Temporal-Relational CrossTransformers for Few-Shot Action Recognition

T. Perrett, A. Masullo, T. Burghardt, M. Mirmehdi, D. Damen
University of Bristol
CVPR 2021

CV/DL reading group on Transformers
April 6th 2021
Sofia Broomé

In a nutshell

- Few-shot action recognition approach building on tuple matching between query and support set
  - Constructs query-specific class prototypes using CrossTransformer method applied to video frames instead of image parts
  - Main result: SotA on action recognition few-shot benchmarks (Kinetics, Something-something, HMDB51 and UCF101)

Few-shot learning (action recognition setting) preliminaries

- You have an (action recognition) dataset with, say, 100 classes
- Train on e.g. 64 of those, validate on 12, test on 24 (i.e. the classes are new and unseen at test-time)
- For each class you have a small labeled support set (K videos, e.g. 5)
- Given a query video, match it to one of the classes
- Overall goal: Train the system to quickly (with few attempts) learn how to match a query to a support class
- Compute query-class distances and pass through cross-entropy loss

- During training, sample X tasks (64 choose 5 = 7M, 24 choose 5 =42k),
  In this paper: X=10 000. Average the results at test time

What are CrossTransformers?

- ”*CrossTransformers: spatially-aware few-shot transfer*”
  by C. Doersch, A. Gupta and A. Zisserman (NeurIPS 2020)

- Idea: Transformer attention **between** spatial locations in query image and spatial locations in support set images

  (hence **cross (?)**)

What are **Temporal-Relational** CrossTransformers?

- Apply CrossTransformer idea to video frames instead
- Sample *tuples* from query video and from all videos in support set, compute attention between these

- Flexible-length ω for tuples (although only 8 frames are sampled from videos here), so $\omega \leq 8$

- Different ω can be combined into Ω set for final classification
- One Transformer per ω
- $\Omega = \{2, 3\}$ works best in this paper

How is this done concretely? For ω = 2

- Query representation Q_p for pair p = (p_1, p_2)

$$Q_p = [\Phi(q_{p_1}) + \mathbf{PE}(p_1), \Phi(q_{p_2}) + \mathbf{PE}(p_2)] \in \mathbb{R}^{2\times D}$$

- One pair m= (m_1, m_2) from support set representation, similarly:

$$S^c_{km} = [\Phi(s^c_{km_1}) + \mathbf{PE}(m_1), \Phi(s^c_{km_2}) + \mathbf{PE}(m_2)] \in \mathbb{R}^{2\times D}$$

Φ is a CNN, c is the class, k is the video

Whole support set for one class:

$$\mathbf{S}^c = \{S^c_{km} : (1 \leq k \leq K) \land (m \in \Pi)\}$$

How is this done concretely? For ω = 2

- Will use these representations in a Transformer query/key/value setting

- To compute query-specific prototypes for each class $\mathbf{t}_p^c$

How is this done concretely? For ω = 2

- Calculate query-specific class prototypes (next slide), $\mathbf{t}_p^c$
  and then: distance between prototype and query representation (u_p)

$$T(Q_p, \mathbf{S}^c) = \|\mathbf{t}_p^c - \mathbf{u}_p\|$$

- Consider multiple pairs from query video, take average distance

$$\mathbf{Q} = \{Q_p : p \in \Pi\}$$

$$T(\mathbf{Q}, \mathbf{S}^c) = \frac{1}{|\Pi|} \sum_{p \in \Pi} T(Q_p, \mathbf{S}^c)$$

Scaled dot-product attention to obtain $\mathbf{t}_p^c$

- Query Υ, key Γ and value Λ linear maps (shared across classes)
  Also query Υ and key Γ are always shared within a class

$$\Upsilon, \Gamma : \mathbb{R}^{2\times D} \mapsto \mathbb{R}^{d_k} \quad \text{and} \quad \Lambda : \mathbb{R}^{2\times D} \mapsto \mathbb{R}^{d_v}$$

- Correspondence between query p and support set tuple km, dot product between key and query:

$$a_{kmp}^c = (\Gamma \cdot L(S_{km}^c)) \cdot (\Upsilon \cdot L(Q_p))$$

L is layer normalization.

- Finally, scaling and softmax:

$$\tilde{a}_{kmp}^c = \frac{\exp(a_{kmp}^c)/\sqrt{d_k}}{\sum_{l,n} \exp(a_{lnp}^c)/\sqrt{d_k}}$$

Scaled dot-product attention to obtain $\mathbf{t}_p^c$ continued.

