Additional Material

Future Work. In the paper, we make decisions online without recovering from errors. Recovery is a very interesting avenue for exploration, including soft assignment as in [13, 14], however these are harder to compare to prior work (e.g. Ego-Topo and Predictabilty from [30]). An exploration of the right metrics and baselines is needed to consider recovery.

These appendices provide additional material on UnweaveNet. Appx. A explains how synthetic stories are constructed, detailing the exact procedure used to sample them. Appx. B covers the procedure used to annotate activity stories, demonstrating the user interface that was developed for this. Appx. D explains the metrics used for evaluating the unweaving task. Appx. E presents additional and extended experimental results of UnweaveNet.

A. Constructing synthetic stories

UnweaveNet is pretrained in a self-supervised manner by learning to unweave synthetic stories constructed via a randomised sampling procedure applied to long video. These synthetic stories aim to pose a similar challenge to unweave as activity stories, albeit in a somewhat label-noisy manner since they are constructed through a fully automated process.

Synthetic stories are composed of a number of randomly sampled subsequences of different lengths, termed synthetic threads, that are randomly woven together. A graphical overview of this process is given in Fig. 5. Synthetic threads are sampled from the same video as sampling them from different videos would result in a story that is trivial to unweave due to the large visual differences between videos. Threads are sampled such that they are at least a minimum distance away from one another and are assumed to depict distinct activities.

Sampling synthetic threads Building a synthetic story starts by obtaining a number of synthetic threads to weave together. These are obtained by sampling a number of sequences of clips of varying lengths from distinct non-overlapping locations within a video. First, the number of clips T that will comprise the story is decided.[4] Then, the number of threads n in the story is sampled from the uniform distribution $\mathcal{U}\{1, N_{\text{max}}\}$ where N_{max} is a specified upper bound. Next, the number of clips m_i comprising thread i is determined by uniformly sampling n non-zero positive integers $(m_i)_{i=1}^{n}$ such that $\sum_{i=1}^{n} m_i = T$ using the method of Smith and Tromble [40]. The starting location of the threads within the video are then sampled randomly from a uniform distribution with the constraint that the threads are at least a minimum distance away from one another. The clips $\mathcal{V}^i = \left(v_t^i\right)_{t=1}^{m_i}$ comprising thread i are then sampled starting from the thread's starting location, obtained in the previous step. Adjacent clips within a thread are separated by a small random gap sampled from $\mathcal{U}\{G_{\text{min}}, G_{\text{max}}\}$ where G_{min} and G_{max} denote the specified minimum and maximum gap, further increasing the difficulty of unweaving the resulting synthetic stories.

Weaving threads Once the threads have been obtained, they need to be woven together to form a synthetic story. This is performed such that the within-thread temporal-ordering of clips is not disrupted. In other words, given a thread i composed of the clips $\left(v_1^i, v_2^i\right)$, v_1^i will always appear before v_2^i in the synthetic story, and so on for threads with more clips. Weaving is accomplished by building a template for how the clips will be ordered relative to one-another, based on the number of clips m_i comprising each thread. First, a vector $q \in \{1..n\}^T$ of repeated thread indices is constructed, where there are m_i repeats of each thread index i:

$$q = (\underbrace{1, \ldots, 1}_{m_1 \text{ times}}, \underbrace{2, \ldots, 2}_{m_2 \text{ times}}, \ \ldots, \ \underbrace{n, \ldots, n}_{m_n \text{ times}}). \tag{10}$$

Next, a random permutation of q is taken, denoted $\tilde{q}$, which forms the template used to weave the threads' clips together into the story v. Each clip v_j in the story can thus be defined as

$$v_j = v_{o_j}^{\tilde{q}_j}, \quad o_j = \sum_{k=1}^{j} \mathbb{I}\left[\tilde{q}_j = \tilde{q}_k\right]. \tag{11}$$

where o_j represents the number of clips in the story from thread $\tilde{q}_j$ up to time j. The following example demonstrates this procedure to aid comprehension. Let the number of total number of clips in the story be $T = 10$, the number of threads be $n = 4$ and the number of clips within each thread be $m = (3, 5, 4)$. The duplicated thread indices vector q is formed:

$$q = (1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3). \tag{12}$$

and is randomly permuted to obtain

$$\tilde{q} = (1, 2, 1, 2, 2, 3, 3, 2, 2, 3, 3, 1), \tag{13}$$

[4] T is fixed across stories to facilitate batch-based training.

