

MOSEL: Inference Serving Using Dynamic Modality Selection

Anonymous ACL submission

Abstract

Rapid advancements over the years have helped machine learning models reach previously hard-to-achieve goals, sometimes even exceeding human capabilities. However, achieving desired accuracy comes at the cost of larger model sizes and increased computational demands. Thus, serving predictions from these models to meet any 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. Our paper introduces a new 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 resource availability, as well as user-defined service level objectives (SLOs). MOSEL exploits modality configurations extensively, improving system throughput by $3.6\times$ with an accuracy guarantee and shortening job completion times by $11\times$ compared to modality-agnostic approaches.

1 Introduction

Recent advancements in Deep Learning has enabled Deep Neural Networks (DNNs), especially Transformers, to far exceed human capabilities in various Computer Vision and Natural Language Processing tasks (He et al., 2015; Wolf et al., 2019). However, the computational requirement of the largest machine learning (ML) models has doubled every few months, resulting in a $1,000,000\times$ increase from 2012 to 2020 (Sevilla et al., 2021). The increasing size of the models presents fundamental challenges in terms of latency and cost when they are commissioned for inference (Romero et al., 2021a; Gunasekaran et al., 2022; Gujarati et al., 2020).

These challenges has driven the development of

inference serving systems. These systems, hosted by cloud providers, deploy ML models to deliver fast and accurate responses to queries. Providers guarantee service level objectives (SLOs) for latency or accuracy while aiming to optimize hardware utilization and maximize throughput.

One approach to mitigate inference overheads and improve throughput is through *accuracy scaling*, which adapts model accuracy to varying query demands. An inference task involves three components: system, model, and input. Prior work focuses on optimizing the *system* and *model* aspects. System optimizations employ techniques such as batching (Ahmad et al., 2024; Choi et al., 2021; Crankshaw et al., 2017a; Shen et al., 2019a), sharing (LeMay et al., 2020; Romero et al., 2021a), and scheduling (Romero et al., 2021a; Ahmad et al., 2024; Crankshaw et al., 2017a). However, these techniques often require additional computational resources or powerful accelerators to handle higher query demands, which may not always be feasible due to the limited availability and flexibility of hardware resources. On the other hand, model optimizations often replace a large model with a more cost-effective variant, typically obtained using ML *compression techniques*, such as distillation (Sanh et al., 2019; Mullapudi et al., 2019), pruning (Lin et al., 2017; Gordon et al., 2020) and quantization (Polino et al., 2018). However, this approach necessitates multiple model replicas, wasting storage space and introducing overhead for switching between replicas and execution backends (Romero et al., 2021a; Ahmad et al., 2024).

In this paper, we propose an orthogonal and complementary perspective on accuracy scaling. In particular, we propose *modulating the input*, specifically via *selectively using* parts of it. We demonstrate its usefulness in the context of *multi-modal learning* (Ngiam et al., 2011; Baltrušaitis et al., 2018), an emerging and important class of ML techniques that combine data from different modal-

ities to provide prediction cooperatively, enhancing prediction accuracy.

As we describe in Section 3, we empirically find that some modalities (e.g., the audio modality in the Textless Vision-Language Transformer (TVLT) model (Tang et al., 2022)) contribute significantly to prediction accuracy without major resource use (e.g., memory) and processing time. In contrast, other modalities (e.g., the video modality in TVLT) consume significant resources and incur latency while only marginally improving accuracy.

We leverage the above insight in inference settings and propose that modalities be selectively enabled or disabled based on application requirements and workload patterns, creating novel opportunities to exploit the trade-off between speed and accuracy that multi-modality presents. We refer to it as *modality selection*, which complements existing accuracy scaling techniques and can be directly applied to the original model.

We assume that queries generally favor higher accuracy whenever resource permits, but can tolerate reduced accuracy for timely responses under resource constraints, provided that SLOs are not violated. This assumption is particularly relevant to applications like recommendation systems (Fang et al., 2018) or real-time applications, where response time outweighs the need for accuracy (Huang et al., 2015).

We build MOSEL, an automated inference serving system for multi-modal models that selects input modalities per request based on user-defined latency and accuracy SLO and system load. Our approach ensures scaling and performance during inference by dividing it into offline and online components. The offline component is designed to quickly generate a rich repository of modality selection strategies, enabling the online component to make informed decisions. For the online component, we ensure that, at inference time, late-enqueued jobs meet their latency requirements. We facilitate jobs ahead in the inference queue by dynamically re-selecting modalities to ease the queueing load; this allows later-enqueued jobs to run at the required accuracy without missing their latency targets.

We evaluate MOSEL on a set of representative multi-modal models that utilize commonly-seen architectures (Transformer (Vaswani et al., 2017), BERT (Devlin et al., 2019), CNN (LeCun et al., 2015)). We show that MOSEL outperforms modality-agnostic approaches in resource utilization and query spike tolerance, reducing job completion

times by up to $11\times$ and handling up to $3.6\times$ more requests with accuracy guarantees. Moreover, MOSEL achieves up to $4.6\times$ throughput when combined with quantization techniques.

2 Background

Multi-modal Learning: Multi-modal learning techniques are shown to surpass unimodal techniques by exploiting the complementary nature of different modalities, such as text, image, audio, and video (Ngiam et al., 2011; Baltrušaitis et al., 2018). The existing techniques can be broadly classified into two categories: early fusion (Snoek et al., 2005; Atrey et al., 2010; Katsaggelos et al., 2015) and late fusion (Snoek et al., 2005; Liu and Yuan, 2018; Abavisani et al., 2019). Early fusion combines modalities at an early stage, blending features before further processing, as seen in TVLT (Tang et al., 2022). Late fusion processes each modality separately and merges outcomes later, exemplified by the Temporal Binding Network (TBN) (Kazakos et al., 2019). Some methods attempt to combine properties from both early and late fusion (Nagrani et al., 2021; Joze et al., 2020; Perez-Rua et al., 2019; Vielzeuf et al., 2018; Xue and Marculescu, 2023; Nagrani et al., 2021). We demonstrate in Figure 1 that multi-modalities present complexities due to varied resource requirements and performance traits.

Inference and its challenges: Increased accuracy of DNNs has led to their wide adoption in real-world applications resulting in increased production costs (Hazelwood et al., 2018; Gupta et al., 2020; Romero et al., 2021b; aws). Inference serving systems use pre-trained ML models for predictions and manage resources to meet diverse user requests and application requirements (Crankshaw et al., 2017b; Reddi et al., 2020; Hsieh et al., 2018; Gog et al., 2022). Additionally, inference serving systems must manage dynamic workloads for cost and resource efficiency (Yadwadkar et al., 2019; Crankshaw et al., 2020, 2017b; Zhang et al., 2023a). The complexity increases when diverse services, each with unique Service Level Objectives (SLOs), contend for shared model resources. Comparing to serving systems that handle uni-modal models, serving multi-modal models with resource and latency-awareness has not been fully explored.

