MMExit: Enabling Fast and Efficient Multi-modal DNN Inference with Adaptive Network Exits

Xiaofeng Hou[1 [0000−0003−4372−7851]], Jiacheng Liu[1 [0000−0003−0378−2311]], Xuehan Tang[1 [0009−0000−4106−2759]], Chao Li[1]✉[[0000−0001−6218−4659]], Kwang-Ting Cheng[2]✉[[0000−0002−3885−4912]], Li Li[1], and Minyi Guo[1 [0000−0003−0034−2302]]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] ACCESS – AI Chip Center for Emerging Smart Systems, InnoHK Centers, The Hong Kong University of Science and Technology

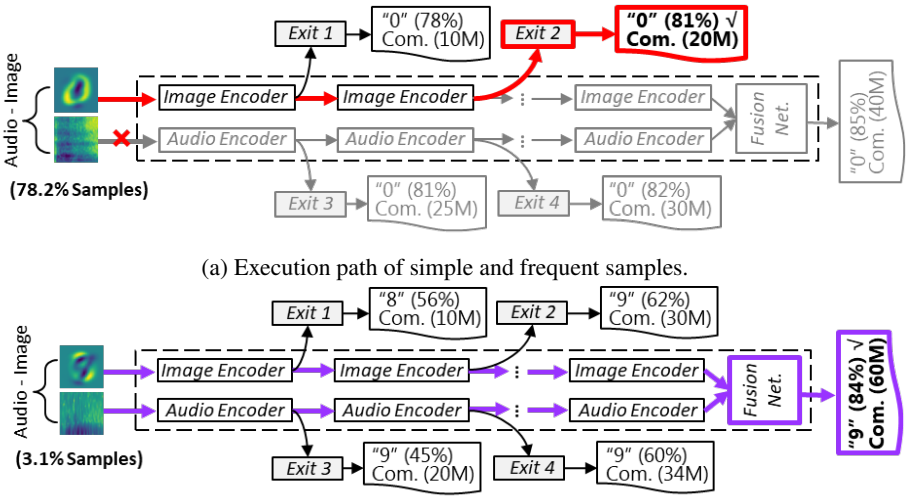
Abstract Multi-modal DNNs have been demonstrated to outperform the best uni-modal DNNs by fusing information from different modalities. However, the performance improvement of multi-modal DNNs is always associated with an incredible increase in computational cost (e.g., network parameters, MAC operations, etc.) to handle more modalities, which ultimately makes them impractical for many real-world applications where computing capability is limited.
In this paper, we propose *MMExit*, a multi-modal exit architecture that allows for computing appropriate modalities and layers to predict results for different data samples. To this end, we define a novel metric called *utility of exit (UoE)* to measure the correlations of performance and computational cost for different exits. We then use an *equivalent modality serialization* method to map the two-dimensional exit space into an equivalent linear space and rank the exits according to their UoE to achieve fast and adaptive inference. To train the *MMExit* network, we devise *a joint loss function* which synthesizes the features of different modalities and layers. Our results show that *MMExit* can slash up to 48.72% of MAC operations with the best performance compared to SOTA multi-modal architectures.

Keywords: Multi-modal DNNs, Energy-efficient AI, Adaptive Inference

1 Introduction

Multi-modal DNNs [16,27,11] have recently attracted lots of attention due to their superior performance. As shown in Figure 1, a multi-modal DNN typically consists of multiple parallel encoder networks that take different modality data as inputs to obtain the modality features and a subsequent fusion and decision network that fuses the different features as well as outputs the final decision. By fusing the information from different modalities, multi-modal DNNs have been demonstrated to outperform the best uni-modal DNNs in many application domains. For example, in multimedia applications, the multi-modal DNNs have been shown to outperform the best uni-modal DNNs by 5% ˜ 30% accuracy through fusing vast amounts of image, video and audio data [2].

Despite the performance advantages, multi-modal DNNs often involve more computational costs such as network parameters and Multiply-Accumulate (MAC) operations [23]. It has been shown that multi-modal DNNs can lead to a $0.1\times$ ˜ $80\times$ increase



(a) Execution path of simple and frequent samples.

(b) Execution path of difficult but less frequent samples.

Figure 1: The execution paths for different data samples in *avmnist* with *MMExit*.

in network parameters compared to uni-modal DNNs [16,17]. This would ultimately increase the latency and energy required by inference tasks. For example, experimental results on powerful servers with 17 GPUs and 32 CPUs installed show that the increased parameters of the multi-modal DNNs can lead to a 10× increase in inference latency and power consumption in affective computing applications [16]. This would further make multi-modal DNNs prohibitive in many real-world scenarios such as next-generation mobile robots where computational capability is limited.

In this paper, we propose *MMExit*, an adaptive multi-modal exit architecture that enables the optimal performance and computational cost tradeoffs in multi-modal DNN inference tasks for different data samples. *MMExit* exploits a unique feature of multi-modal DNNs in that different modalities and layers can provide different levels of confidence at different computational costs. For example, it has been shown that text-based features perform better than visual or auditory features in a multi-modal language-emotion analysis task [1]. Therefore, *MMExit* is designed to predict results for most data samples with a minimal computational cost by exiting from appropriate modality and layers as shown in Figure 1. For very complex data samples, which happen less frequently, *MMExit* would compute more modalities to guarantee better accuracy.

