Low-latency Speculative Inference On Distributed Multi-modal Data Streams

Tianxing Li[#,†], Jin Huang[†], Erik Risinger[†], and Deepak Ganesan[†]
[#]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI
[†]Department of Computer Science, UMass Amherst, Amherst, MA
litianx2@msu.edu,jinhuang@cs.umass.edu,erisinger@umass.edu,dganesan@cs.umass.edu

ABSTRACT

While multi-modal deep learning is useful in distributed sensing tasks like human tracking, activity recognition, and audio and video analysis, deploying state-of-the-art multi-modal models in a wirelessly networked sensor system poses unique challenges. The data sizes for different modalities can be highly asymmetric (e.g., video vs. audio), and these differences can lead to significant delays between streams in the presence of wireless dynamics. Therefore, a slow stream can significantly slow down a multi-modal inference system in the cloud, leading to either increased latency (when blocked by the slow stream) or degradation in inference accuracy (if inference proceeds without waiting). In this paper, we introduce *speculative inference* on multi-modal data streams to adapt to these asymmetries across modalities. Rather than blocking inference until all sensor streams have arrived and been temporally aligned, we impute any missing, corrupt, or partially-available sensor data, then generate a speculative inference using the learned models and imputed data. A rollback module looks at the class output of speculative inference and determines whether the class is sufficiently robust to incomplete data to accept the result; if not, we roll back the inference and update the model's output. We implement the system in three multi-modal application scenarios using public datasets. The experimental results show that our system achieves $7-128\times$ latency speedup with the same accuracy as six state-of-the-art methods.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

KEYWORDS

Edge computing, cloud computing, computation off-loading, deep neural networks

ACM Reference Format:
Tianxing Li[#,†], Jin Huang[†], Erik Risinger[†], and Deepak Ganesan[†]. 2021. Low-latency Speculative Inference On Distributed Multi-modal Data Streams. In *The 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21), June 24–July 2, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3458864.3467884

MobiSys '21, June 24–July 2, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8443-8/21/06…$15.00
https://doi.org/10.1145/3458864.3467884

1 INTRODUCTION

The proliferation of low-power yet data-rich sensors such as cameras, microphones, LIDAR, hyperspectral imagers, and RF imagers, has led to increased use of these sensors on IoT devices. There is also growing evidence that multi-modal inference is more effective than using a single modality in applications such as speech recognition [61], 3D scene understanding [56], health monitoring [27], safety [3], material identification [23], augmented reality [59], robotics [46], and self-driving vehicles [13].

These advantages are also leading to distributed deployments of multi-modal systems, particularly for monitoring complex environments where monitoring needs to be done around corners, through walls, in low light, under foliage, or in other occluded settings [16, 78]. The ability to observe a scene from different vantage points allows us to reconstruct various physical properties of a dynamic scene independently from multiple sensing modalities. This allows us to extract a rich set of parameters about the physical scene from diverse sensors — for example, the location, displacement, speed, direction and material properties of objects may be inferred from various imaging modalities. These parameters can be leveraged to independently cross-check information from alternate sensors from different vantage points, which is particularly important in complex environments. Many applications exhibit such complexity including city-scale real-time analytics about the urban environment, infrastructure, and activity [43, 64], industrial IoT settings for predictive control and fault detection [63], and healthcare settings where data from body-worn sensors (e.g., Fitbit) need to be fused with environmental sensors (e.g., Wi-Fi access points) for measuring the contextual environment around the user [76].

An overarching challenge in these systems is how to process the distributed multi-modal data from these sensors. Advanced sensors are capable of producing extremely large and continuous data streams, and it is impractical to stream all collected data to the cloud for data fusion. At the same time, it is important to fuse these signals and leverage advances in neural networks that have significantly improved in their ability to combine data from multiple modalities.

While a number of recent efforts have been exploring the problem of partitioned machine learning models across IoT devices and the cloud to deal with this issue (e.g. [50]), we face additional challenges in the context of distributed multi-modal inference since data generation rates can vary widely across these data-rich sensor modalities. For example, a video stream requires one or two orders

of magnitude more bandwidth than an audio stream, even though both are considered rich sensors. The underlying assumption in existing data analytic pipelines, however, is that data from individual edge devices will arrive at roughly the same time. But in cases where audio and video streams from distributed nodes need to be fused, audio data about an event is likely to arrive much earlier than video data due to the smaller data size.

