
Peer reviewed version

Link to publication record in Explore Bristol Research
PDF-document

This is the author accepted manuscript (AAM). The final published version (version of record) is available online via [insert publisher name] at [insert hyperlink] . Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available: http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/
Temporal-Relational CrossTransformers for Few-Shot Action Recognition

Toby Perrett  Alessandro Masullo  Tilo Burghardt  Majid Mirmehdi  Dima Damen  
<first>.<last>@bristol.ac.uk  Department of Computer Science, University of Bristol, UK

Abstract

We propose a novel approach to few-shot action recognition, finding temporally-corresponding frame tuples between the query and videos in the support set. Distinct from previous few-shot works, we construct class prototypes using the CrossTransformer attention mechanism to observe relevant sub-sequences of all support videos, rather than using class averages or single best matches. Video representations are formed from ordered tuples of varying numbers of frames, which allows sub-sequences of actions at different speeds and temporal offsets to be compared.

Our proposed Temporal-Relational CrossTransformers (TRX) achieve state-of-the-art results on few-shot splits of Kinetics, Something-Something V2 (SSv2), HMDB51 and UCF101. Importantly, our method outperforms prior work on SSv2 by a wide margin (12%) due to its ability to model temporal relations. A detailed ablation showcases the importance of matching to multiple support set videos and learning higher-order relational CrossTransformers.

1. Introduction

Few-shot methods aim to learn new classes with only a handful of labelled examples. Success in few-shot approaches for image classification [11, 19, 8] and object recognition [26, 15] has triggered recent progress in few-shot video action recognition [31, 32, 3, 27, 4]. This is of particular interest for fine-grained actions where collecting enough labelled examples proves challenging [5, 12, 6].

Recent approaches that achieve state-of-the-art performance [3, 27, 4] acknowledge the additional challenges in few-shot video recognition, due to varying action lengths and temporal dependencies. However, these match the query video (i.e. the video to be recognised) to the single best video in the support set (i.e. the few labelled examples per class), e.g. [27], or to the average across all support set videos belonging to the same class [3, 4]. Inspired by part-based few-shot image classification [8], we consider that, within a few-shot regime, it is advantageous to compare sub-sequences of the query video to sub-sequences of all support videos when constructing class prototypes. This better accumulates evidence, by matching sub-sequences at various temporal positions and shifts.

We propose a novel approach to few-shot action recognition, which we term Temporal-Relational CrossTransformers (TRX). A query-specific class prototype is constructed by using an attention mechanism to match each query sub-sequence against all sub-sequences in the support set, and aggregating this evidence. By performing the attention operation over temporally-ordered sub-sequences...
rather than individual frames (a concept similar to that in many-shot action-recognition works, e.g. [29, 10]), we are better able to match actions performed at different speeds and in different parts of videos, allowing distinction between fine-grained classes. Fig. 1 shows an example of how a query video attends to multiple support set videos using temporally-ordered tuples.

Our key contributions can be summarised as follows:

- We introduce a novel method, called the Temporal-Relational CrossTransformer (TRX), for few-shot action recognition.
- We combine multiple TRXs, each operating over a different number of frames, to exploit higher-ordered temporal relations (pairs, triples and quadruples).
- We achieve state-of-the-art results on the few-shot benchmarks for Kinetics [5], Something-Something V2 (SSv2) [12], HMDB51 [16] and UCF101 [21].
- We perform a detailed ablation, demonstrating how TRX utilises multiple videos from the support set, of different lengths and temporal shifts. Results show that using tuple representations improves over single-frames by 5.8% on SSv2 where temporal ordering proves critical.

2. Related Work

Few-shot classification methods have traditionally fallen into one of three categories - generative [28, 9], adaptation-based [11, 17] and metric-based [24, 20]. Generative methods use examples from the target task to generate additional task-specific training data with which to fine-tune a network. Adaptation-based methods (e.g. MAML [11]) aim to find a network initialisation which can be fine-tuned with little data to an unseen target task. Metric-based methods (e.g. Prototypical [20] or Matching [24] Networks) aim to find a fixed feature representation in which target tasks can be embedded and classified.