Apply same value-transform Λ to query and support set tuple

$$\Upsilon, \Gamma : \mathbb{R}^{2\times D} \mapsto \mathbb{R}^{d_k} \quad \text{and} \quad \Lambda : \mathbb{R}^{2\times D} \mapsto \mathbb{R}^{d_v}$$

$$\mathbf{v}_{km}^c = \Lambda \cdot S_{km}^c$$

$$\mathbf{t}_p^c = \sum_{km} \tilde{a}_{kmp}^c \mathbf{v}_{km}^c$$

$$\mathbf{u}_p = \Lambda \cdot Q_p$$

$$T(Q_p, \mathbf{S}^c) = \|\mathbf{t}_p^c - \mathbf{u}_p\|$$

Final version for all ”cardinalities”: T^{Ω}

- Each TRX T^{ω} includes query, key and value linear maps corresponding to the dimensionality of ω

$$\Upsilon^{\omega}, \Gamma^{\omega} : \mathbb{R}^{\omega \times D} \mapsto \mathbb{R}^{d_k} \quad \textbf{and} \quad \Lambda^{\omega} : \mathbb{R}^{\omega \times D} \mapsto \mathbb{R}^{d_v}$$

- Accumulate distances from the different TRXs:

$$\mathbf{T}^{\Omega}(Q, \mathbf{S}^{c}) = \sum_{\omega \in \Omega} T^{\omega}(\mathbf{Q}^{\omega}, \mathbf{S}^{c\omega})$$

- Query is assigned to class with lowest $\mathbf{T}^{\Omega}$

Experiments

- **Flagship result:** State-of-the-art on 5-way, 5-shot benchmarks

Method	Kinetics	SSv2†	SSv2*	HMDB	UCF
CMN [31]	78.9	-	-	-	-
CMN-J [32]	78.9	48.8	-	-	-
TARN [3]	78.5	-	-	-	-
ARN [27]	82.4	-	-	60.6	83.1
OTAM [4]	85.8	-	52.3	-	-
TRX (Ours)	**85.9**	**59.1**	**64.6**	**75.6**	**96.1**

Table 1: Results on 5-way 5-shot benchmarks of Kinetics (split from [32]), SSv2 (†: split from [32], *: split from [4]), HMDB51 and UCF101 (both splits from [27]).

Experiments

- TRX with different **Ω** values

Cardinalities	Num tuples	Kinetics	SSv2†
Ω={1}	-	85.2	53.3
Ω={2}	28	85.0	57.8
Ω={3}	56	85.6	58.8
Ω={4}	70	84.5	58.9
Ω={2, 3}	84	**85.9**	**59.1**
Ω={2, 4}	98	84.4	58.4
Ω={3, 4}	126	85.3	**59.1**
Ω={2, 3, 4}	154	85.3	58.9

Table 2: Comparing all values of Ω for TRX, noting the number of tuples for each model, given by $\sum_{\omega \in \Omega} |\Pi^{\omega}|$.

- Skipping remaining experiments since they mostly are there to ensure that the method is behaving as desired

Conclusion for CV/DL group

- This was another example of how to apply Transformers-principles in a vision setting
- Videos are represented by ordered tuples of frames at different positions, allowing flexibility in what speeds and temporal offsets we match query with support set
- Quite thorough experimentation shows that the method seems to be doing what it should (qualitative examples, uses multiple support videos, etc)

Discussion points

- Thorough paper, following standard recipe for a CV-paper (currently popular task, slightly new idea, *extensive experimentation* with ablations, beats SotA, well-written)

- ”Temporality”: Representing videos, first by only 8 frames, and then with pairs and triplets, with frame-wise down-sampling

- Toy-flavor of few-shot task and datasets?
  - A robot that encounters unseen objects. How will it find 5 labels?
    (Usually N is small in N-way k-shot classification)
  - Videos are short snippets (temporally trimmed)

- Trained end-to-end with CNN. Missing fixed backbone ablation?

Claim

- Section 4.3.5:
  "Importantly, the method's runtime scales linearly with the number of frames."