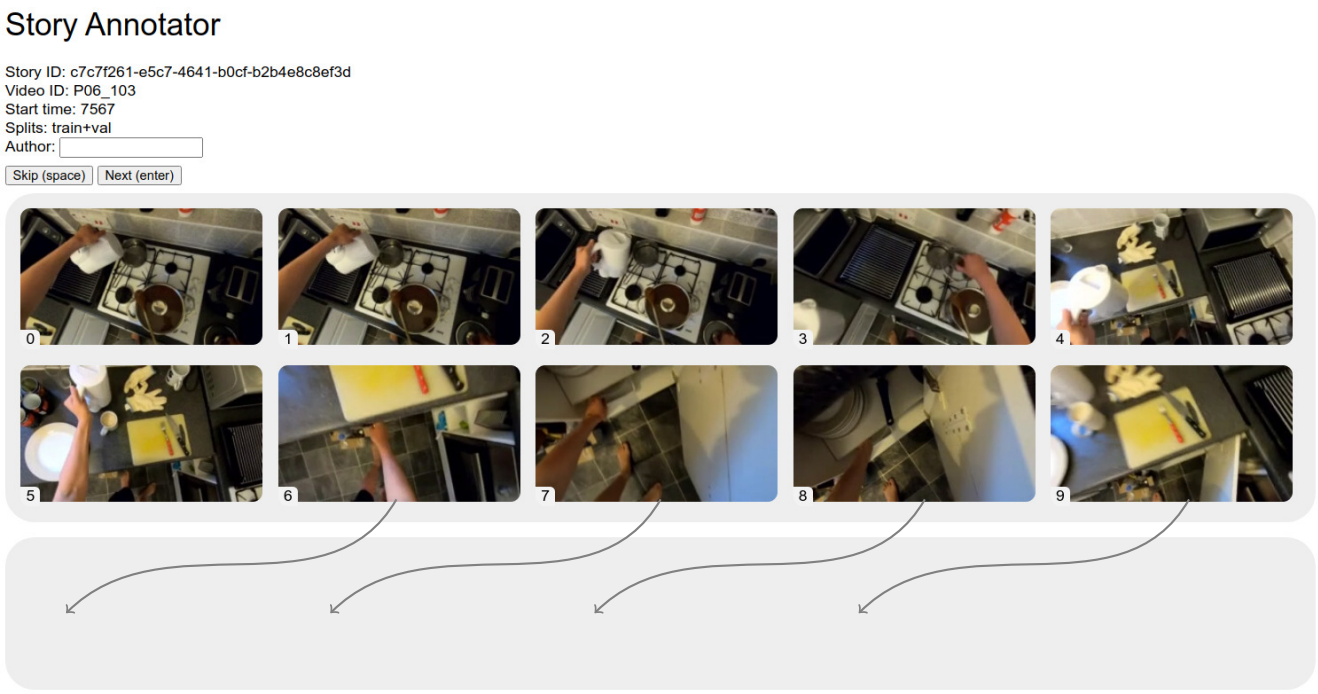

Figure 10. **Story unweaving annotation tool (pre-interaction):** Initial state of the UI when presented with a new video. ⟶ denotes a user dragging a clip into a new thread track (the empty grey rectangle).

which is then used to build the story

$$v = \left(v_1^1, v_1^2, v_2^1, v_2^2, v_3^2, v_1^3, v_2^3, v_4^2, v_5^2, v_3^3, v_4^3, v_3^1\right). \tag{14}$$

Note the within-thread temporal-ordering of clips has not been violated in v.

Error analysis in synthetic stories Studying Table 1 where thread statistics are given, two randomly sampled consecutive clips belong to the same thread 95.2% of the time. Accordingly, we estimate that ∼95% of all C decisions in synthetic stories to be correct. Also, we note that synthetic threads are sampled from distinct locations in the same video to increase the realism. We enforce 60s gap and accordingly estimate 99% of all N decisions to be also correct.