Recent works which perform well on few-shot image classification have found that it is preferable to use a combination of metric-based feature extraction/classification combined with task-specific adaptation [19, 2]. Most relevant to this paper, the recently introduced CrossTransformer [8] uses an attention mechanism to align the query and support set using image patch co-occurrences. This is used to create query-specific class prototypes before classification within a prototypical network [20]. Whilst this is effective for few-shot image classification, one potential weakness is that relative spatial information is not encoded. For example, it would not differentiate between a bicycle and a unicycle. This distinction is typically not needed in [8]'s tested datasets [22], where independent part-based matching is sufficient to distinguish between the classes.

Few-shot video action recognition methods have had success with a wide range of approaches, including memory networks of key frame representations [31, 32] and adversarial video-level feature generation [9]. Recent works have attempted to make use of temporal information. Notably, [3] aligns variable length query and support videos before calculating the similarity between the query and support set. [27] combines a variety of techniques, including spatial and temporal attention to enrich representations, and jigsaws for self-supervision. [4] achieves state-of-the-art performance by calculating query to support-set frame similarities. They then enforce temporal consistency between a pair of videos by monotonic temporal ordering. Their method can be thought of as a differentiable generalisation of dynamic time warping. Note that the above works either search for the single support video [27] or average representation of a support class [3, 4] that the query is closest to. A concurrent work to ours attempts to resolve this through query-centred learning [33]. Importantly, all prior works perform attention operations on a frame level, as they tend to use single-frame representations.

Compared to all prior few-shot action recognition methods, our proposed method attends to all support set videos, using temporal-relational representations from ordered tuples of frames, sampled from the video. By attending to sub-sequences, our method matches actions at different speeds and temporal shifts. Importantly, we use a combination of different CrossTransformers to match tuples of different cardinalities, allowing for higher-order temporal representations. We next describe our method in detail.

3. Method

We propose a method for few-shot action recognition that considers the similarity between an ordered subsequence of frames (referred to as a tuple) to all subsequences in the support set, through multiple CrossTransformer attention modules. This allows the same query video to match to tuples from several support set videos. After stating the problem definition in Section 3.1, for ease of understanding our TRX method, we start from a simplified version, building in complexity and generality up to the full method, which is illustrated in Fig. 2.

In Section 3.2, we consider a single ordered pair of frames, sampled from the query video. We propose a temporal CrossTransformer to compare this query pair to ordered pairs of frames from videos in the support set. This allows the construction of ‘query pair’-specific class prototypes. We then expand to multiple ordered pairs of frames from the query video. Finally, in Section 3.3, motivated by the need to model more complex temporal relationships, we generalise from pairs to tuples. We model a separate temporal CrossTransformer for each tuple cardinality to construct query-cardinality-specific class prototypes. These are combined to classify the query video, based on the distances to all class prototypes.
and representations of the query and support set videos are constructed. These are concatenated representations of pairs/triplets of frames (sampled from the video), temporally-ordered. Two temporal CrossTransformers, \( T^2 \) and \( T^3 \), then construct separate class prototypes for each representation (pairs and triplets) using separate query \( \Upsilon \), key \( \Gamma \) and value \( \Lambda \) linear maps for each transformer. This produces a query-specific class prototype, one per CrossTransformer. The query representation is also passed through the value linear maps \( \Lambda \), and distances are calculated to the prototypes. The distances are averaged, and the query is recognised as belonging to the closest class. Details are in Section 3.

3.1. Problem Formulation

In few-shot video classification, inference aims to classify an unlabelled query video into one of several classes, each represented by a few labelled examples unseen in training, referred to as the ‘support set’. In this paper, we focus on \( K \)-shot where \( K > 1 \), i.e. the support set contains more than one video. Similar to prior works [24, 11, 4, 27], we follow episodic training, i.e. random sampling of few-shot tasks from the training set. For each episode, we consider a \( C \)-way \( K \)-shot classification problem. Let \( Q = \{q_1, \ldots, q_F\} \) be a query video with \( F \) uniformly sampled frames. The goal is to classify \( Q \) into one of the classes \( c \in C \). For the class \( c \), its support set \( S^c \) contains \( K \) videos, where the \( k \)th video is denoted \( S^c_k = \{s^c_{k1}, \ldots, s^c_{kF}\}^2 \).