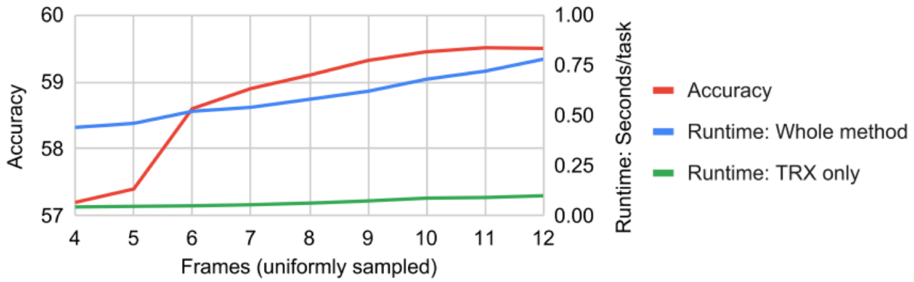
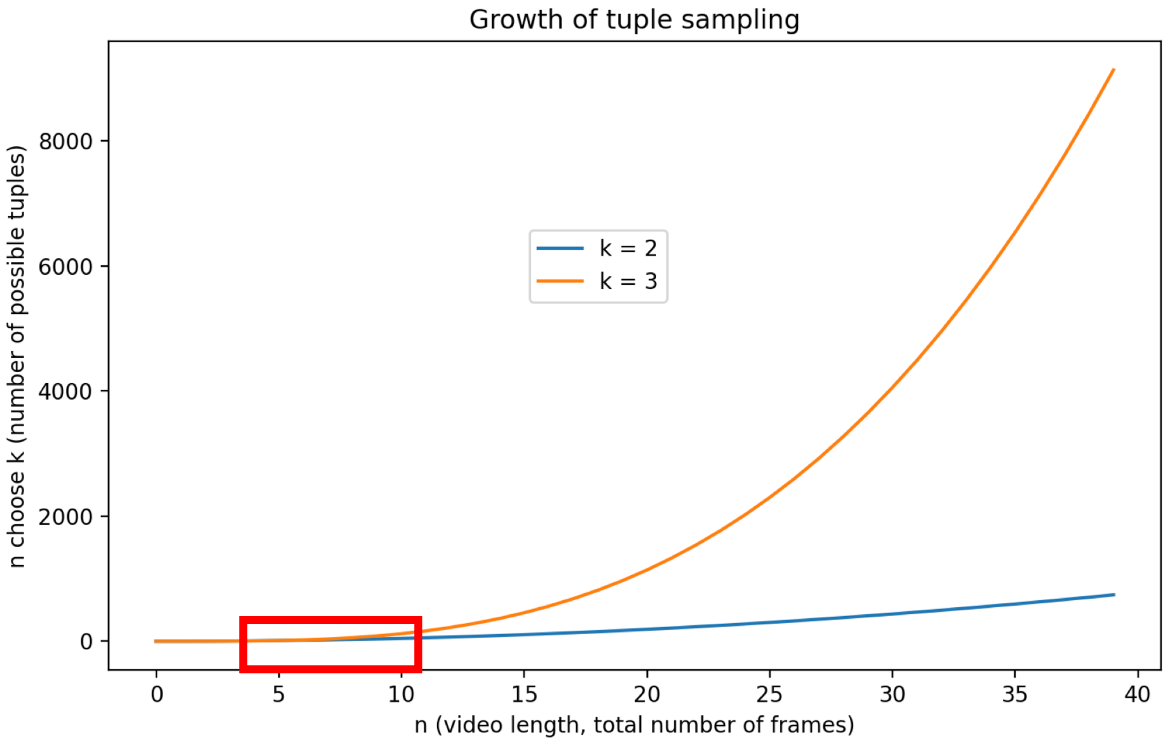
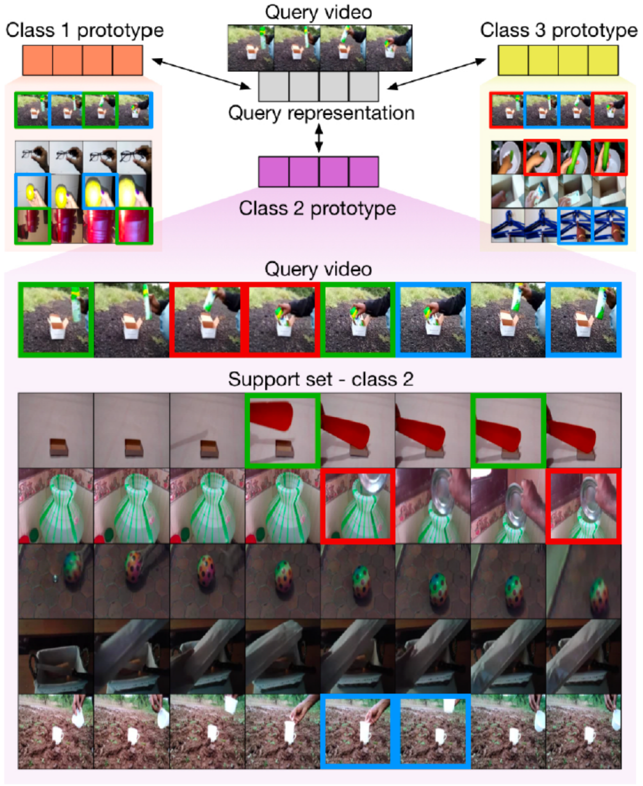


Figure 8: SSv2[†] accuracy (left y-axis) vs runtime analysis (right y-axis in seconds/task) for TRX $\Omega = \{2, 3\}$ as the number of sampled frames varies from 4 to 12 frames.





More discussion?