision. Most adaptations of the Transformer architectures to computer vision, have been slow, although some optimizations, have been successful e.g., for image classification, [4, 41, 6, 25] and for video generation [38].

Applying attention-based architectures to video presents a definite challenge as the model needs to learn dependencies across both the spatial and temporal domains. The Vision Transformer [8] demonstrated how the NLP-specific Transformer architecture can elegantly work for images, by subdividing the input image into non-overlapping patches on a regular grid and feeding them as token embeddings to the Trasnformer, where $O(N^2)$ tokens are used or order of 256 or 1024. [15] relied on the region proposal network to use the detected human and object candidates as tokens, showing that it could be combined with CNNs.

A couple of recent (unpublished) work [2, 3], in the spirit of the Visual Transformer, subdivided the video into token in a 3D grid to capture the video input. This leads to $O(N^3)$ increase in the number of tokens required for learning (typically ~ 25k tokens for 96-frame model). Our work, in contrast, learns the tokens from data which results in a significantly fewer tokens, and more efficient approach. We see that even 8x times fewer tokens (e.g., 512 vs 4096), when learned, are able to capture successfully the information needed for video representation learning.

5 Conclusions

We have presented TokenLearner, a novel approach for space-time representation learning in videos, which adaptively tokenizes the input space. The goal is to learn the important tokens in video frames for the recognition tasks at hand. Our approach is more efficient, than contemporary work, by finding few important space-time tokens which can model interactions within videos. We observe improved accuracies across challenging video understanding tasks. One of the remaining challenges is in learning full spatio-temporal tokens. The current TokenLearner focuses on finding spatial tokens over a sequence of frames, and it could be extended to mine tokens over space-time volumes.

Broader Impact This paper presented a novel approach for space-time representation learning in videos, and it thus has applications in various video recognition systems.

Since video recognition systems are a generic technology with a wide range of potential applications, we cannot be aware of all potential applications. We are, however, conscious that each application has its own merits and societal impacts depending on the intentions of the individuals building and using the system. Finally, we note that training datasets may contain biases that may render the models trained on them unsuitable for certain applications. For example, it is possible that video recognition models may be used to make decisions that impact different groups in society differently.

