OPEL: Optimal Transport Guided ProcedurE Learning

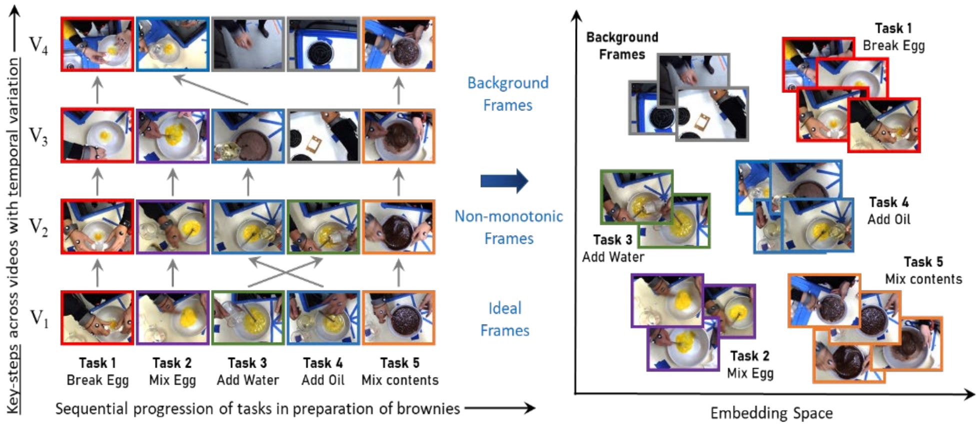

Figure R1: Key-steps required to prepare a brownie, showcasing temporal variations and corresponding key-step alignment challenges, namely (i) background frames (depicted as gray blocks), (ii) non-monotonic frames. OPEL learns an embedding space where corresponding key-steps have similar embeddings while tackling the above challenges.

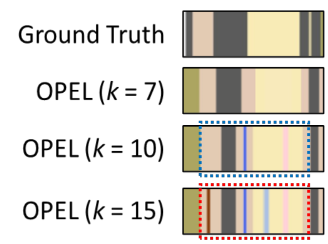

Figure R2: Qualitative analysis on variation of k.

Table R1: Additional ablation study on the effect of k

k	Meccano			EPIC-Tents		
	R	F	IoU	R	F	IoU
7	**60.5**	**39.2**	**20.2**	**23.0**	**20.7**	**10.6**
10	54.9	30.3	17.6	19.8	17.2	9.8
12	50.2	28.1	15.5	18.4	16.5	8.7
15	47.6	26.8	14.8	16.9	15.8	8.2

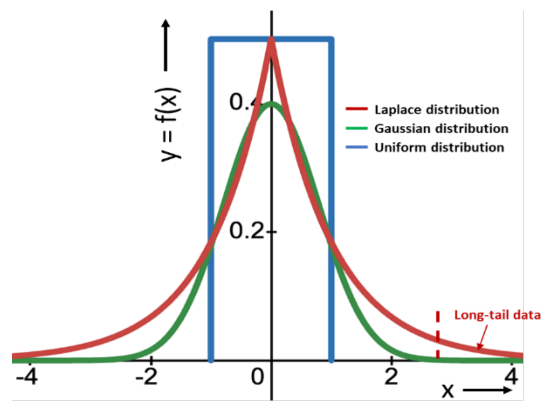

Figure R3: Importance of choosing Laplace distribution as prior.

Codeblock R1: Pytorch Function to determine the sequential ordering of tasks from frame-wise key-step predictions

```
def temporal_order(R, k):
# M: No. of frames
# R: Predicted key-steps of each frame
# k: No. of key-steps   # T: Normalized time
# indices: Final sequential order of task
 M = len(R)
 T = (torch.arange(0, M)+1)/M
 cluster_time = torch.zeros(k)

# Finding the mean time for each cluster and sorting
# them to obtain their sequential order
 for i in range(k):
   cluster_time[i] = T[R==i].mean()
 _, indices = torch.sort(cluster_time)
 return indices
```

Sample Input (R): tensor([6, 2, 1, 3, 5, 1, 1, 1, 1, 6, 0, 4, 6, 1, 1, 3, 0, 4, 0, 4, 5, 5, 5, 1, 3, 2, 0, 4, 3, 6, 0, 1, 2, 4, 2, 3, 5, 4, 6, 2, 5, 1, 2, 4, 3, 2, 2, 3, 4, 1])

Sample Output (indices): tensor([6, 1, 0, 5, 3, 4, 2])

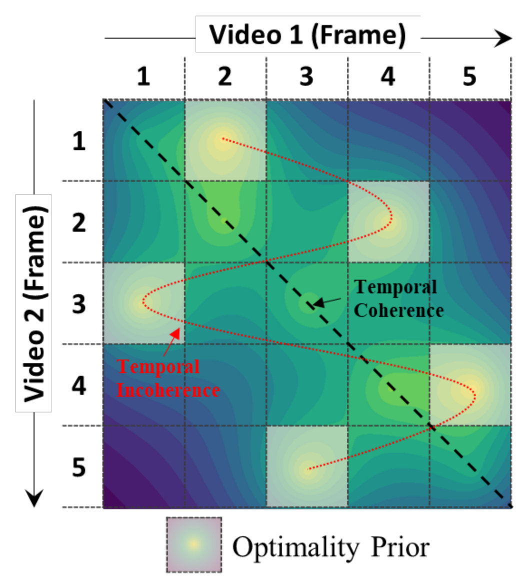

Figure R4: Predictions made only with optimal prior. Temporal prior is needed to preserve temporal coherence among videos.