Exploring Missing Modality in Multimodal Egocentric Datasets
(Supplementary Material)

We present the supplementary results referenced in the main paper. Section 1 presents additional results that were not included in the main manuscript because of space constraints. Section 2 shows additional results and analysis on modal incomplete datasets. Then, in Section 3, we analyze the effect of Bottleneck architecture in missing modalities. Later, in Section 4 we show additional details about Ego4D-AR. Finally, we give extra details on the decoder of our pretraining strategy in Section 5.

1. Additional Results

In this section, we provide additional results on Ego4D-AR verbs and nouns and Epic-Kitchens nouns.

1.1. Ego4D-AR (verbs) and Epic-Kitchens (nouns)

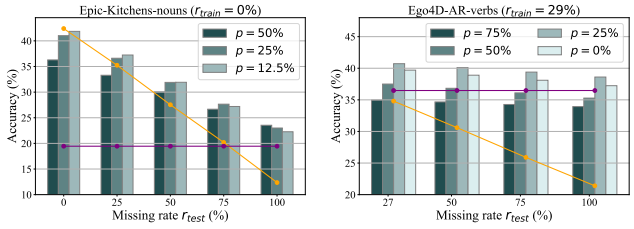

Figure 1. **Modality drop probability p *vs*. accuracy for modal-complete Epic-Kitchens (nouns) and modal-incomplete Ego4D-AR (verbs).** Our MMT effectively learns with missing modalities under several p. The results are consistent with verb classes in Epic-Kitchens and noun classes in Ego4D-AR (main manuscript), *i.e*. smaller p is better for Epic-Kitchens and $p = 25\%$ works the best in Ego4D-AR. For Ego4d-AR verbs, the accuracy is almost the same across all r_{test} for each experiment. As mentioned in Section 4, this is due to the high bias caused by the highly imbalanced distribution of verb classes.

Different values of p in *random-replace* We report the results similarly to the main manuscript. In Figure 1, we show the results using strategy *random-replace* for both datasets. We find that analyzing the results of Ego4D-AR verbs is hard due to the learning issue in the verb prediction. However, we find that for Epic-Kitchens, the observations are consistent with the ones reported in the main manuscript, *e.g*. selecting the appropriate value of p for each dataset (e.g., $p = 25\%$) is crucial to ensure satisfactory performance across all r_{test} instances. This balance allows the model to effectively adapt while still leveraging the benefits of multimodal information. Furthermore, our models significantly outperform the baseline.

Comparison with baselines In Table 1, we report the results with our methods *vs*. the baselines. In Epic-Kitchens, our strategies perform much better than the proposed baselines and outperform the unimodal 19.5% by 3.5% points.

r_{test}	0%	25%	50%	75%	100%
Unimodal	19.5	19.5	19.5	19.5	19.5
Baseline(zeros)	**42.4**	35.2	27.5	20.2	12.4
Baseline(skip)	**42.4**	32.9	22.9	13.1	3.5
Modality Dropping	39.3	35.6	31.1	27.1	22.9
Ours (MMT)	41.0	**36.6**	**31.9**	**27.6**	**23.0**

Table 1. Comparison of our method with baselines on the Epic-Kitchens-nouns dataset. We demonstrate the accuracy of our method *vs*. unimodal performance across different missing modality ratios r_{test}. We show in **bold** the best result and underline the runner-up.

2. Studying modal-incomplete datasets

Recall that Epic-Sounds and Epic-Kitchens have *modal-complete training sets*, and Ego4D-AR has *modal-incomplete training set* with $r_{train} = 29\%$. To study the effect of using MMT in the datasets with different missing modality severity levels, we create several variants of the training sets in each dataset by manually enforcing different $r_{train} > 0$. We do so by randomly shuffling all modal-complete training instances and storing this order in a list. For Ego4D-AR, modal-incomplete instances (29%) are placed at the very start of the list. Following this, we establish our desired missing rate r_{train} by sampling from the start of this list. This method ensures that any increase in r_{train} builds upon the existing set of modal-incomplete instances, meaning that if we increase r_{train}, we add new modal-incomplete instances to those already

