UnweaveNet: Unweaving Activity Stories

Will Price
University of Bristol
will.price@bristol.ac.uk

Carl Vondrick
Columbia University
vondrick.cs.columbia.edu

Dima Damen
University of Bristol
dima.damen@bristol.ac.uk

Abstract

Our lives can be seen as a complex weaving of activities; we switch from one activity to another, to maximise our achievements or in reaction to demands placed upon us. Observing a video of unscripted daily activities, we parse the video into its constituent activity ***threads*** *through a process we call* ***unweaving****. To accomplish this, we introduce a video representation explicitly capturing activity threads called a thread bank, along with a neural controller capable of detecting goal changes and resuming of past activities, together forming* ***UnweaveNet****. We train and evaluate UnweaveNet on sequences from the unscripted egocentric dataset EPIC-KITCHENS. We propose and showcase the efficacy of pretraining UnweaveNet in a self-supervised manner.*

1. Introduction

"It's the morning and you've just walked into the kitchen: you're hungry, sleepy, the kitchen is a mess, but you have a paper to review for CVPR. You put some bread into the toaster, turn the kettle on to make coffee, and in between waiting for the kettle to boil and bread to toast, you clean the dishes. The toast pops up and you put it on a plate, then the kettle boils and you resume making your coffee, switching back and forth as necessary until your breakfast is ready."

As in the storyline described above and depicted in Fig. 1, activities need not be completed over one continuous block of time. Instead they are often paused and interleaved with other activities. This observation gives rise to a new interpretation of video as a *weaving* of activities. Such a perspective supports the distinction between two instances of an activity when the activity is paused and later resumed. This distinction can be important for downstream applications, like assistive technologies which need to differentiate between starting a new task vs. resuming a previously paused one.

This novel view of video leads to the task proposed and tackled in this paper: *unweaving* a video into its constituent activity threads. Like a person reading a story mentally unweaves the story's narrative threads as they unfold, a model unweaving a video does so similarly, processing video *online*, detecting new threads of activity as they appear and updating its representation of previously discovered threads as they are resumed. Following this analogy, videos of activities as referred to as *activity stories*.

This proposed task is related to two previously proposed tasks: event boundary detection and unsupervised activity segmentation. The relationship between unweaving and these other related tasks is summarised in Fig. 2. Event boundary detection [37, 1] aims to detect points in the video where a transition between two events occurs. This task aims to model the experimental observation that humans can consistently detect transitions between events as they watch video online [50, 51, 16]. Typically, these methods are performed online [37, 1], predicting the future video representation, comparing this against the true representation, and measuring the prediction error in order to decide whether a boundary can be detected. Compared to unweaving, event boundary detection focuses on finding the transitions between activities and doesn't support the association between

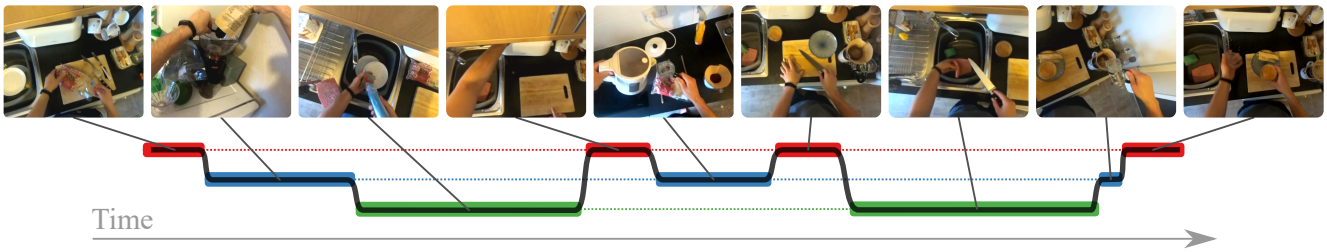


Figure 1. In our daily lives, one switches between activities (*e.g.* making toast, preparing coffee, washing up) to minimize idle time. Such behaviour results in video demonstrating multiple activities woven together. This paper introduces a model that learns to undo this, unweaving video into threads of activity without the need for semantic labels.

Figure 2. **Task comparison**: In *unweaving*, the model has to decide, online, whether the current part of the video is a continuation of the last-seen activity, a resumption of previously paused activity, or a completely new instance of an activity. The figure compares unweaving to two previously studied tasks.

events depicting a paused-and-resumed activity. Also related, unsupervised activity segmentation [22, 36] clusters visual features to produce a segmentation of the video. This task doesn't distinguish between different instances of the same activity, *e.g*. the act of making two cups of tea, one after the other, is the same as making just one. Unweaving is significantly more challenging as it is performed *online*, without specifying the number of activities, nor the duration of the video. Unweaving thus combines the challenges of the two aforementioned tasks.

In addition to introducing the problem of unweaving, this paper proposes a model that learns to unweave video into activity threads. Different threads of activity are modelled in an explicit manner by a *thread bank* that is manipulated by a neural controller as subsequent video is processed. To train UnweaveNet, a self-supervised approach is introduced that leverages within-thread temporal-order consistency to construct *synthetic* visual stories from unlabelled videos for pretraining. The model is then finetuned using a small set of manually annotated stories. The efficacy of this approach is shown experimentally using the unscripted egocentric dataset EPIC-KITCHENS-100 [9].

Our contributions are summarised as: (i) The novel task of unweaving video into its activity threads, online. (ii) A new video representation explicitly modelling video as a set of activity threads operated by a neural controller, which together form *UnweaveNet*. (iii) A self-supervised pretraining approach for UnweaveNet that samples threads from different parts of a long video and synthetically forms woven activity stories. (iv) Labelled annotations of activity threads from videos of the egocentric dataset: EPIC-KITCHENS[1]. (v) An empirical evaluation and ablation study of UnweaveNet.

2. Related work

Event boundary detection In their seminal work, Zacks, Tversky, and Iyer [51] define an event as "a segment of time at a given location that is perceived by an observer to have a beginning and an end". Aakur and Sarkar [1] propose a self-supervised method for detecting event boundaries, by predicting upcoming features. A boundary is detected when the prediction error of the future frame exceeds a dynamically-set threshold. Shou et al. [37] introduce a new dataset for supervised event-boundary detection. They explore detecting event boundaries using both supervised and unsupervised approaches. One of the unsupervised approaches, PredictAbility, measures the change in features about a point in time to detect boundaries.