B. Annotating activity stories

A web-based unweaving tool (shown in Figs. 10 and 11) was developed to enable the annotation of stories within video. The tool randomly samples a sequence of video from a collection of videos and tasks the annotator with manually unweaving the sequence. The sampled sequence of video is broken up into a fixed number of clips which are displayed within a single track that represents a thread (grey box in Fig. 10). The annotator can then drag and drop clips (interaction denoted by the dark grey arrows in Fig. 10) into a new track to form a new thread (the result of which is shown in Fig. 11). Once complete, the tool saves the annotation to a database and ensures that this portion of video is not shown again to another annotator. Annotators were provided with the following instructions to use the tool:

> *Thread annotation:* We have sampled consecutive clips from a video (labelled 0–9) and the interface allows grouping these into activity 'threads' (sequence of clips sharing a common goal). Your task is to identify occurrences where the person is changing the course-of-action or goal (*e.g.* switch from washing-up to cooking, or from preparing-food to returning-items-to-cupboards). Once identified, the interface allows you to group all clips belonging to one goal in the same thread by dragging other clips to a new thread. If the person returns to an earlier goal (*e.g.* Goal switches $A \rightarrow B \rightarrow A$), make sure to keep all A clips together in one thread.
>
> *Using the annotator:* Your task is to drag video clips onto a new track (denoted by the gray background) when you detect a change in the course-of-actions. If the example is ambiguous or you cannot understand the activity from

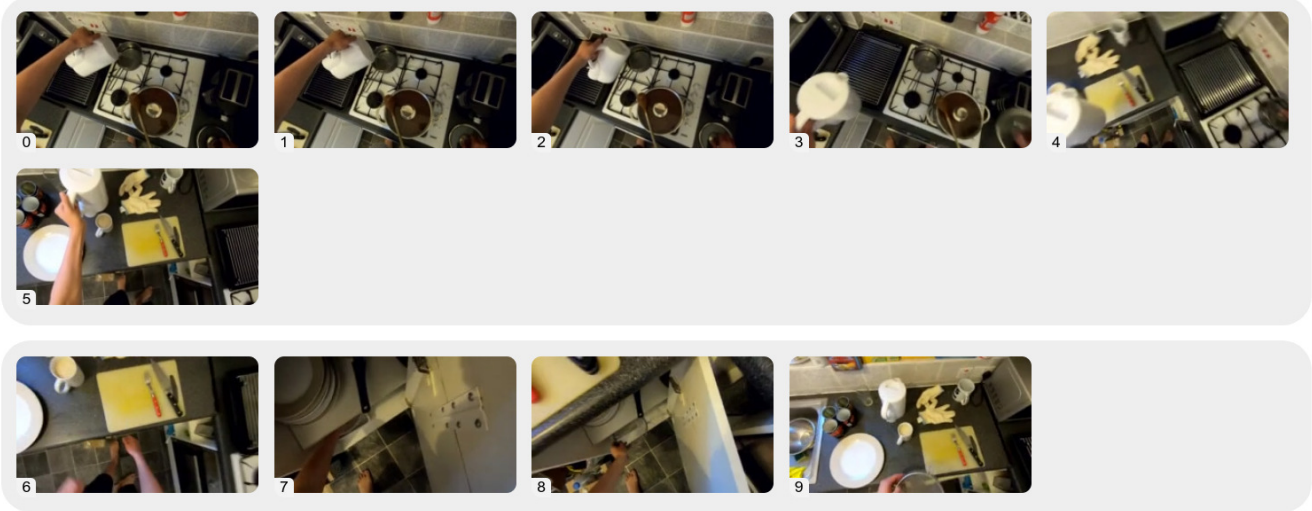

Figure 11. **Story unweaving annotation tool (post-interaction).** State of UI after the annotator has dragged the clips into a new thread (interaction depicted in Fig. 10). The ordering of the clips is always preserved; *i.e.* the tool prevents users from reordering the clips so that one that came at time t in the source video comes before one at $t-1$.

the clips or the videos are too dark to see, click skip (alternatively press `<space>`). Click next (alternatively press `<enter>`) to save and continue on to the next example.

C. Experimental details

This section describes the experimental set up use to train and evaluate UnweaveNet and the baselines.

Data Videos are encoded to 16FPS, resized to a height of 112px, and center-cropped. Synthetic stories are constructed from 1s long clips which are separated from adjacent clips in a synthetic thread by a gap of between 2–4s. To lower the chance of sampling threads sharing the same activity, threads are separated by at least 60s. Each synthetic story is composed of 10 clips and is produced by weaving between 1–4 synthetic threads, unless stated otherwise. Synthetic stories are sampled randomly from the training videos of EPIC-KITCHENS, thus the model is trained on a practically infinite number of synthetic stories. Videos shorter than 3½ mins long are discarded, as they are insufficiently long to sample synthetic stories from. In all experiments using synthetic story pretraining, around 800K synthetic stories are sampled (8M clips, 50K batches, each containing 16 stories).