3 Opportunities & Challenges

Accuracy Across Modalities: In multi-modal DNNs, the importance of each modality can vary

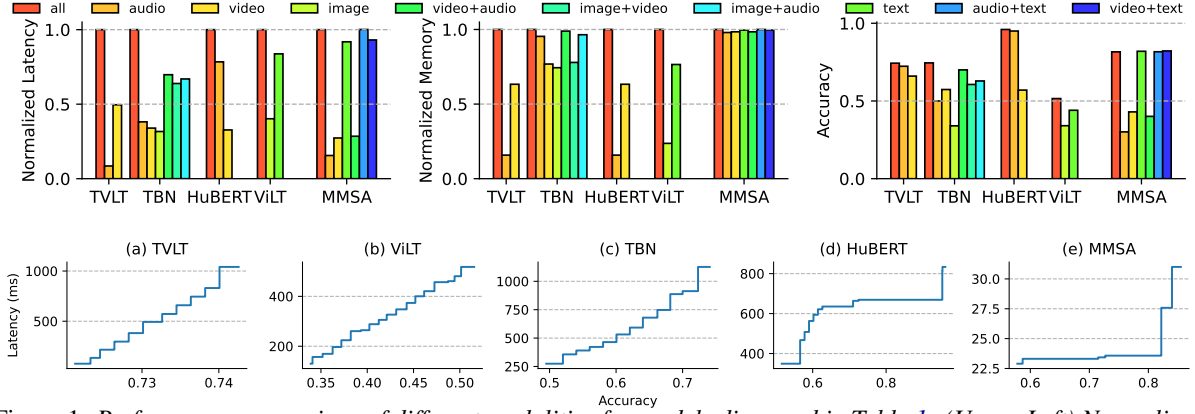


Figure 1: Performance comparison of different modalities for models discussed in Table 1: (Upper Left) Normalized latency for modalities, obtained by dividing each modality’s latency by the modality-agnostic baseline. (Upper Middle) The normalized memory footprint of different modalities. (Upper Right) Accuracy comparison using different modalities. (Bottom) Minimum latency required to achieve different levels of accuracy across various models using combinations of modalities.

based on task, data, and model architecture (Ma et al., 2021, 2022; Tang et al., 2022; Nagrani et al., 2021). Figure 1(left) illustrates that some models, like TVLT, can achieve high accuracy without using all modalities. This shows data of different modalities contribute differently to the model accuracy.

System Implications: Different modalities uniquely impact latency and memory consumption due to their distinct data representations and processing methods. For example, in TVLT, the audio modality is more efficient than video in memory usage and latency, with minimal accuracy trade-offs, shown by Figure 1 (left, middle). Memory consumption scales with sequence length in attention mechanisms (Vaswani et al., 2017), selectively using subset of input modalities means shorter sequences and reduced memory usage. Many recent works (Tang et al., 2022; Shi et al., 2021; Nagrani et al., 2021; Harwath et al., 2016; Lu et al., 2019; Sun et al., 2019) adopt similar attention-based multi-modal models, which can also benefit from using fewer modalities to reduce latency and memory consumption.

Opportunities: Applications provide inference systems with varying SLOs for accuracy and latency. These varying requirements offer opportunities for adaptive multi-modal selection, which previous systems haven’t explored. Modalities can be enabled or disabled based on application needs (e.g., serving latency) and resource availability. For instance, under high load, prioritizing ultra-low latency to prevent resource contention is crucial. In such scenarios, employing only the audio modality in TVLT helps reduce latency by $11\times$ with minimal accuracy loss, as shown in Figure 1. Conversely,

under low load, using both video and audio ensured highest accuracy due to resource availability. We refer to this method as *accuracy scaling* (Ahmad et al., 2024), which adapts the inference accuracy to meet varying query demands. Fully achieving accuracy scaling raises the following challenges.

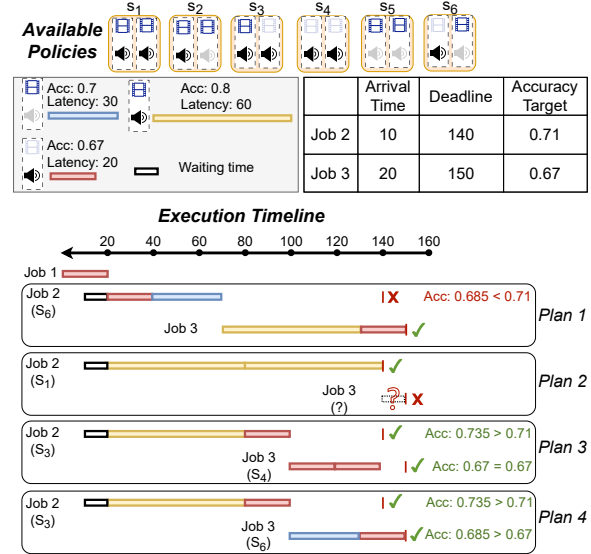


Figure 2: : Job 1 runs from timestamp 0 to 20. Job 2 arrives at timestamp 10 and starts at 20 after existing Job 1 finishes. One of Job 2’s strategies, s_6 , has an accuracy of $\frac{0.67+0.7}{2} = 0.685$, failing to meet its accuracy SLO (Plan 1). Similarly, s_4 also fails with an accuracy of 0.67. Job 3 arrives shortly after 20 with a deadline of 150. If Job 2 selects s_1 , it occupies the system until 140, leaving Job 3 unable to meet its deadline (Plan 2). By selecting a lower accuracy modality, Job 2 can free up resources for Job 3 (Plan 3) allowing Job 3 to use a higher accuracy video modality (Plan 4).

Challenge 1: find optimal modalities to use. Figure 2 illustrates the challenges of multi-modal inference. Each job, consisted of multiple requests

submitted by an application, has specific *accuracy* and *latency* SLOs. Job 1 with an audio modality runs from time 0 to 20, Job 2 arrives at 10 and starts at 20, and Job 3 arrives shortly after 20. All jobs are executed in the order they arrive (a First-In-First-Out, FIFO, manner). Each job requires a *modality selection strategy* to determine the modalities to use for each request. Figure 2 shows six possible strategies for Job 2 or Job 3. For example, S_1 uses both modalities for both requests, while S_4 uses only the audio modality.

The number of strategies can be large and grow exponentially with the number of requests and the number of modalities. For a job with 20 requests and 3 modalities, there are 231 possible strategies. Some strategies may be infeasible, failing to meet accuracy or latency SLOs. For instance, only two of the six strategies for Job 2 satisfy the accuracy SLO (0.71) and the latency SLO (140). To achieve faster model deployment, efficient methods are needed to prune infeasible strategies and estimate latency for feasible ones.