We currently have limited options to deal with such a scenario. A cloud-based inference system can either block the inference pipeline and wait for all inputs to be available or ignore the delay and force output by running inference over a subset of the expected sensor modalities (video in the above example). However, the former introduces indefinite inference latency, whereas the latter sacrifices accuracy and robustness. Our experimental results show that a naive blocking method for audio-video data fusion can lead to tens of seconds of inference latency in event detection, which is not acceptable for many real-time applications. If the cloud ignores the delay and executes incomplete data, accuracy drops can be substantial (up to 35% in our experiments), leading to erroneous decisions and poor user experience.

Existing approaches do not provide a complete solution to the above problem. For example, some efforts have been made to find optimal resource-accuracy trade-offs to reduce latency in single-modality scenarios [29, 39, 84, 85]. These systems can find an optimal configuration by analyzing the impact of configurations/knobs (e.g., resolution) on resource consumption (e.g., bandwidth) and accuracy. However, these will all block inference when employed in a distributed multi-modal context — i.e., if one stream is much larger than the other or if wireless network bandwidth results in one data stream being more impacted than another, the slower stream blocks the entire inference pipeline. Frame sampling [85] and top-K candidate-object detection [35] minimize inference latency by reducing the bandwidth required to upload. However, accuracy suffers significantly with these approaches under poor network conditions. Retraining the deep learning model under varying network dynamics may achieve a better trade-off between accuracy and inference latency. However, such an approach often introduces high training overhead. Our experiment in Sec. 5 shows that it can take several hundred hours (10 times higher than our system) to retrain inference models to account for delays in the data stream.

In this paper, we introduce *speculative inference* on multi-modal data streams to adapt to constrained bandwidth and unpredictable network dynamics. Unlike prior approaches, our system does not modify the deep learning models used in existing multi-modal applications but enables speculative execution of these models without waiting for all inputs to be completely available. Instead of blocking the inference pipeline, our system leverages the fact that different modalities are complementary (e.g., they can provide additional information measured from the same object) and correlate temporally to impute missing, corrupt, or partially-available sensor data, and then generate a speculative inference using the imputed data. We show that most classes are robust to such speculative inference over lower-fidelity data, and only a few classes are more sensitive to such imputation. Thus, rollback is only needed for a small subset of classification categories and can be sufficiently infrequent that average latency can be dramatically improved without compromising inference accuracy. When the inference results are determined to be insufficiently accurate based on prior information, such as the sensitivity of different classification categories to data imputation and volume, our system rolls back the speculative inference and updates the model's output.

Instantiating our idea in practice requires that we address three challenges. First, it is difficult to fully reconstruct the missing data from one modality using only data from another modality due to differences in the underlying sensing principles and potentially discrepant dimensions of the data between modalities. Second, we need to design a computationally-efficient rollback module that can accurately detect when the speculative inference may be insufficiently accurate, which allows us to offer the best trade-off between accuracy and end-to-end latency. Third, we need to deal with the fact that IoT devices are often not well synchronized to a global clock, and timestamps generated on devices cannot be trusted due to clock drift. IoT devices have limited resources and are often duty-cycled; hence clock synchronization methods such as NTP are not ideal for these devices [52]. In addition, IoT devices are often equipped with low-cost clocks that can drift quickly and unpredictably, resulting in transient inaccuracy even if they are periodically re-synchronized.

Our system uses three core techniques to address these challenges. First, we use a conditional Generative Adversarial Network (GAN) model to reconstruct the intermediate features of the multi-modal model, which can be input directly to the multi-modal model to compute the speculative predictions. Second, to design a computationally-efficient rollback mechanism, we observe that many classification categories are more resilient to diminished data quality than others; thus, we only roll back speculative inferences for a subset of categories that are more sensitive to data quality considerations. Third, to align multi-modal data streams, we compute two sets of intermediate feature maps: one generated from a single modality only, and a series of intermediate feature maps generated from multiple modalities under a variety of delay configurations. We then align the data streams by minimizing the distance between the two sets of feature maps.

