

# DeSPITE: Exploring Contrastive Deep Skeleton-Pointcloud-IMU-Text Embeddings for Advanced Point Cloud Human Activity Understanding

## Supplementary Material

	LIPD-Babel-v1	LIPD-Babel-v2
#Sequences Train	502958	403430
#Sequences Test	85551	58802
#Text Train	187641	135699
#Text Test	-	58802

Table 4. Number of total training sequences and frame-wise text labels when considering 24 frame sequences at 10 fps in LIPD-Babel-v1 and LIPD-Babel-v2.

### 6. More Information on LIPD-Babel

LIPD [11] is a large-scale dataset combining LiDAR point clouds, IMU, and skeleton poses. It includes a mix of real and synthetic LiDAR point clouds and IMU measurements, taking advantage of the AMASS [36] motion capture dataset. It combines their own recorded real sequences with ground truth SMPL [63] pose parameters from DIP-IMU [39], LiDARHuman26M [11], AIST++[64], and a subset of AMASS [36] (including ACCAD, BML-Movi, CMU, and TotalCapture(TC) [40]). From this large collection of data, DIP-IMU, TC, the testing set of LiDARHuman26M, and their LIPD test set are used for evaluation, while the remaining data is used as the training set. From the SMPL poses of the AMASS subsets and AIST++, LIPD has generated synthetic point clouds and IMU recordings. For DIP-IMU, they generated synthetic point clouds, and took the real IMU recordings. For LiDARHuman26M, real point clouds are provided, and LIPD generated synthetic IMU recordings. LIPD has provided more details on the generation in the supplementary material of their work but did not publish the code.

While LIPD is a large-scale data for human motion with LiDAR, IMU, and skeletal poses, it is missing activity annotations. Fortunately, the Babel dataset [18] has annotated the AMASS dataset with strong efforts with frame-wise activity annotations. Taking advantage of these annotations, we have merged the Babel annotations into the corresponding AMASS subsets present in LIPD, i.e., ACCAD, BML-Movi, CMU, and TC.

To achieve this, we take advantage of the specific sequence ID's for each AMASS sequence that are stored both in LIPD and in Babel. This allows a unique mapping between both datasets, allowing to add text annotations to the AMASS subset in LIPD. More specifically, all AMASS sequence ids follow the pattern of

“dataset/sequencecategory/posesequence\_poses.npz”.

For example, “ACCAD/Female1Gestures\_c3d/D2-\_Wait\_1\_poses.npz” or “CMU/118/118\_17\_poses.npz”. In LIPD, these sequence ids were modified to follow the pattern “dataset/sequencecategory/posesequence\_stageii”, leading to “ACCAD/Female1Gestures\_c3d/D2-\_Wait\_1\_stageii” “CMU/118/118\_17\_stageii”. Therefore, to achieve the mapping, a reformatting is required by replacing “stageii” with “poses.npz”. Babel follows the exact same format as AMASS, allowing a straightforward mapping from the AMASS subset in LIPD to the respective subset in Babel.

The next difference is the sampling rate of the dataset. AMASS is provided at a higher FPS up to 120FPS, which is much higher than the 10 FPS of LIPD, while Babel is annotated at 30FPS. As a result, we downsample the Babel annotations accordingly to 10FPS to align them with the LIPD dataset. To verify that both datasets are actually temporally aligned after downsampling Babel to 10 FPS, we carefully verified qualitatively that the skeleton poses of the downsampled Babel versions correspond to the poses in the LIPD dataset by plotting them next to each other, and inspecting several sequences from each dataset manually.

When combining Babel and LIPD to LIPD-Babel, we obtain two versions of the dataset. First, LIPD-Babel-v1, which follows the exact test split of LIPD, and LIPD-Babel-v2, which follows the train/val/test split of Babel. LIPD-Babel-v1 allows to evaluate the respective matching and retrieval tasks, while LIPD-Babel-v2 allows to evaluate classification tasks with labels for both training and testing set. More specifically, for LIPD-Babel-v2, the official training split of Babel is used for the training set, and the validation split for the testing set. The annotations for the test split are not publicly available, which is why we use the validation set as the testing set replacement.

Finally, we preprocess the whole dataset into 24-frame sliding window subsequences to ease the training and testing process. A summary of the number of sequences and corresponding text annotations are provided in Table 4. LIPD-Babel-v1 has slightly more sequences than LIPD-Babel-v2 because the test set of Babel is not publicly available, because of which we remove the sequences from ACCAD, BML-movi, CMU, and TC that do not have annotations.