Unlike the previous early-exit architectures for uni-modal DNNs, where exits are explicitly related to the depth of layers, *MMExit* is a **new problem of finding an optimal exit in a 2-dimensional (2D) space composed of modalities and layers**. In this regard, one important challenge is to decide in which modality and layers to exit to reduce the computation cost of the inference task while maintaining high accuracy. To this end, we define a novel metric called *utility of exit (UoE)* to measure the correlations of performance gain and computational effort for different exits. we also use *an equivalent serialization method* to map the 2D exit space into an equivalent linear space which enables us to find the optimal exit fast. Another challenge is how to train the *MMExit*

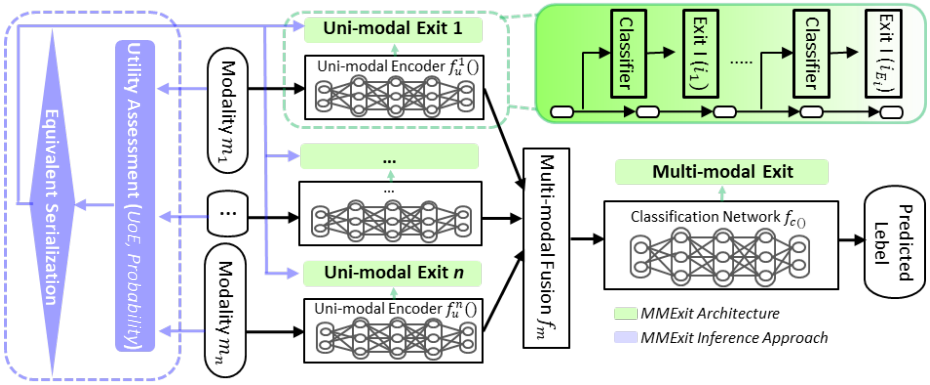


Figure 2: An overview of *MMExit*.

DNNs efficiently. We devise *a joint loss function* which synthesizes the features of different modalities and layers. The experimental results show that *MMExit* can reduce 22.64% ~ 48.72% MACs and 21.44% ~ 45.02% parameters of multi-modal fusion without any performance degradation. To sum up, we make the following contributions:

1. We propose *MMExit*, a multi-modal exit architecture to adaptively reduce the computational cost in multi-modal DNN inference tasks with different data samples.
2. We design a new metric called the utility of exit and the equivalent serialization method to navigate the multi-modal DNN inference tasks to exit adaptively.
3. We define a joint loss function that synthesizes the features of different modalities and layers with a double-stage adaptive re-weighting method to train the *MMExit*.
4. We verify *MMExit* with an extensive number of real-world multi-modal DNN models and datasets based on an open-sourced benchmark.

2 *MMExit*: Architecture Design

2.1 Problem Setup

Background We first briefly introduce the fundamental multi-modal DNN architecture. Without loss of generality, we consider a classification task that leverages the multi-modal DNN to process and fuse the features from n modalities. We use $m_1, ..., m_n$ to denote these modalities. To train the multi-modal DNN, we construct a dataset that contains N data samples denoted as $\mathcal{D} = \{(x^i_{m_1}, x^i_{m_2}, \cdots, x^i_{m_n}, y_i)\}_{i=1}^N$. The goal is to predict the correct label y with the network and dataset. Figure 2 shows the common structure of the multi-modal DNN. It consists of multiple, parallel modality encoder sub-networks as well as a sequential fusion and classification sub-network, i.e., icons highlighted by black border and white background. These encoder sub-networks are responsible for obtaining the representations of different modalities. Typically, they can be implemented with standard uni-modal DNNs, determined by the characteristics of modality [16]. After that, the fusion and classification sub-network is used to merge the representations of all modalities and produce the final prediction.

The *MMExit* Architecture The key behind *MMExit* is that for most data samples, the feature learned from a fraction of modalities is sufficient to produce the final prediction y with high confidence. For example, it is widely accepted that most data samples can be addressed using simple models [8]. In multi-modal settings, some modalities achieve better performance than others in many cases [16].

Therefore, in the *MMExit* network, we obtain the prediction label y through the exits from the modality encoder sub-networks or the exit after fusion. As shown in Figure 2, we define two classes of exits. The first is the encoder exit at each encoder sub-network and the second is the fusion exit at the fusion and classification sub-network. Assuming n modalities in the multi-modal DNN application, any inference task has $n + 1$ exits including n encoder exits and one fusion exit. For the i-th sub-network, we assume the j-th exit point in it is denoted as $e^{(m_i,j)}$. We use a lightweight classification head to transform the features learned at this point into the final predictions.

$$y_e^{(m_i,j)} = f_e^{(m_i,j)}(z_i; \theta_e^{(m_i,j)}) \tag{1}$$

where $y_e^{(m_i,j)}$ is a vector that represents the predicted probability. Then, we calculate the normalized entropy as the confidence of the prediction result from exit $e^{(i,j)}$ as,

$$H(e^{(m_i,j)}) = -\frac{1}{\log(C)} y_e^{(m_i,j)} \log(y_e^{(m_i,j)}) \tag{2}$$

where C is the number of classes in the classification task.