Action segmentation and detection In action *segmentation* [17, 25, 10, 45, 22, 36] the goal is to assign an action label to every frame. In contrast, action *detection* [41, 39, 38, 31] predicts segments of video that possibly overlap. Most efforts for these tasks are supervised.

Kukleva et al. [22] propose an unsupervised method for segmenting video by learning a temporal embedding of frames. First, they train an MLP to regress the position of a frame in the video from which it originates. Intermediate features are extracted and act as the embedding of the frame. The embeddings are then clustered using a constrained optimisation that prevents non-adjacent frames from being assigned to the same cluster. An extension was proposed in [43] that uses the embeddings from a model trained for future feature prediction. Sarfraz et al. [36] also clusters frames, in an unsupervised manner, to form a temporally-weighted distance graph where nodes represent frames and edge weights are determined by the feature dissimilarity and temporal distance. Frames are then clustered iteratively until the desired number of clusters is reached.

Movie scene segmentation A variety of works tackle the problem of segmenting a movie into *scenes*. All existing methods are offline and require specifying the number of scenes into which the movie will be split. Early work by Yeung et al. [48] introduced the concept of a hierarchical *scene transition graph* which splits a movie into acts, scenes, and shots. Cour et al. [7] use the screenplay and closed captions associated with a movie and introduce the problem of *shot threading* to undo the common scenario in which shots from 2 or more cameras are interleaved together. Tapaswi, Bäuml, and Stiefelhagen [42] introduce a method for building a 'StoryGraph', a type of visualisation, originally proposed by the web-comic xkcd [29], where each character in a TV episode is represented as a line on a 2D chart.

More recently, Rao et al. [33] collect a large dataset MovieScenes containing 21k scene segments from 150 movies that are used to supervise their model. These approaches are specific to movies which are made up of scenes and shots. The notion of characters, multiple-cameras, shots, and scenes are not present in daily-activity videos.

Online clustering Unweaving videos can be viewed as a type of online clustering, where the number of clusters is not known ahead of time, nor the number of elements to be

[1] https://github.com/willprice/activity-stories

clustered. Kulshreshtha and Guha [23] investigate online clustering of faces in TV episodes. Faces in the current shot are compared against those seen previously via patch feature similarity and are integrated into the existing closest cluster if the similarity exceeds a threshold.

In Damen et al. [8], egocentric sequences from multiple users are used to cluster activities in an unsupervised manner, using 3D mapping and gaze information.

Nagarajan et al. [30] introduce a method for extracting a topological map of a kitchen environment from a first-person video. Part of their method clusters contiguous portions of video into 'activity-centric zones'. To accomplish this, they train a Siamese network on pairs of video frames to determine whether the frames come from the same zone. The network training is supervised using a heuristic: two frames are considered from the same zone if they are sufficiently close in time or if they have a shared background. The map is constructed by processing the video sequentially, adding nodes and edges as new zones are discovered. This method aims to aggregate subsequences of the video by location rather than by activity.

We compare to both online clustering and EGO-TOPO [30] in our results.

Neural-network controlled machines There is extensive recent work on using neural networks to control data structures [14, 18, 13, 34, 4, 24, 52] such as neural stacks [14, 18], neuro-symbolic stack machines [4], and neural Turing Machines [13, 53]. UnweaveNet follows in this vein by using a neural controller to operate its thread bank. Some of these works [14, 13] use soft operations, where the model performs all operations simultaneously at each computation step with learnt weighting, whereas others [4] employ hard decisions as in UnweaveNet. PtrNets [44] introduced an approach for applying neural networks to seq2seq problems where the output sequence corresponds to locations in the input. Part of UnweaveNet deals with a similar problem. UnweaveNet also shares similarities with the Memory Network [46], however we use an adjustable memory size to support varying numbers of threads.

Video summarisation Another related task is video summarisation [54, 27, 35, 49, 19], which aims to extract highlights that give a condensed overview of the video. Instead, in this work the full video is represented when unweaving an activity story into its constituent activities.

3. Unweaving stories

This section formulates the problem of unweaving (Sec. 3.1); introduces the structured video representation (Sec. 3.2) and the neural controller operating it (Sec. 3.3), together forming UnweaveNet; and concludes with the process used to train the model (Sec. 3.4).

3.1. Problem description

Unweaving is the problem of parsing an arbitrary-length video *online* into N variable-length activity threads, where N is *unspecified* and can vary across videos. Once unwoven, all parts of the video belonging to the same thread should correspond to one activity instance. When the video portrays a switch to a different activity, the ongoing thread should be paused and a different thread started or resumed. Assuming $\hat{n}_t$ threads have been identified up to time t, the task is to decide whether the current video clip v_t is a continuation of an existing thread or the beginning of a new thread.[2]

3.2. Thread bank

Core to our proposal is a structured representation of video, which we call a *thread bank*. This stores the representations of all complete and on-going activity threads discovered in the video as it is processed. New activity threads can be added into the bank as they are discovered and existing threads can be updated by incorporating new clips into them. In its most general form, a representation z_t^i of thread i at time t is produced as an aggregation g of the set of clips $\mathcal{V}_t^i$ currently assigned to the thread:

$$z_t^i = g\left(\mathcal{V}_t^i\right), \qquad g : \mathbb{R}^{|\mathcal{V}_t^i| \times C} \to \mathbb{R}^D \tag{1}$$

where C is the dimension of the clip feature and D the dimension of the thread representation. However, this doesn't quite capture the concept of an activity as an evolving process. Instead, a recurrent function ϕ_{update} is used in place of g, better modelling this perspective by updating the activity representation with information from the latest clip:

$$z_{t+1}^i = \phi_{\text{update}}(v_t, z_t^i), \qquad \phi_{\text{update}} : \mathbb{R}^C \times \mathbb{R}^D \to \mathbb{R}^D. \tag{2}$$

When a new thread is discovered, z_t^i is replaced with an initial learnt empty-thread representation z^*.