## 7. Specific Performance Scores for Matching and Temporal Moment Retrieval

Figure 2 and Figure 3 in the main paper effectively visualizes the differences in the performance of each model and parameters for matching and temporal moment retrieval, which makes a comparison at the scale of the amount of different parameters that we have evaluated easier to see.

In this section of our supplementary material, we provide the specific matching scores between each modality and each dataset through a heatmap in Figure 8 to provide quantitative numbers for future work to compare against our baselines. In the same way, we present the specific temporal moment retrieval scores between each modality and each dataset through a heatmap in Figure 9. These results are the average over the number of subjects for matching and k-shots for temporal moment retrieval. All individual results for each number of subjects for matching, and top-k for temporal moment retrieval are provided in Figures 11-18, and Figures 19-22, respectively.

## 8. A Simple Improved Matching Algorithm to Associate Different Modalities in the Embedding Space

In practice, we observe continuous streams of point cloud, skeleton, and IMU time series data. Therefore, a matching score can be computed not only on similarities between a single query sequence and all possible subsequences of a video, but instead on consecutive subsequences. We define such a matching algorithm in Algorithm 1, where the mean similarity score over several embeddings from a short temporal neighborhood is considered to compute the temporal matching score. More specifically, given  $N$  consecutive subsequences with their respective  $Q$  consecutive query embeddings  $Z_{a,q}^j$ ,  $q \in Q$  from modality  $a$  and  $K$  consecutive candidate embeddings  $Z_{b,k}^j$ ,  $k \in K$  for modality  $b$ ,  $j \leq 0 < N$ . The similarity score between each query  $q \in Q$  and candidate  $p \in P$  is the average over all pair-wise similarities between the consecutive windows. Finally, the assigned match for each  $q \in Q$  is calculated using  $\argmax$  over all candidate similarity scores.

Figure 23 presents the average of the results over all modalities for each dataset, respectively. The results verify that observing more frames leads to improved matching results.

## 9. Additional Results: Retrieval through Natural Language

On LIPD-Babel-v2, we evaluate text-to-motion retrieval against TMR++ [65]. For a fair comparison, we run TMR++ only on the subset of Babel that we use in LIPD-Babel-v2. Note that TMR++ has been trained on the full

**Input:**  $Q, K$  consecutive query/candidate embeddings for respective modality  $a, b$

**Output:** One-to-Many mapping

```

1  $matches \leftarrow \emptyset$ 
2 foreach  $q \in Q$  do
3    $match \leftarrow$ 
      $\arg \max_{k \in K} \left( \frac{1}{N} \sum_{n=1}^N \text{sim}(Z_{a,q}^j, Z_{b,k}^j) \right);$ 
4    $matches \leftarrow matches \cup \{match\};$ 
5 end
6 return  $matches$ 
```

**Algorithm 1:** Matching algorithm for consecutive subsequences

Method	Training Set	R-Top-1
TMR++ Th=0.95	Full Babel	55.54
Ours (Skeleton) Th=0.95	LIPD-Babel-v2	42.55
Ours (Skeleton) Th=0.90		48.92
Ours (IMU) Th=0.95	LIPD-Babel-v2	46.01
Ours (IMU) Th=0.90		46.94
Ours (Pointcloud) Th=0.95	LIPD-Babel-v2	48.68
Ours (Pointcloud) Th=0.90		53.62

Table 5. Text-to-<Skeleton, Pointcloud, IMU> retrieval on LIPD-Babel-v2

Babel dataset, which gives the model itself an advantage over our model.

The results are presented in Table 5. We use the same threshold (Th) as TMR++ to account for semantically similar retrievals. In addition, we report our results for Th=0.90 and Th=0.95 since the CLIP text embedding space may exhibit different semantic relations compared to the text embeddings used by TMR++. Our results show that our performance is promising, but worse than TMR++, showing possibilities for future work to improve the alignment to text.

## 10. Ablation Study on LIPD-Babel-v2 for HAR

We perform a large ablation study between all modalities for downstream classification. We ablate linear/non-linear probing and freezing or fine-tuning each model when training for HAR on LIPD-Babel-v2. The results are presented in Table 6 for IMU, Table 7 for point clouds, and Table 8 for skeletons.