UnweaveNet 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]. The backbone's features are average pooled both spatially and temporally to produce a single vector per clip. Clip and thread features have dimensions $C = D = 256$ respectively and are embedded into to an $E = 256$ dimensional space for $\phi^{\text{linear}}_{\text{select}}$ or $E = 512$ for $\phi^{\text{tran}}_{\text{select}}$. The thread representation update module ϕ_{update} is instantiated using a single layer GRU [5] with a hidden/output dimension of 256. The transformer-based controller $\phi^{\text{tran}}_{\text{select}}$ uses 1 layer, 4 heads, a model dimension of 512, and a feed forward MLP dimension of 2048 with a dropout of 0.2 (applied only during pretraining). The new thread token `[NT]` is defined as a learnt latent vector, and the empty thread representation as $z^* = \mathbf{0}$ (using a learnt latent vector was also experimented with, but it did not yield any improvements). The softmax temperature τ is set to 0.05.

UnweaveNet is pretrained on synthetic stories, generated on the fly, for 50k steps (determined by measuring performance on a validation set of synthetic stories). The learning rate is set to 2×10^{-4} for 25k steps and then dropped to 2×10^{-5} for the remaining 25k steps. Finetuning is conducted using the learning rate 2×10^{-5} and proceeds for up to 1k steps, with early stopping based on validation split performance. Adam [21] is used to optimise the models with a batch-size of 16 stories on 2x NVIDIA RTX-2080 Tis. Each training step takes around 1.3s; therefore pretraining takes around 18 hours and finetuning over 5 random seeds just under 2 hours. Results are reported as the mean over 5 different seeds (consistent across experiments) used in finetuning, starting from a single synthetic story pretrained checkpoint. Horizontal flipping is used for data augmentation, applied consistently to all clips within a story. Other augmentation strategies, including random crops and color distortion, were attempted, but didn't improve performance.

Baselines Both PredictAbility [37] and the online clustering baseline use features extracted from a top-heavy 3D ResNet18 trained with the DPC objective on Kinetics, the same as used for the clip backbone in UnweaveNet. Hyperparameters for the following baselines are chosen by performing a hyperparameter search optimising for the average RI across the validation set. PredictAbility uses a temporal stride of 2, a window of 10 frames each side of the candidate boundary, and a σ of 15 frames for the Laplacian of Gaussian kernel. The online clustering baseline uses a similarity threshold of 0.645 above which the clip is judged a continuation of the thread. EGO-TOPO uses a window size of 8 frames, a lower threshold of 0.4, and an upper threshold of 0.6. The localisation network used is the same as the one released by the authors.[5]

D. Metrics

Rand Index (RI) As unweaving is a form of clustering, a clustering metric is used as the main assessor of the quality of the unwoven threads. The Rand index [32], a frequently used metric to assess the similarity of one clustering to another, fulfils this role. It computes the percentage of correct pair-wise decisions.

The Rand index is best understood by examining the meaning of true/false positives/negatives in this setting. Given a story that is annotated with a ground-truth set of threads G and that has been unwoven by a model into a set of threads P, both defined as partitions over the set of clips comprising the story, the definition of true/false positives/negatives are:

- True positives $\mathrm{TP}(G, P)$: the number of pairs of clips that are in the same thread in G and in the same thread in P.
- False positives $\mathrm{FP}(G, P)$: the number of pairs of clips that are in different threads in G but in the same thread in P.
- True negatives $\mathrm{TN}(G, P)$: the number of pairs of clips that are in different threads in G and also in different threads in P.
- False negatives $\mathrm{FN}(G, P)$: the number of pairs of clips that are in the same thread in G but in different threads in P.

These can then be used to define the Rand index:

$$\mathrm{RI}(G, P) = \frac{\mathrm{TP}(G, P) + \mathrm{TN}(G, P)}{\mathrm{TP}(G, P) + \mathrm{FP}(G, P) + \mathrm{TN}(G, P) + \mathrm{FN}(G, P)}. \tag{15}$$

Note that the denominator in Eq. 15 is equal to $\binom{T}{2}$ where T is the total number of clips. The Rand index is computed for each story and is averaged over all stories in the test set to produce a single score.