Contributions Our main contributions are as follows:

- We present a novel architecture for multi-modal models that enables speculative inference to adapt to fluctuating network bandwidth and asymmetry in data sizes between modalities.
- We propose techniques to effectively align data streams, impute the delayed data, compute speculative inference, and roll back when necessary.
- We implement the system in three multi-modal application scenarios using public datasets to examine the feasibility and performance of speculative inference on multi-modal data streams.
- Experimental results show that the system achieves $7-128\times$ latency speedup with the same accuracy as six state-of-the-art methods, enabling fast and accurate multi-modal learning under fluctuating network bandwidth.

2 THE CASE FOR SPECULATIVE INFERENCE

The notion of *speculative inference* is inspired by speculative execution in modern processors and OS kernels, which is highly effective

at improving performance when different modules operate at different speeds [45, 57]. For example, the CPU is much faster than external physical memory, often executing hundreds of clock cycles before a required value becomes available. During the waiting time, instead of idling, the CPU tries to predict the direction of control flow, checkpoint the calling process's state, and speculatively execute the program based on the prediction. If the prediction is correct, the checkpoint will be removed; otherwise, the CPU will revert the register state to the stored checkpoint and retry the execution.

We observe that the fast and slow data streams in a distributed multi-modal fusion model are analogous to the CPU and memory in computer systems. When data from a fast stream has been received in the cloud, the inference pipeline does not need to stall waiting for the slow stream's data to arrive. Instead, it can use the fast stream's data together with partially received data from the slow stream to impute the rest of the slow stream's data. Then, the cloud can speculatively execute the inference pipeline using the predicted values. As with speculative execution, the speculative inference engine needs to check if the prediction is statistically correct or not, and re-execute the inference if incorrect.

2.1 Challenges

There are several challenges to implementing the imputation-with-rollback scheme that underlies the speculative inference concept.

The first key challenge is in the design of the rollback method. Unlike traditional speculative execution, there is no straightforward way to confirm whether the speculative inference is correct or not. The naïve way to check whether the speculation is accurate is to re-execute the inference when more data has arrived, but this is clearly impractical. Thus, the core challenge with rollback is gauging the accuracy of speculative predictions without re-executing the entire inference pipeline and doing so in a computationally efficient manner.

A second challenge is the design of an imputation method that is broadly applicable across different sensor modalities with different data sizes, different inference tasks, and unpredictable delays arising from dynamic network conditions. In addition, the imputation method should also be lightweight, since the practical deployment of speculative inference may not only be in the deep cloud, but also on more resource-constrained edge clouds and mobile devices.

A third challenge is that the speculative inference method should not make assumptions about the availability of accurate and synchronized timestamps from different devices, and should operate effectively despite time-synchronization errors. IoT devices are generally not tightly synchronized with a global clock, and timestamps on such devices may not be accurate due to clock drift or sensor failure. Because IoT devices are often battery-powered, clock synchronization frequency is low, and many devices may not implement synchronization at all [58]. Thus, a robust inference pipeline must tolerate and deal with typical levels of synchronization inaccuracy.

2.2 Overview of Speculative Inference

We introduce *speculative inference*, which executes a multi-modal deep-learning pipeline without waiting for inputs from all modalities to be available. The system takes data from both fast and slow data streams as inputs, predicts features of the delayed data, speculatively executes the inference pipeline, and rolls back execution if the inference result is statistically likely to be wrong. Compared with state-of-the-art approaches [29, 35, 39, 84, 85], our method further reduces inference latency while maintaining good accuracy.

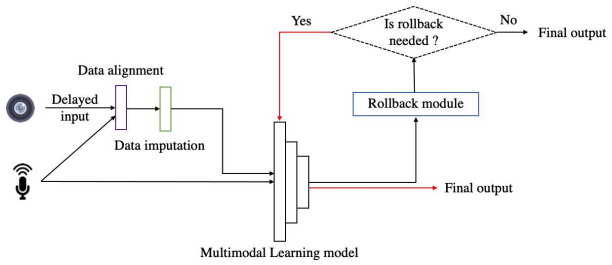


Figure 1: An overview of our system, comprising data imputation, rollback, and data alignment modules.