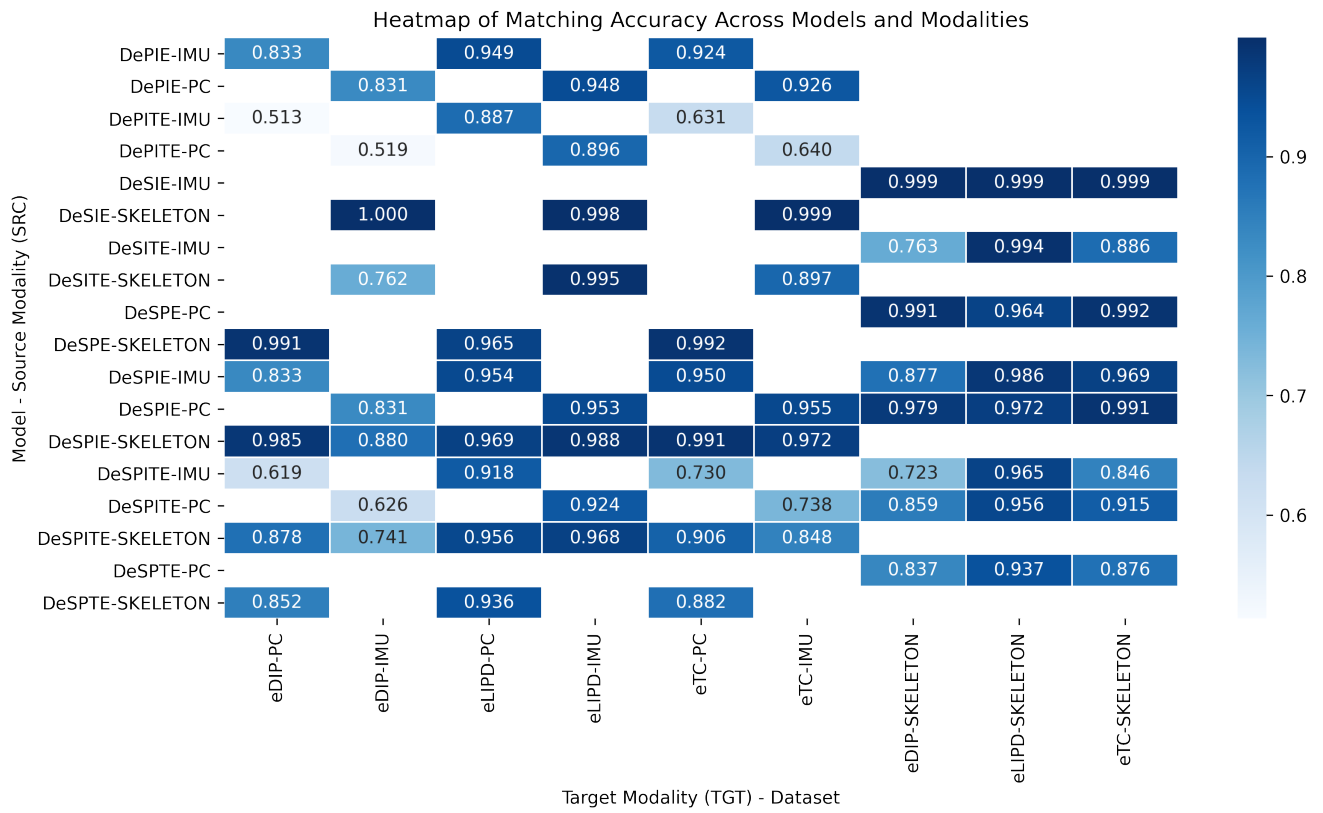


Figure 8. Heatmap to visualize the respective matching results on average across all modalities and datasets at a glance