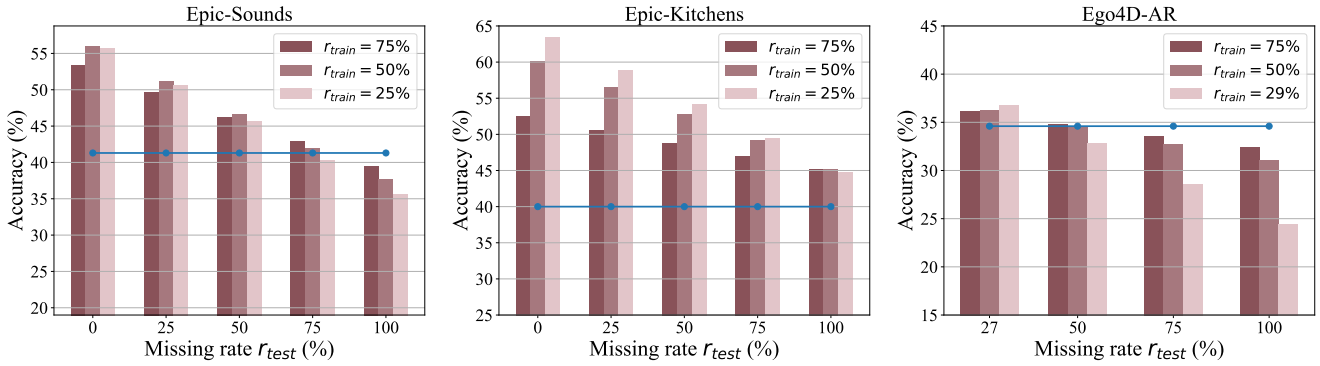

Figure 2. **Missing modality rate in training data r_{train} *vs*. accuracy.** Our MMT effectively deals with missing modalities under several r_{train} regimes. For simplicity, $p = 0$ for all experiments.

included, thereby maintaining a cumulative effect. We also use this protocol for the experiments in Sec. 4.5 and Sec. 4.6 of the main manuscript.

For Epic-Kitchens and Epic-Sounds we use $r_{train} \in \{25\%, 50\%, 75\%\}$, and $r_{train} \in \{29\%, 50\%, 75\%\}$ for Ego4D-AR. Figure 2 shows the results of training our model with MMT in the datasets with different missing modality rates in the training data r_{train}. As can be seen, severe modal incompleteness negatively affects the performance. Nevertheless, overall, our models bring the performance closer to the unimodal for higher r_{test} and still allow the model to benefit from all training samples.

Architecture	**Audio r_{test}**	**Accuracy**
Self-attention	0%	45.3%
	50%	32.7%
	75%	26.7%
	100%	20.5%
Bottleneck	0%	55.5%
	50%	34.3%
	75%	23.7%
	100%	13.6%
Video-only	0% - 100%	41.4%

Table 2. **Bottleneck fusion *vs*. vanilla self-attention with early fusion in Epic-Sounds dataset.**

3. Bottlenecks condensing the information.

As mentioned in the paper, MBT uses a small set of learnable fusion tokens to exchange information between the modalities. One might wonder if the ability of the bottleneck to "condense" the information already plays some role in dealing with missing inputs. To investigate this, Table 2 compares training a vanilla self-attention model with $L_f = 0$ (row 1) with the training MBT with the same fusion layer (row 2). Table 2 shows that vanilla self-attention performs poorly in the multimodal setup with complete modalities, losing 10 points, compared to the bottleneck fusion. Surprisingly, when the modality inputs are incomplete, vanilla self-attention performs better (20.5% vs 13.6% with $r_{test} = 100\%$). This shows that while the bottlenecks are effective for multimodal fusion, they are sensitive to missing modalities, and they must be adapted to address incomplete modalities.

4. Ego4D-AR dataset details

As mentioned in the main manuscript, we curate action recognition clips from the Ego4D Short-term Action Anticipation benchmark [1]. Given a video clip, Short-term Action Anticipation predicts the possible future object interaction and a "time to contact" estimate. The future object interaction is predicted as a noun and verb pair, defining the object and the type of interaction. The "time to contact" is the number of seconds after which the interaction is expected. Given these annotations, we trim the clips centered around the ground-truth "time to contact" and assign them the ground-truth noun and verb annotations. Following that, we obtained 98276 train and 47395 test instances for the action recognition dataset Ego4D-AR. We show the imbalanced verb classes distribution of this dataset in Figure 3.