The expected Rand index $\mathbb{E}_{P \sim R(T)}\left[\mathrm{RI}(G, P)\right]$ for the naïve baselines can be computed by defining their corresponding distribution $R(T)$ over partitions of T clips. Gates and Ahn [11] provide closed form expressions for both naïve baselines. For the single-thread baseline R^1,

$$\mathop{\mathbb{E}}_{P \sim R^1(T)}\left[\mathrm{RI}(G, P)\right] = \frac{\sum_i \binom{|\mathcal{V}^i|}{2}}{\binom{T}{2}} \tag{16}$$

where $|\mathcal{V}^i|$ the number of clips in the i-th thread of partition P. For the random-chance baseline R^{all},

$$\mathop{\mathbb{E}}_{P \sim R^{\mathrm{all}}(T)}\left[\mathrm{RI}(G, P)\right] = \frac{B(T-1)}{B(T)} \frac{\sum_i \binom{|\mathcal{V}^i|}{2}}{\binom{T}{2}} + \left(1 - \frac{B(T-1)}{B(T)}\right)\left(1 - \frac{\sum_i \binom{|\mathcal{V}^i|}{2}}{\binom{T}{2}}\right), \tag{17}$$

where $B(T)$ is the T-th Bell number, which counts how many possible ways there are of partitioning a set of T objects into non-empty subsets.

Teacher-forcing accuracy (TFA) The teacher-forcing accuracy measures the proportion of clip decisions that were made correctly, assuming that all the past decisions up to that point were correct. The teacher-forcing accuracy is broken down by n, the number of threads that have been observed up to and including time t.

To describe how teacher-forcing accuracy is computed, it is necessary to define the set of story prefixes $\mathcal{P}_n$, story subsequences starting from the first clip onwards that contain exactly n threads. These are defined as

$$\mathcal{P}_n = \left\{v_{1:t} \,|\, v \in \mathcal{X} \wedge N(v_{1:t}) = n \wedge 1 \leq t \leq T(v)\right\}, \tag{18}$$

where $\mathcal{X}$ represents the dataset of stories, $N(v_{1:t})$ denotes the number of threads in the ground truth for v up to and including time t, and $T(v)$ denotes the number of clips in v. The teacher-forcing accuracy for clip decisions where exactly n threads have been observed can then be defined as

$$\mathrm{TFA}(\mathcal{P}_n) = \frac{1}{|\mathcal{P}_n|} \sum_{v \in \mathcal{P}_n} \mathbb{I}\left[\hat{y}'_{T(v)}(v) = y_{T(v)}(v)\right] \mathbb{I}\left[|v| > 1\right], \tag{19}$$

[5] EGO-TOPO source code and models: https://github.com/facebookresearch/ego-topo

where $\hat{y}'_t(v)$ denotes the decision produced by the model at time t on v when it is run using teacher-forcing and $y_t(v)$ denotes the ground-truth thread-index at time t in story v. The $|v| > 1$ condition in Eq. 19 removes stories composed of a single clip where all methods will trivially make the correct decision. For UnweaveNet, the thread bank is populated using ϕ_{update} according to the ground-truth thread assignments $y_{1:T(v)-1}$ to produce $z_{T(v)}$. The controller ϕ_{select} is then fed $v_{T(v)}$ and $z_{T(v)}$ to determine $\hat{y}'_{T(v)}(v)$.

An analogous approach is taken for the online clustering baseline, populating the clusters according to the ground-truth. PredictAbility's decision is judged to be correct at each time step if the ground truth has a thread continuation and the model does not detect a transition, or if there is a new thread in the ground truth and the model detects a transition. It is not possible to evaluate the TFA of the EGO-TOPO model with the provided implementation.

To compute the teacher-forcing accuracy for the naïve baselines, the term $\mathbb{I}[\hat{y}'_{T(v)}(v) = y_{T(v)}(v)]$ in Eq. 19 is replaced with the probability $p_t^R(v)$ of selecting the correct thread at time t in video v for a baseline model R. For the single thread baseline,

$$p_t^1(v) = \begin{cases} 1/N(v_{1:t-1}) & y_t \text{ is an existing thread} \\ 0 & y_t \text{ is a new thread.} \end{cases} \tag{20}$$

The reasoning behind the first case in Eq. 20 is that there may be more than one thread in the ground-truth unweaving up to time t. Because of this, the baseline has to make a choice which thread the clip will join, so the choice is made randomly. However, this baseline is unable to start a new thread (case two).

