

# Unsupervised Discovery of Actions in Instructional Videos – Supplemental material

AJ Piergiovanni<sup>1</sup>  
ajpiergi@google.com  
Anelia Angelova<sup>1</sup>  
anelia@google.com  
Michael S. Ryoo<sup>2</sup>  
mryoo@google.com  
Irfan Essa<sup>1</sup>

<sup>1</sup> Google Research  
<sup>2</sup> Robotics at Google

## 1 Supplemental experimental results

Figure 1 visualizes the segmentations inferred by our approach when the number of actions varies from 5 to 11 for the tire changing activity of the NIV dataset. As seen, even when the number of actions is unknown, the algorithm provides meaningful segmentations, for example, by splitting larger actions in two. The boundaries of the automatically segmented actions seem to be consistent with the boundaries of the ground truth actions, as well.

In Table 1, we report the quantitative results corresponding to Figure 7 in the main paper, for future reference. The experiment is done by varying the number of actions to segment the video into and is for the NIV dataset. We find that the model is not overly-sensitive to this hyper-parameter, and that the results are improved, the closer the number of actions is to the ground truth number of actions. Thus, measuring the performance on a validation set will be a good proxy for selecting the best number of actions per dataset.

Figure 2 shows the confusion matrix for the 50 Salads dataset. As seen, actions are well separated from one another. There is confusion among objects (top left portion), e.g., ‘cut

© 2021. The copyright of this document resides with its authors.  
It may be distributed unchanged freely in print or electronic forms.

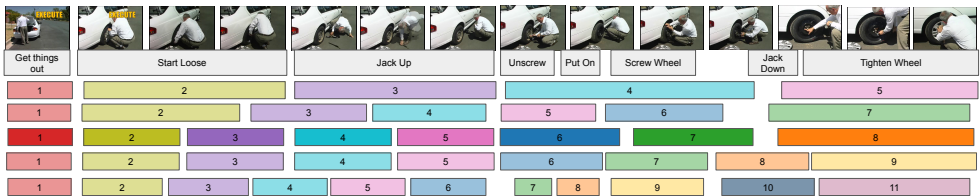


Figure 1: Example segmentation of the change tire activity varying the number of actions from 5 to 11. The segmentations generally match, even when the number of actions does not match the ground truth number.

# Steps	Change Tire (11)	CPR (7)	Repot Plant (7)	Make Coffee (10)	Jump Car (12)
GT Steps	0.60	0.52	0.32	0.37	0.29
5	0.45	0.48	0.25	0.25	0.12
7	0.48	0.52	0.32	0.30	0.18
9	0.54	0.52	0.32	0.35	0.22
11	0.60	0.50	0.30	0.34	0.27
13	0.58	0.48	0.28	0.35	0.26

Table 1: Varying the number of steps used in the model. The number in parenthesis indicates the ground-truth number of steps for each activity. NIV Dataset.

Oracle	Accuracy %
None	33.4
Action	39.8
High-level action	34.8
Temporal	36.8
Ordering	39.4
Object	48.5

Table 2: Comparison of different oracles on 50-salads

cucumber’, ‘cut tomato’ and ‘cut lettuce’ are confused, but actions, e.g., ‘cut’ and ‘peel’ are well separated. This confirms actions are well understood by the model.

## 1.1 Ablation experiments using oracles

To better understand where the model succeeds and fails, we compare effects of adding different oracle information.

First, we compare our model using 5 different oracles. (1) Object oracle that tells us the object of interest for each frame. For example, if cutting cucumber is the action, cucumber is the given object; for frying an egg, egg is given. (2) Action oracle (e.g., cut, peel, mix, etc.). (3) High-level action oracle, i.e., a grouping of several relation actions. For example: ‘prepare salad’ which contains cut, peel, add to bowl actions. (4) Time oracle which gives the start and end times of each action, but not the action itself. (5) Order oracle: gives the previous and next action, but not the current action (only usable for classifying the current frame).

The results are shown in Table 2 for the 50-salads dataset. We find that the model performs quite well in learning the temporal ordering and structure of the data, as the performance only slightly increases when including additional temporal information. Adding perfect object information greatly boosts performance, suggesting that the current model struggles to learn objects.