We observed that the dataset's class distribution is highly imbalanced. To demonstrate this, we plotted the distribution of the top-15 most common verb classes in the training set (see Figure 3). Notably, over 35,000 instances, or 38%, belong to the verb class 'take'. Training with this imbalance is challenging as the model tends to become biased towards this verb class. To address this, we adapted a common strategy: adjusting per-class weights in the cross-entropy loss for verb prediction. Let $|S_i|$ denote the number of samples in the dataset labeled as class i, where $S_i = \{x \in \text{Dataset} \mid \text{label}(x) = i\}$. The weight for class i is then $w_i = 1 - \frac{|S_i|}{|S|}$, meaning the loss is reduced for more common verb classes

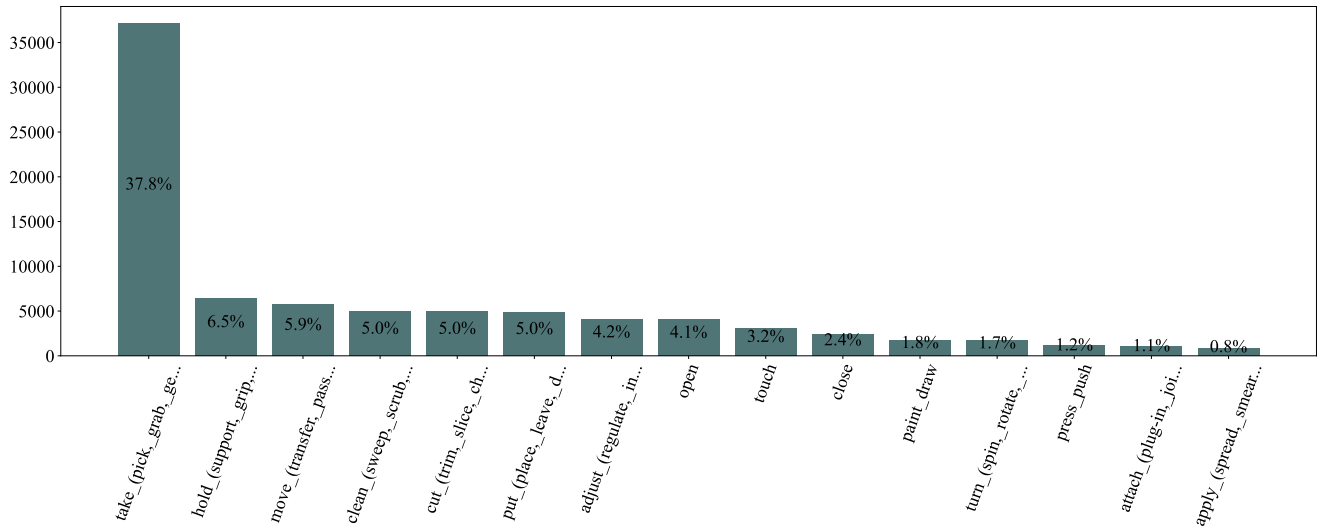

Figure 3. **Distribution of the verb classes in Ego4D-AR.**

and increased for rarer ones. We did not apply this adjustment to noun prediction, as noun label distribution is more balanced. Adjusting per-class weights in the loss function only slightly mitigated bias in the verb distribution. The imbalance still presents a significant learning challenge, resulting in high bias in the model predictions. Understanding this is essential for interpreting the results in the Ego4D-AR experiments.

5. Decoder in the pre-training

We are using Masked-Autoencoder (MAE) [2] for self-supervised pre-training of the MBT backbone. MAE has two parts: encoder and decoder. During training, some input tokens are masked, and only the non-masked tokens are passed to the encoder. Then, learnable masked tokens with shared parameters are appended to the encoded non-masked tokens, and all are passed to the lightweight decoder for reconstructing the original input of the masked token. The reconstruction loss is applied to the masked tokens. After the pre-training, the decoder is discarded, and only the encoder is used for fine-tuning. We provide the encoder details in the main paper and share the decoder details here.

Note that the encoder architecture uses bottleneck fusion, meaning the two modalities can communicate through several learnable tokens. We do not model the multimodal fusion in the decoder part, so each modality has a separate transformer in the decoder. Each decoder is a 4-layer transformer with 16 attention heads and an embedding dimension of 512. We do not share the parameters of the transformers (the same as those in the encoder).

References

[1] Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al.: Ego4d: Around the world in 3,000 hours of egocentric video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18995–19012 (2022) 2

[2] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022) 3