2.2 Discussion on *MMExit*

In the DNN inference process, the optimal exit with minimal computational cost is the earliest one to meet the accuracy. In the previous uni-modal early-exit architecture [24], the accuracy and computational cost of an exit is only related to the depth of layers. Thus, it is easy to find the optimal exit fast in the uni-modal network due to the explicit relationship between different exits. However, *in MMExit architecture, there is no fixed relationship between performance and computational cost of the exits on different modality encoder sub-networks*. Thus, we have to determine which modality should be processed in advance in order to provide the expected prediction results with the least amount of computation. In the training process, the uni-modal exit network only needs to set different weights for different exits. However, *when training the MMExit networks, we must determine the weights for different modalities and exits*, which requires a joint training approach to improve the robustness of *MMExit*.

3 *MMExit*: Adaptive Inference

3.1 Utility Assessment Metric

We define a metric named utility of exit (UoE) to measure the benefit of an exit in terms of its accuracy and computational cost. A larger UoE for an exit indicates the benefit of the performance improvement from the exit outweighs its computational cost. Conversely, a smaller UoE means that the utility of the exit is not good. The mission of

the inference process is to find the optimal exit that has the highest UoE, thus avoiding the waste of computation while satisfying the performance.

To compute the UoE of an encoder exit, for the i-th modality, we denote its modality encoder sub-network as $f_u^i(\cdot)$. We assume modality m_i has e_{m_i} different exits, forming a set of exit classification network, which denoted by $E_{m_i} = \langle f_e^{(m_i,1)}, f_e^{(m_i,2)}, ..., f_e^{(m_i,e_{m_i})} \rangle$. For the j-th exit $f_e^{(m_i,j)} \in E_{m_i}$ of modality encoder sub-network for modality m_i, we assume that it can achieve an accuracy of $a_{m_i}^j$ with a computational cost of $c_{m_i}^j$. Then, we can define the utility of the encoder exit as,

$$\mathcal{U}(e_{m_i}^j) = \lambda a_{m_i}^j - (c_{m_i}^j + \sum_{\substack{f:1\to i-1 \\ g:1\to e_{m_f}}} c_{m_f}^g + \sum_{\substack{f=i \\ g:1\to j-1}} c_{m_f}^g) \tag{3}$$

where λ represents the preference of different applications for performance and computational cost. Notably, we compute the UoE of the fusion exits in a similar way.

Then, we assume that for a standalone exit network a data sample will exit from $f_e^{(m_i,j)}$ with a probability of $p_{m_i}^j$. Thus, in the *MMExit* network, we formulate the probability of a sample exit from y-th exit in x-th modality as follows,

$$P_e(x,y) = \prod_{\substack{i:1\to x-1 \\ j:1\to e_{m_x}}} (1 - p_{m_i}^j) * \prod_{\substack{i=x \\ j:1\to y-1}} (1 - p_{m_i}^j) * p_{m_x}^y \tag{4}$$

3.2 Equivalent Modality Serialization

In the inference process of *MMExit*, the ultimate goal is to generate an order of exits that can maximize the sum of the utility function for all data samples, which is formulated as,

$$\max \sum_{\substack{i:1\to n \\ j:1\to e_{m_i}}} \mathcal{U}(e_{m_i}^j) \tag{5}$$

As shown in Figure 2, the order of exits in a modality encoder sub-network is fixed and the order between the encoder exits and fusion exit is also fixed. We only need to define the execution order of different modalities. Considering the execution order of the modalities as $O = \{o_1, o_2, ..., o_M\}$, the overall target turns to,

$$\arg\max_O \mathcal{U}(O) = \arg\max_O \sum_{\substack{i:1\to M \\ j:1\to e_{m_i}}} P_e(o_i, j) * \mathcal{U}(e_{o_i}^j) \tag{6}$$

This is a hard problem that cannot be solved with a naive method. To order the modalities, the most explicit way is to traverse all orders and select the order which maximizes the utility sum. However, there is a drawback in that the computational process requires traversing all possible modality orders, which leads to unacceptable computational cost when the number of modalities is large. Therefore, we define an equivalent metric ϕ, which defines a fast way to select the optimal modality execution order. Assuming e_{m_i} is equal to e_m for all modalities and $p_{m_i}^j$ is approximately close to p for all exits (the experiments show that our method is able to achieve near-optimal performance even when these assumptions are not satisfied.), the ϕ is formulated as,

$$\phi(i) = (1 - q^{e_m}) * \sum_{j:1 \to e_m} q^{j-1} * p * (\lambda a^j_{m_i} - c^j_{m_i} - \sum_{\substack{f=i \\ g:1 \to j-1}} c^g_{m_f}) - \sum_{j:1 \to e_m} (q^{e_m} * q^{j-1} * p * \sum_{\substack{f=i \\ g:1 \to e_m}} c^g_{m_f}) \tag{7}$$

The validity of the proposed metric ϕ follows the following theorem (the proof is omitted due to the limit of space, but can be easily established using proof by contradiction).

Theorem 1 *Given a modality execution ordering $O = \{o_1, o_2, \cdots, o_n\}$, if it is satisfied that for any i and j ($i \leq j$) we have $\phi(i) \geq \phi(j)$. Then we can conclude that O is the optimal modality execution ordering.*

3.3 *MMExit* Inference Process

In the inference process, the mission of the inference process is to find the optimal exit for a set of data samples. To this end, we can profile the correlation between performance gain and computational cost for each modality by computing the UoE. Then, we rank all the modalities according to their utility functions in an order $O = \{o_1, o_2, \cdots, o_n\}$ with the equivalent serialization method. Notably that the accuracy $a^j_{m_i}$ and the probability $p^j_{m_i}$ can be collected in the training phase. In addition, we leverage a validation set to estimate it and then dynamically change the ordering.