Furthermore, we use a Few-Shot Labeling Oracle to understand the performance of the

# labeled	0	1	2	3	4	5	50
Acc. %	33.4	42.8	44.3	45.2	46.6	47.1	77.6

Table 3: Classification accuracy for different number of labeled examples (50 labels means all examples are labeled). 50-salads.

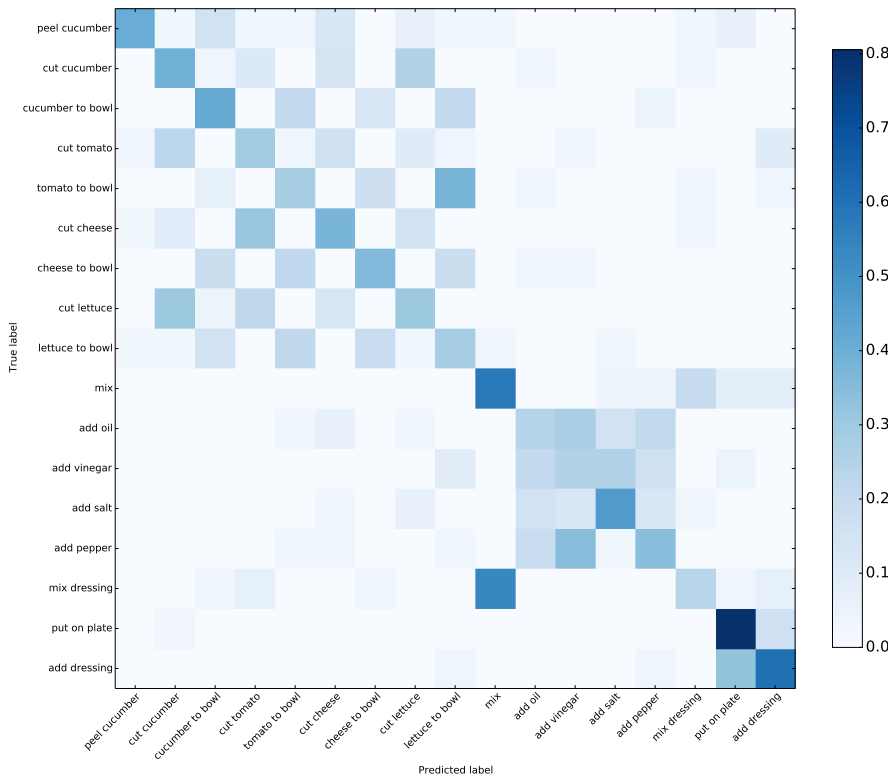


Figure 2: Confusion matrix for the 50 salads dataset. Most of the confusion is around the objects, e.g., ‘cut lettuce’ vs. ‘cut tomato’ and not with the activities themselves, e.g. ‘cut’ and ‘peel’ are not confused.

model. Here we have an oracle that gives  $N$  true examples and the model ‘mines’ the action from the other videos. This allows further analysis of the impact of unsupervised learning. We conduct a set of experiments comparing the unsupervised approach against  $N$  fully-labeled videos given.  $N$  videos are selected at random for supervised learning. The we perform the iterative, unsupervised training method for the remaining videos. The results are averaged over 10 different runs, each with a different set of labeled videos. Table 3 shows the results. We find that adding one true video greatly boosts performance (+9%), and each additional video adds only about 1% to fully supervised performance, showing the strong benefit of the self-labeling approach.

## 2 Method details

### 2.1 Action length equation

As mentioned in the paper, one constraint can require that the average lengths are similar. So we can compute the difference in length compared to the average action length. Computing the average action length cost function can be done as (following the notation from the paper):

$$C_2(S) = \sqrt{\frac{1}{|\mathcal{O}|} \sum_{a \in \mathcal{O}} \left( L(a, S) - \frac{1}{|\mathcal{O}|} (\sum_{i \in \mathcal{O}} L(i, S)) \right)^2}, \quad (1)$$

where  $L(a, S)$  computes the length (i.e., number of frames) labeled as action  $a$  in sequence  $S$ . This function will be minimized when all actions occur for equal number of frames.

