MOSEL: Inference Serving Using Dynamic Modality Selection

Bodun Hu *Le Xu* *Jeongyoon Moon* *Neeraja J. Yadwadkar* *Aditya Akella*
The University of Texas at Austin

Abstract

Rapid advancements over the years have helped machine learning models reach previously hard-to-achieve goals, sometimes even exceeding human capabilities. However, to attain the desired accuracy, the model sizes and in turn their computational requirements have increased drastically. Thus, serving predictions from these models to meet any target latency and cost requirements of applications remains a key challenge, despite recent work in building inference-serving systems as well as algorithmic approaches that dynamically adapt models based on inputs. In this paper, we introduce a form of *dynamism*, modality selection, where we adaptively choose modalities from inference inputs while maintaining the model quality. We introduce MOSEL, an automated inference serving system for multi-modal ML models that carefully picks input modalities per request based on user-defined performance and accuracy requirements. MOSEL exploits modality configurations extensively, improving system throughput by 3.6× with an accuracy guarantee and shortening job completion times by 11×.

1 Introduction

With recent advancements in Deep Learning Models (DNNs), especially Transformers, Machine Learning (ML) has been far exceeding human capabilities in various Computer Vision and Natural Language Processing tasks [28, 69]. However, the computational requirement of the largest ML models/applications has doubled every few months, resulting in a 1,000,000× increase from 2012 to 2020 [58]. The increasing size of the models presents fundamental challenges in terms of latency and cost when they are commissioned for inference [23, 24, 55].

To combat the overheads associated with model sizes, today's services often replace a large model with a single, cheaper variant, typically obtained using ML *compression techniques*, such as distillation [43, 57], pruning [22, 36] and quantization [52]. Although compression techniques are quite successful in achieving their design goals in small and medium-sized models, larger and more sophisticated models (e.g., autoregressive/generative models) face new challenges: it may not always be possible to produce a small enough model that matches (or is close to) the original model in accuracy while meeting the cost constraints for all inputs.

Alternatively, approaches that adapt the models dynamically based on the inputs have been proposed. Such techniques that explore dynamic adaptation opportunities of a model fall into two classes: (i) techniques that reuse the original model and (ii) techniques that rethink the original model. In the former, the key idea is to maintain the model's original architecture and augment it with adaptation capabilities. Examples of these include early-exiting [66, 71, 72, 82], and layer-skipping [65]. The latter techniques modify and/or replace the original model with equivalents that support adaptation; examples in this category include ensembling [10], model merging [49], and mixture-of-experts [20, 53]. Such dynamism offers latency and resource savings [11, 66, 72], but demands new systems designs that are both adaptive and flexible.

In this paper, we introduce a new form of dynamism that falls into the former category. In particular, we propose *modulating the input*, specifically via *selectively using* parts of it. We develop this idea and demonstrate its usefulness in the context of *multi-modal learning* [8, 47], an emerging and important class of ML techniques that combine data of different modalities to provide prediction cooperatively, enhancing ML models' prediction accuracy.

As we describe in Section 2.2, we empirically find that, for inference using such models, some modalities (e.g., the audio modality in the Textless Vision-Language Transformer (TVLT) model [65]) contribute significantly to prediction accuracy while not being the major contributor to resource use (e.g., memory) and processing time, while other modalities (e.g., the video modality in TVLT) consume a significant amount of resources and incur latency while only marginally contributing to prediction accuracy. The observations depend on the inputs being processed and the type of model in use.

We leverage the above insight in inference settings and propose that modalities be selectively enabled or disabled in a given sequence of inference requests, depending on application requirements and workload patterns and distributions, creating novel opportunities to exploit the trade-off between memory/speed and accuracy that multi-modality presents. We refer to this as *modality selection*, and observe that it complements the first set of dynamism techniques above that modulate or change the model being deployed. It can also be directly applied to the original model, and it is complementary to compression and distillation techniques.

We build MOSEL, an automated inference system for multi-modal models that carefully preforms input modalities selection per request based on user-defined performance and accuracy requirements. We develop MOSEL to ensure scaling and performance at inference time by dividing it into offline and online portions. For the former, we develop novel offline profiling strategies that, given a batch of inference requests from a job, help quickly determine at run-time what modalities to use for each request so as to meet the latency requirement at a suitable accuracy. For the latter, we develop dynamic techniques to ensure that, at inference time, late-enqueued jobs don't miss their latency requirements. We do this by facilitating jobs ahead in the inference queue to dynamically reselect modalities to ease the queueing load; this allows later jobs to run at the needed accuracy without missing their latency targets.

We evaluate MOSEL on a set of representative multi-modal models that utilize commonly-seen architectures (Transformer [67], BERT [19], CNN [34]). In our evaluation, MOSEL outperforms *modality-agnostic* approaches in resource utilization and query spike tolerance. MOSEL is capable of reducing job completion time by up to 11× and handling up to 3.6× more requests with accuracy guarantees. Moreover, MOSEL can achieve up to 4.6× throughput when combined with quantization.

2 Background and Motivation

2.1 Background

Inference and its challenges: Inference systems use pre-trained ML models to make predictions or perform tasks based on input data. With the increasing availability of datasets and progress in ML research, ML models have become more accurate and capable, leading to increasing adoptions in production settings [25, 27]. Inferences tasks have shown to dominate ML production costs: [2, 56].

Inference systems typically serve prediction for a variety types of user requests [16, 54, 56]. For inference systems, accuracy and latency are two key factors that impact end-user experiences [56]. Based on deployment scenarios, the requirements imposed by different applications one or the other [29] or both [21] of these factors are vastly different [54]. To complicate the issue even further, inference systems often need to provision for dynamic workloads [15, 16, 74, 79], making dynamic resource provisioning another key requirement for cost/resource effectiveness.

Moreover, the above challenges are magnified when multiple services leverage the same model for inference tasks. Increasing accuracy for one query exercising a shared model directly impacts the availability of resources for subsequent queries, either delaying their processing or affecting their accuracy. Furthermore, some services may prioritize lower latency to ensure faster response times, and others may prefer good throughput, resulting in varying SLOs.