4 *MMExit*: Joint Training

4.1 Joint Loss Function

To train the proposed *MMExit* classification network, we use the cross-entropy between the predicted and real label as the loss function. We assume the loss function for the j-th exit from the i-th modality encoder sub-network and the fusion sub-network is respectively represented as $\mathcal{L}^j_i$ and $\mathcal{L}_c$,

$$\mathcal{L}^j_i(y^j_i, \hat{y}) = -\sum \hat{y} \log(y^j_i); \quad \mathcal{L}_c(y_c, \hat{y}) = -\sum \hat{y} \log(y_c) \tag{8}$$

where $\hat{y}$ is the real label and y^j_i and y_c is the predicted label.

Considering that the *MMExit* network has n predicted labels from the n encoder exits and one predicted label from the fusion exit, we formulate the overall loss function using the weighted sum of the losses from all exits.

$$\mathcal{L} = \sum_{i=1}^{M} \sum_{j=1}^{e_{m_i}} w_{ij} \mathcal{L}^j_i(y, \hat{y}) + \mathcal{L}_c(y, \hat{y}) \tag{9}$$

4.2 Objective Analysis

The training loss of the *MMExit* network is determined by the features of both the modalities and exit layers. We first analyze the effect of the modalities on training loss. We consider the multi-modal classification network $f_c(z; \theta_c)$ with parameters

$\theta_c = \{W \in R^{d_{z_1}+d_{z_2}+\cdots,+d_{z_M}}, b \in R^M\}$ as shown in Section 2, the layer of which can be represented as,

$$f_c(x_i) = W\left[f_u^1\left(\theta_u^1, x_1^i\right) \oplus \cdots \oplus f_u^M\left(\theta_u^1, x_M^i\right)\right] + b \tag{10}$$

It's obvious that the weight matrix W can be split into several blocks represented as $W = [W_{m_1}; W_{m_2}; \cdots; W_{m_M}]$, then we can rewrite the above equation as,

$$f_c(x_i) = W_{m_1} \cdot f_u^1\left(\theta_u^1, x_1^i\right) + \cdots + W_{m_M} f_u^M\left(\theta^v, x_M^i\right) + b. \tag{11}$$

The update of the weight parameter is,

$$W_{m_i}^{t+1} == W_{m_i}^t - \eta \frac{1}{N} \sum_{j=1}^{N} \frac{\partial L}{\partial f_c(x_j)} f_u^i\left(\theta_u^i, x_i^j\right) \tag{12}$$

Then, we can update the overall loss as,

$$\frac{\partial L}{\partial f(x_i)_c} = \frac{e^{(f_c(x_i))_c}}{\sum_{k=1}^{M} e^{(f_c(x_i))_k}} - 1_{c=y_i} \tag{13}$$

where $f(x_i)_c$ is the logits output for class c. It is obvious that the overall gradient will be dominated by the stronger modality (with a smaller gradient), which finally makes the other modalities not converge to the optimal value. To alleviate this, we need to give larger weights to the strong modalities (with lower loss).

Then, we consider the effect of the multi-exit network. We assume training the exits at the sub-network for modality i with loss $\mathcal{L}_i$. Some previous studies have found that training exits sequentially is sub-optimal compared to jointly optimizing all exits [12]. It involves two aspects of the learning objective. On the one hand, the earlier features are not sufficiently predictable and have larger gradients. On the other hand, the earlier part of the network will receive gradient back-propagation from all later exits. The gradient of the s-th block is contributed by the s-th node and the subsequent $(e_{m_i} - s)$ exits denoted as,

$$\nabla_{\theta_u^{i,s}} \mathcal{L}_i = \sum_{j=s}^{e_{m_i}} w_{m_i}^j \nabla_{\theta_u^{i,s}} \mathcal{L}_i^j \tag{14}$$

where $\theta_u^{i,s}$ is the features at s-th block. This illustrates that the earlier exits usually have more weight, making them more important in the optimization process and dominating the training process. So, we need to give smaller weights to the earlier exits, which have higher losses in the training process.

Training *MMExit* contains two conflict objectives in terms of the loss value [20]. For the multi-modal part, the training process can be dominated by strong modalities (with less loss), which suppresses the training of weak modalities and is not conducive to better performance of the overall multi-modal model. For the multi-exit part, its exit structure leads to the fact that the early blocks receive the gradient back-propagate from all the later exits, which leads to its possible domination of the whole training process, resulting in poor performance of the whole network. Traditional adaptive methods tend to solve one of these problems by weighting the losses according to the gradient values (or similar metrics) in various ways. They cannot be directly applied to more complex *MMExit* training.