3.2. Temporal CrossTransformer

We consider the temporal relation of two frames sampled from a video to represent the action, as actions are typically changes in appearance and are poorly represented by a single frame. We thus sample a pair of ordered frames from the query video with indices \( p = (p_1, p_2) \), where \( 1 \leq p_1 < p_2 \leq F \), and define the query representation as:

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

where \( \Phi : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{D} \) is a convolutional network to obtain a \( D \)-dimensional embedding of an input frame, and \( PE(\cdot) \) is a positional encoding given a frame index [23].

We compare the query representation \( Q_{p} \) to all possible pair representations from the support set videos, allowing it to match actions at various speeds and locations within the support set. We define the set of all possible pairs as

\[
\Pi = \{(n_1, n_2) \in \mathbb{N}^2 : 1 \leq n_1 < n_2 \leq F\}.
\]

A single frame-pair representation of video \( k \) in the support set of class \( c \) with respect to the ordered pair of indices \( m = (m_1, m_2) \in \Pi \) is

\[
S^c_{km} = [\Phi(s^c_{k_{m1}}) + PE(m_1), \Phi(s^c_{k_{m2}}) + PE(m_2)] \in \mathbb{R}^{2 \times D}.
\]

The set of all pair representations in the support set for class \( c \) is

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

We propose a temporal CrossTransformer \( T \), based on the spatial CrossTransformer [8], but adapted from image patches to frame pairs, to calculate query-specific class prototypes. The CrossTransformer includes query \( \Upsilon \), key \( \Gamma \) and value \( \Lambda \) linear maps, which are shared across classes:

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

The correspondence between the query pair and pair \( m \) of support video \( k \) in class \( c \) is calculated as

\[
a^c_{kmp} = (\Gamma \cdot L(S^c_{km}))(\Upsilon \cdot L(Q_p)),
\]

where \( L \) is a standard layer normalisation [1]. We apply the Softmax operation to acquire the attention map

\[
\tilde{a}^c_{kmp} = \frac{\exp(a^c_{kmp})}{\sqrt{d_k}}.
\]

This is then combined with value embeddings of the support set \( v^c_{km} = \Lambda \cdot S^c_{km} \), in order to compute the query-specific prototype with respect to the query \( Q_{p} \),

---

\( ^2 \) We assume videos are uniformly sampled to be of the same length \( F \) for simplicity. Alternatively, we could set \( F \) to be the maximum video length and non-existent pairs could be masked out in the attention matrix.
Now that we have a query-specific class prototype, we calculate the embedding of the query $Q_p$ with the value linear map such that $\mathbf{u}_p = \Lambda \cdot Q_p$. This ensures that the query and support representations undergo the same operations. The CrossTransformer $T$ computes the distance between the query and support set $S^c$ by passing $Q_p$ such that

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

(9)

Finding a single pair that best represents the action $Q$ is a difficult problem. Instead, we consider multiple pairs of frames from the query video, such that the query representation is defined as $Q = \{Q_p : p \in \Pi\}$. In Section 4.3.6, we compute exhaustive pairs to random pairs of frames.

To calculate the distance between $Q$ and $S^c$, we accumulate the distances from all query pairs, i.e.

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

(10)

During training, negative query-class distances $T$ are passed as logits to a cross-entropy loss. During inference, the query $Q_p$ is assigned the class of the closest query-specific prototype, i.e. $\text{arg min}_c T(Q_p, S^c)$. Note that the Softmax operation in Eq. 7 is performed separately for each query pair $Q_p$ (i.e. matches are scaled separately for each $p$). Our Temporal CrossTransformer thus accumulates matches between pair representations of the query video (hence the term temporal) and all pairs of frames in the support set.

