A Appendix

A.1 Calibration Error

To estimate both the calibration abilities of each balancing method and individual model we measure the Expected Calibration Error (ECE) [3]. ECE measures the difference between average accuracy and average confidence for datapoints within a narrow confidence interval. We arrange predictions according to per-class confidence scores and categorize them into $K = 15$ bins B_k, each linked to a corresponding equally distant confidence interval. The lower the ECE the better the calibration of the model. With N being the total number of datapoints, ECE can be estimated as:

$$\mathrm{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{N} \cdot |\mathrm{acc}(B_k) - \mathrm{conf}(B_k)| \,. \tag{10}$$

B Implementation

In this section, we provide a detailed account of the implementation steps and choices made in our study. These specifics are crucial for replicating our work and ensuring a fair comparison with previous methods.

B.1 Balancing Hyperparameters

When tuning the hyperparameters of each method, we follow a consistent strategy. Most methods involve adjusting a few key hyperparameters. For example, AGM, PMR, and OGM focus on the α slope of the coefficients and the endpoint of the balancing process. MMCosine includes the magnitude increase scale s, while MSLR incorporates initial coefficients *init* used as priors and specifies the window of epochs for averaging performance k_e. Additionally, MLB has also the α slope and introduces β_{max}, representing the maximum acceleration coefficient value. All hyperparameters used are summarized in Table 2. For AGM, OGM, and PMR, we did not observe any direct benefit from muting the balancing at a specific point, so it is not included in the table. For MLB, we noticed an improvement when β_{max} was set higher than 1, indicating that acceleration was effective. However, values $\beta_{max} \geq 2$ did not show significant differences. We assume that in datasets with more severe overfitting on one of the modalities, this might not hold, and further acceleration might be needed.

We maintained uniform computational limits across all methods, conducting grid searches with an equal number of points. Although we slightly favored MSLR due to its multiple hyperparameters, this choice didn't significantly impact the results. Each method, backbone encoder, and fusion method underwent a separate search, leading to potentially differing hyperparameters in each experiment.

B.2 Training Hyperparameters

For all datasets, we adopt a consistent strategy for defining training hyperparameters. Initially, we conduct a search for the learning rate and weight decay parameters when training each unimodal encoder individually with a linear classification head. Once we establish a suitable combination of these parameters, we apply them to the multimodal models. Interestingly, we find that models are not significantly sensitive to slight variations in these values.

Table 2: Hyperparameters for each dataset (CREMA-D, AVE, and UCF101), backbone encoder (ResNet and Conformer where applicable), balancing method (MSLR, MMCosine, OGM, PMR, AGM, MLB), and fusion method (Late-Linear, MLP, FiLM, Gated, and TF where applicable).

Method	Fusion	Dataset			
		CREMA-D		AVE	UCF
		ResNet	Conformer	ResNet	ResNet
MSLR	Late-Linear	$lr = [0.7, 1.3]$ $k_e = 20$	$lr = [0.9, 1.1]$ $k_e = 20$	$lr = [0.7, 1.3]$ $k_e = 10$	$lr = [1.0, 1.0]$ $k_e = 5$
MMCosine	Late-Linear	$s = 10.0$	$s = 10$	$s = 8$	$s = 15$
OGM	Late-Linear	$\alpha = 1.0$	$\alpha = 1.0$	$\alpha = 0.8$	$\alpha = 1.0$
PMR	Late-Linear	$\alpha = 2.0$	$\alpha = 1.0$	$\alpha = 1.5$	$\alpha = 1.0$
	MLP	-	$\alpha = 0.5$	-	-
AGM	Late-Linear	$\alpha = 3.0$	$\alpha = 3.0$	$\alpha = 1.0$	$\alpha = 1.5$
	MLP	$\alpha - 1.0$	$\alpha = 3.0$	$\alpha = 2.0$	$\alpha = 3.0$
	FiLM	$\alpha = 1.5$	$\alpha = 3.0$	-	-
	Gated	$\alpha = 3.0$	$\alpha = 1.5$	-	-
	TF	$\alpha = 1.5$	$\alpha = 1.5$	-	-
MLB	Late-Linear	$\alpha = 5.0$	$\alpha = 5.0$	$\alpha = 2.0$	$\alpha = 2.0$
	MLP	$\alpha = 5.0$	$\alpha = 3.0$	$\alpha = 1.0$	$\alpha = 4.0$
	FiLM	$\alpha = 0.5$	$\alpha = 4.0$	-	-
	Gated	$\alpha = 5.0$	$\alpha = 5.0$	-	-
	TF	$\alpha = 5.0$	$\alpha = 2.0$	-	-

Consequently, we set our final values uniformly across all datasets: $(1e-3, 1e-4)$ for the ResNet encoders and $(5e-5, 1e-5)$ for Conformer encoders. We utilize the Adam optimizer [4] with default values for (β_1, β_2) as $(0.9, 0.999)$ and ε set to $1e-07$. Batch sizes remain consistent across models using the same backbone encoder within each dataset but differ between the two encoder types. Specifically, we set the batch size to 32 for ResNet and adjust it to 8 for Conformer models to accommodate our computational resources. Model validation occurs once at the end of each epoch. Training continues for a maximum of $1.5k$ epochs, with early stopping implemented at 300 epochs. Despite the large number of epochs allowed, we observed that nearly all runs converged before reaching 350 epochs. We employ a learning rate scheduler [6] based on cosine annealing with warm restarts, where $T_0 = 4$ and $T_{mult} = 2$, supplemented by a constant warming-up stage of 3 epochs. Finally, while the use of gradient clipping [7] could potentially aid in preventing exploding gradients during balancing, we opt not to utilize it in our models to ensure a fair comparison with previous methods, except for AGM [5], where we strictly follow their implementation.