Algorithm 1: *MMExit* Training Algorithm

Input: Trained model $\mathcal{M}$, preference λ, threshold ϵ.
Output: Trained *MMExit* model.

```
for i = 1, ··· , e do
    Get the predictions from the model.
    Calculate the losses and update the running mean according to Eq. (15).
    if i % 2 = 0 then
        Calculate {w_i^{m_i}}_{i=1}^M according to Eq. (16).
        Calculate the loss L_e according to Eq. (17).
    else
        Calculate exit weights according to Eq. (16).
        Calculate the loss L_o according to Eq. (18).
    Update model parameters with gradient descent.
```

4.3 *MMExit* Training Algorithm

Based on previous analysis, we propose a double-stage adaptive re-weighting method to train the *MMExit* network. Firstly, we use the running average of the gradients to weigh the predictive capabilities of different exits. A high gradient always implies a fairly large gap between the predicted label and the true label. For exit j in each modality i, we denote the gradient as $g_{(i,j)}$, then we can formulate the average gradient at step t as,

$$\bar{g}_{(i,j)}^{t} = \alpha \bar{g}_{(i,j)}^{t-1} + (1-\alpha) g_{(i,j)} \tag{15}$$

where α is the weight parameters to control the importance of the current value and previous values. Then, we define the weight for different exits as,

$$w_{i,j}^{t} = \frac{\exp(\bar{g}_{(i,j)}^{t}/\tau)}{(m_i - j + 1)\sum_{x \in \mathcal{Q}} \exp(\bar{g}_{x}^{t}/\tau)} \tag{16}$$

where $\mathcal{Q}$ is the set of exits considered in the update step, τ is the temperature parameter for softmax function. We decrease the temperature parameter during training to help the model focus on some hard parts of the model. With this, we can make better use of the strong modalities and alleviate the effect of the earlier sub-network. Then we can train *MMExit* effectively by balancing different components in the network.

Given the aforementioned discussions, we introduce a novel double-stage cross-training strategy to train the *MMExit* network. The proposed training algorithm is depicted in Algorithm 1. The training process comprises two stages that alternate in consecutive epochs. Specifically, in the even-numbered epochs, the training of the multi-modal part of the network is achieved by using the following model representation,

$$\min \mathcal{L}_e = \sum_{i=1}^{M} w_{i,m^i}^{t} \mathcal{L}_i^{m_i} + \mathcal{L}_c \tag{17}$$

In the odd-numbered epochs, the model trains the remaining exits and is denoted as,

$$\min \mathcal{L}_o = \sum_{i=1}^{M} \sum_{j=1}^{m_i - 1} w_{i,j}^{t} \mathcal{L}_i^{j} \tag{18}$$

Table 1: Description of the multi-modal Datasets used.

Dataset	Data Samples	***Modality* (Encoder sub-network)**	Classes
sarcasm [5]	690	*spoken language* (BERT/GloVe), *visual* (ResNet), *audio* (Librosa)	2
mosi [25]	2,199		5
mosei [25]	22,777		5
avmnist [21]	70,000	*image* (Raw), *audio* (Spectogram)	10

$$\mathcal{L}^t = \begin{cases} \sum_{i=1}^{M} w_{i,m^i}^t \mathcal{L}_i^{m_i} + \mathcal{L}_c, & t\%2 = 1 \\ \sum_{i=1}^{M} \sum_{j=1}^{m_i - 1} w_{i,j}^t \mathcal{L}_i^j, & t\%2 = 0 \end{cases}$$

5 Experiments and Evaluations

5.1 Experiment Setup

Implementation. We conduct our experiment based on 4 representative multi-modal DNNs and datasets provided by the *Multibench* benchmark [16] from real-world applications (details are shown in Table 1). We construct the *MMExit* networks for each dataset by adding 2 exits per modality to their *late fusion (LF)* networks. LF is the most fundamental method that combines multiple modalities with the concatenation operation. We implement the exits by using one linear layer, which only results in an extra computational cost of less than 0.02% to produce the prediction label. We train and run all these models on a server with one GeForce RTX 2080Ti GPU. We run each experiment 5 times with different random seeds for reliability. Notably that it is easy to apply our *MMExit* method to other state-of-the-art multi-modal networks such as *MIM*, *TF* and *LRTF* to reduce their computational effort. However, for the space limitation, we only apply the *MMExit* to the *Humor Knowledge enriched Transformer (HKT)* [9], which is one of the latest multi-modal transformer networks and omit the most content of the integration of *MMExit* with other fusion methods.

Baselines and the state-of-the-art. We use the ***late fusion (LF)*** method as our baseline. We also compare *MMExit* with both the uni-modal methods and the most representative multi-modal methods. In each of the ***uni-modal models (Uni1 ~ Uni3)***, we only use the encoder sub-network of one modality and connect it to the classification network to obtain the output predictions. Among these multi-modal methods, ***Tensor fusion network (TF)*** [26] uses tensor outer product to fuse information from different modalities. ***Low rank tensor fusion network (LRTF)*** [18] leverages a modality-specific set of low-rank factors to improve the efficiency of tensor fusion. ***Multiplicative interaction model (MIM)*** [13] further generalizes the tensor products to capture and learn the interactions between different modalities. We also implement another multi-exit method called ***RExit***, which inserts exits without *the equivalent serialization* optimization.