The formulation for the random chance baseline is simpler, since the probability that the correct decision is made is uniform across the possible options: $p_t^{\text{all}} = 1/(N(v_{1:t-1}) + 1)$. Note that the denominator is $N(v_{1:t-1}) + 1$ and not $N(v_{1:t-1})$ as there is the additional option of starting a new thread at each time-step.

Difference in number of threads (ΔN) It is informative to know whether the model oversegments, creating more threads than in the ground truth, or undersegments, creating fewer threads than in the ground truth. Computing the difference $\Delta N = \hat{N} - N$ between the number of threads $\hat{N}$ detected by a model and the number of threads N in the ground truth reveals this.

For the single-thread baseline, the number of predicted threads is always set to one. For the random-chance baseline, a closed form expression can be derived using Stirling numbers of the second kind which compute the number of ways to partition n items into k non-empty subsets denoted $S(n, k)$. The number of threads this baseline produces over a video with T clips is simply a weighted average over partitions of size n:

$$\hat{N}^{\text{all}} = \frac{1}{\sum_{n=1}^{T} S(T, n)} \sum_{n=1}^{T} S(T, n) n. \tag{21}$$

This metric is computed for each story and averaged across all stories to produce a single score across the dataset.

E. Additional results

Additional qualitative results We present an additional selection of successful predictions by UnweaveNet Fig. 12. We provide the full, non-truncated, version of Fig. 7 in Fig. 13.

How do the scenario loss weights affect UnweaveNet's behaviour? Section 3.4 noted that it was necessary to weight different scenarios in the loss function. The effects of the scenario loss weights α_s used in Eq. 8 on the model's behaviour are investigated in Fig. 14. When an equal weighting for all scenarios is used (left), the model is heavily biased towards continuing threads. To mitigate this, a higher weighting $(\alpha_{\text{C}}, \alpha_{\text{R}}, \alpha_{\text{N}}) = (1, 100, 10)$ is used for resume and new thread scenarios (right) roughly proportional to their inverse frequency in the training split. The higher weighting for these scenarios improves their performance at the cost of the continue thread scenario. As anticipated, resuming threads after a break is the hardest scenario to tackle. Two additional configurations for $(\alpha_{\text{C}}, \alpha_{\text{R}}, \alpha_{\text{N}})$ are shown in the center of the figure between the two extremes of an equal weighting for each scenario (left) and heavily weighting towards resume and new thread scenarios (right). Overall, the proposed non-uniform weightings perform better in terms of RI (75.1%) than using equal scenario weightings (67.9%).

How does the performance vary for different story lengths? To showcase that Unweavenet is robust to story length, we analyse the results for stories of varying lengths. In Fig 15, we bin stories by # of clips in the story and compare RI for each bin across the baselines and UnweaveNet. Results demonstrate that UnweaveNet is robust to story length whereas online clustering struggles.

