# Appendix

## A. Dual Optimization Derivation

We provide a complete proof of the dual problem derivation as mentioned in section 3.3 in our manuscript. We will start with a simpler problem without introducing the demonstration weights and then we will add them to the problem as we go on.

The goal here is to derive the dual problem for calculating the model $p(t|s)$ by choosing the model that has the maximum entropy with the FEM as a constraint.

$$\begin{aligned} \max_{p(t|s)\in\mathbb{R}^{\mathcal{S}\times\mathcal{T}}} \quad & H(p(t|s)) \equiv -\sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} \tilde{p}(s)p(t|s)\log p(t|s) \\ \text{s.t.} \quad & \mathbb{E}_p[f_i] - \mathbb{E}_{\tilde{p}}[f_i] = 0 \quad i = 1,\dots,n \\ & \sum_{t\in\mathcal{T}} p(t|s) - 1 = 0 \quad \forall\, s \in \mathcal{S} \end{aligned}$$

To derive the dual problem we will use the Lagrange method for convex optimization problems.

$$\Lambda(p,\lambda,\mu) \equiv H(p) + \sum_{i=1}^{N}\lambda_i\Big(\mathbb{E}_p[f_i] - \mathbb{E}_{\tilde{p}}[f_i]\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s\Big(\sum_{t\in\mathcal{T}} p(t|s) - 1\Big)$$

Where $\lambda_i$, $\mu_s$ are the Lagrangian's multiplier corresponding to each constraint.

$$\Lambda(p,\lambda,\mu) \equiv -\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{t\in\mathcal{T}} p(t\mid s)\log p(t\mid s) + \sum_{i=1}^{N}\lambda_i\Big(\mathbb{E}_p[f_i] - \mathbb{E}_{\tilde{p}}[f_i]\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s\Big(\sum_{t\in\mathcal{T}} p(t|s) - 1\Big) \tag{1}$$

By Differentiating the Lagrangian with respect to primal variables $p(s|t)$ and letting them be zero, we obtain:

$$\frac{\partial\Lambda}{\partial p(t|s)} = -\sum_{s\in\mathcal{S}}\tilde{p}(s)\Big(1 + \sum_{t\in\mathcal{T}}\log p(t|s)\Big) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{t\in\mathcal{T}} f(s,t)\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s \tag{2}$$

$$-\sum_{s\in\mathcal{S}}\tilde{p}(s)\Big(1 + \sum_{t\in\mathcal{T}}\log p(t|s)\Big) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{t\in\mathcal{T}} f(s,t)\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s = 0 \tag{3}$$

$$\sum_{s\in\mathcal{S}}\tilde{p}(s)\left(-1 - \sum_{t\in\mathcal{T}}\log p(t|s) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{t\in\mathcal{T}} f(s,t)\Big) + \mu_s\right) = 0 \tag{4}$$

Assuming $\tilde{p}(s) \neq 0$,

$$\log p(t|s) = \sum_{i=1}^{N}\lambda_i\Big(f_i(s,t)\Big) + \mu_s - 1 \tag{5}$$

$$p(t|s) = \exp\left(\sum_{i=1}^{N}\lambda_i\Big(f_i(s,t)\Big)\right)\cdot\exp\Big(\mu_s - 1\Big) \tag{6}$$

Since $\sum_{t\in\mathcal{T}} p(t|s) = 1$

$$\sum_{t\in\mathcal{T}} \exp\Big(\sum_{i=1}^{N} \lambda_i\Big(f_i(s,t)\Big)\Big) \cdot \exp\Big(\mu_s - 1\Big) = 1 \tag{7}$$

$$\frac{1}{\sum_{t\in\mathcal{T}} \exp\Big(\sum_{i=1}^{N} \lambda_i\Big(f_i(s,t)\Big)\Big)} = \exp\Big(\mu_s - 1\Big) = (z_\lambda(s))^{-1} \tag{8}$$

By substituting in Eq. 6 we will get the following Eq. which we are using to predict the next sub-task distribution based on the current state features.