Multimodal Learning: Multimodal learning techniques are shown to surpass the abilities of unimodal techniques by exploiting the complementary nature of different modalities, such as text, image, audio, and video, in input data sources [8, 47]. Multimodal machine learning techniques rely on ways to fuse data from diverse modalities [8]. The existing approaches for fusion can be broadly classified into two categories: early fusion [7, 31, 63] and late fusion [4, 37, 63]. Early fusion consolidates modalities at an early stage, blending features prior to further processing. For example, TVLT [65] converts video and audio data into sequences of patches and concatenates them before the transformer layer. On the other hand, late fusion preserves separate pathways for each modality and merges the outcomes later. For example, Temporal Binding Network (TBN) [32] processes RGB, Flow, and audio separately, then combines them with mid-level fusion alongside sparse temporal sampling of fused representations. Some fusion methods attempt to combine properties from both early and late fusion [30, 46, 46, 51, 68, 73].

2.2 Characteristics of multi-modality ML models

We now characterize the trade-offs in multi-modal inference, using them to motivate techniques for achieving better inference latency and resource-efficiency.

Accuracy Across Modalities: Multimodal deep neural networks (DNNs) can achieve higher accuracy than unimodal models by combining multiple modalities, such as visual, audio, and textual data. However, the relative importance of each modality may depend on various factors, such as the task, the data, and the model architecture. Previous studies [40, 41, 46, 65] have shown that different modalities can have different effects on the performance of multimodal models on different validation datasets. Figure 1 shows the average accuracy of different multimodal models using all possible combinations of modalities. We observe that some models (TVLT) are capable of achieving high accuracy without using all the available modalities in input data.

Performance Implications: Different modalities have different data representations and processing methods. Thus, along with their impact on model accuracy, different modalities im-

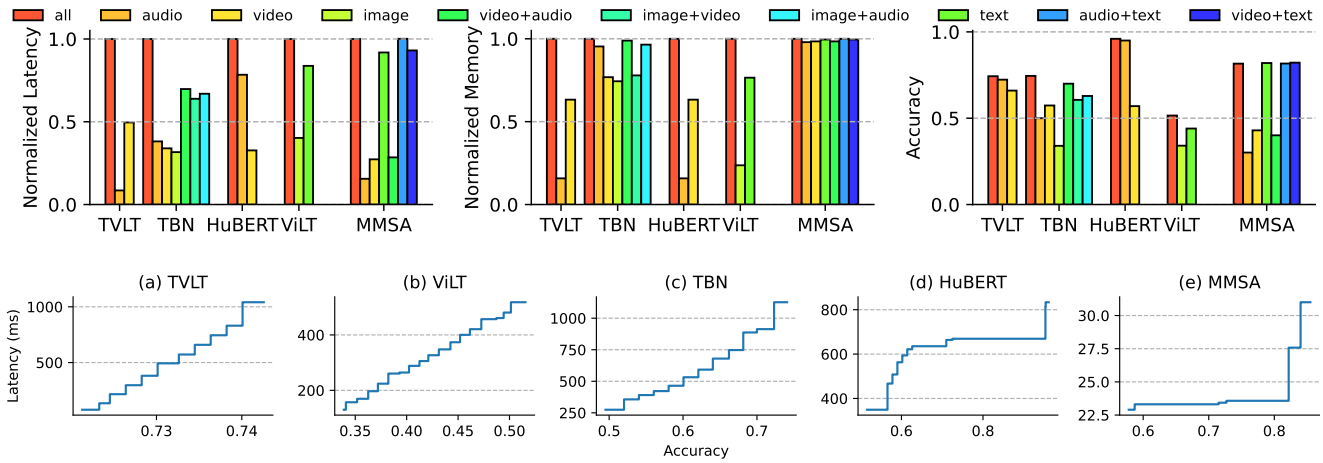

Figure 1: *Performance comparison of different modalities for models discussed in Table 1: (Upper Left) The normalized latency for different modalities. (Upper Middle) The normalized memory footprint of different modalities. (Upper Right) Accuracy comparison using different modalities. (Bottom) Minimum latency required to achieve different levels of accuracy across various models using combinations of modalities.*

pact the latency and memory consumption of multimodal models differently. Figure 1 compares the inference latency and memory consumption of different models. We observe that for TBN, the video modality is more costly than the other modalities in terms of both time and space, due to the large data size of the temporal dimension. TBN uses late-fusion, and so most of the computation happens before fusion. This implies that adding more modalities are likely to increase latency and resource consumption.

For the attention-based models such as the TVLT [65] (shown in Figure 1), we find that the audio modality is more efficient than the video modality in terms of memory usage and processing latency, with only a minor trade-off in accuracy. The memory consumption of TVLT [65] depends on the attention mechanism [67], which scales quadratically with the sequence length. It concatenates different modalities into one sequence, which grows longer as more modalities are added. Thus, using selected few modalities reduces sequence length which results in reduced overall memory consumption of the model. Shorter sequence length also means less computation for serving an inference from the model. A similar approach of using attention-based multi-modal models is adopted by many recent works [26, 39, 46, 61, 64, 65], which may also achieve lower latency and memory consumption by utilizing fewer modalities.

Opportunities: The diverse requirements of applications in terms of the desired accuracy, memory availability, and target latency across different modalities offer interesting possibilities for adaptive multimodal selection, which previous inference-serving systems have not considered. In particular, modalities can be selectively enabled or disabled depending on the application requirements and availability of underlying resources. For instance, for TVLT, as shown in Figure 1, under high load conditions, we can use only the audio modality and achieve a 11× latency reduction without sacrificing much accuracy. Whereas, when system load is low, both video and audio modalities can be used to obtain the highest accuracy, at the cost of higher latency. Figure 1 (Bottom) shows the optimal modality choices for minimizing the latency of 32 requests with a given accuracy objective. Section 4.2 discusses how to generate these modality choices.

3 Challenges

Typically, applications provide the inference systems with SLOs such as accuracy and latency. We define a *job* as a set of requests. Each job has a specified accuracy and latency constraint. *Effective accuracy* is the average accuracy that is achieved across all requests in the job at inference time. This must equal or exceed the job's specified accuracy requirement. Similarly, all requests in the job must be completed within the specified latency.

3.1 Challenges in Exploiting Multiple Modalities

We illustrate the challenges in exploring multiple modalities using Figure 2. Here we have three jobs. Job 1 has one request, and Jobs 2 and 3 have two each. Job 1's request only has the audio modality, whereas the other jobs' requests have both modalities. Job 1 runs from time 0 to 20, and Job 2 arrives at time 10. Job 3 arrives soon after. Requests in jobs are executed in a FIFO manner.