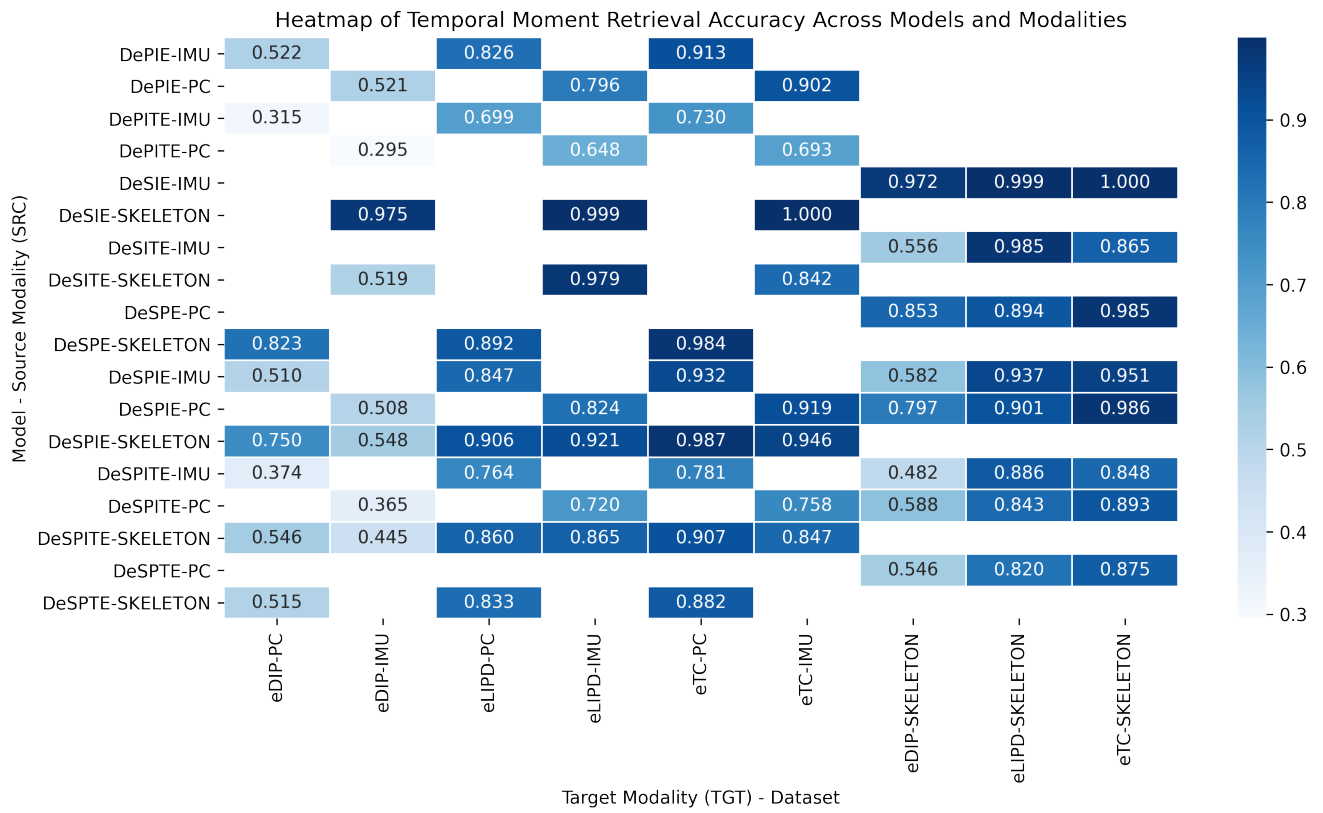


Figure 9. Heatmap to visualize the respective temporal moment retrieval results on average across all modalities and datasets at a glance

Table 6. All IMU HAR classification results on the Babel-LIPD-v2-CLS action recognition dataset, segment-level accuracy  $Acc(seg)$  is reported.

model	Skeleton	PC	IMU	Text	Probing	Fine tuning	Projection Head	$Acc(seg) \uparrow$
Random Init			✓		✓		linear	43.01
Random Init			✓		✓		non-linear	57.12
Random Init			✓			✓	linear	65.62
Random Init			✓			✓	non-linear	64.26
PIE		✓	✓		✓		linear	60.21
PIE		✓	✓		✓		non-linear	62.05
PIE		✓	✓			✓	linear	68.32
PIE		✓	✓			✓	non-linear	67.50
SIE	✓		✓		✓		linear	44.20
SIE	✓		✓		✓		non-linear	56.62
SIE	✓		✓			✓	linear	66.44
SIE	✓		✓			✓	non-linear	67.06
SPIE	✓	✓	✓		✓		linear	58.29
SPIE	✓	✓	✓		✓		non-linear	60.95
SPIE	✓	✓	✓			✓	linear	67.28
SPIE	✓	✓	✓			✓	non-linear	69.21
PITE		✓	✓	✓	✓		linear	61.14
PITE		✓	✓	✓	✓		non-linear	59.21
PITE		✓	✓	✓		✓	linear	68.54
PITE		✓	✓	✓		✓	non-linear	66.63
SITE	✓		✓	✓	✓		linear	56.69
SITE	✓		✓	✓	✓		non-linear	56.95
SITE	✓		✓	✓		✓	linear	66.86
SITE	✓		✓	✓		✓	non-linear	67.10
SPITE	✓	✓	✓	✓	✓		linear	62.06
SPITE	✓	✓	✓	✓	✓		non-linear	59.76
SPITE	✓	✓	✓	✓		✓	linear	68.08
SPITE	✓	✓	✓	✓		✓	non-linear	68.40