Figure 1 shows the main modules involved in speculative inference over a multi-stream, multi-modal data pipeline, including a data imputation module, a rollback module, and a data alignment module. The *data imputation* module leverages a conditional Generative Adversarial Network (GAN) to reconstruct features of any missing, corrupt, or partially-available sensor data. The *rollback module* provides a lightweight rollback mechanism that leverages the fact that many classification categories do not need data from all modalities to be present at high fidelity, and only a small number of categories are sensitive to the effects of speculative inference. The module, therefore, learns how robust the different classification categories are to the availability of data from different modalities, and triggers the rollback operation judiciously. The *data alignment module* aligns multi-modal data streams without using timestamps or injecting any synchronization pulses or frames, which is more general than state-of-the-art methods[1, 41, 86]. This module leverages a series of generative models to generate (a) intermediate features using the faster data stream (e.g., audio) only and (b) a series of intermediate features using both fast and slow data streams with different delay configurations. It aligns the two data streams by searching for the minimum $L1$ distance between (a) and (b).

Applications: Speculative inference can benefit data-driven applications that involve the fusion of multi-modal data under latency constraints. Wirelessly networked distributed smart camera and microphone systems are common in security, automation, biometric and environmental applications [22, 81]. Speculative inference is an effective way to fuse such multi-modal signals in real-time while being robust to bandwidth differences. Wireless health is another domain where speculative inference can be useful. For example, an adverse health event detection system might fuse information from a body-worn sensor like a smartwatch and a remote monitoring sensor at home to determine the likelihood that a serious event has occurred [76]. Real-time actuation can also benefit from these techniques. For example, robotic navigation or grasping in occluded conditions can leverage multiple modalities from different vantage points to complete difficult tasks in complex environments [24].

3 DESIGN

Speculative inference on multi-modal data streams consists of three core modules — the data imputation module, the rollback module,

and the data alignment module. Together, these modules optimize the accuracy of multi-model models while minimizing end-to-end latency under variable network dynamics.

We focus on the two-modality case (e.g., video and audio) because there is a dearth of distributed multi-modal datasets involving more than two modalities[8]. We expect that our methods will generalize to more modalities, but we leave this to future work when labeled datasets of such scenarios are more widely available. In this work, we assume that data from the two modalities is transmitted from two separate devices to the cloud using any common network protocol (e.g., TCP, UDP). We also assume that the IoT clocks are not always going to be time synchronized, either because time synchronization has been turned off to reduce resource usage or because the clock crystals used on these devices have a high drift-rate, resulting in transient loss of synchronization.

3.1 Data Imputation Module

We propose the use of a conditional GAN to predict the features of missing or delayed input data from one modality. A GAN trains two separate neural networks simultaneously: a generator G learns to map from a latent space to a data distribution of interest, and a discriminator learns to D distinguish between real samples and synthetic samples produced by the generator. To effectively train the generator, the discriminator provides feedback to the generator about the quality of the synthetic samples, and the generator imputes better samples to try to fool the discriminator. During training, the generator and the discriminator play a competitive game, resulting in synthetic samples that are very close to real samples. A vanilla generator usually takes a noise vector as input, but when conditioned generation is needed, the noise vector can be an explicit source [60]. The generator then leverages receptive fields to denoise and densify the input samples by referring to the generator's spatial or temporal contexts [51]. In our system, the generator leverages data from both fast and slow streams to learn and predict the features of one modality from the raw data of another.

Directly applying the GAN in our system does not work well due to the difference in data sizes and underlying sensing modalities. For example, a raw audio stream is a 2D vector and contains rich temporal information, whereas raw video is a 3D vector which contains rich spatial and temporal information. Also, the noise distributions of audio and video data are different due to inherent differences between the sensors and their respective noise sources. Therefore, training a GAN that is capable of generating realistic raw sensor data from partially received multi-modal sensor data is extremely challenging. Recent efforts have made some progress in generating images from audio signals [14, 74]. But the predicted images lose significant detail compared with the originals, which degrades performance in predictive applications.