Modality search for one job: To select modalities adaptively, we need a suitable *modality selection strategy* for each job. This is a policy that determines which modalities should be used for each request in a job. Figure 2 illustrates the six possible modality selection strategies for a request in Job 2 or Job 3. For example, in S1, both requests use both modalities,

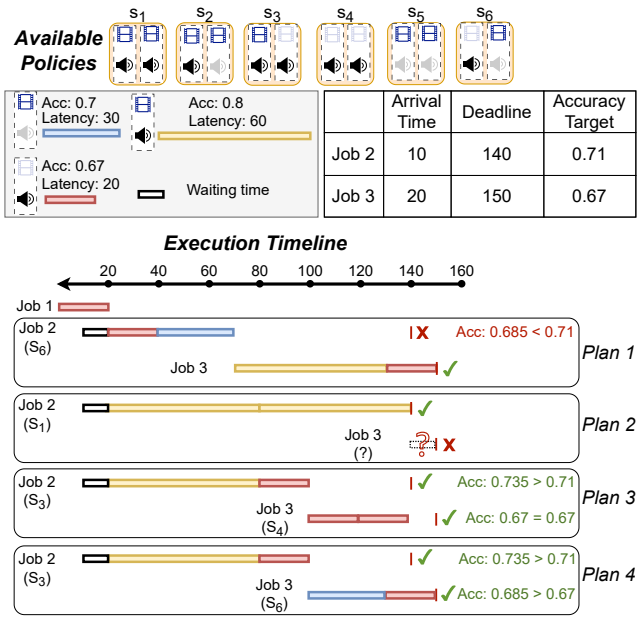

Figure 2: *: (a) Job 1, which has only one audio modality, executes from timestamp 0 to 20. (b) Job 2 arrives at timestamp 10 and executes from timestamp 20, after job 1 completes. One of job 2's modality selection strategy, s_6, shown in Plan 1, has an effective accuracy of* $\frac{0.67+0.7}{2} = 0.685$*, making it unable to satisfy the accuracy requirement of job 2 (Plan 1). Similarly, s_4 also fails to satisfy the accuracy requirement with an accuracy of* 0.67*. (c) Job 3 also arrives at a timestamp a little after 20 and has a deadline constraint of 150. However, if job 2 selects s_1, it will occupy the system until timestamp 140, leaving no feasible modality selection strategy for job 3 to meet its deadline (Plan 2). (d) When job 2 selects a modality that yields the lowest accuracy for one of its requests, the leftover system resource could aid job 3 (Plan 3). (e) In fact, the additional resource could be used to further increase the accuracy of job 3 by using the video modality with higher accuracy (Plan 4).*

whereas in S4 both requests use only the audio modality.

The set of strategies can be large, making choosing among them challenging. The set grows proportional to the number of requests in a job and exponential in the number of modalities per request. For a job with 20 requests and 3 modalities, there are 231 possible modality selection strategies. Also, some of these strategies are infeasible because they may not meet the accuracy and latency constraints of the job. For example, of the six strategies, only two are valid for Job 2, as they satisfy the job's effective accuracy requirement (0.71) without violating the latency constraint (140).

Therefore, given the large search space of possible strategies, we need efficient methods to prune infeasible strategies and estimate the latency of the feasible ones, especially for jobs with low latency requirements.

Planning for multiple jobs: Figure 2 illustrates that multiple strategies can yield valid effective accuracy. But we note that some strategies that create opportunities for a job potentially come at the cost of other jobs. In particular, greedily increasing accuracy for a job comes at the cost of increased resource consumption that may in turn hurt other jobs. This is illustrated by Plan 2 in Figure 2, which offers great accuracy for Job 2 by selecting both modalities for both requests (effective accuracy of 0.8) and finishing exactly by 140 time units, but it leaves no room for Job 3 to finish by its deadline. On the other hand, by lowering accuracy for some jobs, we are left with extra resources that can be used to improve the outcomes for other jobs; e.g., in Plan 3, we use just the audio modality for one of Job 2's requests, yielding an effective accuracy of 0.735, which allows Job 3 to start at time 100 and use the audio modality for both its requests in order to finish by time 140 with an accuracy of 0.67. In fact, we can do better for Job 3 – by picking a higher-accuracy modality (video) for one of its requests, Job 3 achieves an effective accuracy of 0.685 (Plan 4), while finishing at its deadline of 150.

The upshot is that we may have to look for less-than-optimal strategies for some jobs in the queue to enable other later-coming jobs to meet their objectives.

To tackle the challenges for models that dynamically adapt to input data, we need techniques that adapt to the changing SLO requirements and query load across jobs. To tackle such dynamics, existing inference serving systems leverage various techniques, including autoscaling [6, 42], model switching [55, 80], batching [17], predictive serving [23] and preemption [79]. Inference systems for multi-modality such as [35] focus on speculatively executing modalities. All of these techniques are agnostic to input data modalities and to the possibility of exploiting them for performance and efficiency.

4 MOSEL Overview

4.1 Design Goals

We design MOSEL to meet several goals. First, MOSEL should automate modality selection: users should not specify modality choices for jobs; they should only focus on high-level performance, costs, or accuracy requirements. Second, to make modality selection practical during deployment, the system should strive to minimize the overhead of modality selection. Third, given the dynamic variation of the system load, we need to track the system state and adjust the modality selection plan accordingly, ensuring that jobs continue to meet their objectives as best as possible. In this section, we give an overview of our approach that meets these goals and addresses the challenges presented in Section 3.

Figure 3 illustrates the two stages of MOSEL's approach: *offline profiling* and *online optimization*. We outline these next, and discuss their algorithmic details in Section 5.

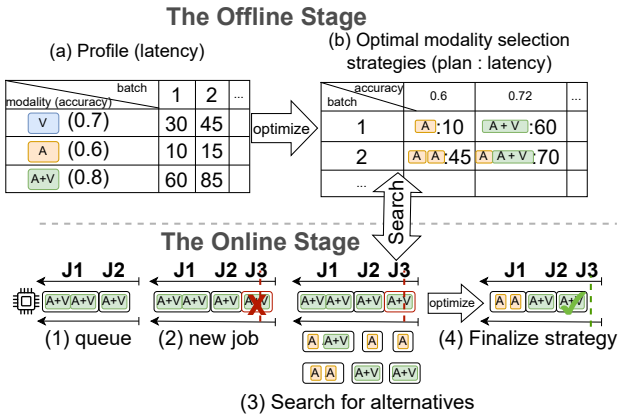

Figure 3: *MOSEL first construct optimized modality selection strategy matrix based on model profile offline. Then it uses this matrix to dynamically derive modality deployment strategies for different jobs online.*

4.2 The Offline Stage

As discussed in Section 3.1, a single job could have various modality selection strategies. Some of these strategies may fail to meet a job's latency goal or SLO. To narrow down the search space, MOSEL needs to find a modality selection strategy with the *low enough* latency, while still satisfying the given accuracy requirement.