5.2 Visualization

We first illustrate that *MMExit* has the ability to adaptively exit from appropriate modalities and network layers for different data samples. In Figure 3, we plot the results for the

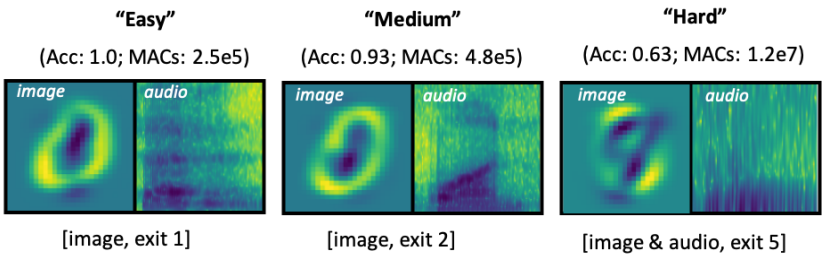
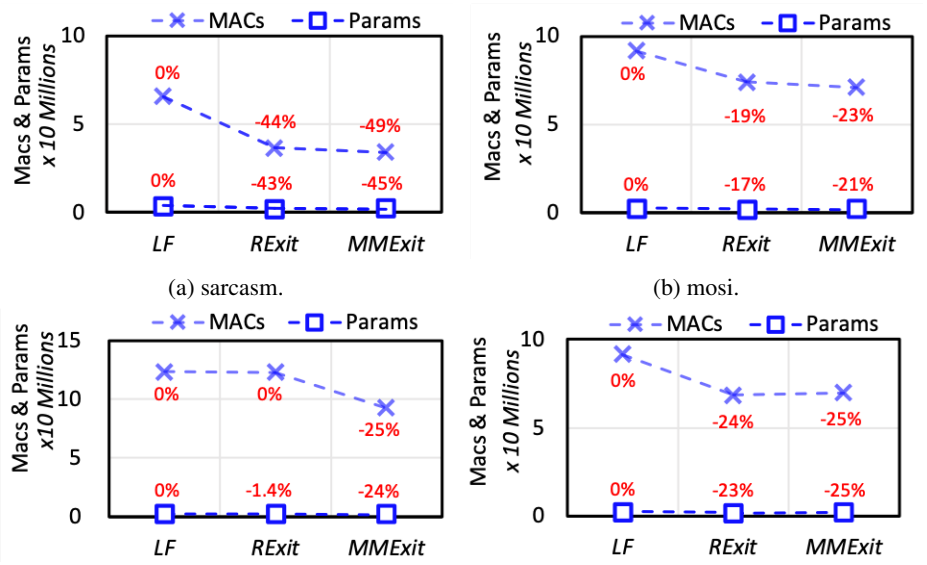


Figure 3: Visualization of *MMExit* under the *avmnist* dataset.

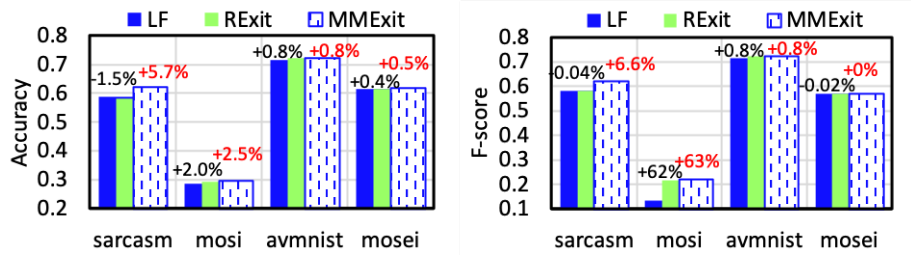


(a) sarcasm. (b) mosi.

(c) avmnist. (d) mosei.

Figure 4: Computational effort under the two exit schemes.

avmnist dataset which classifies data samples based on two modalities including *image* and *audio*. The data samples of *image* are represented in pixels, and the data samples of *audio* are represented with a 112×112 spectrogram. First of all, we can see that for a very **"Easy"** sample, *MMExit* is able to perform an accurate recognition at the first exit which significantly saves the computational effort. For more complex sample, *MMExit* extracts more features from the modality of *image* by exiting later at the second exit to obtain the prediction result with high accuracy. For both **"Easy"** and **"Medium"** samples, they can be classified accurately by exiting from different layers of the *image* modality. However, for some **"Hard"** samples, which happens less frequently, it is difficult to obtain its correct label only from the *image* modality. In this case, *MMExit* has to complete both *image* and *audio* modalities to compute the final prediction.



(a) Accuracy. (b) F1-score.

Figure 5: Performance comparison of the two exit schemes.

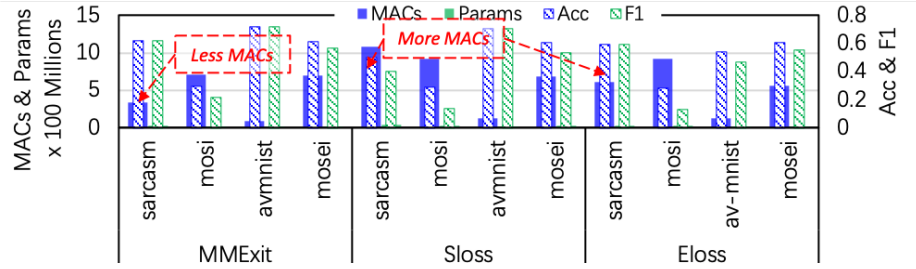


Figure 6: The impact of different training algorithms on the performance gain and computational effort of *MMExit*.

5.3 Ablation Study

Utility Analysis: To evaluate the equivalent serialization method, We compare both the performance and computation cost including network parameters and MAC operations under *RExit* and *MMExit*. As shown in Figure 4, both the *RExit* and *MMExit* can reduce the number of MACs and parameters of the *LF* baseline method. For these datasets, *MMExit* can reduce the computation load by 23% ˜ 49%. Although *RExit* can reduce 1.4% ˜ 44% computational effort as well, *RExit* cannot guarantee the performance (i.e., accuracy and F-scores) with unawareness of the trade-off between the accuracy gain and computational effort as shown in Figure 5. Overall, the proposed *MMExit* can always find the optimal tradeoffs between computation load and performance.