To address this challenge, we propose to impute missing or incomplete data by generating *intermediate feature maps*, rather than attempting to generate realistic raw sensor data. This method is inspired by computer vision applications in which intermediate features from pre-trained deep neural networks are used to guide the generator in a GAN [9, 37, 75]. Figure 2 illustrates a typical audio-visual multi-modal model that has been widely used in the computer vision and deep learning communities [32, 68], which

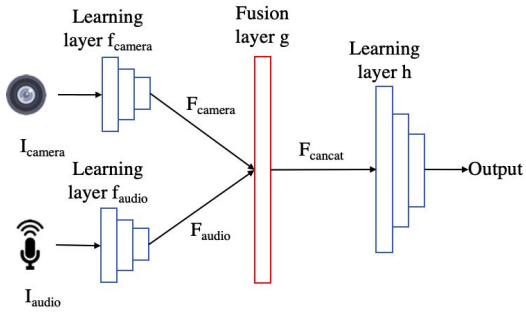


Figure 2: A typical audiovisual multi-modal model. The learning layers f_{camera} and f_{audio} take the raw sensing data as input and compute intermediate feature maps $\mathbf{F_{camera}}$ and $\mathbf{F_{audio}}$. Then the fusion layer g concatenates $\mathbf{F_{camera}}$ and $\mathbf{F_{audio}}$ to generate another intermediate feature map $\mathbf{F_{concat}}$. Finally, the learning layer h uses $\mathbf{F_{concat}}$ to compute the inference class.

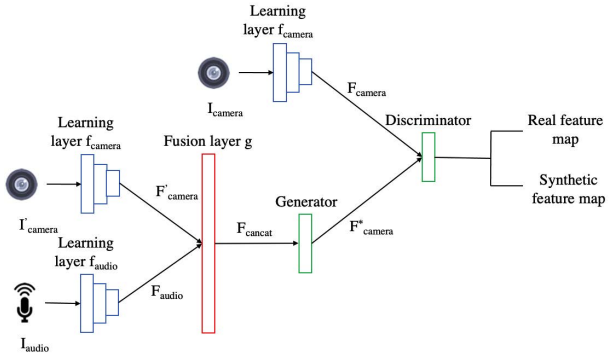


Figure 3: An overview of the data imputation module in our system. We first extract intermediate feature map $\mathbf{F_{camera}}$ from a pre-trained multi-modal model, which will be the discriminator's input. We then compute the concatenated feature map $\mathbf{F_{concat}}$ within the learning model as the generator's input.

takes raw sensor data from a camera and a microphone as inputs, and computes intermediate feature maps $\mathbf{F_{camera}}$ and $\mathbf{F_{audio}}$ via the feature extraction layers f_{camera} and f_{audio}, respectively. A fusion layer, g, concatenates $\mathbf{F_{camera}}$ and $\mathbf{F_{audio}}$ to generate an intermediate feature map, $\mathbf{F_{concat}}$. Finally, the prediction layer h uses $\mathbf{F_{concat}}$ to compute the final output, which can be derived by Equation 1.

$$\text{Output} = h(g(f_{camera}(\mathbf{I_{camera}}), f_{audio}(\mathbf{I_{audio}})) \quad (1)$$

Since the bandwidth requirement for audio is less than that of video, it is typically the faster of the two data streams to arrive for inference in the cloud. Thus, our system aims to impute the features of the slower video stream. We design our GAN models to leverage the features from audio and incomplete video data to generate synthetic feature maps of the full video stream. Specifically, we use audio features $\mathbf{F_{audio}}$ and features $\mathbf{F'_{camera}}$ from video data that is incomplete due to transmission delays to mimic the features $\mathbf{F_{camera}}$ of the complete video stream. Figure 3 shows an overview of our data imputation module.

We use the intermediate feature map $\mathbf{F_{camera}}$ prior to the fusion layer as the input to the discriminator because f_{camera} acts as a feature extraction and denoising module, so $\mathbf{F_{camera}}$ has a higher signal-to-noise ratio than the raw sensing data $\mathbf{I_{camera}}$. We leverage the feature map $\mathbf{F_{concat}}$ after the fusion layer g as the generator's

input to better utilize the correlated cross-modal information, because simply concatenating the intermediate feature maps $\mathbf{F_{camera}}$ and $\mathbf{F_{audio}}$ does not extract correlations between audio and visual content. We observe that many multi-modal models use Long Short Term Memory (LSTM) networks or Bidirectional Gated Recurrent Unit (BGRU) networks in the fusion layer, which jointly models temporal dynamics [32, 68].