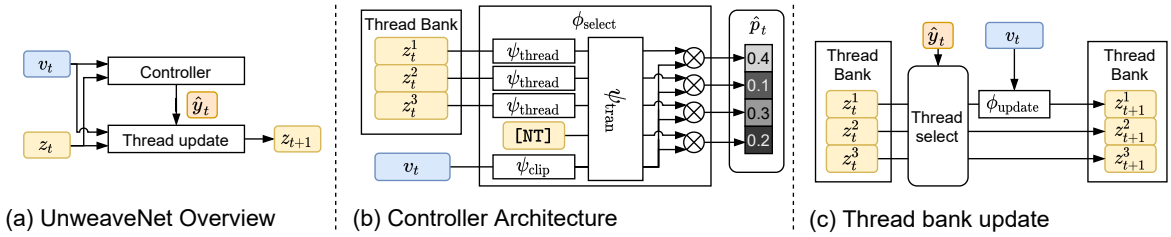
The state of the thread bank at time t and $t+1$ can be related as follows. Let $\hat{y}_t$ be the thread to which v_t will be added; for UnweaveNet, this is decided by its neural controller (described in Sec. 3.3). The representations within the updated thread bank z_{t+1} are related to the previous previous representations z_t as follows

$$z_{t+1}^i = \begin{cases} \phi_{\text{update}}(v_t, z_t^i) & i = \hat{y}_t \leq \hat{n}_t \\ \phi_{\text{update}}(v_t, z^*) & i = \hat{y}_t = \hat{n}_t + 1 \\ z_t^i & \text{otherwise.} \end{cases} \tag{3}$$

When $t = 1$, the thread bank is empty, thus $\hat{n}_1 = 0$.

While the number of threads in the bank can vary, each thread's representation is fixed in size, thus the model's complexity is linear in the number of threads rather than number of clips. Since the number of clips greatly exceeds the number of threads, this keeps the representation compact.

[2] The clip is assumed short enough to belong to one thread, leaving the exploration of clip length and multi-thread clips to future work.



(a) UnweaveNet Overview (b) Controller Architecture (c) Thread bank update

Figure 3. **(a) UnweaveNet overview:** at each timestep, UnweaveNet receives a clip v_t and considers the current thread bank state z_t; the controller determines whether that clip is a continuation of the ongoing thread, a resumption of a past thread, or the start of a new thread and updates the thread bank accordingly. **(b) Controller architecture:** v_t and z_t are embedded into a common space by linear layers ψ_{clip} and ψ_{thread} and fed into a transformer encoder. The clip embedding is compared against the thread embeddings and the new thread token [NT] to determine the probability of the clip joining an existing/new thread. **(c) Thread bank update:** Given a thread to update, determined by $\hat{y}_t = \text{argmax}_i\, p_t^i$, ϕ_{update} incorporates v_t into the current thread representation z_t^i to produce the updated representation z_{t+1}^i.

3.3. Neural controller

In order to construct the thread bank representation, a neural controller manipulates the thread bank as new clips are considered from the video (Fig. 3a). Given a new clip, the controller determines whether the clip is the beginning of a new thread or whether it is a continuation or a resumption of an existing thread (Fig. 3b). Once the decision has been made, the thread bank is updated (Fig. 3c) and the process iterates.

UnweaveNet uses a neural network ϕ_{select} to implement the controller. It is fed the new clip v_t and the current thread bank state z_t and is tasked with calculating the probabilities $p_t \in [0,1]^{\hat{n}_t+1}$ of how likely it is that v_t is the continuation of the ongoing thread, the resumption of a past thread, or the start of a new thread. Specifically, $p_t^{1:\hat{n}_t}$ contains the probabilities of v_t continuing/resuming existing threads and $p_t^{\hat{n}_t+1}$ is the probability of v_t starting a new thread. The controller ϕ_{select} computes a vector of logits

$$l_t = \phi_{\text{select}}(v_t, z_t), \quad \phi_{\text{select}} : \mathbb{R}^C \times \mathbb{R}^{\hat{n}_t \times D} \to \mathbb{R}^{\hat{n}_t+1}, \tag{4}$$

which are softmaxed (with temperature τ) to compute p_t

$$p_t^i = e^{l_t^i/\tau} / \textstyle\sum_{j=1}^{\hat{n}_t+1} e^{l_t^j/\tau}. \tag{5}$$

The decision is then determined by $\hat{y}_t = \text{argmax}_i\, p_t^i$.

To obtain $l_t^{1:\hat{n}_t}$, we learn a space in which the clip is closest to the thread it belongs to. Both the clip and threads are embedded into this space through linear projections $\psi_{\text{clip}} : \mathbb{R}^C \to \mathbb{R}^E$ and $\psi_{\text{thread}} : \mathbb{R}^D \to \mathbb{R}^E$. The cosine similarity between the clip and each thread embedding is measured to produce the scores $l_t^{1:\hat{n}_t}$ for how likely it is that the clip belongs to each thread. We also learn a latent similarity score $l^{\text{NT}} \in \mathbb{R}$, which acts as a threshold that a clip-thread similarity must exceed if the clip is to be deemed a continuation. This gives rise to the linear controller

$$\phi_{\text{select}}^{\text{linear}}(v_t, z_t)_i = \begin{cases} \cos\left(\psi_{\text{clip}}(v_t), \psi_{\text{thread}}(z_t^i)\right) & i \le \hat{n}_t \\ l^{\text{NT}} & \text{otherwise.} \end{cases} \tag{6}$$

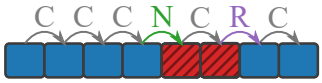


Figure 4. **Different decision scenarios during the unweaving process**: Continue thread, New thread, Resume thread. Each square depicts a single clip which is coloured by the thread it belongs to.