Table 7. All point cloud HAR classification results on the Babel-LIPD-v2-CLS action recognition dataset, segment-level accuracy Acc(Seg) is reported.

model	Skeleton	PC	IMU	Text	Probing	Fine tuning	projection	$Acc(seg) \uparrow$
Random Init		✓				✓	linear	65.69
Random Init		✓				✓	non-linear	67.38
Random Init		✓			✓		linear	51.90
Random Init		✓			✓		non-linear	61.98
PIE		✓	✓			✓	linear	65.96
PIE		✓	✓			✓	non-linear	69.52
PIE		✓	✓		✓		linear	68.27
PIE		✓	✓		✓		non-linear	68.36
SPE	✓	✓				✓	linear	66.65
SPE	✓	✓				✓	non-linear	66.13
SPE	✓	✓			✓		linear	63.55
SPE	✓	✓			✓		non-linear	64.17
SPIE	✓	✓	✓			✓	linear	67.51
SPIE	✓	✓	✓			✓	non-linear	66.93
SPIE	✓	✓	✓		✓		linear	67.06
SPIE	✓	✓	✓		✓		non-linear	66.41
SPTE	✓	✓		✓		✓	linear	67.43
SPTE	✓	✓		✓		✓	non-linear	67.30
SPTE	✓	✓		✓	✓		linear	69.31
SPTE	✓	✓		✓	✓		non-linear	68.66
PITE		✓	✓	✓		✓	linear	68.84
PITE		✓	✓	✓		✓	non-linear	69.50
PITE		✓	✓	✓	✓		linear	70.04
PITE		✓	✓	✓	✓		non-linear	69.00
SPITE	✓	✓	✓	✓		✓	linear	69.00
SPITE	✓	✓	✓	✓		✓	non-linear	68.04
SPITE	✓	✓	✓	✓	✓		linear	67.06
SPITE	✓	✓	✓	✓	✓		non-linear	66.32

Table 8. All skeleton HAR classification results on the Babel-LIPD-v2-CLS action recognition dataset, segment-level accuracy Acc(Seg) is reported.

model	Skeleton	PC	IMU	Text	Probing	Fine tuning	projection	$Acc(seg) \uparrow$
Random Init	✓					✓	linear	67.90
Random Init	✓					✓	non-linear	68.23
Random Init	✓				✓		linear	59.71
Random Init	✓				✓		non-linear	60.59
SIE	✓		✓			✓	linear	67.79
SIE	✓		✓			✓	non-linear	70.44
SIE	✓		✓		✓		linear	50.50
SIE	✓		✓		✓		non-linear	57.14
SPE	✓	✓				✓	linear	69.06
SPE	✓	✓				✓	non-linear	70.14
SPE	✓	✓			✓		linear	58.55
SPE	✓	✓			✓		non-linear	59.93
SPIE	✓	✓	✓			✓	linear	68.31
SPIE	✓	✓	✓			✓	non-linear	67.47
SPIE	✓	✓	✓		✓		linear	61.76
SPIE	✓	✓	✓		✓		non-linear	63.64
SPTE	✓	✓		✓		✓	linear	69.20
SPTE	✓	✓		✓		✓	non-linear	69.01
SPTE	✓	✓		✓	✓		linear	65.84
SPTE	✓	✓		✓	✓		non-linear	65.93
SITE	✓		✓	✓		✓	linear	68.14
SITE	✓		✓	✓		✓	non-linear	69.64
SITE	✓		✓	✓	✓		linear	63.83
SITE	✓		✓	✓	✓		non-linear	63.93
SPITE	✓	✓	✓	✓		✓	linear	70.64
SPITE	✓	✓	✓	✓		✓	non-linear	69.91
SPITE	✓	✓	✓	✓	✓		linear	67.20
SPITE	✓	✓	✓	✓	✓		non-linear	66.77