We generate suitable modality selection strategies *offline* by constructing *profiles* of the accuracy and latency of each modality using validation datasets. In particular, we evaluate and record the latency of various combinations of modalities under different batch sizes for a given model. Figure 3(a) shows the latency for each combination of modalities and batch size as an entry in a table.

Next, MOSEL uses the accuracy and latency measurements to construct a matrix of modality selection strategies for different job sizes and accuracy constraints for a given model, as shown in Fig 3(b). Each matrix entry shows the best modality combinations for minimizing latency and meeting accuracy needs for a given job size (number of requests), as well as the latency of the chosen combination. The construction is again offline and only happens once before the model is deployed. MOSEL uses a non-linear integer program (NILP) to construct this matrix; more details can be found in Section 5.1.

4.3 The Online Stage

Once a model is deployed, the system queues incoming jobs. Further, MOSEL prioritizes and orders jobs by their deadlines, as shown Figure 3(1) (bottom half; leftmost panel). Each new job adopts the strategy with the highest accuracy by default.

MOSEL monitors the queued jobs and detects if incoming new jobs may suffer from deadline violations. If a job risks missing its deadline, as shown in Figure 3(2), its preceding jobs need to change their modality selection strategies, potentially sacrificing accuracy, but finishing faster and thus reducing the wait time for the job at risk of deadline violation.

To aid this, MOSEL looks up the pre-computed modality selection strategy matrix for all possible strategies for each queued job, as shown in Figure 3(3). Note that we do not consider the resource availability of other jobs when choosing a strategy. We only consider the latency and accuracy constraints of each job, assuming it can start right away without any interference.

At this point, MOSEL reassigns a strategy for each job, avoiding deadline violation and shortening queue delay for all jobs, as shown in Figure 3(4). If the queue is relatively empty, or contains few jobs, MOSEL will attempt to increase the accuracy for all queued jobs by progressively trying modality strategies with higher accuracy for each of them. The modality strategy reassigning process is formulated as an INLP, details in Section 5.2.

5 Formulation

This section describes our nonlinear integer programming (NILP) formulation that identifies the best modality selection strategies to reduce latency and meet accuracy requirements for different job sizes. Then, we explain how we prevent deadline violations and guarantee a fair accuracy distribution among various jobs.

5.1 Offline Optimal Strategy Generation

As discussed in 2.2, different modalities exhibit varying resource consumption patterns. Consequently, such variation would also persist given different batch sizes. For a given model supporting n modalities and batch size of b, we measure the latency for each modality combination and batch size, yielding, in total, $b(2^n-1)$ results. We denote the results collectively as the set $\mathcal{D}$. $\mathcal{D}_{ij}$ represents the latency using modality combination i with batch size j.

To produce the modality selection strategies with the lowest latency for a given job size that satisfies an accuracy SLO α, we divide the job into multiple batches, each using a different modality selection strategy from $\mathcal{D}$. We aim to find the modality combinations and the exact number of requests they should cover, jointly denoted as $\{I, \mathcal{J}\}$, such that the total latency:

$$\sum_{i,j \in I,\mathcal{J}} D_{ij}$$

is minimized, subject to the following constraints: (**1**) The sum of all requests must be equal to the total number of requests. (**2**) The average accuracy of all requests using different modality combinations must exceed the user-specified accuracy objective α. We use $acc(i)$ to denote the effective average accuracy achieved by a modality selection strategy i, and $|\mathcal{R}|$

as the size of a given job. Formally, we have:

$$|\mathcal{R}| = \sum_{j \in \mathcal{J}} j \tag{1}$$

$$\sum_{i,j \in I,\mathcal{J}} acc(i)\, j \geq \alpha |\mathcal{R}| \tag{2}$$

When the target model is deployed, it may receive inference jobs with different sizes and accuracy requirements. Running the NILP entirely online may increase the risk of deadline violations due to the solver's overhead, especially for inference tasks with strict latency constraints. Since the optimization process depends only on accuracy and profiled latency $\mathcal{D}$, we can generate the optimal modality selection strategies for the most common job sizes and accuracy goals completely offline. Formally, given N possible request sizes and A accuracy requirements, we optimize for each of the $N \cdot A$ combinations. This process has negligible overheads.

5.2 Online Modality Selections and Adjustment

We formulate the problem of `MOSEL` selecting alternative modality strategies for jobs in a queue when an incoming job is at risk of deadline violation as an integer non-linear program (INLP), which is a type of optimization problem that involves nonlinear functions and integer variables. Section 7 demonstrates the evaluation setup and the performances of our `optimized` modality-aware approach.

To ensure that no job in the queue misses its deadline (shown in Figure 3), we use a variable T to represent the maximum allowed time budget. If `MOSEL` detects a deadline violation, it sets T to the difference between the violator's deadline and the start time of the most recent job. Otherwise, it sets T to the difference between the deadline of the last job in the queue and the start time of the most recent job. This means that all jobs before the violator must have a modality selection strategy that results in a total latency lower than T.

Each job can have multiple modality selection strategies that satisfy the minimum accuracy objective, as shown in Figure 3(3). These strategies are precomputed in the offline stage. `MOSEL` selects one strategy for each job in the queue, depending on whether there is a deadline violation or not. If there is a violator, `MOSEL` selects one strategy for all jobs before the violator. If no job misses its deadline, `MOSEL` selects one strategy for all jobs in the queue. We denote the set of all such strategies as S. We also denote the set of all jobs before the violator as J. The goal is to select a strategy for each job in J that minimizes the total latency and maximizes the total accuracy.

The objective function that our INLP maximizes is the average accuracy for all requests, defined as:

$$\sum_{s,j \in S,J} acc(s) \cdot |j|$$

We use $l(s)$ to represent the execution latency of a strategy. To select a strategy for each job that fits within the total latency budget, we add the following constraint to our INLP: the sum of the execution latency of the selected strategies for all jobs must not exceed the total time budget T:

$$\sum_{s \in S} l(s) \leq T$$

Algorithm 1 Random modality strategy selection

```
1:  function RandSelect(job, QmodalityStrategies)
2:      S ← {}
3:      if hasDeadlineViolation(jobQ) then
4:          J ← jobsBeforeViolator(jobQ)
5:      else
6:          J ← jobQ
7:      end if
8:      deadline ← overhead(J) + currentTime
9:      while deadline > violatorDeadline do
10:         j ← randomJob(J)
11:         s ← modalityStrategies(|j|,
12:                 accuracy(j))
13:         deadline ← update(deadline, s)
14:         S.append(s)
15:     end while
16:     return S
17: end function
```

A Greedy Heuristic. The time taken to solve each instance of our INLP is long (up to 70 ms) making it challenging to use in an online fashion for jobs with low latency requirements. To address this issue, we also propose a greedy heuristic that adapts the accuracy of enqueued jobs by randomly applying modality selection strategies with the lowest latency and accuracy above the minimum accuracy requirements for jobs before the deadline violator until the queue wait time for the violator is within its deadline. The steps are described in Algorithm 1. We present the evaluation setup and the performance of the `random` heuristic in Section 7.