Challenge 2: handle resource contention. Figure 2 illustrates that multiple strategies can yield valid 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: it 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 improve 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

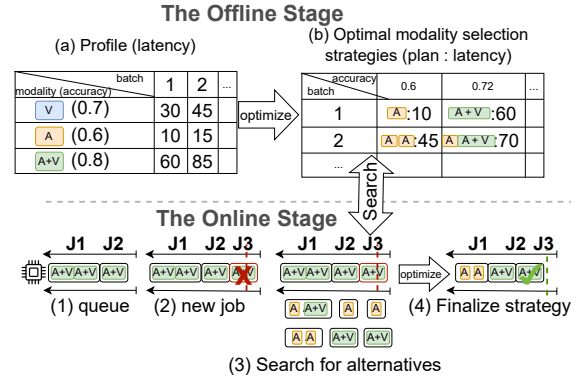


Figure 3: MOSEL During the offline phase, MOSEL first (a) profiles latency for different accuracy-batch size pairs, then (b) constructs the optimized modality selection strategy matrix based on profile. In the online phase, it uses this matrix to dynamically derive modality selection strategies for different jobs.

techniques that adapt to the changing SLOs and query load across jobs. Existing inference serving systems leverage various techniques, including autoscaling (Microsoft Azure; Amazon Web Services), model switching (Romero et al., 2021a; Zhang et al., 2020), batching (Crankshaw et al., 2017a), predictive serving (Gujarati et al., 2020) and preemption (Zhang et al., 2023a). Inference systems for multi-modality such as (Li et al., 2021) focus on speculatively executing modalities using augmented data. All of these techniques are agnostic to input data modalities and to the possibility of exploiting them for efficiency.

4 MOSEL Overview

4.1 Design Goals

We design MOSEL to achieve three key goals. First, MOSEL should automate modality selection, allowing users to only focus on high-level SLOs. Second, MOSEL should dynamically scale inference accuracy in response to varying system loads, maximize accuracy whenever possible while ensuring SLO compliance. Finally, MOSEL should easily integrate with existing inference systems for ease of use and adoption. This section provides an overview of our approach to meeting these goals and addressing the challenges outlined in Section 3.

Figure 3 illustrates the two stages of MOSEL’s approach: *offline profiling* and *online optimization*. The offline stage generates potential modality selection strategies, thereby preparing the system for varying operational scenarios. During the online stage, the system selects from these pre-computed strategies in real-time, adjusting modal-

ity choices to scale accuracy based on system load and SLOs of active jobs. This two-stage process ensures that MOSEL can minimize job deadline violations, enhance inference accuracy, and boost overall throughput - all without direct user intervention in modality selection.

4.2 The Offline Stage

MOSEL’s offline stage generates a repository of potential modality selection strategies. As discussed in Section 3, each job has specific SLOs for *accuracy* and *latency*. Moreover, the number of requests submitted by different users can vary, resulting in variations in batch sizes. Consequently, MOSEL must prepare diverse modality strategies to accommodate diverse request volumes and accuracy demanded by users.

However, exhaustively exploring every potential possible modality selection strategy to identify those that fulfill the specified criteria is not practical. Consider a model with three modalities and jobs sizes ranging from 1 to 64; this results in approximately 400,000 distinct strategies, taking up to 25 hours just for profiling. Moreover, increasing the jobs sizes and the number of modalities significantly escalates the complexity of the search space.

Therefore, we adopt an alternative approach: profiling *individual* modality combinations and leveraging the profiled data to *synthesize* optimized modality selection strategies. Taking the same model for example, we can form seven distinct combinations by selecting one, two, or three modalities. This requires profiling only $448 = 64 \times 7$ instances — a dramatic reduction from the exhaustive method. This approach decreases the profiling workload by a factor of $890\times$, making it significantly more efficient. We profile each instance for latency, documenting the accuracy and batch size. The latency for each batch-modality pair is stored as an entry in a profile table (Figure 3(a)).

MOSEL uses the profile table to construct a matrix of modality selection strategies for different job sizes and accuracy constraints for a given model, as shown in Figure 3(b). MOSEL first defines a range of possible job sizes, as well as a range of potential accuracy SLOs. Then, it uses an integer non-linear program (INLP) solver to generate an optimal strategy with minimal latency for a given accuracy-job size pair, represented by an entry in Figure 3(b). The construction happens in a one-time offline phase before the model is deployed.

More details can be found in Section 5.1.

4.3 The Online Stage

MOSEL’s online stage dynamically selects modality selection strategies, generated during offline stage, for each job in real-time. As outlined in Section 3, MOSEL’s goal is to *scale* accuracy for all requests, **maximizing** it during low system load and **balancing** it against higher loads, ensuring compliance with user-defined accuracy and latency SLOs. This requires an *ongoing update* of modality strategies in response to the fluctuating system load and the SLOs of active jobs.

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

MOSEL monitors the queued jobs and detects if incoming new jobs may suffer from deadline violations. For a given job, MOSEL calculates the total latency by adding the latency of the existing modality selection strategy used by the job and the total latency of all preceding jobs. It then checks whether the sum would exceed the given job’s latency SLO. If a job risks missing its deadline, as shown in Figure 3(2), MOSEL adjusts the modality selection strategies for all preceding jobs, potentially sacrificing accuracy, in order to reduce the wait time for the job at risk of a deadline violation.

When a job is detected to be at risk of a deadline violation, MOSEL considers the violator and all its preceding jobs as candidates for potential modality selection strategy changes. For each candidate job, MOSEL selects from the pre-computed modality strategies generated during the offline stage, whose accuracy are greater than the accuracy SLO specified for each job, as shown in Figure 3(3). MOSEL then takes all such strategies for all candidate jobs, and inputs them into an INLP solver, which reassigns a strategy for each candidate job, as shown in Figure 3(4). If the solver fails to find a solution, it means MOSEL is unable to reduce the queue time further without violating the accuracy SLO, and it will drop the job at risk of a deadline violation.

This approach allows MOSEL to dynamically adjust modality selection strategies to accommodate varying system loads. If the queue becomes relatively empty or contains few jobs, MOSEL will attempt to increase the accuracy for all queued jobs by progressively trying higher-accuracy modality strategies for each queued job. The modality strat-

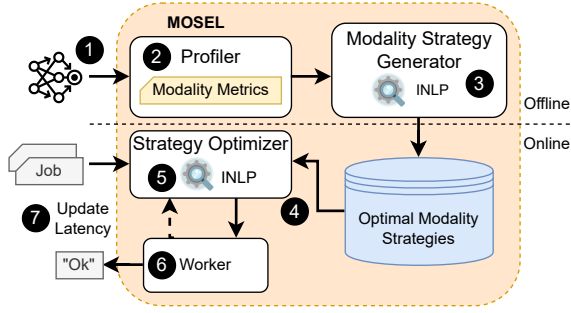


Figure 4: MOSEL Workflow

egy reassigment process is formulated as an INLP, detailed in Section 5.2.

5 Formulation

This section outlines how we identify optimal modality selection strategies to minimize latency while meeting accuracy SLOs for different job sizes. We also explain how we dynamically adjust these strategies in real-time to adapt to varying system resources.

5.1 Offline Optimal Strategy Generation

For a given model supporting n modalities and batch size from 1 to b , we profile the latency for each modality combination and batch size, yielding, in total, $b(2^n - 1)$ results, collectively denoted as the set \mathcal{D} (represented by Figure 3 (a)). \mathcal{D}_{ij} represents the latency using modality combination i with batch size j .