We contrast this approach with a contextualised approach by feeding a transformer encoder ψ_{tran} with a sequence composed of the embedded clip and thread representations, and a new-thread token [NT] $\in \mathbb{R}^E$. The transformer ψ_{tran} takes this input sequence $\left[\psi_{\text{clip}}(v_t), \psi_{\text{thread}}(z_t^1), \ldots, \psi_{\text{thread}}\left(z_1^{\hat{n}_t}\right), [\text{NT}]\right]$ and produces a corresponding output sequence $\left[\tilde{v}_t, \tilde{z}_t^1, \ldots, \tilde{z}_t^{\hat{n}_t+1}\right]$. These contextualised vectors (denoted by tilde) form a new embedding space in which the clip $\tilde{v}_t$ is compared against all threads $\tilde{z}_t^{1:\hat{n}_t}$ *and* the new-thread token $\tilde{z}_t^{\hat{n}_t+1}$ to form the vector of logits:

$$\phi_{\text{select}}^{\text{tran}}(v_t, z_t)_i = \cos(\tilde{v}_t, \tilde{z}_t^i). \tag{7}$$

This process using $\phi_{\text{select}}^{\text{tran}}$ is graphically depicted in Fig. 3b.

3.4. Training

UnweaveNet is trained *end-to-end*, including the backbone used to extract clip features, thus the clip and thread representations are optimised along with the controller parameters. The decisions made by ϕ_{select} are supervised using teacher forcing [47, 12], used for training language models; at each time step, z_t is populated according to the ground-truth clip-thread assignments $y_{1:t-1}$. A loss is then imposed on the output p_t (Eq. 4) with the correct decision y_t.

Due to the imbalance in the decisions, we weight three mutually exclusive scenarios in the loss (Fig. 4): starting a new thread (N), continuing the currently-active thread (C), and resuming a paused thread after a gap of more than one clip (R). Each scenario $s \in \{\text{C}, \text{N}, \text{R}\}$ is given a positive weight α_s, and we train with a focal loss [26] that causes hard examples to have a larger impact on the gradient than easy examples. Let S be a function that given $y_{1:t}$ determines the

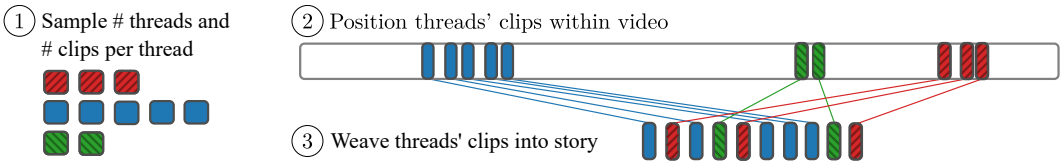


Figure 5. **Synthetic story construction:** 1 The number of threads is sampled, then the quantity of clips comprising each thread are sampled. 2 The threads are randomly positioned within the video, where clips within a thread are separated by a small random gap. 3 Finally, the threads' clips are randomly woven together into a synthetic story.

scenario s at time t. The loss (with focal hyperparameter γ) for a single story takes the form:

$$\mathcal{L} = -\sum_t \alpha_{S(y_{1:t})} \left(1 - p_t^{y_t}\right)^\gamma \log p_t^{y_t}. \tag{8}$$

The loss is averaged over all stories within the batch and back-propagated to train UnweaveNet.

This section formalised the problem of video unweaving (Sec. 3.1) and proposed a model, UnweaveNet, for solving it. UnweaveNet builds up a structured representation of video, called a thread bank, as it processes a streaming video (Sec. 3.2). A neural controller (Sec. 3.3) determines whether a clip belongs to an existing thread, or is the beginning of a new, and updates the thread bank accordingly. UnweaveNet is trained through a teacher-forcing set-up (Sec. 3.4).

4. Obtaining stories

Exploring unweaving requires a dataset of untrimmed videos with interleaved activity instances. We use the large-scale unscripted egocentric dataset EPIC-KITCHENS [9], where videos capture people in their own kitchens, capturing their activities over a three day period using a head-mounted camera. The dataset contains videos of participants switching back and forth between activities, making it a suitable source for obtaining interesting activity stories to *unweave*.

First, the untrimmed nature of the dataset is leveraged in making *synthetic stories*, that can be acquired *without* any annotations for pretraining UnweaveNet. Second, a sample of the dataset is annotated with activity threads providing *activity stories* for finetuning and evaluation purposes.

Synthetic stories We propose a method for pretraining UnweaveNet in a self-supervised manner by constructing synthetic stories through a randomised sampling procedure applied to long video. We sample threads from the same video as threads from different videos would be trivial to unweave. Given a number of threads N, we sample N sequences of clips of varying lengths, each from distinct locations within the same video, to produce synthetic threads.

Similar to the assumption in instance discrimination [2, 3], we assume these threads depict distinct activities. To further increase the likelihood of this, we enforce a minimum separation between threads. We then randomly interleave these together, respecting the arrow-of-time of the clips within each thread, to produce a synthetic story (depicted in Fig. 5). Full details on this sampling process are given in Appx. A.

In our setup, each synthetic story is composed of 10 clips woven from 1–4 threads. Synthetic stories are sampled randomly, per batch, from the dataset's training videos. Thus, the model is trained on a practically infinite number of synthetic stories. In all experiments, we sample 800K synthetic stories (8M clips, 50k batches, each containing 16 stories).

Activity-story annotation Synthetic stories contain visual discontinuities and synthetic threads aren't always composed of clips from a single activity due to the random sampling process. Thus, a model trained solely on synthetic stories falls short of being able to unweave natural video into activity threads. Consequently, a small dataset of manually annotated activity stories was collected for finetuning and evaluation purposes (further details are given in Appx. B).

Overall, 15k clips were annotated (4.2 hours) across 448 videos from EPIC-KITCHENS into their activity threads. Of these clips, 9.5k are for training, 3.8k for validation, and 1.8k for testing. The activity stories comprising the training and validation set were collected by 7 volunteer annotators, all consisting of 10 clips. A sample of each annotator's stories were checked for correctness. For testing, we manually collected stories of varying lengths (from 5 to 26 clips). The training stories come from videos in the training split of the EPIC-KITCHENS action-recognition challenge, and the test and validation stories come from videos from the combined test and validation splits. Statistics on the number of stories by number of threads are given in Tab. 1. Note that *threads are not annotated with any semantic labels*, the only metadata annotated is which clips belong together within a thread.