Training of GAN models The objective function considers losses from two sources: a conditional GAN, $\mathcal{L}_{cGAN}$, and a perceptual loss, $\mathcal{L}_p$. We leverage a conditional GAN to model the conditional distribution of the real intermediate feature map $\mathbf{F_{camera}}$ from the complete video input $\mathbf{I_{camera}}$ given the intermediate feature map $\mathbf{F_{concat}}$, which is generated from the full audio data $\mathbf{I_{audio}}$ and partial video data $\mathbf{I'_{camera}}$. The conditional GAN loss can be expressed as:

$$\mathbf{F_{camera}} = f_{camera}(\mathbf{I_{camera}}) \tag{2}$$

$$\mathbf{F_{concat}} = g(f_{camera}(\mathbf{I'_{camera}}), f_{audio}(\mathbf{I_{audio}})) \tag{3}$$

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{\mathbf{F_{camera}}}[\log D(\mathbf{F_{camera}}, \mathbf{y})] + \mathbb{E}_{\mathbf{F_{concat}}}[\log(1 - D(G(\mathbf{F_{concat}}, \mathbf{y}))] \tag{4}$$

where $\mathbf{y}$ is the ground truth and the generator G tries to minimize this objective function against an adversarial discriminator, D, that tries to maximize it [38]. Then, we follow [40, 51] and introduce a perceptual loss, $\mathcal{L}_p$, in our objective function to compare high-level differences and stabilize GAN training:

$$\mathcal{L}_p(G) = \mathbb{E}[\sum_{n=1}^{N} ||V^{(n)}(G(\mathbf{F_{concat}})) - V^{(n)}(\mathbf{F_{camera}})||_1] \tag{5}$$

where V is a pre-trained VGG network that helps to quantify the perceptual differences of the content between our intermediate feature maps. In the VGG network, each layer n measures different levels of perception.

Our final objective, therefore, is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_p(G) \tag{6}$$

where λ is a hyper-parameter for regularization.

Offline Training We first extract intermediate feature map $\mathbf{F_{camera}}$ within the multi-modal model. $\mathbf{F_{camera}}$ will then be the discriminator's input. We next compute the concatenated feature map $\mathbf{F_{concat}}$ within the learning model as the generator's input. We modify the quality (e.g., resolution) and the quantity (e.g., delay time) of video data $\mathbf{I'_{camera}}$ and train a series of GANs ($G_{ijk}, i \in [1, n], j \in [1, r], k \in [1, d]$) to adapt to fluctuating network bandwidths, where n, r, d are the number of frame rate settings, resolution settings, and delay configurations, respectively. The definition of the delay configuration $k \in [1, d]$ is the number of the video frames received when audio data is fully received. Therefore, the number of GANs is $n \times r \times d$. For example, for the speech-recognition model in Sec. 5.1, the number of GANs is $4 \times 2 \times 25 = 200$. The data sizes of the intermediate features that our models learn to reconstruct are significantly smaller than those of the raw data, resulting in significantly lower training overhead than the state of the art [28, 66]. Experimental results in Sec. 5.6 show that our system reduces the training overhead by 80-90%, while still retaining good accuracy. For each resolution setting $j \in [1, r]$, we use an Image Expansion to transform the input resolution to a model-specific resolution (e.g. 256×256 for speech recognition, 224×224 for event detection); for each delay configuration $k \in [1, d]$, we pad the input with empty frames to reach the full frame count per sample (e.g. 29 frames for speech recognition, 300 frames for event detection).

Our system does not require the retraining of models pre-trained for individual inference tasks. Rather, it utilizes pre-trained models by replacing $\mathbf{F_{camera}}$ with $\mathbf{F^*_{camera}}$, which is the output of the generator G. As a result, our system can also be used to impute other sensing modalities (e.g., accelerometer) by retraining the set of GANs, which only takes 8–20 hours while retaining high accuracy.

Online Inference As sensing data arrives in the cloud, our system first uses the data alignment module in Sec 3.3 to align the data streams. Then it uses the corresponding GAN $G_{i^*j^*k^*}$ to compute a synthetic feature map $\mathbf{F^*_{camera}}$ via Eq. 7. Since only a single GAN needs to be loaded into memory at one time, the resource overhead is very small (< 100 MB memory in the GPU). The speculative inference can be derived via Eq. 9.