Joint training algorithm: To verify the effectiveness of the training algorithm, we compare it with two commonly-used training strategies. In Figure 6, the *Eloss* represents the ones which treat and train all the exits equally with the same weight [22]. The *Sloss* stands for the ones which group different layers and assign different groups with static weights increasing from previous to latter groups. In the figure, the more overlap of two bars means requiring more computational effort to obtain the performance gains. For example, *Eloss* and *Sloss* explicitly consumes more MACs than *MMExit* . We can see that *MMExit* always intends to guarantee higher accuracy and less computation load compared with other training loss functions under all the datasets.

5.4 Performance Evaluation

In Table 2, we compare the performance of *MMExit* with both the uni-modal and multi-modal methods under various datasets. It is evident that *MMExit* obtains better

Table 2: Accuracy and weighted F1 score of the 4 datasets.

	sarcasm		**mosei**		**mosi**		**avmnist**	
	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
Uni1	0.536	0.538	0.573	0.422	0.286	0.130	0.651	0.649
Uni2	0.470	0.440	0.575	0.420	0.289	0.137	0.421	0.421
Uni3	0.613	0.611	0.612	0.571	0.287	0.128	-	-
TF	0.535	0.492	0.612	0.567	0.287	0.130	0.712	0.710
LRTF	0.467	0.364	0.591	0.484	0.287	0.128	0.715	0.714
MIM	0.455	0.352	0.611	0.557	0.285	0.128	0.716	0.714
LF	0.588	0.583	0.614	0.570	0.288	0.134	0.717	0.715
MMExit	**0.622**	**0.622**	**0.617**	**0.570**	**0.295**	**0.220**	**0.722**	**0.720**

Table 3: The benefits of applying *MMExit* to *HKT* on sarcasm dataset.

Methods	**Accuracy**	**F1 Score**	**Parameters (MB)**	**Time (ms)**
HKT	0.7647	0.7639	12.12	28.43
MMExit+HKT	0.7941	0.7941	8.63	21.73
Improvements	**+0.0294**	**+0.0302**	**-3.50 (28.9%)**	**-6.7 (23.6%)**

output predictions than the uni-modal models by fusing multiple modalities. In addition, *MMExit* can achieve the same or even higher accuracy and F-scores than the *LF* baseline method in all the scenarios. It also achieves the best performance compared to the most state-of-the-art multi-modal methods. It is notable that *MMExit* can be easily applied to more advanced multi-modal networks such as *MIM*, *LRTF* and *HKT* to reduce their computational effort. Table 3 shows that applying the *MMExit* to *HKT* can reduce its parameters, thus significantly reducing the inference latency and improving the performance. Overall, *MMExit* can reduce the computational effort of multi-modal networks without any performance degradation.

5.5 Reduction of Computation

An important design objective of *MMExit* is to reduce the computational effort of the existing multi-modal methods. In this part, we compare the MACs and parameters of different methods. As shown in Figure 7, the *MMExit* reduces 22.64% ˜ 48.72% MACs and 21.44% ˜ 45.02% parameters of the *LF* method. It even consumes less MACs and parameters that the uni-modal networks. For example, *MMExit* has 13.0% less MACs than the uni-modal network for the *mosi* dataset. Combined with the results in Table 2, *MMExit* offers the probability to improve the existing model in terms of performance and efficiency, which is important for real-world deployments. Moreover, the reduction in computation complexity would lead to additional benefits such as speeding up the inference processes as shown in Table 3.

6 Related Work

Multi-modal DNNs: Multi-modal deep neural networks [27] are designed to merge complementary information from various modalities like text, audio, image, etc. They have been demonstrated to outperform the uni-modal networks in many application

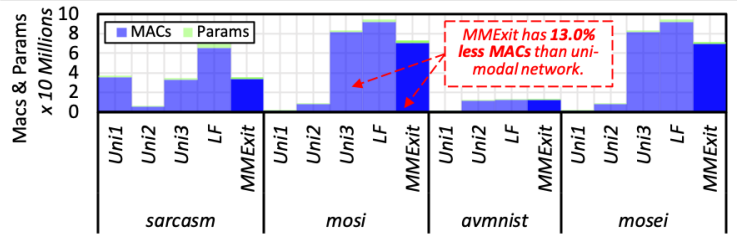


Figure 7: Comparison of computational effort.

fields [16]. The most typical multi-modal architecture consists of multiple heterogeneous encoders to obtain representations of different modalities. These representations are then fused using either early fusion methods [19] or late fusion methods [3]. Recently, multi-modal transformers [14] are proposed, which are powerful but computationally intensive, using only transformers to obtain and fuse multi-modal features. *MMExit is orthogonal to all these methods. It can be used to reduce their computational effort.*

Early Exit Neural Network: Early exit which has been extensively studied for uni-modal DNN inference tasks [8], is the most similar to our work. Compared to other state-of-the-art neural network (NN) compression methods such as pruning [6] and quantization [4], early exit [22,8,15,7] aims to reduce the computation of network layers adapted to different inference tasks, thus making DNNs more applicable in some resource-limited application scenarios. For example, some previous work leverages early exit [22,8,10] to adapt edge DNN tasks to resource-limited AIoT devices. *MMExit is a new adaptive neural architecture for multi-modal DNNs.*

7 Conclusion

While multi-modal DNNs have culminated in significant accuracy gain, they also lead to an explosive increase in computational cost, which would hinder their deployment in many real-world applications. To address this, we propose a novel multi-modal exit architecture called *MMExit*. To the best of our knowledge, it is the first multi-modal exit network that provides adaptive inference with minimal computational effort. *MMExit* shows great potential in applying multi-modal networks to the next-generation resource-constrained scenarios such as smart networking devices, mobile robots, etc.