	# Threads		
Split	1	2	3
Train	718	201	32
Val	211	94	46
Test	50	50	50
Total	979	345	128

Table 1. EPIC-KITCHENS activity-story dataset by # of threads.

5. Experiments

This section evaluates UnweaveNet on the EPIC-KITCHENS activity-story dataset, demonstrating the

model's capabilities both qualitatively and quantitatively. Its performance is compared to a number of baselines and design decisions are motivated through ablation studies. The section is structured as follows: Sec. 5.1 introduces the baselines; Sec. 5.2 explains the evaluation metrics; Sec. 5.3 presents the main results; and Sec. 5.4 ablates the components in UnweaveNet.

5.1. Experimental setup & Baselines

This section briefly outlines the key points of the experimental setup (comprehensive details can be found in Appx. C) and explains the different baselines. As unweaving is a new concept, there are no existing works to directly compare against. Accordingly, a variety of baselines are either proposed or adapted from prior work. Two non-learnt naïve baselines are provided to give a lower bound on performance. As unweaving is an inherently online process, methods for offline unsupervised action segmentation are excluded.

Experimental setup Videos are encoded to 16FPS, resized to a height of 112px, and center-cropped. Horizontal flipping is used during training for data augmentation. The backbone network used to extract clip features is a top-heavy 3D ResNet-18 pretrained on Kinetics [20] using the DPC self-supervised objective [15]. A single-layer GRU [6] is used as ϕ_{update} and a single-layer transformer encoder with 4 heads is used for $\phi_{\text{select}}^{\text{tran}}$.

Naïve baselines The simplest baseline assigns all clips to a single thread, and hence is referred to as the *single-thread* baseline. Naturally this baseline will perform optimally in the case of all clips belonging to the same thread. An alternate, non-learnt baseline predicts all possible partitions of the clips as equally likely, including new threads. This is termed the *random-chance* baseline.

Online clustering This baseline clusters clips online into threads by measuring feature similarities. The similarity s_t^i of clip v_t to each thread $\mathcal{V}_t^i$ detected up to time t is

$$s_t^i = \frac{1}{|\mathcal{V}_t^i|} \sum_{v_j \in \mathcal{V}_t^i} \cos(v_t, v_j) \tag{9}$$

The clip is assigned a candidate thread $\text{argmax}_i\, s_t^i$ to join and if $\max_i s_t^i$ is beyond a specified threshold, then the clip continues that thread, otherwise a new thread consisting only of the clip v_t is started. The threshold is trained optimally on the validation set.

PredictAbility [37] This is a model designed for event segmentation adapted to perform unweaving. Recall that event segmentation aims to find the boundaries between events. This model detects event boundaries where there is a large change in feature representation over time. These boundaries are snapped to clip edges which splits the video into threads (without any resumed threads).

EGO-TOPO [30] This is a model designed to produce a graph structure of the egocentric video, where subsequences captured in the same physical location comprise a node. Edges are formed when the video depicts a transition from one location to another. Thus EGO-TOPO models the video by the topological locations depicted within. Nevertheless, it can still be viewed as a form of video unweaving that creates threads by location. As this model operates on frames rather than clips, a majority voting strategy is used to map from the frames assigned to nodes in the graph to the clips comprising threads in the unweaving.

5.2. Metrics

To measure performance on the unweaving task, we report the following metrics (further details, with metric equations, are given in Appx. D):

- The *Rand Index (RI)* [32], often used in clustering problems, which compares the estimated pair-wise grouping of clips to the ground truth.
- The *Teacher-Forcing Accuracy (TFA)*, which reports accuracy of the decisions produced by ϕ_{select} at each timestep t when we populate the thread bank according to the ground truth $y_{1:t-1}$. This allows us to evaluate the model's performance at each timestep without confounding the results by erroneous past decisions.
- ΔN, the difference in the number of predicted threads to the ground truth. This allows us to compare methods on whether new threads are created too readily, or too infrequently.

5.3. Results

The results for unweaving the stories from the EPIC-KITCHENS activity-story test set are presented in Tab. 2, comparing the non-learnt and learnt baselines to UnweaveNet. UnweaveNet's performance is reported under three different training regimes: synthetic stories only (SS), activity stories only (AS), and pretrained on synthetic stories then finetuned on the activity stories (SS+AS). UnweaveNet performs well compared to the baselines. The full model that is pretrained on synthetic stories and finetuned on activity stories (SS+AS) outperforms the baselines on all averaged metrics. EGO-TOPO performs best out of the baselines, however this is primarily due to its strong performance on the single-thread examples which demonstrate a sole activity, typically in one location[3]. Compared to UnweaveNet (when trained with activity stories), all the learnt baselines have a tendency to create more threads than exist in the ground truth as evidenced by the positive ΔN values. UnweaveNet also suffers this issue, albeit to a lesser extent.

The benefit of synthetic story pretraining is evident com-

[3] It is not possible to evaluate the TFA on the EGO-TOPO model with the provided implementation.

Model	Supervision	RI (↑)				TFA (↑)				$\Delta N(\rightarrow 0 \leftarrow)$			
		1	2	3	Avg	1	2	3	Avg	1	2	3	Avg
Single thread	-	-	52.4	35.3	62.6	-	41.9	26.5	56.2	-	-1.0	-2.0	-1.0
Chance*	-	18.9	48.4	60.5	42.6	50.0	36.0	26.7	37.6	3.7	3.0	3.8	3.5
PredictAbility [37]	Self-supervised	47.2	58.8	73.5	59.9	54.6	44.4	39.3	46.1	1.2	**0.5**	0.5	0.8
Online Clustering	Self-supervised	66.8	60.4	64.3	64.0	**92.4**	60.5	47.5	63.7	1.0	0.9	0.3	0.7
EGO-TOPO [30]	Action boundaries	83.9	66.2	64.9	71.7	-	-	-	-	0.6	0.7	1.6	0.9
UnweaveNet	SS	63.8	68.6	**74.8**	69.1	78.3	53.1	61.9	66.5	1.5	2.4	2.0	2.0
UnweaveNet	AS	81.8	58.4	60.9	67.0	83.0	51.3	37.6	57.3	0.6	**0.5**	**-0.1**	**0.3**
UnweaveNet	SS+AS	**84.1**	**70.1**	71.3	**75.1**	85.2	**71.9**	**74.3**	**77.5**	0.5	0.7	0.3	0.5