### 3.3. Temporal-Relational CrossTransformers

A shortcoming of the above method is that an ordered pair of frames might not be the best representation of an action, particularly when fine-grained distinctions between the classes are required. Consider two classes: “picking an object up” vs “moving an object”. To discern between these two actions, a method would require at least three frames - i.e. whether the object is put down eventually or remains in hand. Similarly, consider full-body actions such as “jumping” vs “tumbling”. This highlights the need to explore higher-order temporal representations.

We extend the Temporal CrossTransformer to a Temporal-Relational CrossTransformer (TRX) by considering a sub-sequence of ordered frames of any length. We use $\omega$ to indicate the length, or cardinality, of a tuple. For example, $\omega=2$ for a pair, $\omega=3$ for a triple. We generalise to possible tuples for any $\omega$, such that

$$\Pi^\omega = \{(n_1, ..., n_\omega) \in \mathbb{N}^\omega : \forall i(1 \leq n_i < n_{i+1} \leq F)\}$$

(11)

The associated query representation with respect to the tuple with indices $p = (p_1, ..., p_\omega) \in \Pi^\omega$, generalising the pair representation in Eq. 1, is

$$Q_p^\omega = [\Phi(q_{p_1}) + \Lambda \cdot Q_p(p_1), ..., \Phi(q_{p_\omega}) + \Lambda \cdot Q_p(p_\omega)] \in \mathbb{R}^{\omega \times D}.$$  

(12)

This is done similarly for the support set representations.

We define the set of cardinalities as $\Omega$. For example, pairs, triples and quadruples of frames would be $\Omega=\{2, 3, 4\}$. We use one TRX per cardinality, as parameters can only be defined for a known input dimensionality (e.g. Eq. 5). Each TRX $T^\omega$ includes query, key and value linear maps corresponding to the dimensionality of $\omega$:

$$T^\omega, \Gamma^\omega : \mathbb{R}^{\omega \times D} \rightarrow \mathbb{R}^{d_k} \text{ and } \Lambda^\omega : \mathbb{R}^{\omega \times D} \rightarrow \mathbb{R}^{d_v}.$$  

(13)

Each $T^\omega$ outputs the distance between the query and support set with respect to tuples of cardinality $\omega$. We then accumulate distances from the various TRXs, such that:

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

(14)

Note that averaging the outputs from each TRX first (as in Eq. 10 for $\omega=2$) balances the varying number of tuples for each $\omega$. As with a single TRX, during training, the negative distance for each class is passed as the logit to a cross-entropy loss. During inference, the query is assigned the class which is closest to the query with respect to $T^\Omega$.

**Summary:** TRX in its complete form considers a set of cardinalities $\Omega$. For each $\omega \in \Omega$, different linear maps of corresponding dimensions are trained (Eq. 13). These are trained jointly using a single cross-entropy loss, that uses the summed distances (Eq. 14), where the gradient is backpropagated for each $\omega$ (Eq. 10). The gradient is accumulated from each TRX, through the tuple representations, and backpropagated through a convolutional network to update frame representations. TRX is trained end-to-end with shared backbone parameters for all $\omega \in \Omega$, and all tuples.

### 4. Experiments

#### 4.1. Setup

**Datasets.** We evaluate our method on four datasets. The first two are Kinetics [5] and Something-Something V2 (SSv2) [12], which have been frequently used to evaluate few-shot action recognition in previous works [31, 3, 27, 4]. SSv2, in particular, has been shown to require temporal reasoning (e.g. [30, 18, 14]). We use the few-shot splits for both datasets proposed by the authors of [31, 32] which are publicly accessible. In this setup, 100 videos from 100 classes are selected, with 64, 12 and 24 classes used for train/val/test. We also provide results for the few-shot split of SSv2 used by [4] which uses 10x more videos per class in the training set. Additionally, we evaluate our method on HMDB51 [16] and UCF101 [21], using splits from [27].