Acknowledgements

This work is supported in part by the National Key R&D Program of China under grant No.2021ZD0110104, and the National Natural Science Foundation of China under grant No.62122053. It was also partially supported by ACCESS - AI Chip Center for Emerging Smart Systems, InnoHK funding, Hong Kong SAR. We thank all the anonymous reviewers for their valuable feedback.

References

1. Akhtar, M.S., Chauhan, D.S., Ghosal, D., Poria, S., Ekbal, A., Bhattacharyya, P.: Multi-task learning for multi-modal emotion recognition and sentiment analysis. In: NAACL-HLT (2019)
2. Arevalo, J., Solorio, T., Montes-y Gómez, M., González, F.A.: Gated multimodal units for information fusion. In: ICLR (2017)
3. Bach, F.R., Lanckriet, G.R., Jordan, M.I.: Multiple kernel learning, conic duality, and the smo algorithm. In: ICML (2004)
4. Bhattacharjee, A., et al.: Mime: adapting a single neural network for multi-task inference with memory-efficient dynamic pruning. DAC (2022)
5. Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R., Poria, S.: Towards multimodal sarcasm detection (an _obviously_ perfect paper). In: ACL (2019)
6. Choi, K., Yang, H.: A gpu architecture aware fine-grain pruning technique for deep neural networks. In: Euro-Par (2021)
7. Cui, W., Zhao, H., Chen, Q., Wei, H., Li, Z., Zeng, D., Li, C., Guo, M.: Dvabatch: Diversity-aware multi-entry multi-exit batching for efficient processing of dnn services on gpus. In: USENIX ATC (2022)
8. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. In: TPAMI (2021)
9. Hasan, M.K., Lee, S., Rahman, W., Zadeh, A., Mihalcea, R., Morency, L.P., Hoque, E.: Humor knowledge enriched transformer for understanding multimodal humor. In: AAAI (2021)
10. Hou, X., Liu, J., Tang, X., Li, C., Chen, J., Liang, L., Cheng, K.T., Guo, M.: Architecting efficient multi-modal aiot systems. In: ISCA (2023)
11. Hou, X., Xu, C., Liu, J., Tang, X., Sun, L., Li, C., Cheng, K.T.: Characterizing and understanding end-to-end multi-modal neural networks on gpus. In: IEEE CAL (2022)
12. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. In: ICLR (2018)
13. Jayakumar, S.M., Czarnecki, W.M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y.W., Harley, T., Pascanu, R.: Multiplicative interactions and where to find them. In: ICLR (2020)
14. Kim, W., Son, B., Kim, I.: Vilt: Vision-and-language transformer without convolution or region supervision. In: ICML (2021)
15. Laskaridis, S., Kouris, A., Lane, N.D.: Adaptive inference through early-exit networks: Design, challenges and directions. In: MobiSys (2021)
16. Liang, P.P., Lyu, Y., Fan, X., Wu, Z., Cheng, Y., Wu, J., Chen, L.Y., Wu, P., Lee, M.A., Zhu, Y.: Multibench: Multiscale benchmarks for multimodal representation learning. In: NeurIPS (2021)
17. Liu, J., Hou, X., Tang, F.: Fine-grained machine teaching with attention modeling. In: AAAI (2020)
18. Liu, Z., Shen, Y., Lakshminarasimhan, V.B., Liang, P.P., Zadeh, A., Morency, L.P.: Efficient low-rank multimodal fusion with modality-specific factors. In: ACL (2018)
19. Natalia, N., Christian, W., Graham, W.T., Florian, N.: Multi-scale deep learning for gesture detection and localization. In: ECCV (2014)
20. Peng, X., Wei, Y., Deng, A., Wang, D., Hu, D.: Balanced multimodal learning via on-the-fly gradient modulation. In: CVPR (2022)
21. Pham, H., Liang, P.P., Manzini, T., Morency, L.P., Póczos, B.: Found in translation: Learning robust joint representations by cyclic translations between modalities. In: AAAI (2019)
22. Scardapane, S., Scarpiniti, M., Baccarelli, E., Uncini, A.: Why should we add early exits to neural networks? In: Cognitive Computation (2020)
23. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks. In: Synthesis Lectures on Computer Architecture (2020)

24. Teerapittayanon, S., McDanel, B., Kung, H.T.: Branchynet: Fast inference via early exiting from deep neural networks. In: ICPR (2016)
25. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. In: TPAMI (2016)
26. Zadeh, A., Chen, M., Poria, S., Cambria, E., Morency, L.P.: Tensor fusion network for multimodal sentiment analysis. In: EMNLP (2017)
27. Zhang, C., Yang, Z., He, X., Deng, L.: Multimodal intelligence: Representation learning, information fusion, and applications. In: JSTSP (2020)