Table 2. Unweaving performance (5 run average) on the activity-story test set for naïve no-learning baselines (top section), learnt baselines (middle section) and UnweaveNet (bottom section) with/without pretraining on synthetic stories (SS) and finetuning on activity stories (AS). Metrics are described in Sec. 5.2. Performance is broken down by the number of threads in the test story (specified below each metric heading). Chance* refers to a random partition for RI and ΔN, and a random decision at each step for teacher-forcing accuracy (TFA).

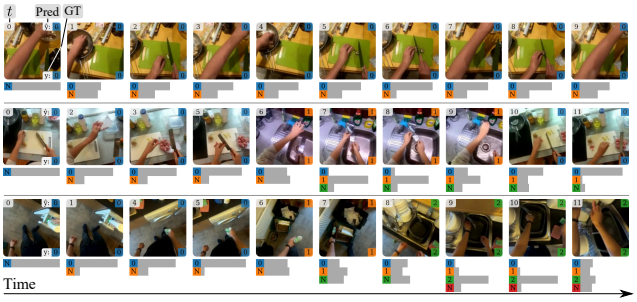


Figure 6. **Qualitative examples demonstrating UnweaveNet successfully unweaving 3 activity stories**. Decision probabilities p_t are shown below each clip as a bar chart (N denotes a new thread). Top right corner indicates predicted thread, bottom right–ground-truth thread, and top left–clip index. **Top** (1 thread story): chopping mushrooms. **Middle** (2 thread story): dicing meat (clips 0–5, 10–11) and rinsing cleaver (6–9). **Bottom** (3 thread story): setting up washing machine (0–5), throwing bottle into recycling (6–7) and washing hands (8–11).

paring AS to SS+AS under both the RI and TFA metrics. The average ΔN is closer to the ideal (0) for the AS model compared to the SS+AS model. This can be attributed to the synthetic stories having more threads than the activity stories, thus predisposing the model to creating more threads. Interestingly, the average RI is higher, 69.1% vs. 67.0%, when *only* pretraining on synthetic stories (SS) than when only training on activity stories (AS); though the SS model oversegments the video into too many threads as can be seen from the higher average ΔN (2.0 vs. 0.3) and higher 3-thread RI (74.8% vs. 60.9%).

Qualitative examples of the unweavings produced by UnweaveNet are provided in Fig. 6, demonstrating the model's capability to unweave stories with varying numbers of threads. Figure 7 demonstrates how UnweaveNet is capable of unweaving a long sequence of 40 clips, demonstrating the ability of the model to create new threads of

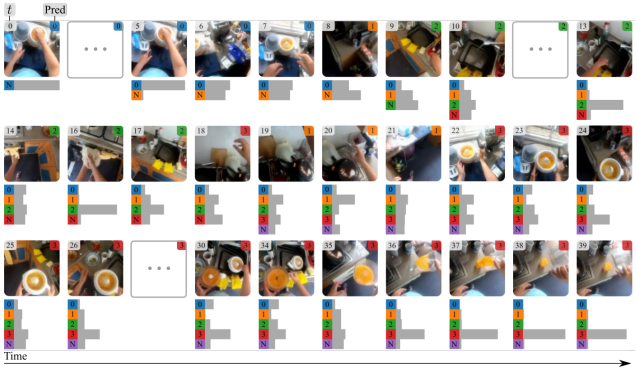


Figure 7. UnweaveNet represents this 40 clip sequence as 4 threads: juicing the oranges (0–7), washing hands (9–17), getting a glass (19–21), and serving the orange juice (22–39).

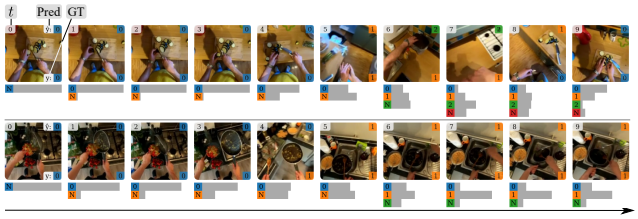


Figure 8. UnweaveNet's failure modes. **Top (over-segmentation)**: UnweaveNet separates the chopping activity (clips 0–4) from cleaning (putting peelings into the bin) (clips 6–7) and correctly resumes the first thread (clip 9). However, an additional incorrect thread is created (clips 5 and 8)c to capture the transition. **Bottom (late-starts)**: two threads are recognised: serving food (clips 0–4) and washing pan (clips 5–9). However UnweaveNet leaves the serving thread one clip later than in the ground truth (clip 4).

varying lengths. Figure 8 demonstrates two failure modes of UnweaveNet: oversegmenting into too many threads and transitioning between threads slightly later than in the ground truth. Further qualitative results can be found in Appx. E.

Figure 9 shows how TFA varies during online predictions

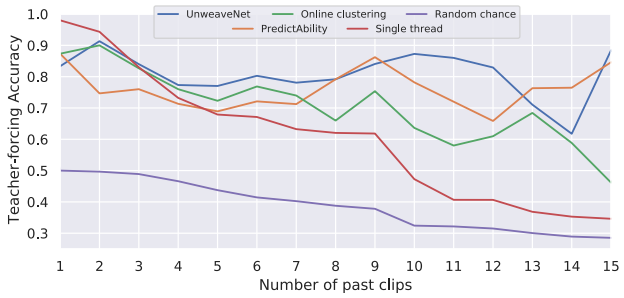


Figure 9. Teacher-forcing accuracy by number of clips observed

as more clips are observed. Initially, the single thread baseline has an easy task since few stories this short have more than a single thread, but from 4 clips onwards, UnweaveNet's performance gap over this baseline steadily increases, and the performance is robust as more clips are considered. UnweaveNet outperforms the online clustering baseline from observing 4 clips onwards. The teacher-forcing accuracy of the PredictAbility model is quite high due to two facts: the model cannot resume threads, and resuming a threads is a rarer event than continuing or starting a thread, therefore the model has fewer choices to take at each step and performs well on the more frequent scenarios.