6 `MOSEL` Implementation

`MOSEL` is implemented in 3k lines of Python code. The offline profiler uses Pytorch [50] to execute ❶ DNNs on the GPU and profile ❷ system metrics through CUDA API. We use GEKKO [9] to generate ❸ the offline modality selection strategies. GEKKO is an optimizer that solves large-scale mixed-integer and differential algebraic equations with nonlinear programming solvers. The generated selection strategies can be stored in a single pickle object. When a model is deployed on an accelerator, a monitor process and a worker process are launched. The monitor process buffers incoming jobs, ❹ retrieves the generated modality plans, then ❺ uses

GEKKO to finalize the modality plan for each job, and puts the job into a FIFO queue shared with the worker process. The GEKKO optimizer can take up 70 ms seconds to generate a solution. Therefore, the monitor process enqueues enough jobs to compensate for the optimizer overhead. In addition, to avoid overflowing the FIFO queue, the monitor process buffers more jobs and pauses periodically. The worker process polls the FIFO queue and executes the ❻ the jobs. It also reports the latest ❼ execution latency metrics back to the monitor process, so that it can estimate available system resources more accurately.

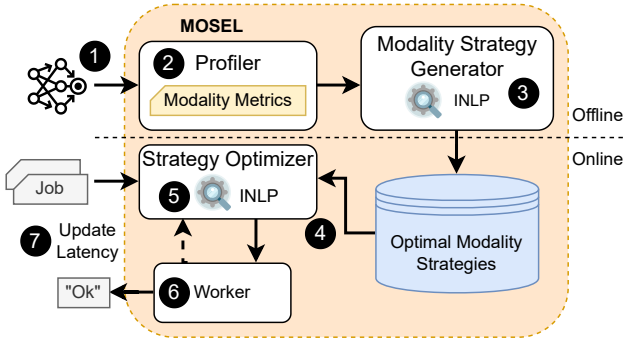

Figure 4: *MOSEL Workflow*

7 Evaluation

To evaluate our implementation, we conduct experiments using realistic workloads and address the following questions:
Q1*: What are the benefits of modality-aware optimizations?* (Section 7.1)
Q2*: How compatible is MOSEL with existing model optimization techniques?* (Section 7.2)
Q3*: Is MOSEL resilient towards profiling error?* (Section 7.4)

Unless specified otherwise, our experiments use the following configurations.

Experimental Setup All measurements are conducted on real hardware using a NVIDIA Tesla A100 GPU with 1.48 GHz shader clock and 80GB DRAM. The CPU is an Intel Xeon Silver 4314 running at 2.40GHz (128GiB DRAM). We used NVIDIA driver version 525.85 and CUDA version 12.0. The PyTorch version is 2.1.0. The operating system is Ubuntu 22.04.1 LTS with 5.15.0 kernel.

Models. Table 1 summarizes the five models used for evaluation; all are pre-trained using PyTorch. The models differ in size and fusion strategy. All models are finetuned on the task-specific datasets, and preloaded onto the GPU before evaluation.

Workloads. We conducted experiments using both synthetic and real-world query patterns. For synthetic workloads, we generated queries with constant loads and a fixed time interval. For real-world workloads, we used the timing information from a Twitter trace, collected over a month in 2018 [3]. Previous work on inference serving has shown this trace reflects realistic inference workloads with diurnal patterns and unexpected spikes [78]. For each experiment, we randomly selected a day out of the month from the Twitter trace.

7.1 MOSEL with production workload

To show that through dynamic modality selection mechanism, MOSEL improves the throughput, utilization, and reduces SLO violations under heavy load.

Experimental setup. We evaluated various models, which are summarized in Table 1. To account for the different processing latency of each model, we adjusted the query per second (QPS) accordingly. The Twitter trace is mapped to a minimum of 5 QPS. We set the maximum QPS for each model based on its capacity to process requests within one second without missing any deadline. The maximum QPS for TVLT, AVHuBERT, TBN, MMSA, and ViLT is set to 60, 20, 40, 100, and 40, respectively. These values were 2× the number of maximum requests each model is capable of handling under one second. We generated requests for each job following a normal distribution, with a mean of 1 and a standard deviation of 6, until the total number of requests matches the QPS. We randomly assigned the accuracy of each job within the range of the model's performance, which was determined by the lowest and highest accuracy that the model can achieve using different modalities.

We used two types of approaches, modality-aware and modality-agnostic, to optimize the online performance of the models. We implemented four different policies for these approaches. The modality-aware approach includes: (a) `optimized`: described in Section 5.2, it attempts to utilize available resources to achieve the highest accuracy across all enqueued jobs; (b) `random`: described by Algorithm 1, this policy randomly selects different jobs from the queue and applies the fastest strategy that meets the accuracy SLO. It repeats this process until no deadline violation can occur;(c) `aggressive`: different to the random policy, it selects *all* enqueued jobs and applies the fastest strategy that satisfies the accuracy SLO. (d) The modality-agnostic policy (`none`) performs modality modification, and serves as the baseline.

Results and discussion. Figure 5 compares MOSEL, with dynamic modality selection. The throughput across all models is higher than the modality-agnostic method. TVLT, AvHUBERT, TBN, MMSA, and ViLT on average achieved 5.3× 2.2×, 3.1×, 1.12×, and 4.3× higher throughput, respectively. When the request arrival rate is low, both the modality-aware and modality-agnostic approaches have similar throughput. On the other hand, the modality-aware methods can handle higher request arrival rates, while the modality-agnostic method suffers from high processing latency and fluctuation. This is due to the high processing latency of the model. MMSA has much lower processing latency across all modalities, leading to close performance among different modality

Task	Dataset	Model	Modalities	Fusion
Sentiment Analysis	MOSEI [77]	TVLT [65]	audio, video	Early
Speech Recognition	LRS3 [5]	AVHuBERT [62]	audio, video	Early
Action Recognition	EPIC-KITECHENS [18]	TBN [32]	audio, video, image	Late
Sentiment Analysis	MOSEI [77]	Self-MM [76]	text, audio, video	Late
Multi-Label Classification	MM-IMDb [48]	ViLT [33]	text, image	Early

Table 1: *Tasks, datasets used for finetuning and evaluation, model architectures, model sizes, modalities used, fusion strategy*

strategies.