B.3 Dataset Splits

It is important to note that some of the previous methods introduced in the paper [5, 8, 10, 11], based on the official code they shared, seem to have used the test set as the validation set on CREMA-D, tuning model parameters directly on the test set. Additionally, the same actors

were included in different sets, leading to further overfitting. Therefore, the observed performance difference in our results compared to the initially reported results of these papers is attributed to the fact that we validate the models with a separate validation set, comprising around 10% of the total data, and we report results on the test set using 3-fold cross-validation with the splits provided by Goncalves *et al.* [2].

C Coefficients Analysis

Several methods, including MLB, estimate coefficients for each modality that are exploited to balance the multimodal learning. In Figure 7, we compare the values of the k_i coefficients for the different methods to illustrate the effect caused by each one of them. This visualization demonstrates that MLB is the only one that upon convergence its coefficients are equalizing and approach value 1 phasing out the balancing when it's no longer necessary.

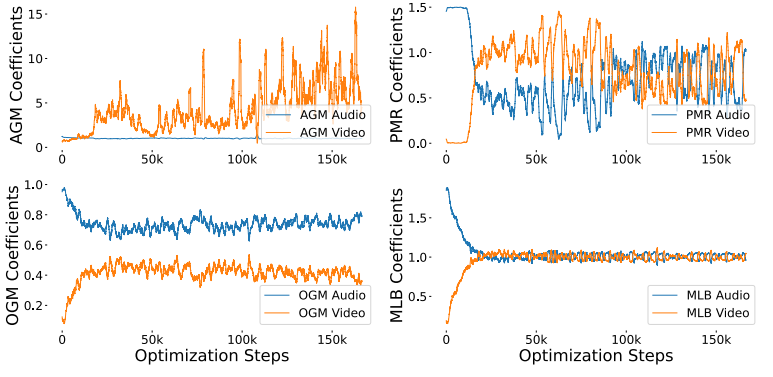

Figure 7: Comparison of balancing coefficients evolution of coefficients that amplify the contribution of each unimodal encoder's learning, smoothed over a 1k-point kernel. It illustrates that while all methods provide some balancing, only Multi-Loss Balanced (MLB) achieves self-muting when modalities have converged.

D Computational Complexity Analysis

The computational complexity of the proposed balancing method, MLB, is introducing negligible additional overhead compared to both training without any balancing and existing multimodal learning approaches. To quantify this, we analyze the complexity associated with the forward and backward passes of the multimodal encoders $f_1, f_2, \ldots, f_M$, with computational costs denoted by C_{f_i} and C_{b_i}, respectively, alongside the common network f_v with complexity C_v.

Methods classified under gradient balancing techniques (Section 2.1) typically do not introduce significant additional computation during network passes, rendering the balancing computations negligible. Approaches in the second category (Section 2.2), which incorporate additional classification heads, do introduce some computational overhead. However, these heads usually consist of linear layers or prototype networks, whose computational costs are minimal compared to the overall network.

To illustrate this point, consider a ResNet-18 encoder used in our experiments, which requires approximately $1.8 \times N$ GFlops per forward pass for a batch size of N on the video modality. In contrast, a linear classifier mapping from a 512-dimensional feature space to C classes involves only $512 \times N \times C$ flops, on the order of thousands of flops, rendering its cost negligible in comparison to the rest of the network.

Notable exceptions include methods like PMR, which necessitate prototype estimation at the end of each epoch, and the approach by Wang et al. [9], which computes balancing coefficients on a separate validation set each epoch, introducing additional overhead. The AGM method represents the most computationally expensive approach, as it requires $2^M - 1$ forward passes, resulting in a training complexity of $O((2^M - 1) \times (C_{f_i}) + C_{b_i})$.

In contrast, MLB adds only the computational cost of the linear classifiers, the estimation of additional losses, and the computation of balancing coefficients. These operations require only thousands of flops, and thus are negligible relative to the overall model size. Consequently, in practical terms, these computations do not result in any noticeable increase in training time. Similarly, the memory overhead introduced by MLB remains minimal, comparable to the most efficient methods reviewed.

E Tabular Presentation of Results

The results from Figure 3 are also provided in table format to facilitate direct access and use by future research. Table 3 presents the accuracy of each balancing method across various datasets and backbone encoders using Late-Linear fusion. Additionally, Table 4 details the findings from Figure 5, illustrating the performance of different fusion methods on the CREMA-D dataset.