---

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Kinetics</th>
<th>SSv2( ^\dagger )</th>
<th>SSv2( ^\ast )</th>
<th>HMDB</th>
<th>UCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMN [31]</td>
<td>78.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMN-J [32]</td>
<td>78.9</td>
<td>48.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TARN [3]</td>
<td>78.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ARN [27]</td>
<td>82.4</td>
<td>-</td>
<td>60.6</td>
<td>83.1</td>
<td>-</td>
</tr>
<tr>
<td>OTAM [4]</td>
<td>85.8</td>
<td>-</td>
<td>52.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TRX (Ours)</td>
<td>85.9</td>
<td>59.1</td>
<td>64.6</td>
<td>75.6</td>
<td>96.1</td>
</tr>
</tbody>
</table>

**Evaluation.** TRX particularly benefits from the presence of a number of videos in the support set (i.e. few-shot rather than one-shot). We thus evaluate our method on the standard 5-way 5-shot benchmark, and report average results over 10,000 tasks randomly selected from the test sets. We provide an ablation on X-shot, including one-shot results in the ablation and appendices for completeness.

**Baselines.** We give comparisons against four seminal and recent works [31, 3, 27, 4], which reported state-of-the-art performance. We primarily use the large-scale datasets Kinetics and SSv2 because TRX is particularly beneficial when temporally ordered tuples can assist the discrimination between classes. It also outperforms prior work on HMDB51 and UCF101 by 15% and 13% respectively. On Kinetics, it exceeds the state-of-the-art (by 0.1%). Kinetics is more of an appearance-based dataset when used as a few-shot benchmark, where ordering is less important and single frame representation can be sufficient. We ablate this in Section 4.3.2.

Figure 3 shows qualitative results, highlighting tuple matches between the query and support set for \( \Omega=\{2, 3\} \). For each subfigure, we show query pairs (top) and triplets (bottom) with their corresponding tuples (same colour) in the support set. For example, the red pair of frames in the first example (frames 1 and 2) gets the maximum attention when compared to the second support set video (frames 2 and 3). We select three tuples to highlight in each case. The figure shows that tuples match to different videos in the support set, as well as tuples of varying positions and frame differences. A failure case (Fig. 3c) matches pair/triplet frames from the query “failing to put something into something because it doesn’t fit”, with pairs/triplets of the support set class “put something upright on the table”. In each example, the putting action is correctly matched, but the query is closest to the wrong prototype.

### 4.3. Ablations

Our motivation in proposing TRX is the importance of representing both the query and the support set videos by tuples of ordered frames, and that class prototypes should be constructed from multiple support set videos. We showcase this motivation experimentally through several ablations. We specifically evaluate: (4.3.1) the impact of \( \Omega \), (4.3.2) the importance of ordered frames in the tuple, (4.3.3) the importance of multiple videos in the support set, and (4.3.4) whether tuples at various locations and frame positions are being matched within TRX. Additionally, (4.3.5) we compare performance and runtime as the number of sampled frames changes, and (4.3.6) compare exhaustive to random tuples, showcasing the potential to compress TRX models without a significant drop in performance. We primarily use the large-scale datasets Kinetics and SSv2\( ^\dagger \) for these ablations using the splits from [31, 32].

#### 4.3.1 TRX with different \( \Omega \) values

In our comparative analysis in Tab. 1, we reported results using \( \Omega=\{2, 3\} \). This is the combined TRX of pair and triplet frames demonstrated in Fig. 2. We now evaluate each cardinality of \( \Omega \in \{1, 2, 3, 4\} \) independently as well as all their combinations on both datasets.

In Tab. 2, results demonstrate significant improvement in SSv2 moving from single frame comparisons to pair...
(a) SSv2†: Throwing something in the air and letting it fall.
(b) Kinetics: Cutting watermelon.
(c) SSv2† (False Positive): Query GT: Failing to put S into S because S does not fit, Support Set: putting something upright on the table.