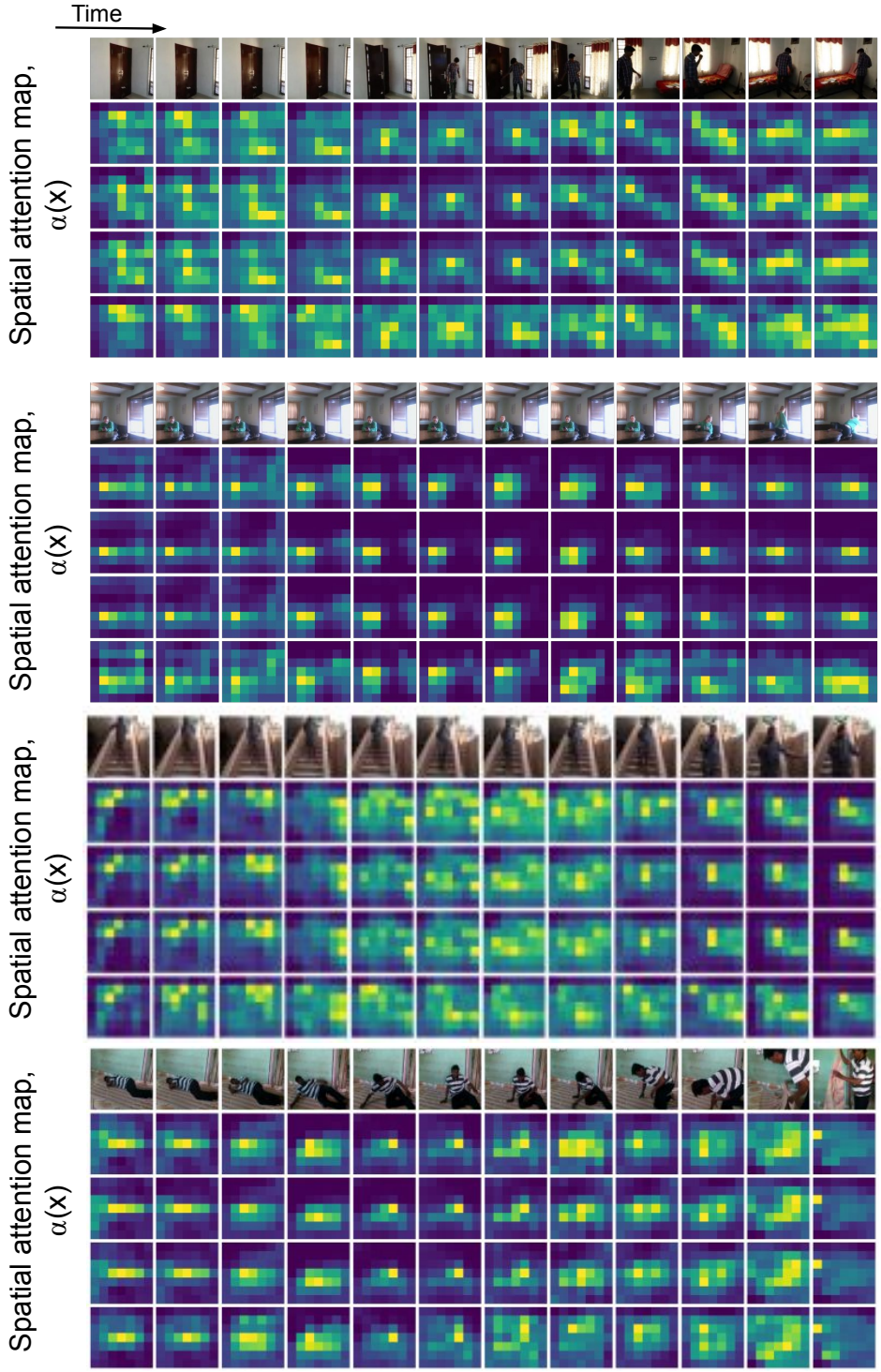

Figure 3: Visualization of the spatial attention maps for the tokenizations. Attention maps for four among a total of eight learned tokens are shown.

References

[1] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. D. Fauw, L. Smaira, S. Dieleman, and A. Zisserman. Self-supervised multimodal versatile networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021.

[3] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] J. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations (ICLR)*, 2020.

[7] A. Diba, M. Fayyaz, V. Sharma, M. Paluri, J. Gall, , R. Stiefelhagen, and L. V. Gool. Holistic large scale video understanding. *arXiv preprint arXiv:1904.114511*, 2019.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. Activitynet: A large-scale video benchma rk for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.

[10] C. Feichtenhofer. X3D: expanding architectures for efficient video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[12] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3468–3476, 2016.

[13] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4768–4777, 2017.

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[15] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the ICCV Workshop on Action, Gesture, and Emotion Recognition*, volume 2, page 4, 2017.

[17] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles. Action genome: Actions as composition of spatio-temporal scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[20] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[21] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfruend, C. Vondrick, et al. Moments in time dataset: one million videos for event understanding. *arXiv preprint arXiv:1801.03150*, 2018.

[22] A. Piergiovanni, A. Angelova, and M. S. Ryoo. Evolving losses for unsupervised video representation learning. 2020.

[23] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo. Evolving space-time neural architectures for videos. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[24] A. Piergiovanni and M. S. Ryoo. AViD dataset: Anonymized videos from diverse countries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[25] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[26] M. S. Ryoo, A. Piergiovanni, J. Kangaspunta, and A. Angelova. AssembleNet++: Assembling modality representations via attention connections. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[27] M. S. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova. AssembleNet: Searching for multi-stream neural connectivity in video architectures. In *International Conference on Learning Representations (ICLR)*, 2020.

[28] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[29] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 568–576, 2014.

[30] A. Srinivas, T. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[31] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[32] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3d: generic features for video analysis. *CoRR, abs/1412.0767*, 2(7):8, 2014.

[33] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[35] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[36] X. Wang and A. Gupta. Videos as space-time region graphs. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 399–417, 2018.

[37] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[38] D. Weissenborn, O. Täckström, and J. Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations (ICLR)*, 2020.

[39] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick. Long-term feature banks for detailed video understanding. *arXiv preprint arXiv:1812.05038*, 2018.

[40] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[41] H. Zhao, J. Jia, and V. Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Added the discussion in the conclusion
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We do not see any negative societal impact. We tried our best to use unbiased datasets.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We plan to release the code together with the final release of the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] They are included in Appendix in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We measured the mean and the standard deviation of our approach and reported them.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] They are included in Appendix in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]