Table 3: Accuracy presented in table format for each dataset (CREMA-D, AVE, and UCF101) and backbone encoder (ResNet and Conformer where applicable), balancing method (MMCosine, MSLR, OGM, PMR, AGM, MLB) using the Late-Linear fusion method.

Method	CREMA-D		AVE	UCF
	ResNet	Conformer	ResNet	ResNet
Video	55.4$_{\pm 2.3}$	69.4$_{\pm 2.8}$	62.6$_{\pm 1.0}$	30.3$_{\pm 1.5}$
Audio	60.6$_{\pm 3.0}$	76.8$_{\pm 2.0}$	45.7$_{\pm 1.6}$	38.3$_{\pm 0.8}$
Joint Training	62.6$_{\pm 1.4}$	75.1$_{\pm 1.7}$	66.7$_{\pm 1.5}$	47.7$_{\pm 1.5}$
MMCosine	58.8$_{\pm 1.6}$	73.5$_{\pm 2.1}$	66.7$_{\pm 0.9}$	46.9$_{\pm 3.6}$
MSLR	56.5$_{\pm 2.2}$	77.1$_{\pm 2.4}$	66.8$_{\pm 1.7}$	47.9$_{\pm 3.8}$
OGM	65.6$_{\pm 3.8}$	82.4$_{\pm 1.0}$	67.9$_{\pm 0.5}$	51.8$_{\pm 1.9}$
PMR	53.7$_{\pm 2.3}$	80.5$_{\pm 0.2}$	37.4$_{\pm 7.5}$	40.9$_{\pm 2.0}$
AGM	69.3$_{\pm 1.4}$	78.5$_{\pm 2.6}$	65.8$_{\pm 2.7}$	50.5$_{\pm 1.5}$
Multi-Loss	69.2$_{\pm 1.8}$	82.6$_{\pm 0.9}$	70.1$_{\pm 0.9}$	51.1$_{\pm 1.8}$
MLB	**71.4**$_{\pm 1.9}$	**85.2**$_{\pm 0.9}$	**70.6**$_{\pm 1.4}$	**52.0**$_{\pm 2.2}$

Table 4: Accuracy presented in table format for the different fusion methods (MLP, FiLM, Gated, TF) on CREMA-D dataset.

Method	MLP		FiLM		Gated		TF	
	ResNet	Conformer	ResNet	Conformer	ResNet	Conformer	ResNet	Conformer
Joint Training	62.6$_{\pm 1.4}$	75.1$_{\pm 1.7}$	64.8$_{\pm 1.1}$	73.8$_{\pm 1.5}$	59.1$_{\pm 4.7}$	71.7$_{\pm 2.5}$	58.7$_{\pm 2.2}$	71.0$_{\pm 2.5}$
AGM	69.3$_{\pm 1.4}$	78.5$_{\pm 2.6}$	57.8$_{\pm 1.3}$	68.9$_{\pm 0.6}$	55.7$_{\pm 4.7}$	69.0$_{\pm 0.7}$	54.0$_{\pm 4.5}$	70.9$_{\pm 0.7}$
Multi-Loss	69.2$_{\pm 1.8}$	82.6$_{\pm 0.9}$	68.3$_{\pm 2.9}$	84.7$_{\pm 0.7}$	70.7$_{\pm 2.4}$	83.3$_{\pm 1.1}$	**69.7**$_{\pm 3.4}$	83.4$_{\pm 1.5}$
MLB	**71.4**$_{\pm 1.9}$	**85.2**$_{\pm 0.9}$	**72.2**$_{\pm 0.5}$	**84.4**$_{\pm 0.5}$	**71.1**$_{\pm 1.7}$	**84.2**$_{\pm 1.4}$	**69.7**$_{\pm 1.1}$	**85.5**$_{\pm 0.9}$

Appendix References

[1] Yunfeng Fan et al. “PMR: Prototypical Modal Rebalance for Multimodal Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20029–20038.

[2] Lucas Goncalves et al. “Versatile Audio-Visual Learning for Handling Single and Multi Modalities in Emotion Regression and Classification Tasks”. In: *arXiv preprint arXiv:2305.07216* (2023).

[3] Chuan Guo et al. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[4] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[5] Hong Li et al. “Boosting Multi-modal Model Performance with Adaptive Gradient Modulation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 22214–22224.

[6] Ilya Loshchilov and Frank Hutter. “Sgdr: Stochastic gradient descent with warm restarts”. In: *arXiv preprint arXiv:1608.03983* (2016).

[7] Tomáš Mikolov et al. “Statistical language models based on neural networks”. In: *Presentation at Google, Mountain View, 2nd April* 80.26 (2012).

[8] Xiaokang Peng et al. “Balanced multimodal learning via on-the-fly gradient modulation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8238–8247.

[9] Weiyao Wang, Du Tran, and Matt Feiszli. “What makes training multi-modal classification networks hard?” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12695–12705.

[10] Ruize Xu et al. “MMCosine: Multi-Modal Cosine Loss Towards Balanced Audio-Visual Fine-Grained Learning”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.