The submitted requests, with an accuracy SLO α , and the number of requests $|\mathcal{R}|$, are divided into multiple batches, denoted as \mathcal{J} , each using a different modality selection strategy from \mathcal{D} . For each batch, we aim to find the modality combination \mathcal{I} , such that the total latency of all batches $\sum_{i,j \in \mathcal{I}, \mathcal{J}} \mathcal{D}_{ij}$ is **minimized**, subject to two **constraints**: (1) The sum of all batch sizes must be equal to the total number of requests: $|\mathcal{R}| = \sum_{j \in \mathcal{J}} j$, and (2) The average accuracy of all batches must exceed the user-specified accuracy SLO α . We use $acc(i)$ to denote the accuracy achieved by a modality selection strategy i . Formally, we have: $\sum_{i,j \in \mathcal{I}, \mathcal{J}} acc(i)j \geq \alpha|\mathcal{R}|$

The INLP solver requires only three components to function: the profiled results \mathcal{D} , the request size \mathcal{R} , and the accuracy SLO α . This enables us to precompute the modality selection strategies completely offline, reducing the risk of deadline violations once a model is deployed. Formally, given N possible request sizes and A accuracy requirements, we optimize for each of the $N \cdot A$ combinations. The optimal strategies are denoted as \mathcal{P} , where \mathcal{P}_{ij}

represents the optimal strategy for a request of size i with accuracy SLO j (shown by Figure 3 (b)). This process has negligible overheads.

5.2 Online Modality Selections and Adjustment

Once a model is deployed, incoming requests are enqueued. To ensure no job misses its deadline (shown in Figure 3), we use T to represent the maximum allowed time budget. If MOSEL detects a deadline violation, T is set to the difference between the violator’s deadline and the start time of the most recently executed job. Otherwise, T is set to the difference between the last job’s deadline and the start time of the most recent job.

MOSEL selects one strategy from \mathcal{P} for each job in the queue, based on the available time budget T . We denote the set of all such strategies as \mathcal{S} , and the set of all involved jobs as J . Our goal is to select a strategy for each job in J such that the total accuracy is **maximized**: $\sum_{s,j \in \mathcal{S}, J} acc(s) \cdot |j|$.

We use $l(s)$ to represent the execution latency of a strategy. To ensure the total latency fits within the budget T , we add this **constraint** to our INLP: $\sum_{s \in \mathcal{S}} l(s) \leq T$.

6 MOSEL Implementation

MOSEL is implemented in 3k lines of Python code. The offline profiler uses Pytorch (Paszke et al., 2019) to execute ① DNNs on the GPU and profile ② system metrics through CUDA API. We use GEKKO (Beal et al., 2018) to generate ③ 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 strategies are stored in a single pickle object. During model deployment, the monitor process buffers incoming jobs, ④ retrieves the generated modality plans, ⑤ uses GEKKO to finalize the modality plan for each job, and puts the job into a FIFO queue shared with the worker process. To handle the GEKKO’s overhead (which takes up to 80 ms), the monitor process enqueues enough jobs to compensate for the optimizer overhead. The worker process polls the FIFO queue, executes ⑥ the jobs, and reports the latest ⑦ execution latency metrics back to the monitor process for accurate resource estimation.

7 Evaluation

To evaluate our implementation, we conduct experiments using realistic workloads and address the

507 following questions:

508 **Q1:** *What are the benefits of modality-aware opti-* 558
509 *mizations?* (Section 7.1) 559

510 **Q2:** *Is MOSEL resilient towards profiling error?* 560
511 *(Section 7.2)* 561

512 Unless specified otherwise, our experiments use the 562
513 following configurations. We explore MOSEL’s com- 563
514 patibility with existing model optimization tech- 564
515 niques in Appendix C. 565

516 **Experimental Setup** All measurements are con- 566
517 ducted on real hardware using a NVIDIA Tesla 567
518 A100 GPU (80GB DRAM) and an Intel Xeon Sil- 568
519 ver 4314 CPU (2.40GHz, 128GB DRAM). We used 569
520 NVIDIA driver version 525.85, CUDA 12.0, and 570
521 PyTorch 2.1.0. The operating system is Ubuntu 571
522 22.04.1 LTS with 5.15.0 kernel. 572

523 **Models.** Table 1 summarizes the five pretrained Py- 573
524 Torch models used for evaluation. The models dif- 574
525 fer in size and fusion strategy. All models are fine- 575
526 tuned on the task-specific datasets and preloaded 576
527 onto the GPU before evaluation. 577

528 **Workloads.** We conducted experiments using both 578
529 synthetic and real-world query patterns. For syn- 579
530 thetic workloads, we generated queries with con- 580
531 stant loads at fixed intervals. For real-world work- 581
532 loads, we used timing information from a month- 582
533 long 2018 Twitter trace (twi, 2018), which reflects 583
534 realistic inference workloads with diurnal patterns 584
535 and spikes (Zhang et al., 2019). For each exper- 585
536 iment, we randomly selected a day from the Twitter 586
537 trace. 587

538 7.1 MOSEL with production workload

539 Here we show that dynamic modality selection en- 591
540 ables MOSEL to improve throughput and utilization 592
541 while reducing SLO violations under heavy load. 593

542 **Experimental setup.** We evaluated various models 594
543 summarized in Table 1. To account for the vary- 595
544 ing processing latency, we adjusted the query per 596
545 second (QPS) for each model. The Twitter trace 597
546 was mapped to a minimum of 5 QPS. We set the 598
547 maximum QPS based on each model’s capacity to 599
548 process requests within one second without miss- 600
549 ing deadlines: TVLT (60), AVHuBERT (20), TBN 601
550 (40), MMSA (100), and ViLT (40). These values 602
551 are twice the maximum requests each model can 603
552 process per second. Requests were generated fol- 604
553 lowing a normal distribution, with a mean of 1 and 605
554 a standard deviation of 6, until the total number of 606
555 requests matches the QPS. We randomly assigned 607
556 each job an accuracy SLO within the model’s per- 608
557 formance range, based on the lowest and highest

achievable accuracy using different modalities.

We used four different policies: (a) optimized: (Section 5.2) uses available resources to achieve the highest accuracy for all enqueued jobs; (b) random (Algorithm 1) selects jobs randomly from the queue and applies the fastest strategy meeting the accuracy SLO, repeating until no deadline violations occur; (c) aggressive applies the fastest strategy satisfying the accuracy SLO to all enqueued jobs, **regardless** of deadline violations; and (d) none (modality-agnostic) performs **no** modality modification and serves as the baseline.

Results and discussion. Figure 5 shows dynamic modality selection results in higher throughput for all models compared to the modality-agnostic approach. TVLT, AvHUBERT, TBN, MMSA, and ViLT achieved throughput increase of $5.3\times$, $2.2\times$, $3.1\times$, $1.12\times$, and $4.3\times$, respectively. At low request arrival rate, both the modality-aware and modality-agnostic approaches have similar throughput. However, the modality-aware methods can handle higher request arrival rates, while the modality-agnostic method suffers from high processing latency and fluctuation. Note that MMSA has consistently low processing latency across all modalities, resulting in similar performance among different modality strategies.