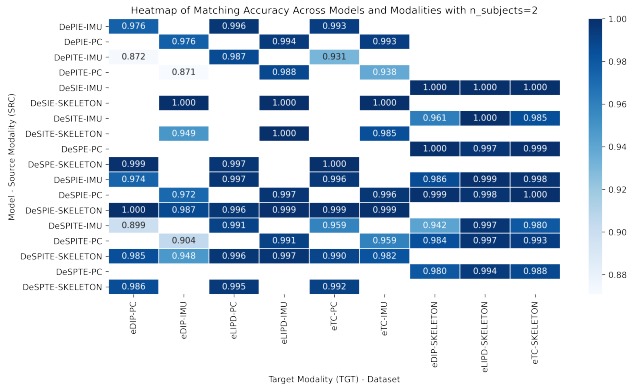


Figure 10. Matching, subjects=2

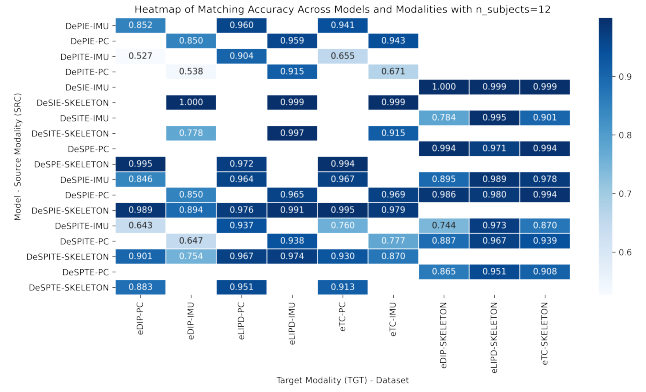


Figure 13. Matching, subjects=12

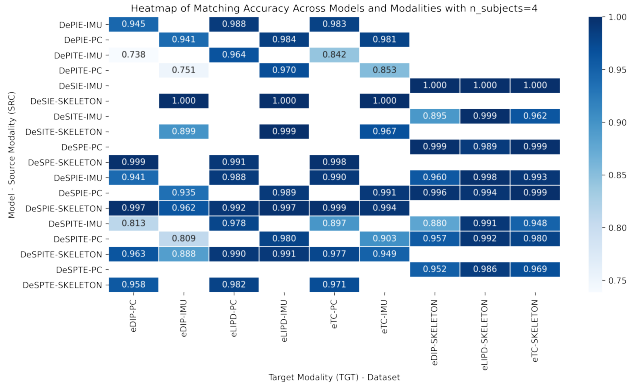


Figure 11. Matching, subjects=4

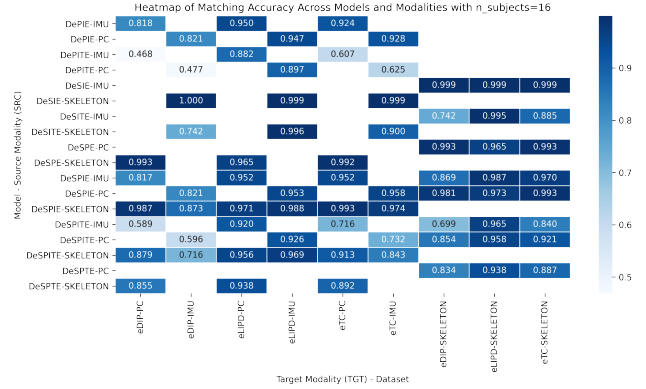


Figure 14. Matching, subjects=16

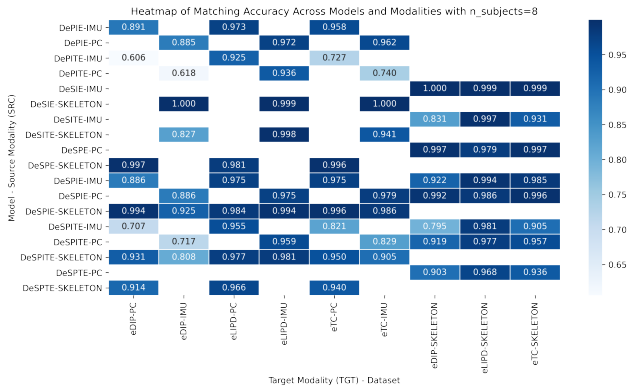


Figure 12. Matching, subjects=8

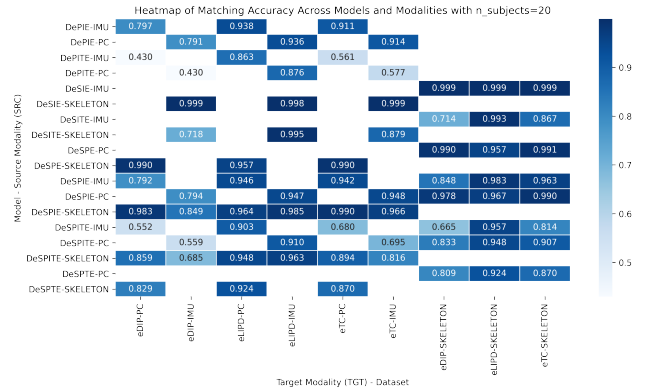


Figure 15. Matching, subjects=20