Figure 5 also shows that all modality-aware techniques achieve much fewer SLO violations compared to the modality-agnostic approach. The optimized policy has 25%, 18%, 17%, 15%, and 4% lower average SLO violation ratio compared to the modality-agnostic approach for TVLT, VilT, TBN, AVHuBERT, MMSA, respectively. The optimized policy has in general a higher SLO violation ratio compared to the aggressive and random policy, due to the online optimizer overheads. It compensates for the slightly higher violation ratio by having higher average accuracy and more even accuracy distributions across jobs, as shown in Figure 6.

7.2 Complimenting Existing Approaches

Next, we show that MOSEL can be seamlessly incorporated into existing model optimization techniques to further improve inference throughput, addressing .

Experiment Setup. We use quantization to show how modality-aware techniques can be combined with other model optimization techniques to further reduce inference latency and satisfy SLO objectives. Quantization is a technique that reduces the precision of numerical values in a model, from high-precision data types to low-precision data types [45]. This technique can reduce memory footprint, as well as speed up the inference process. We use two data types for evaluation: `float32`, and `float16`. To study the effect under various loads, we select a range of QPS for each model. For modality selection, we use the `optimized` policy. The maximum QPS is set to the point where the deadline violation ratio reaches 99%.

Results and discussion. Figure 7 shows that quantization allows all models to handle higher QPS before the deadline violation ratio rises sharply. For example, when using only quantization, AVHuBERT fails to increase its processing capability; when both quantization and dynamic modality selection are used, AVHuBERT can process up to 7× more requests before reaching a 99% violation ratio. This shows our approach is compatible with existing model optimization approaches and can improve their performances.

7.3 MOSEL's decision overheads

In the offline profiling stage, MOSEL performs two tasks: (a) it measures the latency of different modalities under various batch sizes, and (b) it generates the optimal modality selection strategies for each batch size. Table 2 shows the median latency of these tasks and the speedup achieved by MOSEL over a brute-force search. Generating a single optimal modality offline selection strategy takes only 12ms.

Profile(s)	Optimize(s)	Speedup
32	45	31×

Table 2: The amoutnt of time TVLT spends in both system metrics profiling and modality generations, as well as speedup compared to brute force search for optimal modality generations.

In the online stage, MOSEL does two things: (a) it searches for the pre-computed optimal modality strategies that match the SLOs of each enqueued job, and (b) it finds the best modality selection strategy for each job. The optimizer's overhead varies from 12 ms to 80 ms. Note that this not on the critical path on job execution, as we overlap the optimization process with the job execution by having enough jobs enqueued by worker, as discussed in Section 6.

7.4 Ablation Study

In this section, we show how variations in both offline and online optimization phases can affect the inference process.

Experimental setup. To evaluate the impact of the offline optimization, we first generate optimized modality selection strategies, discussed in Figure 4.2. We then vary the latency from 20% to 250% of the original latency, to simulate the potential discrepancy between the estimated latency and the actual inference latency on real hardware. To evaluate the impact on accuracy, we also use TVLT with fixed QPS of 40. We use `optimized` strategy for all experiments.

During the online phase, MOSEL's overhead can vary significantly depending on how long it takes to finalize an optimized plan for all jobs in the queue. Therefore, we impose different time constraints on the online optimizer, ranging from 10 ms to 120 ms, to study how it affects the performance of MOSEL.

Results and discussion. As Figure 8 shows, both throughout and accuracy can tolerate errors in latency estimation. All models can tolerate underestimated latency and maintain throughput. TVLT, AVHuBERT, and MMSA and tolerate up to 50% latency overestimation with negligible sacrifice in throughput. Since it's rare to obverse such discrepancy in inference infrastructures [23], we believe MOSEL is robust

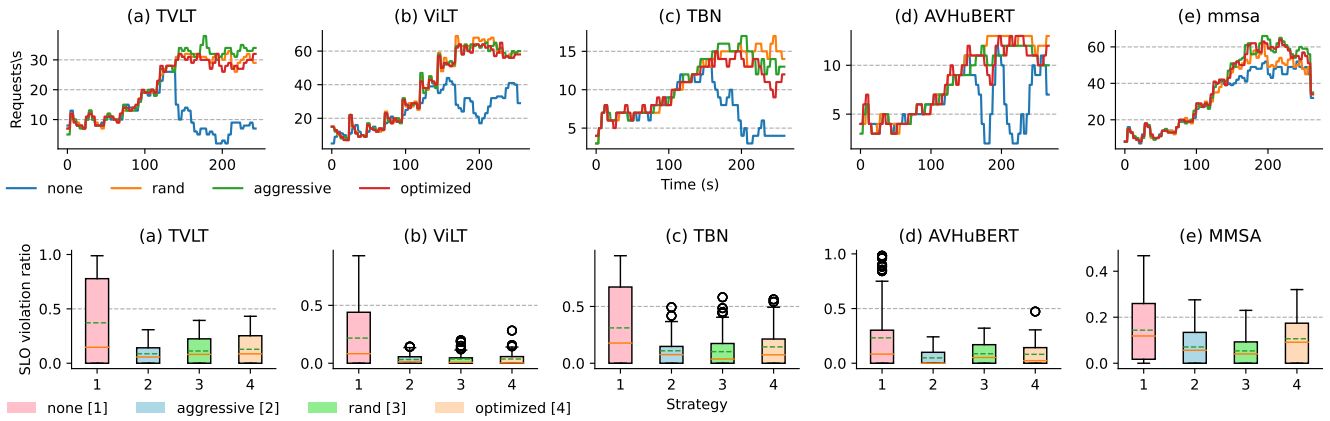

Figure 5: *Throughput and SLO violation ratio (number of SLO violations by total number of requests), profiled every 4 seconds. Each box shows the outlier, median, mean, 25%, and 75% quartiles.*

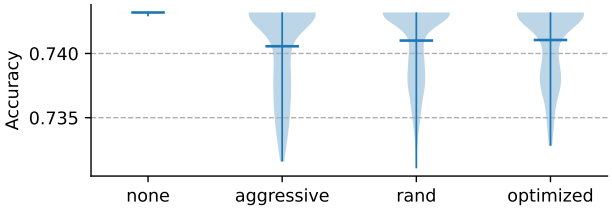

Figure 6: Accuracy distribution of TVLT with average accuracy of all jobs using different modality strategies.

against estimation errors in most scenarios. The changing accuracy, as shown in Figure 8, is attributed to the system having false impression of resources due to overestimation, thus dropping jobs prematurely.