$$p^*(t|s) = (z_\lambda(s))^{-1} \cdot \exp\Big(\sum_{i=1}^{N} \lambda_i\Big(f_i(s,t)\Big)\Big) \tag{9}$$

Finally, the dual problem will be:

$$-\Big\{\max_{\lambda}\ \Lambda(\lambda) \equiv -\sum_{s\in\mathcal{S}} \tilde{p}(s)\log z_\lambda(s) + \sum_{i=1}^{N} \lambda_i \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} \tilde{p}(s,t) f(s,t)\Big\} \tag{10}$$

Now we will add the demonstration weights $w$ and have our empirical distributions be parameterized by it.

$$\begin{aligned} \tilde{p}_w(t|s) &= \frac{1}{M}\sum_{d=1}^{D} w_d \cdot \tilde{p}(t|s,d) \\ \tilde{p}_w(s) &= \frac{1}{M}\sum_{d=1}^{D} w_d \cdot \tilde{p}(s,d) \end{aligned} \tag{11}$$

Where $D$ is the total number of demonstrations, and $M$ should be $\sum_{d=1}^{D} w_d$. Which is the minimum number of demonstrations that we can trust in the given set.

We can start presenting the primal formulation by substituting from Eq. 11 to Eq. .

$$\begin{aligned} \min_{w\in\mathbb{R}^D} \max_{p(t|s)\in\mathbb{R}^{\mathcal{S}\times\mathcal{T}}} \quad H(p(t|s)) &= -\sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} p(t|s)\log p(t|s)\sum_{d=1}^{D} w_d\cdot\tilde{p}(s,d) \\ \text{s. t.} \quad & \sum_{d=1}^{D} w_d \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} f_i(s,t)\tilde{p}(s,d)\Big(p(t|s) - \tilde{p}(t|s,d)\Big) = 0, \quad i = 1,\dots,N \\ & \sum_{t\in\mathcal{T}} p(t|s) - 1 = 0, \quad \forall s\in\mathcal{S} \\ & \sum_{d=1}^{D} w_d = M, \quad w_d \geq 0, \quad \forall d\in\mathcal{D}, \quad w_d \leq 1 \quad \forall d = 1,\dots,D \end{aligned} \tag{12}$$

To solve this problem we will use the Lagrange multiplier as we did before and we can continue from Eq. 10 by adding the new weight terms from Eq. 11.

$$\min_{w} \quad -\left\{ \max_{\lambda} \quad \Lambda(\lambda) \equiv -\frac{1}{M}\sum_{d=1}^{D} w_d \sum_{s\in\mathcal{S}} \tilde{p}(s,d)\log z_\lambda(s) + \frac{1}{M}\sum_{d=1}^{D} w_d \sum_{i=1}^{N} \lambda_i \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} \tilde{p}(t|s,d) f(s,t) \right\}$$
$$\text{s.t.} \quad \sum_{d=1}^{D} w_d = M$$
$$w_d \geq 0 \quad \forall d \in \mathcal{D} = 1....D$$
$$w_d \leq 1 \quad \forall d \in \mathcal{D} = 1, \ldots, D \tag{13}$$

By moving the negative sign to inside we will reach our final optimization dual problem.

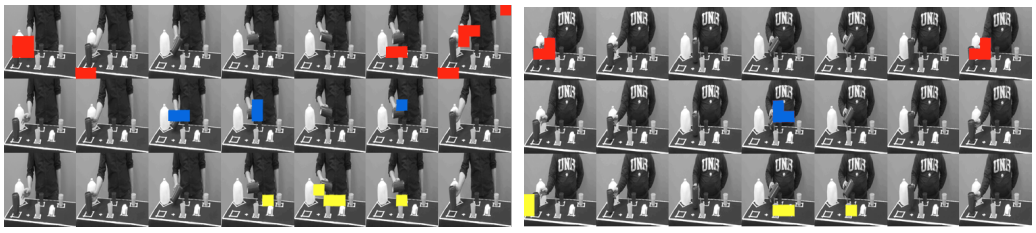
$$\min_{w\in\mathbb{R}^D, \lambda\in\mathbb{R}^N} \Lambda(\lambda, w) \equiv -\frac{1}{M}\sum_{d=1}^{D} w_d \Big( -\sum_{s\in\mathcal{S}} \tilde{p}(s,d)\log z_\lambda(s) + \sum_{i=1}^{N} \lambda_i \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} \tilde{p}(t|s,d) f(s,t) \Big)$$
$$\text{s. t.} \quad \sum_{d=1}^{D} w_d = M, \quad w_d \geq 0 \quad \forall d \in \mathcal{D}, \quad w_d \leq 1 \quad \forall d \in \mathcal{D} \tag{14}$$

