A. Pseudo-code for ASOT

In Alg. 1, we present pseudo-code around the numerical solver for our action segmentation optimal transport (ASOT) algorithm. Our approach is based on projected mirror descent, similar to Peyre et al. [30]. We initialize the coupling $\mathbf{T}$ element-wise as $\mathbf{T}_{ij} = 1/(NK)$. Each iteration then involves two steps, given by

1. **Update step (Lines 6–16):** The first-order update step is computed as a mirror descent step under the KL-divergence. The update step at iteration t is given by $\hat{\mathbf{T}}^{t+1} = \mathbf{T}^t \odot \exp(-\phi \nabla^t_{\mathbf{T}} \mathcal{F}_{\text{ASOT}}(\mathbf{C}, \mathbf{T}^t))$, where $\phi > 0$ is a step size and

$$\mathcal{F}_{\text{ASOT}}(\mathbf{C}, \mathbf{T}, \epsilon) := \mathcal{F}_{\text{FGW}}(\mathbf{C}, \mathbf{T}) + \lambda \mathrm{D}_{\text{KL}}(\mathbf{T}^\top \mathbf{1}_n \| \mathbf{q}) - \epsilon H(\mathbf{T}). \quad (8)$$

2. **Projection step (Lines 10–15):** The projection step involves ensuring the polytope constraints are satisfied. This involves simply rescaling the rows of $\mathbf{T}$, to satisfy the frame-wise marginal constraint, i.e., $T^{t+1}_{ij} = p_i \hat{T}^{t+1}_{ij} / (\hat{\mathbf{T}}^{t+1} \mathbf{1}_K)_i$.

Algorithm 1 Action Segmentation Optimal Transport

Input: Video frame to action class affinity matrix $\mathbf{C}^k \in \mathbb{R}^{N \times K}$, derived from a video encoder and hyperparameters $\alpha \in [0,1]$, $r \in [0,1]$, $\lambda \geq 0$, $\epsilon > 0$, $\phi > 0$, $n_{\text{iter}} > 0$.

Output: Soft assignment probabilities $\mathbf{T}^\star \in \mathbb{R}_+^{N \times K}$.

1: **function** ASOT($\mathbf{C}^k$, α, r, λ, ϵ, ϕ, n_{iter})
2: $\quad \mathbf{p} \leftarrow \frac{1}{N} \mathbf{1}_N$ ▷ Frame marginals
3: $\quad \mathbf{q} \leftarrow \frac{1}{K} \mathbf{1}_K$ ▷ Balanced action marginals
4: $\quad$ Construct $\mathbf{C}^v$, $\mathbf{C}^a$ using (5).
5: $\quad \mathbf{C} = \{\mathbf{C}^k, \mathbf{C}^v, \mathbf{C}^a\}$
6: $\quad \mathbf{T}^0 \leftarrow \mathbf{p} \otimes \mathbf{q}$ ▷ $\otimes$ denotes the outer product
7: $\quad$ **for** $t \leftarrow 0$ to $n_{\text{iter}} - 1$ **do**
8: $\quad\quad \hat{\mathbf{T}}^{t+1} = \mathbf{T}^t \odot \exp(-\phi \nabla_{\mathbf{T}} \mathcal{F}_{\text{ASOT}}(\mathbf{C}, \mathbf{T}^t))$
9: $\quad\quad T^{t+1}_{ij} = p_i \hat{T}^{t+1}_{ij} / (\hat{\mathbf{T}}^{t+1} \mathbf{1}_K)_i \quad \forall i, j$
10: $\quad$ **end for**
11: $\quad$ **return** $\mathbf{T}^{n_{\text{iter}}}$
12: **end function**

We provide a full reference implementation for ASOT and training code at https://github.com/mingu6/action_seg_ot.

B. Additional Details for Unsupervised Experiment

For both the training (pseudo-labelling) and inference phases, we use 25 iterations for projected mirror descent (see Sec. A). We use the same Gromov-Wasserstein (GW) radius parameter r across training and inference, which is set at 0.04 for all datasets except for desktop assembly, which is set at 0.02. We observed that desktop assembly has smaller ground truth segments relative to video length, benefiting a lower value for r.