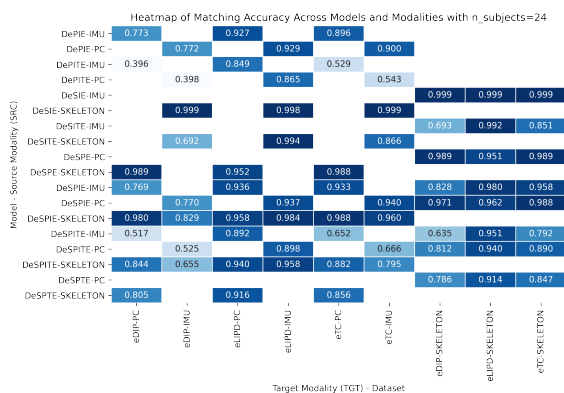


Figure 16. Matching, subjects=24

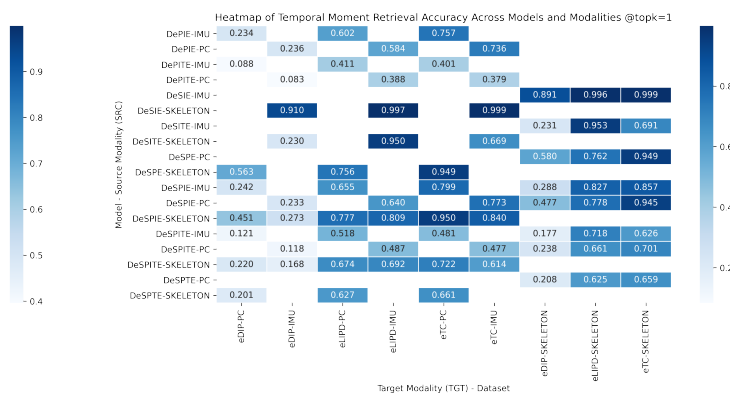


Figure 19. Temporal Moment Retrieval, topk=1

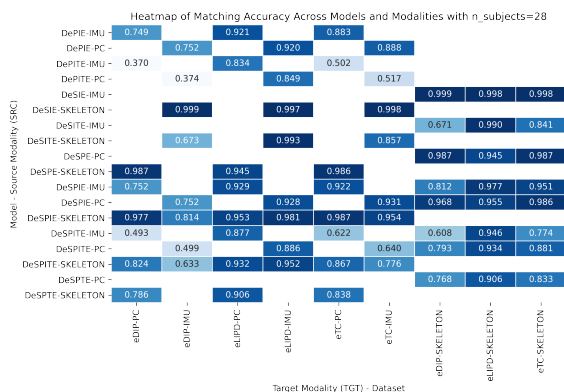


Figure 17. Matching, subjects=28

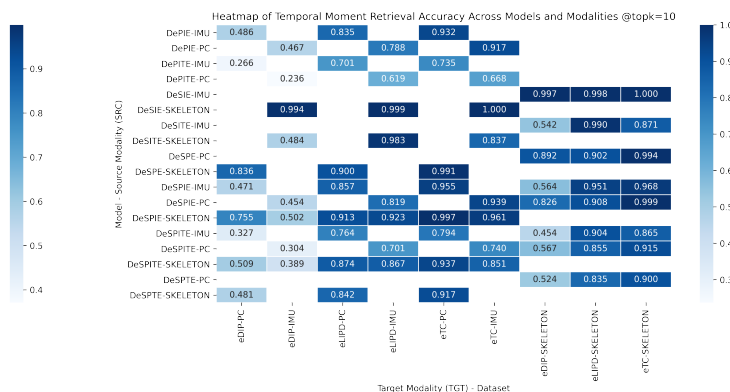


Figure 20. Temporal Moment Retrieval, topk=10

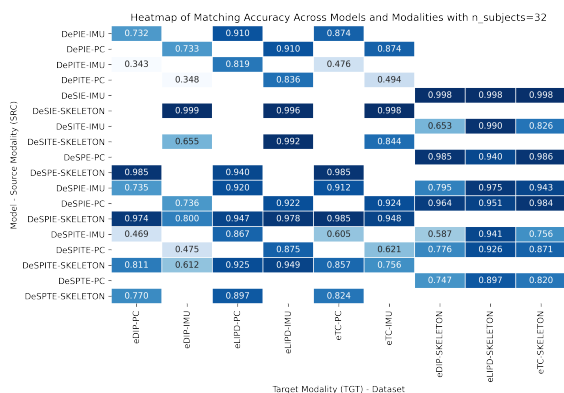


Figure 18. Matching, subjects=32