Where $z_\lambda(s) = \sum_{t\in\mathcal{T}} \exp\left(\sum_{i=1}^{N} \lambda_i f_i(s,t)\right)$ is a normalization constant.

We solve the non-convex quadratic problem in Equation14 using the Sequential Quadratic Programming (SQP) algorithm [1]; The SQP algorithms generalize Newton's method for constrained optimization problems. In each iteration, the Hessian of the Lagrangian function is approximated in a quasi-Newton style. The algorithm then solves the resulting quadratic program and finds the next iteration using the line search procedure. This algorithm may not converge to the global minimum, but our experiments find it to perform very well.

## B. Visualization of Temporally Ranked Features

We provide a visualization and analysis of some of the features we generated and used in our model. We contrast some of the highly ranked temporally-grounded features and identified by our novel ranking system. We investigate two examples from four of the actions in our tea making task: 'add water' (Fig. 1), 'add sugar' (Fig. 2), 'stir' (Fig. 3), and 'add milk' (Fig. 4). In all figures the background video has been depicted in grey scale in order to highlight the location of feature expression.



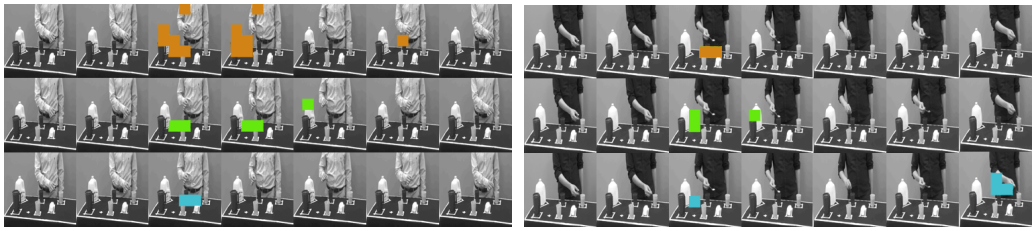
(a) Example 1 (b) Example 2

Figure 1: Visual analysis of the features in the context of the 'Add Water' action.

Temporally-grounded features captured more specific features related to the task being demonstrated. In the 'add water' task we see that the features capture picking up and returning the pitcher (red) and

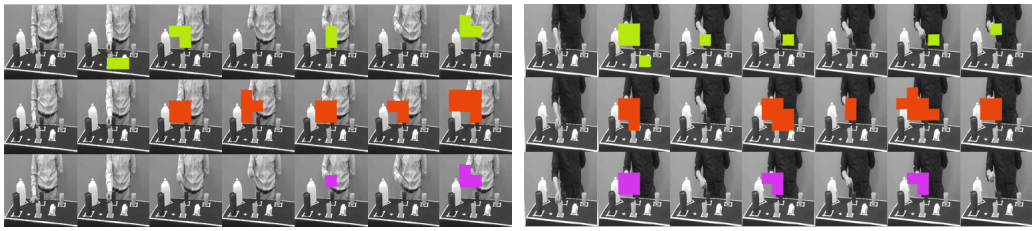
[1](Implementation:https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#bsgppl4)

features for the pouring of the water (blue and yellow). The features in the 'add sugar' action captured those frames where the individual deposited the spoon into the mug (blue) and the reorientation (green) and return of the spoon (orange). The temporally-grounded features in the 'stir' action were intermittently expressed as the stir action was performed (lime and purple) capturing the cyclical pattern of the participants hand.