### 2.2 Cross-Video Matching details

We here detail the losses or constraints used when learning across videos. A cross-video matching term is added to encourage the model to generalize from other videos, and is especially useful in videos which have the same actions but in partially ordered sequences (e.g., break egg, heat pan vs. heat pan, break egg).

Given a video segment the model labeled as an action  $f_a$  from one video, a segment  $\hat{f}_a$  the model labeled as the same action from a second video, and a segment  $f_b$  the modeled labeled as a different action from any video, we can measure the cross-video similarity using standard methods, such as a triplet loss

$$\mathcal{L}_T(f_a, \hat{f}_a, f_b) = \|f_a - \hat{f}_a\|_2 - \|f_a - f_b\|_2 + \alpha, \quad (2)$$

or a contrastive loss

$$\mathcal{L}_C(f_a, \hat{f}_a, f_b) = \frac{1}{2} \|f_a - \hat{f}_a\|_2 + \frac{1}{2} \max(0, \alpha - \|f_a - f_b\|_2). \quad (3)$$

These two functions capture similar properties but in slightly different ways. The triplet loss maximizes the difference between anchor (e.g.,  $f_a$ ) and positive ( $\hat{f}_a$ )/negative ( $f_b$ ) distance. While the contrastive loss maximizes the distance between  $f_a$  and  $f_b$  separately from minimizing the distance between  $f_a$  and  $\hat{f}_a$ . This results in slightly different cross-video matching metrics.

Function	Use	NIV	50-Salads	Brkfst	Method	Lrn	NIV	50-Salads	Brkfst
None	N/A	0.420	33.4	31.7	Avg.	no	0.420	33.4	31.7
triplet	cost	0.485	37.8	37.8	Gaussian	no	0.418	32.8	34.5
contr.	cost	0.478	37.9	36.4	Poisson	no	0.435	33.6	32.8
triplet	loss	0.492	38.4	37.5	Gaussian	yes	0.432	35.7	36.5
contr.	loss	0.478	39.2	38.4	Poisson	yes	0.447	37.9	37.9
triplet	both	0.442	35.7	36.9					
contr.	both	0.448	36.2	35.2					

Table 4: Cross video matching<sup>3</sup>Table 5: Different length models, including learned and fixed lengths per action.<sup>1</sup>

As these functions are differentiable, we can directly add this to the loss function (Eq. 4 in the main paper) or to the cost function (e.g. as an additional cost term similar to  $C_2$  in Eq. 2 in the main paper). It can also be added to both. By adding this to the cost function, we are ensuring that the chosen labeling of the videos is most consistent for feature representations. By adding it to the loss function, we are encouraging the learned representations to be similar for the actions with the same selected labels and different for other actions. We analyze the effect of these in Table 4.

**Methods for cross-video matching.** In Table 4, we compare the results for the different methods of cross-video matching on the 50-salads dataset. We compare both the triplet loss and the contrastive loss (see sup. material) using them as part of the cost function, training loss function or both. We find that using the contrastive as part of the training loss performs the best, as this further encourages the learned representation to match the chosen labels.

**Methods for length modeling.** In Table 5, we compare the different methods to model the length of each action. We find that learning the length of each action is most beneficial.

### 3 Implementation Details

The model is implemented in PyTorch. The pretrained models are on Kinetics-600 with overlapping classes removed, as is standard practice for unsupervised approaches (see Section 5 below for all removed classes). As base networks, which are needed to obtain initial features from the videos, we use and compare VGG [2], I3D [2] and AssembleNet [6]. These cover a wide range of networks previously used for video understanding (e.g. I3D), for unsupervised video segmentation, where VGG is often used, and current state-of-the-art models (AssembleNet). Our main model uses the AssembleNet backbone as a more contemporary model, which contains ResNet blocks of interleaved spatial and 1d temporal convolutions. It is equivalent in the number of parameters to a ResNet-50 (2+1)D network.

We used all three constraints ( $C_1$ ,  $C_2$ ,  $C_3$ ) with the weights set as described in Section 3.2 of the main paper. We used cross-video matching in the loss function with the triplet loss formulation. We used the learned Poisson version of length modeling. These corresponded to the best values in each of Tables 4 and 5 here and Table 5 of the paper.