$$\mathbf{F^*_{camera}} = G_{i^*j^*k^*}(g(f_{camera}(\mathbf{I'_{camera}}), f_{audio}(\mathbf{I_{audio}}))) \tag{7}$$

$$\mathbf{F_{audio}} = f_{audio}(\mathbf{I_{audio}}) \tag{8}$$

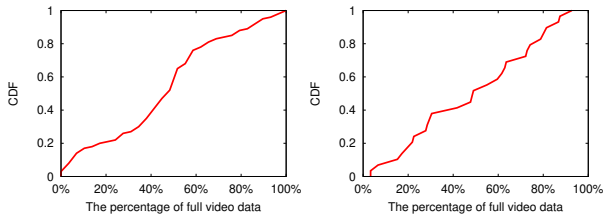
$$\text{Output} = h(g(\mathbf{F^*_{camera}}, \mathbf{F_{audio}})) \tag{9}$$

3.2 Rollback Module

So far, we have described how the system imputes incomplete data in the slow data stream and speculatively infers the class output without waiting for all inputs to be available. But what do we do if the speculative inference is insufficiently accurate to use in a particular application? The *rollback module* helps our system to achieve a critical balance between high accuracy and latencies incurred by discarding already-computed inferences. The key question is: how can the speculative inference module ensure that its output is suitable for downstream processing, and when should it decide to roll back and wait for more (or higher-quality) data? To ensure that latency is minimized, this decision must be made rapidly, without waiting for all inputs to arrive.

A simple observation inspires our design: some prediction classes are resilient to incomplete data in one sensor modality, whereas others are more sensitive to incomplete or lower-quality data. Resilience to data quality issues depends on both the duration and the complexity of a classifiable event. For example, based on the complexity of lip movement, an audiovisual lipreading model can recognize the word "under" with less visual data than is required for the word "allegations." Due to event duration, an audiovisual event detector can recognize "dog bark" with less visual data than that for "toilet flush." For activity recognition, some activities like sitting and standing are easier to infer with incomplete data than others, such as discriminating between modes of transportation. Therefore, the rollback module will green-light predictions for certain inference classes if the speculative inference is statistically likely to achieve the desired accuracy. For other inference classes, the rollback module will hold onto the inference and wait for additional data to verify the output.

To validate this observation, we conduct an experiment using two public datasets and multi-modal models in Sec. 5. In this experiment, we enforce various delay configurations on the validation data



(a) Speech recognition (b) Event detection

Figure 4: The CDF of the minimum percentage of full video data that achieves the best accuracy.

and measure the minimum data requirements to achieve the best accuracy. Figure 4 shows the CDF of the minimum ratio of full video data to achieve the highest accuracy. We make two observations:

- For about 50% the inference classes, the speculative inference can achieve the best accuracy with less than 50% of the full video data in both application scenarios. This validates our assumption that most classes are robust to the speculative inference over low fidelity data. For these inference classes, if the cloud has already received half of the total video frames in a prediction window (e.g., one second for speech recognition and ten seconds for event detection), the rollback module can release the lock and output the result of the speculative inference.
- For less than 10% of inference classes, the cloud needs to receive over 90% of total video frames to achieve the best accuracy in both application scenarios. For these inference classes, if the cloud receives less than 90% of the total video frames, it will discard the speculative inference and wait for the additional video frames. Once the cloud receives the additional data, the rollback module will re-compute and output the inference.

Our system considers latency as the main optimization objective and accuracy as the constraint. The objective of the rollback module is shown in Eq. 10:

$$\mathbf{I}^{*}_{\mathbf{camera}}, l^{*}, s^{*} = \underset{\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}, l, s}{\arg\min} Size(\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}})$$
$$s.t.\ Acc_{max} - Acc(G_{ijl}, \mathbf{I}_{\mathbf{audio}}, \mathbf{I}'_{\mathbf{camera}} + \mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}) <= x\% \quad (10)$$

where G_{ijl} is the GAN model, $\mathbf{I}_{\mathbf{audio}}$ is the audio data,and $\mathbf{I}'_{\mathbf{camera}}$ is the delayed video data; $\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}$ is additional video data acquired by the rollback module, and the total length and specifications are l and s. The rollback module aims to achieve the desired accuracy ($x\%$ less than the maximum accuracy Acc_{max}) with minimum data size $Size(\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}})$.