Figure 5 also shows that all modality-aware techniques have significantly fewer SLO violations compared to the modality-agnostic approach. The optimized policy achieves 25%, 18%, 17%, 15%, and 4% lower average SLO violation ratios for TVLT, ViLT, TBN, AVHuBERT, MMSA, respectively. Note the optimized policy has a slightly higher SLO violation ratio compared to the aggressive and random policy for models like TVLT and MMSA, due to processing latency being close the online optimizer latency. MOSEL compensates for this with higher accuracy and more consistency accuracy distributions across jobs, as shown in Figure 6. For larger models, the online optimizer overhead is negligible.

600 7.2 Resilience to Variations

601 In this section, we show how variations in offline 602
603 and online optimizations can affect the inference 604
605 process. 606

607 **Experimental setup.** To evaluate the impact of the 608
609 offline optimization on accuracy and throughput, 610
611 we generate optimized modality selection strate- 612
613 gies, discussed in Section 4.2. We then vary the 614
615 latency from 20% to 250% of the original latency

Task	Dataset	Model	Modalities	Fusion
Sentiment Analysis	MOSEI (Zadeh et al., 2018)	TVLT (Tang et al., 2022)	audio, video	Early
Speech Recognition	LRS3 (Afouras et al., 2018)	AVHuBERT (Shi et al., 2022)	audio, video	Early
Action Recognition	EPIC-KITECHENS (Damen et al., 2022)	TBN (Kazakos et al., 2019)	audio, video, image	Late
Sentiment Analysis	MOSEI (Zadeh et al., 2018)	Self-MM (Yu et al., 2021)	text, audio, video	Late
Multi-Label Classification	MM-IMDb (Ovalle et al., 2017)	ViLT (Kim et al., 2021)	text, image	Early

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

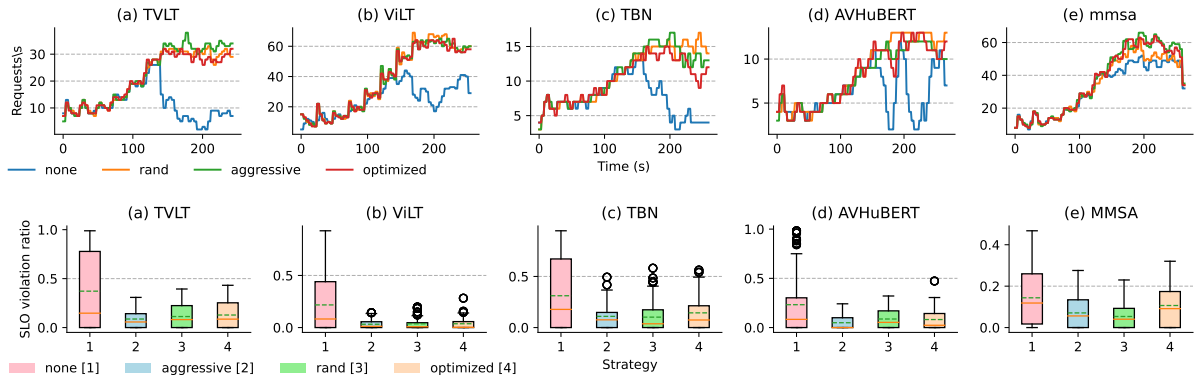


Figure 5: Throughput and SLO violation ratio (number of SLO violations by total number of requests), profiled every 4 seconds. Each box shows the outlier, median, mean, 25%, and 75% quartiles.

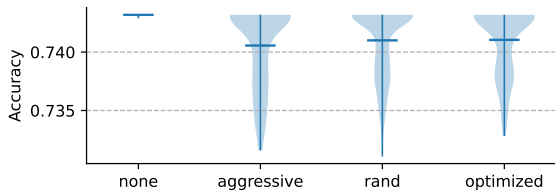


Figure 6: Accuracy distribution of TVLT with average accuracy of all jobs using different modality strategies.

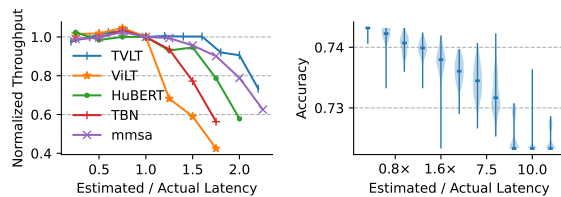


Figure 7: Left: demonstrating the effect of the deviation between the expected and actual execution latency on models' normalized throughput. The discrepancy is calculated by estimated latency over actual latency. Right: accuracy distribution under different discrepancy between estimated and actual execution latency for TVLT. The discrepancy is calculated by estimated latency over actual latency.

to simulate discrepancies between estimated and actual inference latency on real hardware. Using TVLT with a fixed QPS of 40, we apply optimized strategy for all experiments.

Results and discussion. As Figure 7 shows, 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 observe such discrepancy in inference infrastructures (Gujarati et al., 2020), we believe MOSEL is robust against estimation errors in most scenarios. The changing accuracy, as shown in Figure 7, is attributed to the system having false impression of resources due to overestimation, thus dropping jobs prematurely.

8 Conclusions

We modulate the input to a model at inference time to achieve accuracy scaling. We show the benefits of this approach in multi-modal inference. We highlight the key challenges and present practical solutions within MOSEL. We believe that input data modulation, combined with model and system optimization, opens new possibilities in inference literature. Modifying the input data can lead to significant benefits across the inference serving stack, including reduced network bandwidth, lower pre-processing costs, energy efficiency, and reduced operating costs. We envision MOSEL being applied to many scenarios with high input data variability that require adaptive optimizations.

640 Limitations

641 MOSEL presents two limitations in order to lever-
642 age the opportunity (§3) in a profitable way. First,
643 MOSEL only considers the strategies that select the
644 same modality for every request in a single job.
645 This may lead to sub-optimal decisions. For exam-
646 ple, Plan 4 in Figure 2 cannot be chosen. However,
647 this design choice is inevitable otherwise the of-
648 fline phase incurs prohibitive profiling costs. In the
649 online phase, MOSEL may adopt a greedy heuristic
650 that could be sub-optimal. We introduce it because
651 solving the optimization problem online imposes
652 a non-negligible latency overhead. We empirically
653 show that the proposed heuristic works well and is
654 close to the solver-based approach.

655 References

656 Accelerating inference with sparsity using the nvidia
657 ampere architecture and nvidia tensorrt. [https://](https://shorturl.at/wCHI3)
658 shorturl.at/wCHI3.

659 Deliver high performance ml inference with
660 aws inferentia. [https://d1.awsstatic.com/](https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf)
661 [events/reinvent/2019/REPEAT_1_Deliver_](https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf)
662 [high_performance_ML_inference_with_AWS_](https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf)
663 [Inferentia_CMP324-R1.pdf](https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf).

664 2018. Twitter traces. [https://](https://archive.org/details/archiveteam-twitter-stream-2018-04)
665 [archive.org/details/](https://archive.org/details/archiveteam-twitter-stream-2018-04)
666 [archiveteam-twitter-stream-2018-04](https://archive.org/details/archiveteam-twitter-stream-2018-04).