5.4. Ablation studies

Several ablation studies are conducted to determine the impact of the components of UnweaveNet on its behaviour. Each ablation study aims to answer a specific question.

How to best construct synthetic stories? Having established that pretraining on synthetic stories is beneficial (as was shown in Tab. 2), the best way of constructing them is investigated. There are two hyperparameters to tune: the gap between clips within a synthetic thread and the number of synthetic threads forming the story. As the gap between clips in a thread increases, the visual similarity between adjacent clips decreases, making the task of associating the clips together harder. Table 3a shows that increasing this gap up to 2 seconds is beneficial, but beyond this we observe a degradation. Using a random gap between 2–4 seconds as an augmentation strategy was found to further boost performance over a fixed clip gap. This is the default configuration used throughout the remainder of the ablation studies.

When constructing synthetic stories, the number of threads is sampled uniformly from 1 to a maximum $N_{\max}$. Table 3b shows the RI increases as $N_{\max}$ is increased up to 4 threads, beyond which the performance decreases. This drop can be attributed to the fact that threads are composed of fewer clips as the number of threads is increased in addition to the increased risk that some threads overlap in activity.

How to implement ϕ_{select} and ϕ_{update}? The two versions of ϕ_{select} introduced in Sec. 3.3, $\phi_{\text{select}}^{\text{linear}}$ and $\phi_{\text{select}}^{\text{tran}}$, are compared in Tab. 4. The transformer based model proves superior to the linear embedding. In a similar manner, two versions of ϕ_{update} are compared using the recurrent update module based on a GRU vs. a linear projection of the last clip of each thread. For the latter, the new clip representation overwrites the previous thread representation when performing an update. The results demonstrate a small but consistent improvement when using the GRU update module.

Clip gap (s)	RI (↑)
0	60.6±0.6
1	73.1±0.3
2	74.6±0.7
4	73.7±0.7
2–4	**75.1**±0.5

(a) Clip gap within thread.

Max # threads	RI (↑)
2	69.0±0.6
3	72.7±0.6
4	**75.1**±0.5
5	74.0±0.8
6	73.5±0.8

(b) Max # threads.

Table 3. Synthetic story formation ablation study.

ϕ_{select}	ϕ_{update}	RI (↑)
Linear Embedding	Last clip	73.7±0.7
Linear Embedding	GRU	74.3±0.3
Transformer	Last clip	74.8±0.3
Transformer	GRU	**75.1**±0.5

Table 4. UnweaveNet architectural choices.

An additional study on the effect of the weight in the loss function can be found in Appx. E.

6. Conclusion

This paper introduced video *unweaving*, the task of parsing a video online into its constituent activity *threads*, accomplished by the introduction of a novel representation that models ongoing activity, operated by a neural controller, together called UnweaveNet. UnweaveNet can handle resuming a thread when the video depicts a switch from one thread to a previously observed thread. Moreover, it can be applied to variable-length videos, with memory requirements scaling linearly in the number of threads. A dataset of activity stories was annotated and used to evaluate how UnweaveNet can be pretrained through self-supervision by sampling synthetic stories from untrimmed videos.

UnweaveNet has potential applications in assistive technologies as the activities are perceived online. By focusing the experiments on egocentric footage, UnweaveNet is more suitable for sousveillance [28], one's ability to monitor her/his activities, than surveillance, remote monitoring of others' activities. However, in principle, the same approach can be adapted for monitoring other people's activity.

Acknowledgements. This work used public dataset and was supported by EPSRC Doctoral Training Program, EPSRC UMPIRE (EP/T004991/1), and NSF NRI Award #2132519.

References

[1] Sathyanarayanan N. Aakur and Sudeep Sarkar. "A Perceptual Prediction Framework for Self Supervised Event Segmentation". In *CVPR*. 2019 (pages 1, 2).

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In *ICML*. 2020 (page 5).

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. "Improved Baselines with Momentum Contrastive Learning". In *arXiv:2003.04297*. 2020 (page 5).

[4] Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. "Compositional Generalization via Neural-Symbolic Stack Machines". In *NeurIPS*. 2020 (page 3).

[5] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In *SSST*. 2014 (page 13).

[6] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation". In *EMNLP*. 2014 (page 6).

[7] Timothee Cour, Chris Jordan, Eleni Miltsakaki, and Ben Taskar. "Movie/Script: Alignment and Parsing of Video and Text Transcription". In *ECCV*. 2008 (page 2).

[8] Dima Damen, Teesid Leelasawassuk, Osian Haines, Andrew Calway, and Walterio Mayol-Cuevas. "You-Do, I-Learn: Discovering Task Relevant Objects and their Modes of Interaction from Multi-User Egocentric Video". In *BMVC*. 2014 (page 3).

[9] Dima Damen et al. "Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100". In *IJCV* (2021) (pages 2, 5).

[10] Yazan Abu Farha and Jurgen Gall. "MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation". In *CVPR*. 2019 (page 2).

[11] Alexander J. Gates and Yong-Yeol Ahn. "The Impact of Random Models on Clustering Similarity". In *JMLR* 1 (2017) (page 14).

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016 (page 4).

[13] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines". In *arXiv:1410.5401*. 2014 (pages 3, 11).

[14] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. "Learning to Transduce with Unbounded Memory". In *NeurIPS*. 2015 (pages 3, 11).

[15] Tengda Han, Weidi Xie, and Andrew Zisserman. "Video Representation Learning by Dense Predictive Coding". In *NeurIPS*. 2019 (pages 6, 13).

[16] Bridgette M. Hard, Barbara Tversky, and David S. Lang. "Making Sense of Abstract Events: Building Event Schemas". In *Memory & Cognition* (2006) (page 1).