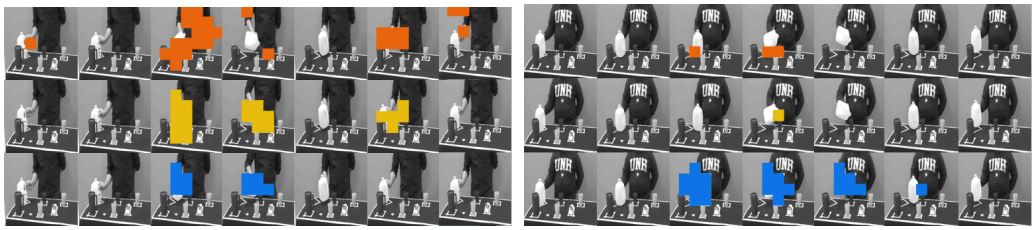
(a) Example 1 (b) Example 2

Figure 2: Visual analysis of the features in the context of the 'Add Sugar' action.



(a) Example 1 (b) Example 2

Figure 3: Visual analysis of the features in the context of the 'Stir' action.



(a) Example 1 (b) Example 2

Figure 4: Visual analysis of the features in the context of the 'Add Milk' action.

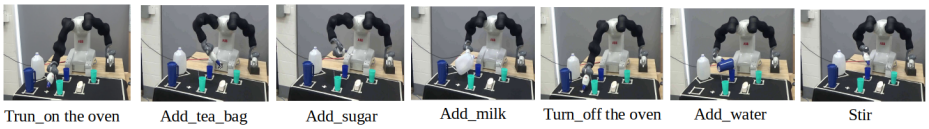
## C. Dataset Visualization Examples

We provide a visualization for the *tea-making* task, Fig. 5 Shows the three different sequences we used in our evaluation. Fig.5.a shows seq_1: Turn on the oven, Add tea bag, Add sugar, Add milk, Turn off the oven, Add water, Stir. Fig.5.b shows seq_2: Turn on the oven, Add sugar, Add milk, Add tea bag, Turn off oven, Add water, Stir. Fig.5.c shows seq_3: Turn on the oven, Add sugar, Add milk, Add tea bag, Turn off oven, Stir.

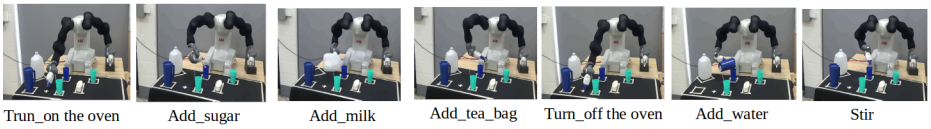
## D. Algorithm and Implementation Details

### D.1 Incorrect Demonstration Detection Pseudo Code

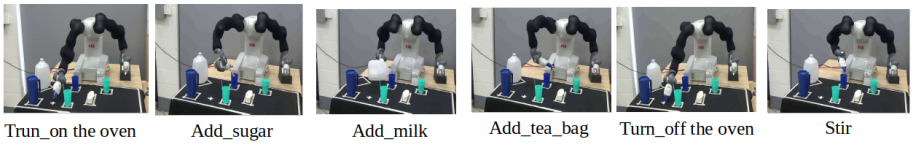
In this section, we present a pseudo-code for our framework presented in Algorithm 1. Starting from step1: by doing the semi-supervised video segmentation, to generate consistent labels without the need for manual labeling of each frame. Step2: Generate a set of ranked features represents each video segment. Step3: Calculate the required variables from the given dataset and solve the optimization equation.




(a) Representative frames from a Seq_1 demonstration.




(b) Representative frames from a Seq_2 demonstration.




(c) Representative frames from a Seq_3 demonstration.

Figure 5: Example frames from different types of video demonstrations.

**Algorithm 1** Incorrect visual demonstration framework

**Step 1: Sub-task segmentation**

$\{s, t_j\}_{j=1}^{q}$ ▷ Segment each video in the demonstration set $\mathcal{D}$ into segments with labels $t$ , the number of segments in each video is not fixed.