667 M. Abavisani, H. Joze, and V. M. Patel. 2019. Im-
668 proving the performance of unimodal dynamic hand-
669 gesture recognition with multimodal training. In
670 *2019 IEEE/CVF Conference on Computer Vision and*
671 *Pattern Recognition (CVPR)*, pages 1165–1174, Los
672 Alamitos, CA, USA. IEEE Computer Society.

673 Triantafyllos Afouras, Joon Son Chung, and An-
674 drew Zisserman. 2018. Lrs3-ted: a large-scale
675 dataset for visual speech recognition. *arXiv preprint*
676 *arXiv:1809.00496*.

677 Sohaib Ahmad, Hui Guan, Brian D. Friedman, Thomas
678 Williams, Ramesh K. Sitaraman, and Thomas Woo.
679 2024. *Proteus: A high-throughput inference-serving*
680 *system with accuracy scaling*. In *Proceedings of the*
681 *29th ACM International Conference on Architectural*
682 *Support for Programming Languages and Operat-*
683 *ing Systems, Volume 1, ASPLOS '24*, page 318–334,
684 New York, NY, USA. Association for Computing
685 Machinery.

686 Amazon Web Services. Amazon SageMaker. [https://](https://aws.amazon.com/sagemaker/)
687 aws.amazon.com/sagemaker/.

688 Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El
689 Saddik, and M. Kankanhalli. 2010. *Multimodal fu-*
690 *sion for multimedia analysis: a survey*. *Multimedia*
691 *Systems*, 16:345–379.

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

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

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

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017.
A survey of model compression and acceleration for
deep neural networks. *arXiv preprint*.

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

Y. Choi, Y. Kim, and M. Rhu. 2021. *Lazy batching: An*
sla-aware batching system for cloud machine learn-
ing inference. In *2021 IEEE International Sympo-*
sium on High-Performance Computer Architecture
(HPCA), pages 493–506, Los Alamitos, CA, USA.
IEEE Computer Society.

Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey
Zumar, Ion Stoica, Joseph Gonzalez, and Alexey
Tumanov. 2020. *Inferline: Latency-aware provision-*
ing 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. Association for Computing Machinery.

Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J.
Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017a.
Clipper: A Low-Latency online prediction serv-
ing system. In *14th USENIX Symposium on Net-*
worked Systems Design and Implementation (NSDI
17), pages 613–627, Boston, MA. USENIX Associa-
tion.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and
Kristina Toutanova. 2019. *Bert: Pre-training of deep*

748	bidirectional transformers for language understanding . <i>Preprint</i> , arXiv:1810.04805.	806
749		807
750	Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision . <i>Proceedings of the 24th Annual International Conference on Mobile Computing and Networking</i> .	808
751		809
752		810
753		811
754		812
755	Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. 2022. D3: a dynamic deadline-driven approach for building autonomous vehicles. In <i>Proceedings of the Seventeenth European Conference on Computer Systems</i> , pages 453–471.	813
756		814
757		815
758		816
759		817
760		818
761	Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning . In <i>Proceedings of the 5th Workshop on Representation Learning for NLP</i> , pages 143–155. Online. Association for Computational Linguistics.	819
762		820
763		821
764		822
765		823
766		824
767	Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. 2020. Serving dnns like clockwork: Performance predictability from the bottom up . In <i>14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)</i> , pages 443–462. USENIX Association.	825
768		826
769		827
770		828
771		829
772		
773		
774	Jashwant Raj Gunasekaran, Cyan Subhra Mishra, Prashanth Thinakaran, Bikash Sharma, Mahmut Taylan Kandemir, and Chita R Das. 2022. Cocktail: A multidimensional optimization for model serving in cloud . In <i>19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)</i> , pages 1041–1057.	830
775		831
776		832
777		833
778		834
779		835
780		836
781	Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim Hazelwood, Mark Hempstead, Bill Jia, et al. 2020. The architectural implications of facebook’s dnn-based personalized recommendation. In <i>2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)</i> , pages 488–501. IEEE.	837
782		838
783		839
784		840
785		841
786		842
787		843
788		844
789	David Harwath, Antonio Torralba, and James Glass. 2016. Unsupervised learning of spoken language with visual context . In <i>Advances in Neural Information Processing Systems</i> , volume 29. Curran Associates, Inc.	845
790		846
791		847
792		848
793		849
794	Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. 2018. Applied machine learning at facebook: A data-center infrastructure perspective. In <i>2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)</i> , pages 620–629. IEEE.	850
795		851
796		852
797		853
798		854
799		855
800		
801	Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification . In <i>2015 IEEE International Conference on Computer Vision (ICCV)</i> , pages 1026–1034.	856
802		857
803		858
804		859
805		
	Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In <i>13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)</i> , pages 269–286.	
	Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. 2015. Tencentrec: Real-time stream recommendation in practice . In <i>Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD ’15</i> , page 227–238, New York, NY, USA. Association for Computing Machinery.	
	H. Vaezi Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida. 2020. Mmtm: Multimodal transfer module for cnn fusion . In <i>2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)</i> , pages 13286–13296, Los Alamitos, CA, USA. IEEE Computer Society.	
	Aggelos K. Katsaggelos, Sara Bahaadini, and Rafael Molina. 2015. Audiovisual fusion: Challenges and new approaches . <i>Proceedings of the IEEE</i> , 103(9):1635–1653.	
	Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. 2019. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In <i>Proceedings of the IEEE/CVF International Conference on Computer Vision</i> , pages 5492–5501.	
	Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In <i>International Conference on Machine Learning</i> , pages 5583–5594. PMLR.	
	Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. <i>nature</i> , 521(7553):436.	
	M. LeMay, S. Li, and T. Guo. 2020. Perseus: Characterizing performance and cost of multi-tenant serving for cnn models . In <i>2020 IEEE International Conference on Cloud Engineering (IC2E)</i> , pages 66–72, Los Alamitos, CA, USA. IEEE Computer Society.	
	Tianxing Li, Jin Huang, Erik Risinger, and Deepak Ganesan. 2021. Low-latency speculative inference on distributed multi-modal data streams. In <i>Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services</i> , pages 67–80.	
	Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime neural pruning. <i>Advances in neural information processing systems</i> , 30.	
	Mengyuan Liu and Junsong Yuan. 2018. Recognizing human actions as the evolution of pose estimation maps . In <i>2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition</i> , pages 1159–1168.	