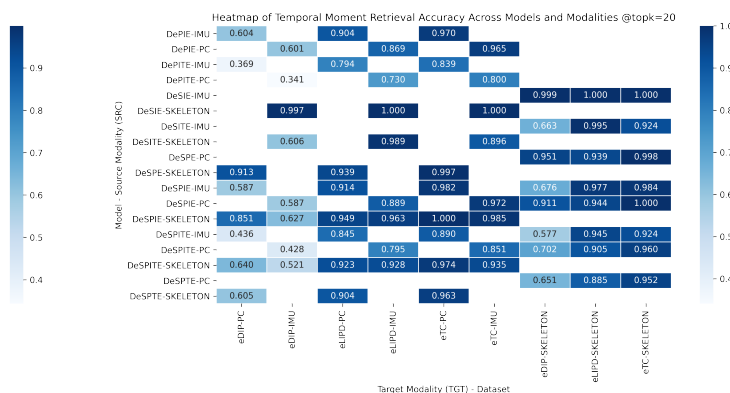


Figure 21. Temporal Moment Retrieval, topk=20

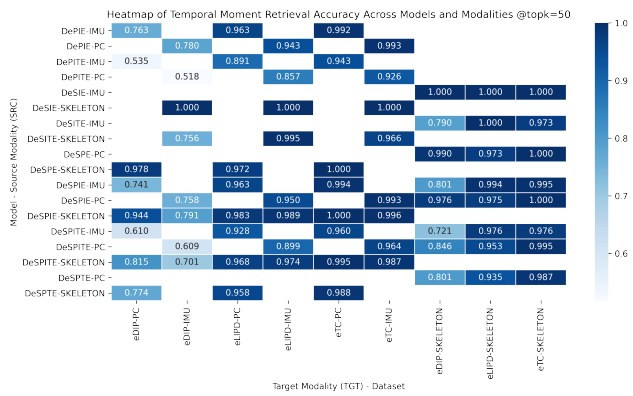


Figure 22. Temporal Moment Retrieval, topk=50

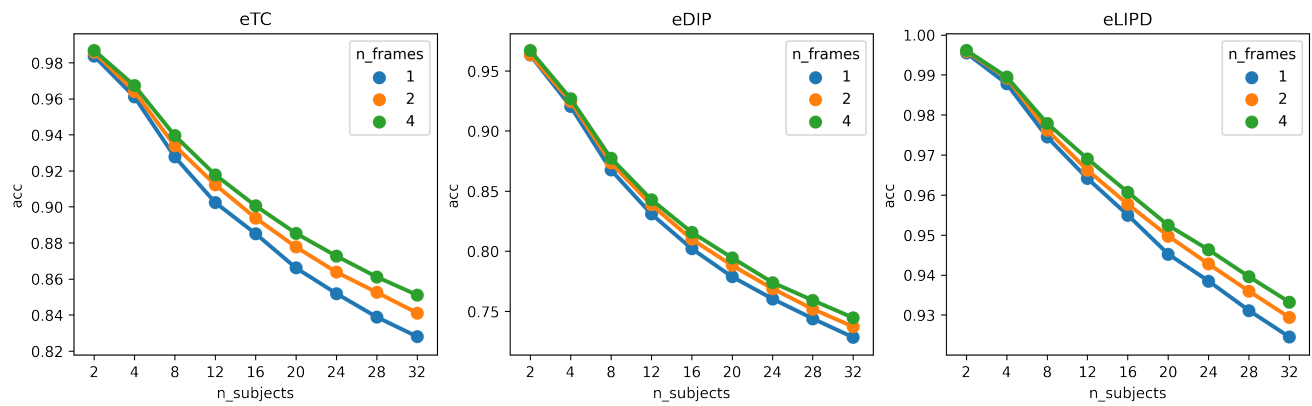


Figure 23. Performance of computing matching scores based on 1,2, or 4 consecutive windows. The matching scores are presented averaged over each model and modality combination to show the effectiveness of matching based on consecutive windows.