Figure 3: Examples for TRX with \( \Omega = \{2, 3\} \). Colour-matching pairs (top) and triplets (bottom) are shown between the query and support set videos from one class. Three tuples are highlighted in each subfigure (red, green and blue). This figure demonstrates maximum attention matches to several videos in the support set, at different relative and absolute positions.

<table>
<thead>
<tr>
<th>Cardinalities</th>
<th>Num tuples</th>
<th>Kinetics</th>
<th>SSv2†</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Omega = {1} )</td>
<td>-</td>
<td>85.2</td>
<td>53.3</td>
</tr>
<tr>
<td>( \Omega = {2} )</td>
<td>28</td>
<td>85.0</td>
<td>58.8</td>
</tr>
<tr>
<td>( \Omega = {3} )</td>
<td>56</td>
<td>85.6</td>
<td>58.8</td>
</tr>
<tr>
<td>( \Omega = {4} )</td>
<td>70</td>
<td>84.5</td>
<td>59.9</td>
</tr>
<tr>
<td>( \Omega = {2, 3} )</td>
<td>84</td>
<td>85.9</td>
<td>59.1</td>
</tr>
<tr>
<td>( \Omega = {2, 4} )</td>
<td>98</td>
<td>84.4</td>
<td>58.4</td>
</tr>
<tr>
<td>( \Omega = {3, 4} )</td>
<td>126</td>
<td>85.3</td>
<td>59.1</td>
</tr>
<tr>
<td>( \Omega = {2, 3, 4} )</td>
<td>154</td>
<td>85.3</td>
<td>58.9</td>
</tr>
</tbody>
</table>

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

comparisons, of (+4.5%). The performance increases further for triplets (+1.0%) and only marginally again for quadruples (+0.1%). Combining two CrossTransformers \( \Omega = \{2, 3\} \) performs best. Using all cardinalities \( \Omega = \{2, 3, 4\} \) results in a slight drop in performance (-0.2%). Comparatively, differences are smaller on Kinetics, and moving to quadruples drops the performance significantly (-1.4%) compared to the best TRX combination \( \Omega = \{2, 3\} \).

<table>
<thead>
<tr>
<th>Method</th>
<th>Kinetics</th>
<th>SSv2†</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Omega = {2, 3} ) order reversed</td>
<td>85.9</td>
<td>51.3</td>
</tr>
<tr>
<td>( \Omega = {2, 3} )</td>
<td>85.9</td>
<td>59.1</td>
</tr>
</tbody>
</table>

Table 3: Results assess the importance of temporal ordering. When the tuple orders are reversed for the query video, a large drop is observed for SSv2†, but not for Kinetics.

The improvement using TRX with the multiple cardinalities \( \Omega = \{2, 3\} \) over frame-based comparisons (\( \Omega = \{1\} \)) is demonstrated per-class in Fig. 4. For SSv2, some classes see little improvement (e.g. “scooping something up with something”, “opening something”), whereas others see a greater than 10% improvement (e.g. “pretending to take something from somewhere”, “putting something next to something”). Aligned with the overall results on Kinetics, Fig. 4 shows modest improvements per-class, including marginal drop in some classes.

4.3.2 The impact of ordered tuples

Up to this point, we have made the assumption that tuples should be temporally ordered to best represent actions. We
evaluate the extreme scenario, where frames in the support set are temporally ordered, but frames in the query take the reverse order during inference only. Table 3 shows a large drop for the reversed query sets on SSv2 (-7.8%). Supporting our prior observation that it is more an appearance-based dataset, no drop is observed for Kinetics.

4.3.3 Matching to multiple support set videos

Our motivation for using CrossTransformers is that query tuples would match to tuples from multiple support set videos in order to create the query-specific class prototype. Note that this is not regularised during training - i.e. there is no encouragement to use more than one support set video.

Figure 5 ablates TRX (Ω={1}) and Kinetics. True/False Positive/Negative query percentages are shown for Ω={2} and Ω={4}.