ASOT pseudo-labelling. The ASOT hyperparameters used to generate pseudo-labels during the training phase of our unsupervised action segmentation pipeline are described as follows. The GW structure weight α is set to 0.3 for all datasets except for Breakfast where $\alpha = 0.4$ and furthermore, entropy regularization weight $\epsilon = 0.07$ for all datasets. Table 4 presents the remaining hyperparameters.

ASOT inference. The hyperparameters used to generate segmentations from embeddings learned from our unsupervised learning pipeline are described as follows. First, we set the unbalanced weight λ to a low value (0.01) for all datasets. At inference, ASOT will segment according to the underlying learned embeddings, and will not encourage a more balanced assignment to action classes. Second, we set $\alpha = 0.6$ for all datasets except for Breakfast where $\alpha = 0.7$, because a higher value for α results in stronger temporal consistency due to heavier weighting of the GW structure objective. Finally, we set $\epsilon = 0.04$, since lower levels of entropy regularization yields sharper segmentations.

Representation learning. We set the learning rate at 10^{-3} and weight decay to 10^{-4} for all datasets. We sample a batch of 2 videos, where for each video, we sample 256 frames randomly by partitioning each video into 256 uniform intervals and sampling a single frame from each interval. Our MLP architecture has a single hidden layer with ReLU activations. The hidden layer size is 128 for all datasets, with an output feature dimension of 40, with the exception of YouTube Instructions (YTI), which have sizes 32 and 32, respectively. Output features are l_2-normalized before applying ASOT. The smaller model size from YTI arises from the significantly higher dimensional input features compared to the remaining datasets.

	BF	YTI	FS (M)	FS (E)	DA
Unbalanced (λ)	0.1	0.12	0.15	0.11	0.16
Global prior (ρ)	0.2	0.15	0.15	0.15	0.25
Num. epochs	15	10	30	30	30

Table 4. Hyperparameter settings for ASOT used to generate pseudo-labels in the *training phase* of our unsupervised action segmentation pipeline. FS (M) and FS (E) represent the 50 Salads Mid and Eval splits, respectively.

Dataset	50 Salads (Mid)	50 Salads (Eval)	Desktop Ass.
MoF (%)	(43.0, 1.9)	(56.4, 2.8)	(62.9, 2.8)

Table 5. MoF results for five runs recorded as (mean, std. err.).

C. Experimental Setup (Supervised)

For the supervised example, the MS-TCN [28] architecture outputs logits $\mathbf{L} \in \mathbb{R}^{N \times K}$ directly instead of video frame and action embeddings. We can transform these logits into a cost matrix $\mathbf{C}^k$ using

$$C_{ij}^k = 2\left(1 - \frac{L_{ij} - L_{\min}}{L_{\max} - L_{\min}}\right), \tag{9}$$

where $L_{\max} \coloneqq \max_{i,j} L_{ij}$ and $L_{\min} \coloneqq \min_{i,j} L_{ij}$. This is a simple method for converting the logits which model a frame/action affinity into a cost matrix with elements scaled between $[0, 2]$.