[17] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. "Connectionist Temporal Modeling for Weakly Supervised Action Labeling". In *ECCV*. 2016 (page 2).

[18] Armand Joulin and Tomas Mikolov. "Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets". In *NeurIPS*. 2015 (page 3).

[19] Y. Jung, Donghyeon Cho, Dahun Kim, S. Woo, and In-So Kweon. "Discriminative Feature Learning for Unsupervised Video Summarization". In *AAAI*. 2019 (page 3).

[20] Will Kay et al. "The Kinetics Human Action Video Dataset". In *arXiv:1705.06950*. 2017 (pages 6, 13).

[21] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In *ICLR*. 2015 (page 13).

[22] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. "Unsupervised Learning of Action Classes With Continuous Temporal Embedding". In *CVPR*. 2019 (page 2).

[23] Prakhar Kulshreshtha and Tanaya Guha. "An Online Algorithm for Constrained Face Clustering in Videos". In *ICIP*. 2018 (page 3).

[24] Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. "Neural Random-Access Machines." In *ICLR*. 2016 (page 3).

[25] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. "Temporal Convolutional Networks for Action Segmentation and Detection". In *CVPR*. 2017 (page 2).

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. "Focal Loss for Dense Object Detection". In *CVPR*. 2017 (page 4).

[27] Behrooz Mahasseni, Michael Lam, and S. Todorovic. "Unsupervised Video Summarization with Adversarial LSTM Networks". In *CVPR* (2017) (page 3).

[28] Steve Mann, Jason Nolan, and Barry Wellman. "Sousveillance: Inventing and Using Wearable Computing Devices for Data Collection in Surveillance Environments." In *Surveill. Soc.* 3 (2003) (page 8).

[29] Randall Munroe. *Movie Narrative Charts*. xkcd. 2009. URL: https://xkcd.com/657/ (page 2).

[30] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. "Ego-Topo: Environment Affordances From Egocentric Video". In *CVPR*. 2020 (pages 3, 6, 7, 11).

[31] Aj Piergiovanni and Michael Ryoo. "Temporal Gaussian Mixture Layer for Videos". In *ICML*. 2019 (page 2).

[32] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In *J. Am. Stat. Assoc.* 336 (1971) (pages 6, 14).

[33] Anyi Rao, Linning Xu, Yu Xiong, Guodong Xu, Qingqiu Huang, Bolei Zhou, and Dahua Lin. "A Local-to-Global Approach to Multi-Modal Movie Scene Segmentation". In *CVPR*. 2020 (page 2).

[34] Scott E. Reed and Nando de Freitas. "Neural Programmer-Interpreters". In *ICLR*. 2016 (page 3).

[35] Mrigank Rochan, L. Ye, and Y. Wang. "Video Summarization Using Fully Convolutional Sequence Networks". In *ECCV*. 2018 (page 3).

[36] M. Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. "Temporally-Weighted Hierarchical Clustering for Unsupervised Action Segmentation". In *CVPR*. 2021 (page 2).

[37] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. "Generic Event Boundary Detection: A Benchmark for Event Segmentation". In *ICCV*. 2021 (pages 1, 2, 6, 7, 14).

[38] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. "CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos". In *CVPR*. 2017 (page 2).

[39] Zheng Shou, Dongang Wang, and Shih-Fu Chang. "Temporal Action Localization in Untrimmed Videos via Multi-Stage Cnns". In *CVPR*. 2016 (page 2).

[40] Noah A. Smith and Roy W. Tromble. "Sampling Uniformly from the Unit Simplex". In *Johns Hopkins Univ. Tech Rep* (2004) (page 11).

[41] Chen Sun, Sanketh Shetty, Rahul Sukthankar, and Ram Nevatia. "Temporal Localization of Fine-Grained Actions in Videos by Domain Transfer from Web Images". In *ICM*. 2015 (page 2).

[42] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. "StoryGraphs: Visualizing Character Interactions as a Timeline". In *CVPR*. 2014 (page 2).

[43] Rosaura G. VidalMata, Walter J. Scheirer, Anna Kukleva, David D. Cox, and Hilde Kuehne. "Joint Visual-Temporal Embedding for Unsupervised Learning of Actions in Untrimmed Sequences". In *WACV*. 2021 (page 2).

[44] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer Networks". In *NeurIPS*. 2015 (page 3).

[45] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. "Boundary-Aware Cascade Networks for Temporal Action Segmentation". In *ECCV*. 2020 (page 2).

[46] Jason Weston, Sumit Chopra, and Antoine Bordes. "Memory Networks". In *ICLR*. 2015 (page 3).

[47] Ronald J. Williams and David Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". In *Neural Computation* 2 (1989) (page 4).

[48] Minerva M. Yeung, Boon-Lock Yeo, Wayne H. Wolf, and Bede Liu. "Video Browsing Using Clustering and Scene Transitions on Compressed Sequences". In *Multimedia Computing and Networking*. 1995 (page 2).

[49] L. Yuan, Francis E. H. Tay, Ping Li, Li Zhou, and Jiashi Feng. "Cycle-SUM: Cycle-Consistent Adversarial LSTM Networks for Unsupervised Video Summarization". In *AAAI* (2019) (page 3).

[50] J. M. Zacks, T. S. Braver, M. A. Sheridan, D. I. Donaldson, A. Z. Snyder, J. M. Ollinger, R. L. Buckner, and M. E. Raichle. "Human Brain Activity Time-Locked to Perceptual Event Boundaries". In *Nat Neurosci* 6 (2001) (page 1).

[51] Jeffrey M. Zacks, Barbara Tversky, and Gowri Iyer. "Perceiving, Remembering, and Communicating Structure in Events." In *J. Exp. Psychol. Gen.* 1 (2001) (pages 1, 2).

[52] Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus. "Learning Simple Algorithms from Examples". In *ICML*. 2016 (page 3).

[53] Wojciech Zaremba and Ilya Sutskever. "Reinforcement Learning Neural Turing Machines". In *arXiv:1505.00521*. 2016 (page 3).

[54] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. "Video Summarization with Long Short-Term Memory". In *ECCV*. 2016 (page 3).