8 Related Work

System-level dynamic optimization [17] proposes dynamic input batching to improve serving throughput by amortizing GPU kernel execution costs across multiple requests. It dynamically selects the largest profitable batch size that meets latency constraints.

Serving systems dynamically assign GPUs to jobs based on their SLOs and request rates. Some of them [14, 60, 75] consider GPU sharing to improve GPU utilization and goodput. [79] proposes burst-tolerant resource provisioning by mapping multiple jobs to a group of resources at runtime. [79] argues that preemption is necessary to maximize a serving system's goodput and their system makes preemption decisions at runtime providing formal guarantees on goodput.

[55] introduces a new dynamism layer, model-variants. A user specifies a task, accuracy, and latency requirements, and the proposed serving system automatically and dynamically explores the accuracy-latency tradeoff space of model-variants for the same task. [12] generates cost-effective LLM cascade execution plans, leveraging different cost-accuracy characteristics of different LLMs.

[35] focused on dealing with the delayed communication of input data in the case of multi-modal inference on streaming sensor data. Their proposed approach generates an input modality that is delayed based on the available input using a generative adversarial network (GAN) instead of waiting for the delayed input. They assume that dropping a modality always causes a significant accuracy drop.

Model-level optimization A number of ML compression techniques [13] including pruning [1, 70] and quantization [45] reduce both a model's memory and computational costs by reducing model weights or precision. They are usually applied before deployment, but recent work shows that runtime quantization bit-width decision is beneficial [38].

Early exiting [66, 71, 72, 82] adds task-specific layers (e.g. classification) to existing models, and stops inference early based on a given confidence level. [81] uses layer skipping and output verification for LLMs. It dynamically skips layers to reduce per-token inference time.

In mixture-of-experts (MoE) models [59], a model is partially activated during its forward pass. A gating network selects the expert networks that will be activated based on input. This architecture allows a model's parameters to scale while avoiding the prohibitive forward pass execution costs of a dense model with the same number of parameters.

Data multiplexing [44] adds multiplexing and demultiplexing layers at the beginning and end of the original model. The former transforms inputs into a succinct encoding and the latter does the opposite at the output. This improves throughput as the original model only runs on the more succinct encoding space. This technique is complementary to our approach that *drops* portions of the input data.

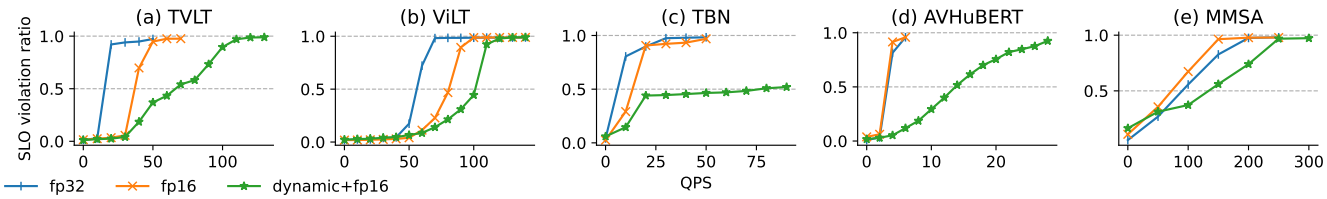

Figure 7: *SLO violation ratio using FP32, FP16, and dynamic modality selection combined with FP16.*

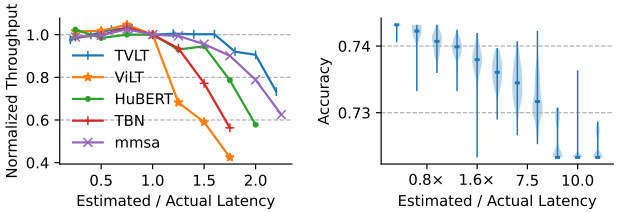

Figure 8: *Left: demonstrating the effect of the deviation between the expected and actual execution latency on models' normalized throughput. The discrepancy is calculated by estimated latency over actual latency. Right: accuracy distribution under different discrepancy between estimated and actual execution latency for TVLT. The discrepancy is calculated by estimated latency over actual latency.*

9 Conclusions

We present a new approach for dynamism, where we modulate the input to a model at inference time by selectively dropping portions of it. We should how the benefits of doing this in multi-modal inference. We highlight the key challenges that arise in leveraging this idea and present practical solutions to overcome them within our system, `MOSEL`.

We believe that input data modulation along with model optimization provides new possibilities for existing inference literature. The ability to modify the input data can lead to significant benefits across the entire inference serving stack, including reduced network bandwidth, lower preprocessing costs, energy efficiency, and reduced operating costs. We envision that `MOSEL` can be applied to a wide range of scenarios where input data variability is high and requires adaptive optimizations.

References

[1] Accelerating inference with sparsity using the nvidia ampere architecture and nvidia tensorrt. https://shorturl.at/wCHI3.

[2] Deliver high performance ml inference with aws inferentia. https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf.

[3] Twitter traces. https://archive.org/details/archiveteam-twitter-stream-2018-04, 2018.

[4] M. Abavisani, H. Joze, and V. M. Patel. Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1165–1174, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society.

[5] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Lrs3-ted: a large-scale dataset for visual speech recognition. *arXiv preprint arXiv:1809.00496*, 2018.

[6] Amazon Web Services. Amazon SageMaker. https://aws.amazon.com/sagemaker/.

[7] Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El Saddik, and M. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, 16:345–379, 2010.

[8] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[9] Logan D. R. Beal, Daniel C. Hill, R. Abraham Martin, and John D. Hedengren. Gekko optimization suite. *Processes*, 6(8), 2018.

[10] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018.

[11] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018.

[12] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *CoRR*, abs/2305.05176, 2023.

[13] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks, 2017.

[14] Seungbeom Choi, Sunho Lee, Yeonjae Kim, Jongse Park, Youngjin Kwon, and Jaehyuk Huh. Serving heterogeneous machine learning models on multi-gpu servers with spatio-temporal sharing. In Jiri Schindler and Noa Zilberman, editors, *2022 USENIX Annual Technical Conference, USENIX ATC 2022, Carlsbad, CA, USA, July 11-13, 2022*, pages 199–216. USENIX Association, 2022.