During evaluation, we use a greedy rule selection method to pick the rule at each time step, so only one sequence is generated for each sample. We note that other methods are possible, such as generating multiple sequences and picking the best one. Using the greedy method, it is possible that it generates missing or repeated actions. However, since the cost function is not used during evaluation, we observe that this has minimal impact on the model.

<sup>1</sup>To isolate the effect, Table 4 uses the length model without learning, and Table 5 uses no cross video matching.

Method	Model Length (sec)	GT Length (sec)
Baseline Avg.	18	15
Gaussian (no learning)	16	15
Gaussian (learned)	14	15
Poisson (no learning)	16	15
Poisson (learned)	13	15

Table 6: Statistics on the learned average action length and GT length on the NIV dataset.

**Input Features.** In the experiments, as mentioned, we use VGG, I3D and AssembleNet as initial features. VGG and I3D use RGB inputs, while AssembleNet (by network design) uses RGB and optical flow as input. The optical flow is computed over RGB inputs on the fly. The CTC and ECTC methods, which are also compared in the paper, use IDT features [8] features on the 50-salads and Breakfast datasets and AssembleNet on the NIV dataset, unless otherwise noted.

**Specific Model Details.** We provide specific details about the model size. For the various experiments,  $|\mathcal{H}|$ , the size of the set of states, was set to 50 for all experiments and datasets. Changing this value did not significantly impact performance as long as it was greater than the expected number of outputs  $|\mathcal{O}|$ .  $\mathcal{R}$ , the set of transition rules, was set to 3 per-state, a total of 150, which is again fixed for all experiments. We use this strategy to be consistent across experiments; this can be further tuned for specific dataset to improve performance. We set  $M = 32$ , we note that we found the model was not sensitive to this setting, provided it was larger than 8.

**Evaluation Metrics details.** We follow all previously established protocols for evaluation in each dataset. We first use the Hungarian algorithm to map the predicted action symbols to action classes in the ground truth. Since different metrics are used for different datasets we report the previously adopted metrics per dataset. Specifically, for NIV, we predict a temporal interval for each action, then compute the F1 score if the interval falls within a ground truth interval (following [10]). For 50-salads, we compute the mean-over-frames (MoF) which is the per-frame accuracy for each frame. For Breakfast, we report both the MoF and Jaccard measure, following previous works [8, 9, 10].

## 4 Model Prediction Analysis

To show the model is not learning trivial solutions, here we provide some analysis of the model predictions. In Table 6 and 7, we show the difference between the predicted length and true action length.

## 5 Excluded Kinetics classes

We removed some classes from the Kinetics dataset, used to pretrain the models to obtain the initial features, in order to avoid overlap with the actions we are trying to discover. We also provide a list of some similar classes we left in Kinetics.

1. cooking egg
2. scrambling eggs

Method	Action	Model Length (sec)	GT Length (sec)
Gaussian (learned)	Take Plant	22	18
Gaussian (learned)	Use Soil	8	13
Gaussian (learned)	Loosen Roots	12	16
Gaussian (learned)	Place Plant	7	5
Gaussian (learned)	Add Soil	17	22
Gaussian (learned)	Water Plant	15	15

Table 7: Statistics on the learned average action length and GT length on the Repot a plant action in NIV dataset. Each action has a different length and the Gaussian model is able to learn this.

3. preparing salad
4. making a sandwich

Similar actions left in:

1. peeling apples/potatoes (similar to 50-salads peeling cucumber)
2. cutting apple/orange/watermelon/pineapple (similar to 50-salads cutting cucumber, cutting tomato, and cutting cheese)
3. changing wheel (similar to NIV changing a tire)
4. planting trees (similar to NIV repotting plant)
5. frying vegetables (similar to Breakfast frying an egg)

## References

- [1] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4575–4583, 2016.
- [2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [3] De-An Huang, Li Fei-Fei, and Juan Carlos Nieves. Connectionist temporal modeling for weakly supervised action labeling. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [4] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12066–12074, 2019.
- [5] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [6] Michael S. Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SJgMK64Ywr>.
- [7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [8] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE, 2011.