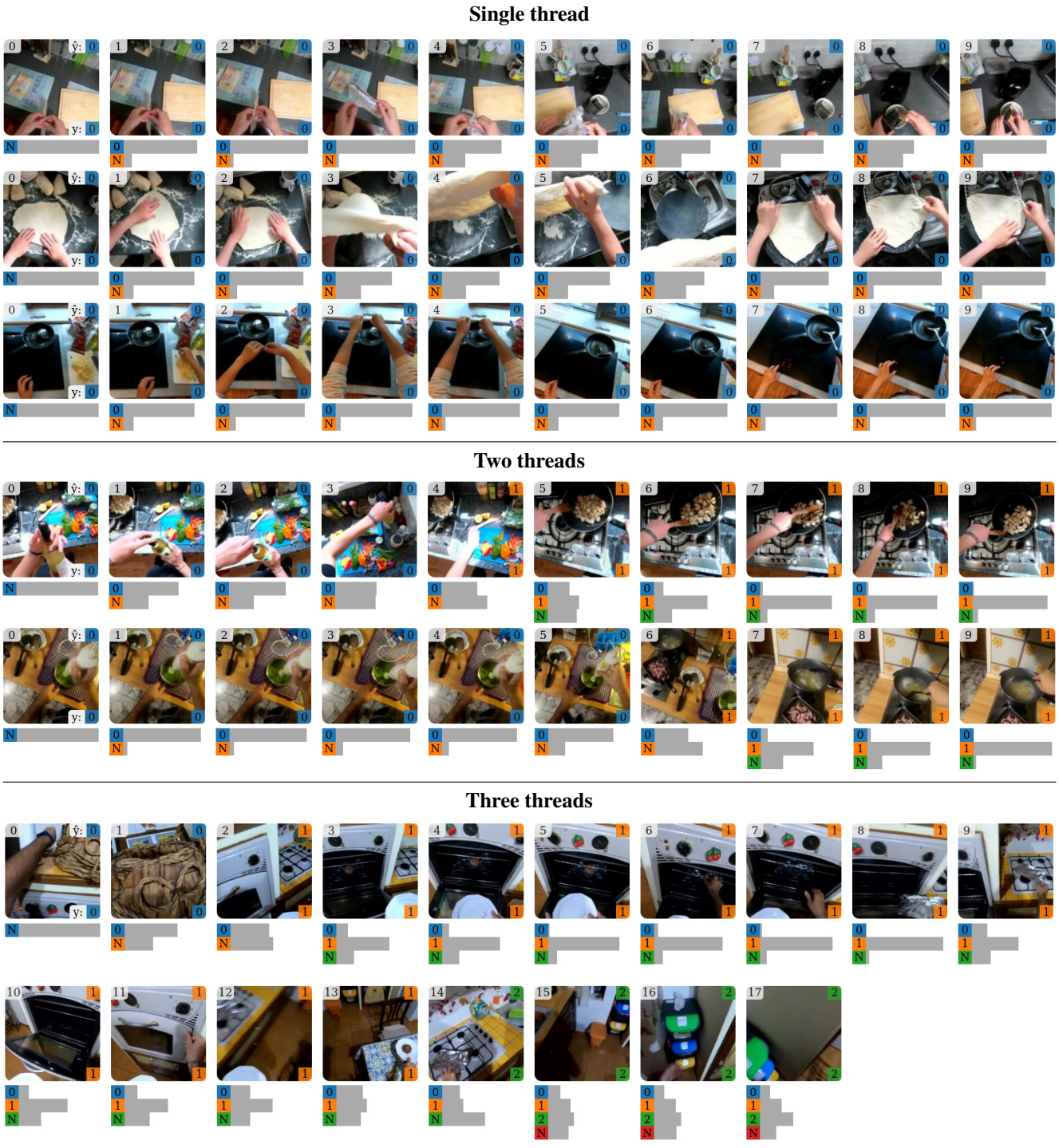

Figure 12. Additional examples demonstrating UnweaveNet successfully unweaving videos. See Fig. 6 for a legend. **Single thread (top)** row one: preparing grated cheese; row two: preparing pizza dough; row three: stir frying herbs. **Two threads (middle)**: row one: closing and putting away bottle, stirring contents of pan; row two: blending soup, stirring pasta. **Three threads (bottom)**: turning oven off, serving baked potato, recycling used tin foil.