**Step 2: Feature generation**

**for** each video segment **do**

  $s = \{f_n\}_{n=1}^{N}, \{t_j\}_{j=1}^{q}$ ▷ From each video segment an $N$ ranked features are extracted and paired with the next sub-tasks label based on the sequence of each demonstration $t$.

**Step 3: Calculate the demonstrations weighs**

**for** each video $d$ **do**

  **for** each video segment $s$ **do**

    count how many times $s$ appears in each video $d$ to calculate $\tilde{p}(s, d)$.

    **for** each task $t$ **do**

      count how many times the next task $t$ will appear given $s, d$ to calculate $\tilde{p}(t|s, d)$

      **for** each feature in the feature set $f$ **do**

        build the feature matrix with shape $(f \times s \times t)$

Solve the optimization function in Equation 14 ▷ Matlab toolbox has been used to solve this problem [1]

Using the set of generated weights $w$ determine the class of the demonstrations using any clustering algorithms.

### D.2 Behavioral Cloning (BC, BC-RNN, R-MAXENT, DEMO-DICE)

Behavioral Cloning (BC) is a one of main method for learning a policy from a set of demonstrations. It trains a policy $\pi_x(s)$ to clone the actions in the dataset via the objective:

$$arg\min_x \mathbb{E}_{(s,a)\sim\mathcal{D}}||\pi_x(s) - a||^2 \tag{15}$$

BC-RNN is a strong variant of BC that uses a Recurrent Neural Network (RNN) as the policy network, which allows the policy to model temporal dependencies in the data through the recurrent hidden state. The network is trained on length-$T$ temporal sequences of data $(s_t, a_t, ..., s_{t+T}, a_{t+T})$. The network predicts the sequence of actions using the sequence of states as input. At test-time, the RNN policy network is unrolled one-step at a time, $a_t, h_{t+1} = \pi_x(s_t, h_t)$ where $h$ is the RNN hidden state. The hidden state is refreshed every $T$ steps. We levered the code code available for both of these approaches here [2].

R-MAXENT leverage the incorrect demonstrations for policy learning but assign poor weights to them in the learning process. We leveraged the available code here [3].

DEMO-DICE implementation is available by the author here [4].

## E. More Experimental Results

### E.1 Clustering of the Generated Weights Results

We are using a k-means algorithm [43] to cluster the weights and the threshold is chosen as the middle point between the resulting means. The number of clusters is preset to 2 in case of optimal/incorrect demonstrations and 3 in case optimal, sub-optimal, and incorrect videos. We did run two other clustering algorithms and all came with the same thresholds (Gaussian mixture model [43] and Mini-Batch K-Means [44]). Table 1 presents the results for calculating the threshold for one of the experiments where we have 2 type of demonstrations – correct and incorrect.

| Algorithm | Cluster_1 mean | Cluster_2 mean | Threshold |
|---|---|---|---|
| KNN | 0.28 | 0.8 | 0.52 |
| Gaussian mixture model | 0.27 | 0.8 | 0.52 |
| Mini-Batch K-Means | 0.28 | 0.8 | 0.52 |

Table 1: Clustering of weights generated by the proposed framework

### E.2 Robustness with respect to Video Segmentation Performance

To study the effect of the video segmentation accuracy over the performance of our framework, we ran our framework with video segmentation algorithms [38]. We investigated three data points: 100% accuracy achieved through manual video segmentation, 83% accuracy through the best performing segmentation algorithm in [38], and 75 % accuracy through the same algorithm in [38] but with less training. Table 2 shows that the accuracy of the proposed framework saturates with a reasonably good video segmentation performance (i.e. 83% and 100% ) but drops moderately with the performance of the video segmentation algorithm.

| Segmentation accuracy | Overall Framework accuracy |
|---|---|
| 100% | 90 % |
| 83% | 90% |
| 75% | 85% |

Table 2: Robustness with respect to the performance of the video segmentation algorithm

[2] https://robomimic.github.io/study/

[3] https://github.com/mostafa-hussein/max-ent

[4] https://github.com/KAIST-AILab/imitation-dice