4.3.4 Visualising tuple matches

In addition to matching multiple support set videos, we visualise the tuple matches between the queries and the support set. Given a query tuple (row) and a support set tuple (col), we sum the attention values over the test set, and then normalise per row. Fig. 7 shows the summed attention values between all sets of pairs. While query pairs match baseline as the number of shots increases. For our proposed model Ω={2, 3}, we report improvements of +3.9%, +7.3%, +7.9% and +10.3% for 2-, 3-, 4- and 5-shots comparatively. Note that using a single frame representation also improves over CMN, by a smaller but significant margin. This ablation showcases TRX’s ability to utilise tuples from the support set as the number of videos increases.

To analyse how many support videos are used, we train TRX with pairs (Ω={2}) and quadruples (Ω={4}) on SSv2 and Kinetics. For each query tuple, we find the support set tuple with the maximum attention value. We then count the number of support set videos per class which contain at least one maximal match, and average over all test tasks. Figure 6 presents the results for true and false, positive and negative, results. The figure demonstrates that TRX successfully matches the query to tuples from multiple videos in the support set. Most queries (> 50%) match to 2-3 videos in the support set. Very few queries match to all 5 videos in the support set, particularly for higher cardinality tuples. A similar distribution is seen for both datasets, however for SSv2, more true positive queries are matched to a single video in the support set.
frequently to corresponding support set pairs (i.e. same frame positions) in the support set, pairs are also matched to shifted locations (e.g. [1,2] with [2,4]) as well as significantly different frame distances (e.g. [6,7] with [1,7]).

4.3.5 Varying the number of frames

All previous results sample 8 frames from each video, in the query and support set. This allows us to directly compare to previous few-shot works that all consider 8 uniformly sampled frames [32, 3, 4]. To demonstrate TRX is scalable, we plot $\Omega=\{2,3\}$ results on SSv2 for the 5-way 5-shot task, sampling different numbers of frames (4-12), and compare the accuracy and runtime in Fig. 8. Accuracy is comparable for $\geq 6$ frames. Importantly, the method’s runtime scales linearly with the number of frames. The TRX component only contributes a margin of the runtime and memory requirements of the network, with the ResNet-50 backbone dominating both needs.

4.3.6 Random tuples in TRX

All the above experiments have used exhaustive sets of tuples, e.g. every possible pair $(n_1, n_2)$ such that $1 \leq n_1 < n_2 \leq F$ for $w = 2$. To explore the impact of randomly sampling tuples, we experiment with 20, 40, 60 and 80% of tuples retained for $\Omega=\{2\}$, $\{3\}$ and $\{4\}$, as well as a combined $\Omega=\{2,3\}$. We report four runs for each percentage, each with a different random selection of tuples.

Fig. 9 shows that while retaining all tuples gives the best performance, some of the runs produce results comparable to exhaustive tuple selections for $\Omega=\{2,3\}$ and even outperform these for $\Omega=\{4\}$. The performance degrades quicker for $\Omega=\{2\}$. The associated Tab. 4 compares the corresponding GPU usage. This shows it is possible to utilise fewer resources with comparable performance. A method for selecting tuples that maintain performance is left for future work.

5. Conclusion

This paper introduced Temporal-Relational CrossTransformers (TRX) for few-shot action recognition. TRX constructs query-specific class prototypes by comparing the query to sub-sequences of all support set videos. To model temporal relationships, videos are represented by ordered tuples of frames, which allows sub-sequences of actions at different speeds and temporal offsets to be compared. TRX achieves state-of-the-art results on the few-shot versions of four datasets: Kinetics, Something-Something V2, HMDB51 and UCF101. An extensive set of ablations shows how TRX observes multiple support set videos, the importance of tuple representations over single-frame comparisons, and the benefits of exploiting tuples of different cardinalities. As future work, we aim to explore spatio-temporal versions of TRX.

Acknowledgements Publicly-available datasets were used for this work. This work was performed under the SPHERE Next Steps EPSRC Project EP/R005273/1. Damen is supported by EPSRC Fellowship UMPIRE (EP/T004991/1).
References