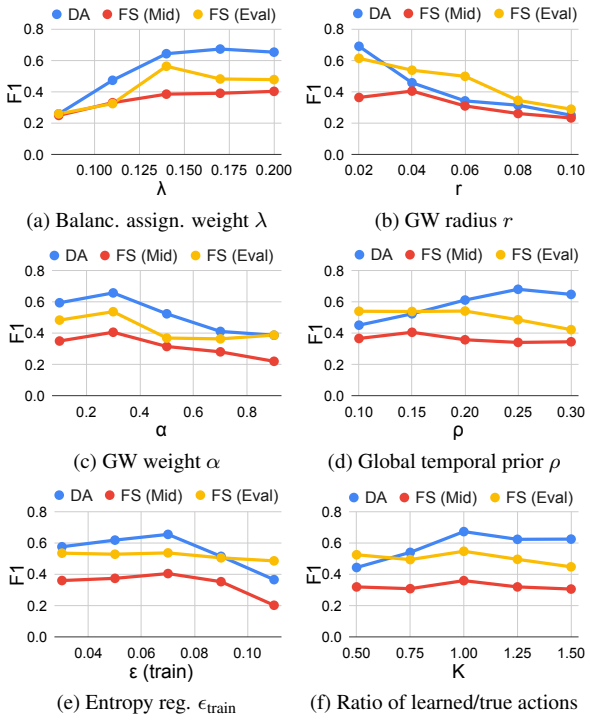
We then apply ASOT to the cost matrix derived per video with hyperparameters $\lambda = 0.05$, $\alpha = 0.4$, $\epsilon = 0.06$ and $r = 0.01$. We evaluate our method using 5-fold cross validation, similar to [28].

D. Additional Sensitivity Analysis

In addition to the sensitivity analysis under the MoF metric presented in Fig. 5, we present additional results for F1-score and mIoU in Fig. 7 and 8, respectively. The results follow very similar trends to MoF, however it is notable in Fig. 8a that mIoU performance collapses to almost 0 for $\lambda = 0$, whereas MoF does not (see Fig. 5a). This is especially notable for the FS (Eval) dataset. This discrepancy between MoF and mIoU suggests the presence of unbalanced classes within the datasets, especially FS (Eval).

E. Reproducibility

Our results in Tab. 1 and 2 were produced using one run, consistent with prior works in unsupervised action segmentation [22, 23, 26, 36, 40, 42]. To show the robustness of our methods, we used five runs on the 50 Salads and Desktop Assembly datasets with different random seeds. Differing random seeds impact the network initialization and sampling of batch indices for our training pipeline. We report the results of this experiment in Table (mean, std. err.) for the MoF metric (cf. Tab. 5).

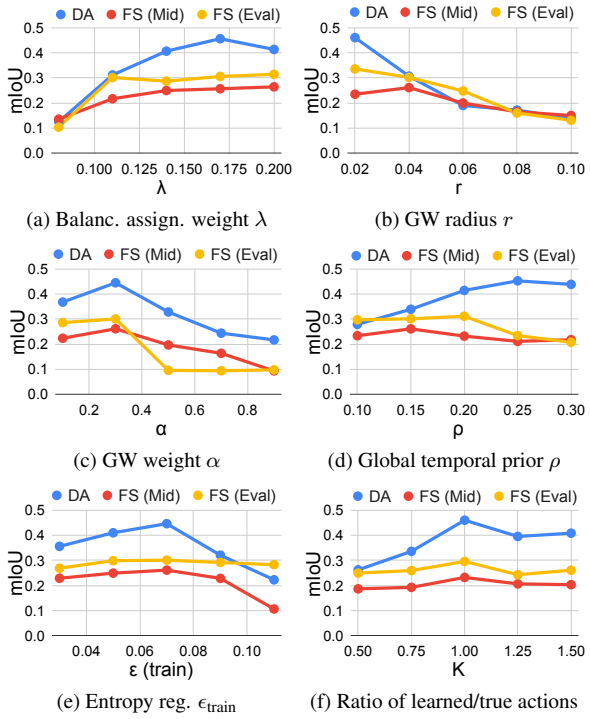


(a) Balanc. assign. weight λ (b) GW radius r

(c) GW weight α (d) Global temporal prior ρ

(e) Entropy reg. ϵ_{train} (f) Ratio of learned/true actions

Figure 7. Sensitivity analysis reporting F1-score on ASOT hyperparameters.



(a) Balanc. assign. weight λ (b) GW radius r

(c) GW weight α (d) Global temporal prior ρ

(e) Entropy reg. ϵ_{train} (f) Ratio of learned/true actions

Figure 8. Sensitivity analysis reporting mIoU-score on ASOT hyperparameters.