860	Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. 2022. Instance-aware dynamic neural network quantization . In <i>IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022</i> , pages 12424–12433. IEEE.	915
861		916
862		917
863		918
864		919
865		920
866	Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks . Curran Associates Inc., Red Hook, NY, USA.	921
867		922
868		923
869		924
870	Mengmeng Ma, Jian Ren, Long Zhao, Davide Testuggine, and Xi Peng. 2022. Are multimodal transformers robust to missing modality? In <i>2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)</i> , pages 18156–18165.	925
871		
872		
873		
874		
875	Mengmeng Ma, Jian Ren, Long Zhao, Sergey Tulyakov, Cathy Wu, and Xi Peng. 2021. Smil: Multimodal learning with severely missing modality . <i>Preprint</i> , arXiv:2103.05677.	929
876		930
877		931
878		932
879	Microsoft Azure. Azure Machine Learning. https://azure.microsoft.com/en-us/products/machine-learning .	933
880		934
881		935
882	Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. 2019. Online model distillation for efficient video inference. In <i>Proceedings of the IEEE/CVF International Conference on Computer Vision</i> , pages 3573–3582.	936
883		937
884		938
885		939
886		940
887	Vishvak Murahari, Carlos E Jimenez, Runzhe Yang, and Karthik R Narasimhan. 2022. DataMUX: Data multiplexing for neural networks . In <i>Thirty-Sixth Conference on Neural Information Processing Systems</i> .	941
888		942
889		943
890		944
891		945
892	Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. <i>arXiv preprint arXiv:2106.08295</i> .	946
893		947
894		948
895		949
896	Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. 2021. Attention bottlenecks for multimodal fusion . In <i>Advances in Neural Information Processing Systems</i> , volume 34, pages 14200–14213. Curran Associates, Inc.	950
897		951
898		952
899		953
900		954
901	Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In <i>Proceedings of the 28th international conference on machine learning (ICML-11)</i> , pages 689–696.	955
902		956
903		957
904		958
905		959
906	John Edison Arevalo Ovalle, Tamar Solorio, Manuel Montes-y-Gómez, and Fabio A. González. 2017. Gated multimodal units for information fusion . In <i>5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings</i> . OpenReview.net.	960
907		961
908		
909		
910		
911		
912	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. 2019. Pytorch: An imperative style, high-performance deep learning library . <i>CoRR</i> , abs/1912.01703.	962
913		963
914		964
		965
		966
	Juan-Manuel Perez-Rua, Valentin Vielzeuf, Stephane Pateux, Moez Baccouche, and Frederic Jurie. 2019. Mfas: Multimodal fusion architecture search . In <i>2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)</i> , pages 6959–6968.	967
		968
		969
	Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. <i>arXiv preprint arXiv:1802.05668</i> .	
	Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. Mlperf inference benchmark. In <i>2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)</i> , pages 446–459. IEEE.	
	Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021a. {INFaaS}: Automated model-less inference serving. In <i>2021 USENIX Annual Technical Conference (USENIX ATC 21)</i> , pages 397–411.	
	Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021b. {INFaaS}: Automated model-less inference serving. In <i>2021 USENIX Annual Technical Conference (USENIX ATC 21)</i> , pages 397–411.	
	Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. <i>arXiv preprint arXiv:1910.01108</i> .	
	Jaime Sevilla, Pablo Villalobos, and Juan Cerón. 2021. Parameter counts in Machine Learning. https://www.lesswrong.com/posts/GzoWcYibWYwJva8aL/parameter-counts-in-machine-learning .	
	Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer . In <i>5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings</i> . OpenReview.net.	
	Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019a. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. <i>SOSP '19</i> , pages 322–337.	
	Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019b. Nexus: A gpu	

970	cluster engine for accelerating dnn-based video analysis. In <i>Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP '19</i> , page 322–337, New York, NY, USA. Association for Computing Machinery.	
971		
972		
973		
974		
975	Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. 2021. Learning audio-visual speech representation by masked multimodal cluster prediction. In <i>International Conference on Learning Representations</i> .	
976		
977		
978		
979		
980	Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. 2022. Learning audio-visual speech representation by masked multimodal cluster prediction. <i>arXiv preprint arXiv:2201.02184</i> .	
981		
982		
983		
984	Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. 2005. Early versus late fusion in semantic video analysis. In <i>Proceedings of the 13th annual ACM international conference on Multimedia</i> , pages 399–402.	
985		
986		
987		
988		
989	C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. 2019. Videobert: A joint model for video and language representation learning. In <i>2019 IEEE/CVF International Conference on Computer Vision (ICCV)</i> , pages 7463–7472, Los Alamitos, CA, USA. IEEE Computer Society.	
990		
991		
992		
993		
994		
995	Zineng Tang, Jaemin Cho, Yixin Nie, and Mohit Bansal. 2022. TvlT: Textless vision-language transformer. <i>Advances in Neural Information Processing Systems</i> , 35:9617–9632.	
996		
997		
998		
999	Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2017. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. <i>arXiv e-prints</i> , arXiv:1709.01686.	
1000		
1001		
1002		
1003	Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. <i>Advances in neural information processing systems</i> , 30.	
1004		
1005		
1006		
1007		
1008	Valentin Vielzeuf, Alexis Lechervy, Stéphane Pateux, and Frédéric Jurie. 2018. Centralnet: a multi-layer approach for multimodal fusion. <i>Preprint</i> , arXiv:1808.07275.	
1009		
1010		
1011		
1012	Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. <i>arXiv preprint arXiv:1910.03771</i> .	
1013		
1014		
1015		
1016		
1017		
1018	Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. 2023. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. <i>Preprint</i> , arXiv:2309.10285.	
1019		
1020		
1021		
1022		
1023		
	Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In <i>Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics</i> , pages 2246–2251, Online. Association for Computational Linguistics.	1024 1025 1026 1027 1028 1029
	Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In <i>Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume</i> , pages 91–104, Online. Association for Computational Linguistics.	1030 1031 1032 1033 1034 1035 1036
	Zihui Xue and Radu Marculescu. 2023. Dynamic multi-modal fusion. <i>Preprint</i> , arXiv:2204.00102.	1037 1038
	Neeraja J Yadwadkar, Francisco Romero, Qian Li, and Christos Kozyrakis. 2019. A case for managed and model-less inference serving. In <i>Proceedings of the Workshop on Hot Topics in Operating Systems</i> , pages 184–191.	1039 1040 1041 1042 1043
	Peifeng Yu and Mosharaf Chowdhury. 2020. Fine-grained gpu sharing primitives for deep learning applications. <i>Proceedings of Machine Learning and Systems</i> , 2:98–111.	1044 1045 1046 1047
	Wenmeng Yu, Hua Xu, Ziqi Yuan, and Jiele Wu. 2021. Learning modality-specific representations with self-supervised multi-task learning for multimodal sentiment analysis. In <i>Proceedings of the AAAI conference on artificial intelligence</i> , volume 35, pages 10790–10797.	1048 1049 1050 1051 1052 1053
	AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In <i>Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 2236–2246.	1054 1055 1056 1057 1058 1059 1060
	Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. 2019. Mark: Exploiting cloud services for Cost-Effective, SLO-Aware machine learning inference serving. In <i>2019 USENIX Annual Technical Conference (USENIX ATC 19)</i> , pages 1049–1062, Renton, WA. USENIX Association.	1061 1062 1063 1064 1065 1066
	Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. 2023a. SHEPHERD: Serving DNNs in the wild. In <i>20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)</i> , pages 787–808, Boston, MA. USENIX Association.	1067 1068 1069 1070 1071 1072
	Jeff Zhang, Sameh Elnikety, Shuayb Zarar, Atul Gupta, and Siddharth Garg. 2020. {Model-Switching}: Dealing with fluctuating workloads in {Machine-Learning-as-a-Service} systems. In <i>12th USENIX Workshop on Hot Topics in Cloud Computing (Hot-Cloud 20)</i> .	1073 1074 1075 1076 1077 1078

1079 Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen,
1080 Gang Chen, and Sharad Mehrotra. 2023b. [Draft
1081 & verify: Lossless large language model ac-
1082 celeration via self-speculative decoding](#). *CoRR*,
1083 abs/2309.08168.