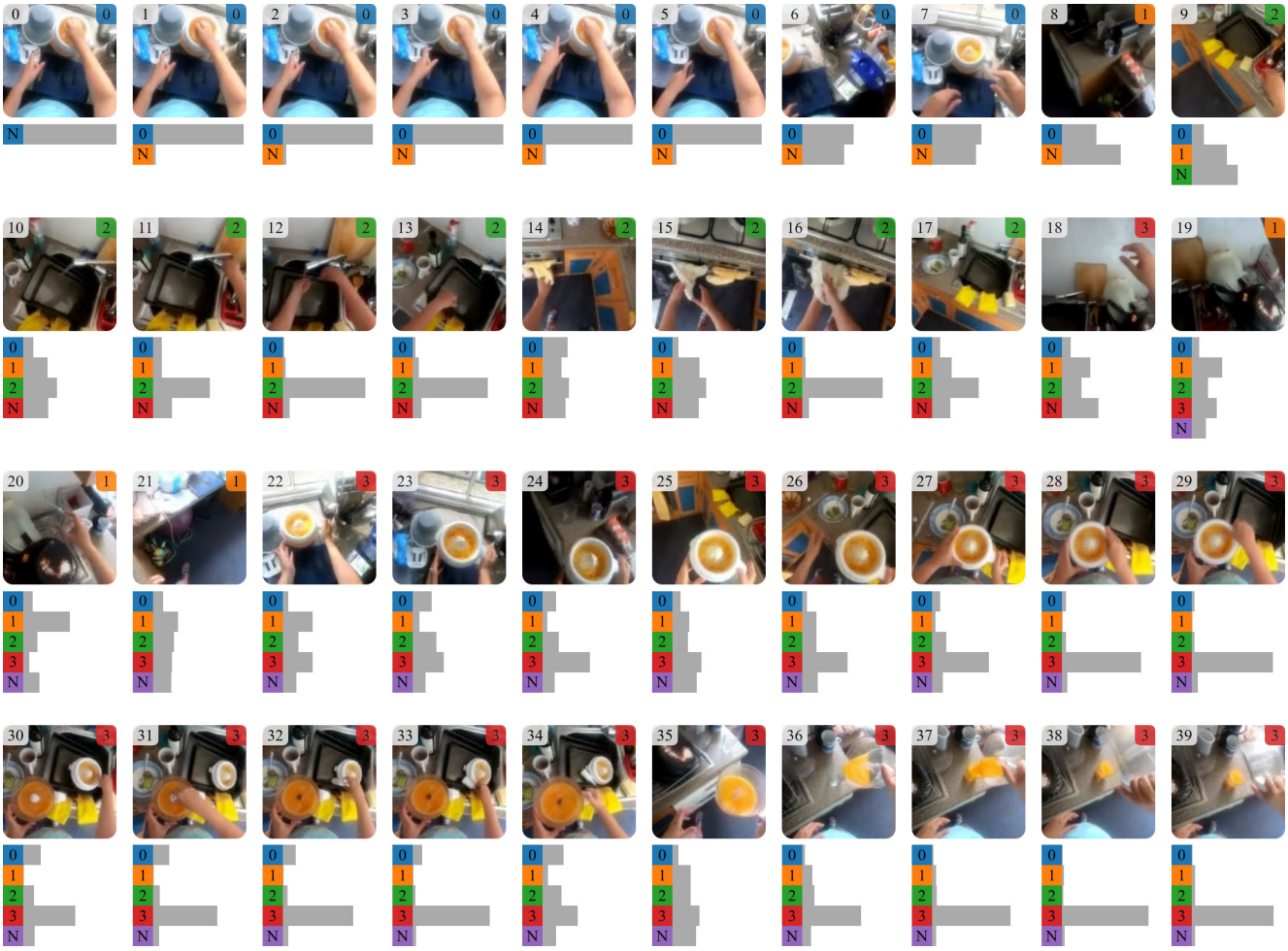

Figure 13. Untruncated version of Fig. 7. UnweaveNet represents this video as 4 threads: juicing the oranges (0–7), washing hands (9–17), getting a glass (19–21), and removing the juicer and serving the orange juice (22–39).

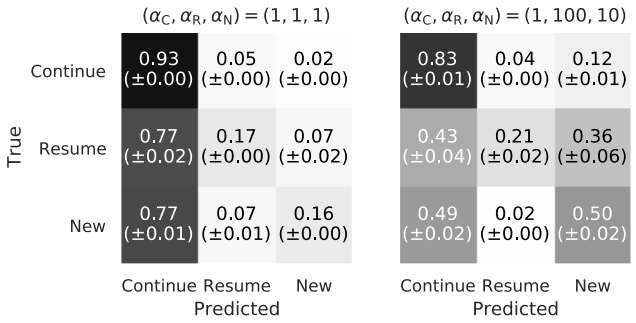

Figure 14. Scenario confusion matrix as we vary the scenario's weight α_s used in the loss when using teacher forced history.

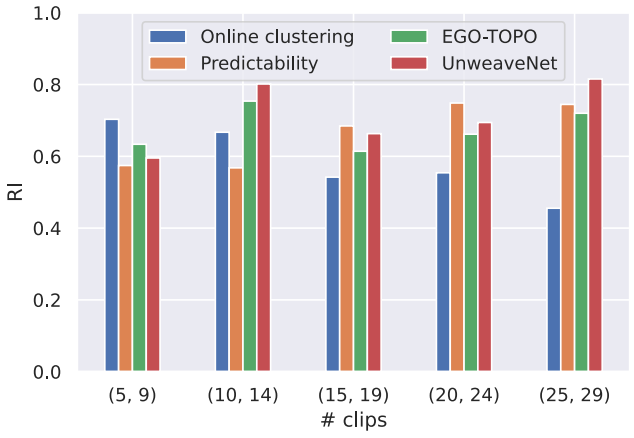

Figure 15. RI for binned stories by number of clips. Plot highlights UnweaveNet is robust to length of stories.