[15] Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey Zumar, Ion Stoica, Joseph Gonzalez, and Alexey Tumanov. Inferline: Latency-aware provisioning and scaling for prediction serving pipelines. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, SoCC '20, page 477–491, New York, NY, USA, 2020. Association for Computing Machinery.

[16] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. Clipper: A {Low-Latency} online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 613–627, 2017.

[17] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A Low-Latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 613–627, Boston, MA, March 2017. USENIX Association.

[18] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55, 2022.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[20] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient Sparse Training with Mixture-of-Experts. *Proceedings of Machine Learning and Systems*, 5, 2023.

[21] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. D3: a dynamic deadline-driven approach for building autonomous vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 453–471, 2022.

[22] Mitchell Gordon, Kevin Duh, and Nicholas Andrews. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online, July 2020. Association for Computational Linguistics.

[23] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving dnns like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.

[24] Jashwant Raj Gunasekaran, Cyan Subhra Mishra, Prashanth Thinakaran, Bikash Sharma, Mahmut Taylan Kandemir, and Chita R Das. Cocktail: A multidimensional optimization for model serving in cloud. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1041–1057, 2022.

[25] Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim Hazelwood, Mark Hempstead, Bill Jia, et al. The architectural implications of facebook's dnn-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 488–501. IEEE, 2020.

[26] David Harwath, Antonio Torralba, and James Glass. Unsupervised learning of spoken language with visual context. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[27] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[29] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost.

In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, 2018.

[30] H. Vaezi Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida. Mmtm: Multimodal transfer module for cnn fusion. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13286–13296, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[31] Aggelos K. Katsaggelos, Sara Bahaadini, and Rafael Molina. Audiovisual fusion: Challenges and new approaches. *Proceedings of the IEEE*, 103(9):1635–1653, 2015.

[32] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5492–5501, 2019.

[33] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

[34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[35] Tianxing Li, Jin Huang, Erik Risinger, and Deepak Ganesan. Low-latency speculative inference on distributed multi-modal data streams. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 67–80, 2021.

[36] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[37] Mengyuan Liu and Junsong Yuan. Recognizing human actions as the evolution of pose estimation maps. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1159–1168, 2018.

[38] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. Instance-aware dynamic neural network quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12424–12433. IEEE, 2022.

[39] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[40] Mengmeng Ma, Jian Ren, Long Zhao, Davide Testuggine, and Xi Peng. Are multimodal transformers robust to missing modality? In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18156–18165, 2022.

[41] Mengmeng Ma, Jian Ren, Long Zhao, Sergey Tulyakov, Cathy Wu, and Xi Peng. Smil: Multimodal learning with severely missing modality, 2021.

[42] Microsoft Azure. Azure Machine Learning. https://azure.microsoft.com/en-us/products/machine-learning.

[43] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3573–3582, 2019.

[44] Vishvak Murahari, Carlos E Jimenez, Runzhe Yang, and Karthik R Narasimhan. DataMUX: Data multiplexing for neural networks. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

[45] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[46] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14200–14213. Curran Associates, Inc., 2021.

[47] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[48] John Edison Arevalo Ovalle, Thamar Solorio, Manuel Montes-y-Gómez, and Fabio A. González. Gated multimodal units for information fusion. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[49] Arthi Padmanabhan, Neil Agarwal, Anand P. Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. GEMEL: model merging for memory-efficient, real-time video analytics at the edge. *CoRR*, abs/2201.07705, 2022.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[51] Juan-Manuel Perez-Rua, Valentin Vielzeuf, Stephane Pateux, Moez Baccouche, and Frederic Jurie. Mfas: Multimodal fusion architecture search. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6959–6968, 2019.

[52] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[53] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR, 17–23 Jul 2022.

[54] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459. IEEE, 2020.

[55] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411, 2021.

[56] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411, 2021.

[57] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[58] Jaime Sevilla, Pablo Villalobos, and Juan Cerón. Parameter counts in Machine Learning. https://www.lesswrong.com/posts/GzoWcYibWYwJva8aL/parameter-counts-in-machine-learning, 2021.

[59] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[60] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 322–337, New York, NY, USA, 2019. Association for Computing Machinery.

[61] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. Learning audio-visual speech representation by masked multimodal cluster prediction. In *International Conference on Learning Representations*, 2021.

[62] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. Learning audio-visual speech representation by masked multimodal cluster prediction. *arXiv preprint arXiv:2201.02184*, 2022.

[63] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, 2005.

[64] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[65] Zineng Tang, Jaemin Cho, Yixin Nie, and Mohit Bansal. Tvlt: Textless vision-language transformer. *Advances in Neural Information Processing Systems*, 35:9617–9632, 2022.

[66] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. *arXiv e-prints*, page arXiv:1709.01686, September 2017.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[68] Valentin Vielzeuf, Alexis Lechervy, Stéphane Pateux, and Frédéric Jurie. Centralnet: a multilayer approach for multimodal fusion, 2018.

[69] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[70] Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity, 2023.

[71] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online, July 2020. Association for Computational Linguistics.

[72] Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online, April 2021. Association for Computational Linguistics.

[73] Zihui Xue and Radu Marculescu. Dynamic multimodal fusion, 2023.

[74] Neeraja J Yadwadkar, Francisco Romero, Qian Li, and Christos Kozyrakis. A case for managed and model-less inference serving. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 184–191, 2019.

[75] Peifeng Yu and Mosharaf Chowdhury. Fine-grained gpu sharing primitives for deep learning applications. *Proceedings of Machine Learning and Systems*, 2:98–111, 2020.

[76] Wenmeng Yu, Hua Xu, Ziqi Yuan, and Jiele Wu. Learning modality-specific representations with self-supervised multi-task learning for multimodal sentiment analysis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10790–10797, 2021.

[77] AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246, 2018.

[78] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. MArk: Exploiting cloud services for Cost-Effective, SLO-Aware machine learning inference serving. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1049–1062, Renton, WA, July 2019. USENIX Association.

[79] Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. SHEPHERD: Serving DNNs in the wild. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 787–808, Boston, MA, April 2023. USENIX Association.

[80] Jeff Zhang, Sameh Elnikety, Shuayb Zarar, Atul Gupta, and Siddharth Garg. {Model-Switching}: Dealing with fluctuating workloads in {Machine-Learning-as-a-Service} systems. In *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*, 2020.

[81] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *CoRR*, abs/2309.08168, 2023.

[82] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020.