1084 Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian
1085 McAuley, Ke Xu, and Furu Wei. 2020. [Bert loses
1086 patience: Fast and robust inference with early exit](#). In
1087 *Advances in Neural Information Processing Systems*,
1088 volume 33, pages 18330–18341. Curran Associates,
1089 Inc.

1090 A Greedy Heuristic

Algorithm 1 Random modality strategy selection

```
1: function RAND( $jobQ, \mathcal{P}$ )  
2:    $S \leftarrow \{\}$   
3:   if deadlineViolation( $jobQ$ ) then  
4:      $J \leftarrow \text{jobsBeforeViolator}(jobQ)$   
5:   else  
6:      $J \leftarrow jobQ$   
7:   end if  
8:    $deadline \leftarrow \text{overhead}(J) + currTime$   
9:   while  $deadline > violatorDeadline$  do  
10:     $j \leftarrow \text{randomJob}(J)$   
11:     $s \leftarrow \mathcal{P}_{|j|, accuracy(j)}$   
12:     $deadline \leftarrow \text{update}(deadline, s)$   
13:     $S.append(s)$   
14:   end while  
15:   return  $S$   
16: end function
```

1091 During the online stage (discussed in Sec-
1092 tion 5.2), the INLP solver may take up to 70 ms
1093 to assign modality selection strategies for each en-
1094 queued job, posing challenges for jobs with ex-
1095 tremely low latency SLOs, such as MMSA. To ad-
1096 dress this issue, we propose a greedy heuristic that
1097 adapts the accuracy of enqueued jobs by randomly
1098 applying the fasted modality selection strategies
1099 meeting the minimum SLO for jobs preceding the
1100 deadline violator. We repeat this process until the
1101 total queue wait time is within the violator’s dead-
1102 line. The steps are described in Algorithm 1. We
1103 present the evaluation setup and the performance
1104 of the random heuristic in Section 7.

1105 B Related Work

1106 **System-level dynamic optimization** (Crankshaw
1107 et al., 2017a) proposes dynamic input batching to
1108 improve serving throughput by amortizing GPU
1109 kernel execution costs across multiple requests. It
1110 dynamically selects the largest profitable batch size
1111 that meets latency constraints.

Serving systems dynamically assign GPUs to 1112
jobs based on their SLOs and request rates. Some 1113
of them (Shen et al., 2019b; Yu and Chowdhury, 1114
2020; Choi et al., 2022) consider GPU sharing to 1115
improve GPU utilization and goodput. (Zhang 1116
et al., 2023a) proposes burst-tolerant resource pro- 1117
visioning by mapping multiple jobs to a group of 1118
resources at runtime. (Zhang et al., 2023a) ar- 1119
gues that preemption is necessary to maximize a 1120
serving system’s goodput and their system makes 1121
preemption decisions at runtime providing formal 1122
guarantees on goodput. 1123

(Romero et al., 2021a) introduces a new dy- 1124
namism layer, model-variants. A user specifies a 1125
task, accuracy, and latency requirements, and the 1126
proposed serving system automatically and dynam- 1127
ically explores the accuracy-latency tradeoff space 1128
of model-variants for the same task. (Chen et al., 1129
2023) generates cost-effective LLM cascade ex- 1130
ecution plans, leveraging different cost-accuracy 1131
characteristics of different LLMs. 1132

(Li et al., 2021) focused on dealing with the 1133
delayed communication of input data in the case 1134
of multi-modal inference on streaming sensor data. 1135
Their proposed approach generates an input modal- 1136
ity that is delayed based on the available input using 1137
a generative adversarial network (GAN) instead of 1138
waiting for the delayed input. They assume that 1139
dropping a modality always causes a significant 1140
accuracy drop. 1141

Model-level optimization A number of ML com- 1142
pression techniques (Cheng et al., 2017) includ- 1143
ing pruning (Xia et al., 2023; sem) and quantiza- 1144
tion (Nagel et al., 2021) reduce both a model’s 1145
memory and computational costs by reducing 1146
model weights or precision. They are usually ap- 1147
plied before deployment, but recent work shows 1148
that runtime quantization bit-width decision is ben- 1149
eficial (Liu et al., 2022). 1150

Early exiting (Xin et al., 2020; Zhou et al., 2020; 1151
Teerapittayanon et al., 2017; Xin et al., 2021) adds 1152
task-specific layers (e.g. classification) to exist- 1153
ing models, and stops inference early based on a 1154
given confidence level. (Zhang et al., 2023b) uses 1155
layer skipping and output verification for LLMs. It 1156
dynamically skips layers to reduce per-token infer- 1157
ence time. 1158

In mixture-of-experts (MoE) models (Shazeer 1159
et al., 2017), a model is partially activated during 1160
its forward pass. A gating network selects the ex- 1161
pert networks that will be activated based on input. 1162
This architecture allows a model’s parameters to 1163

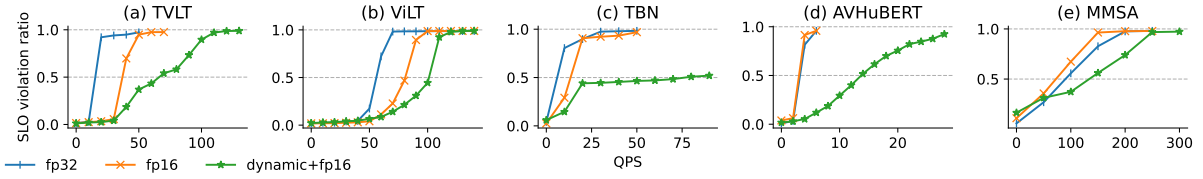


Figure 8: SLO violation ratio using FP32, FP16, and dynamic modality selection combined with FP16.

scale while avoiding the prohibitive forward pass execution costs of a dense model with the same number of parameters.

Data multiplexing (Murahari et al., 2022) 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.

C Complimenting Existing Approaches

We show that MOSEL can be seamlessly incorporated into existing model optimization techniques to further improve inference throughput.

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 SLOs. Quantization reduces the precision of numerical values in a model (Nagel et al., 2021), reducing memory footprint and speed up the inference process. We perform evaluation using two data types: float32, and float16. To study the effects under varying system loads, we select a range of QPS for each model. For modality selection, we use the optimized policy, employing the INLP solver during the online stage (as discussed in Section 5.2). The maximum QPS is set where the deadline violation ratio reaches 99%.

Results and discussion. Figure 8 shows that quantization allows all models to handle higher QPS before the deadline violation ratio increases significantly. For instance, AVHuBERT, when solely using quantization, fails to increase its processing throughput. However, with the combined use of quantization and dynamic modality selection, AVHuBERT can process up to 7× more requests before reaching a 99% violation ratio. This shows our approach is complimentary to existing model optimization techniques and can significantly improve inference processing throughput.

D 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 amount 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 is 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. For models with extremely low processing latency, we also propose a heuristic based method with lower latency, discussed in Appendix A.