Alternatively, we can also optimize accuracy against a desired latency. The objective function is shown in Eq. 11:

$$\mathbf{I}^{*}_{\mathbf{camera}}, l^{*}, s^{*} = \underset{\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}, l, s}{\arg\min}(1 - Acc(G_{ijl}, \mathbf{I}_{\mathbf{audio}}, \mathbf{I}'_{\mathbf{camera}} + \mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}))$$
$$s.t.\ Size(\mathbf{I}^{\mathbf{r}}_{\mathbf{camera}}) <= y \quad (11)$$

where y is the maximum number of additional frames after the speculative inference.

Algorithm 1 shows the details of the rollback module using the objective function in Eq. 10. Once the cloud receives the audio data $\mathbf{I}_{\mathbf{audio}}$ and partial video data $\mathbf{I}'_{\mathbf{camera}}$, it counts the available video

Algorithm 1: Rollback module.

input : 1) A validation accuracy lookup table of C labels, n frame rates, r resolutions, d delay configurations: $AccTable[1:C, 1:n, 1:r, 1:d]$;
2) Data imputation modules: $G = \{G_{ijk} | i \in [1,n], j \in [1,r], k \in [1,d]\}$;
3) The number of available video frames: m;
4) Audio data: $\mathbf{I}_{\mathbf{audio}}$;
5) Partial Video data: $\mathbf{I}'_{\mathbf{camera}}$;

output : Inference class c

```
c ← ∅;
// Get multiple inference classes using different
   delay configurations
for k ← 1 to m do
    c.append(get_inference_class(G_ijk, I_audio, I'_camera,k));
end
c = voting(c);
// Check if the speculative inference achieves the
   desired accuracy. When x=0, the desired accuracy is
   the best accuracy of the corresponding inference
   class.
if max(AccTable[c, i, j, 1:N]) − AccTable[c, i, j, m] <= x%
 then
    output c;
end
else
    // Search for the next speculative inference that
       achieves the desired accuracy with minimum data
       size.
    [I*_camera, l*] = get_more_frames(AccTable, c, i, j, m)
    // Re-compute the speculative inference and output
       the inference classes.
    c = get_inference_class(G_ijl*, I_audio, I'_camera + I*_camera)
    output c;
end
```

frames m in the current time window. The module then uses the audio data $\mathbf{I}_{\mathbf{audio}}$, m image frames, and m generators $G_{ijk}(k \in [1,m])$, each of which uses the intermediate result $\mathbf{F}_{\mathbf{concat,k}}$ generated from the audio data and the video data $\mathbf{I}'_{\mathbf{camera,k}}$ from k frames. We compute m inference classes and generate the speculative inference via voting. For this we utilize an accuracy table $AccTable$, evaluated on validation data, to check if the speculative inference achieves the desired accuracy. The inference result will be accepted if the accuracy of the speculations on which that output depends has historically been above the desired accuracy or if the bandwidth is sufficient. Otherwise, based on the current network bandwidth, the cloud uses $AccTable$ to search for the next spatial configuration and frame count that will achieve the desired accuracy while minimizing the size of transferred data. Finally, the cloud rolls back the inference, waits for additional data, and updates the model's inference result when sufficient data is available.

The rollback module assumes that the distribution of the testing data is similar to the distribution of the validation data. However, we observe that if the test and validation data are limited (e.g., less than 5% of data is available to train $AccTable$), the rollback module tends to be erroneous. We also observe that the distribution of input data may vary over time. For these scenarios, the rollback

module can periodically update *AccTable* in the cloud when as data becomes available.

3.3 Data Alignment Module

This module aims to align multi-modal data streams without using timestamps or injecting synchronization pulses, as the former can be incorrect due to sensor failure or clock drift [83] and the latter may not be available under certain scenarios (e.g., low light or noisy environments). The key idea is to re-use the data imputation module in Sec. 3.1 to generate (a) a synthetic feature map $\mathbf{F}^*_{\mathbf{camera,0}}$ generated by audio-only data, and (b) a series of synthetic feature maps $\mathbf{F}^*_{\mathbf{camera,k}}$ generated by both audio and video data with multiple delay configurations k. $\mathbf{F}^*_{\mathbf{camera,0}}$ and $\mathbf{F}^*_{\mathbf{camera,k}}$ can be derived by Eq. 12 and 13.

$$\mathbf{F}^*_{\mathbf{camera,0}} = G(g(f_{audio}(\mathbf{I_{audio}}))) \tag{12}$$

$$\mathbf{F}^*_{\mathbf{camera,k}} = G_k(g(f_{camera}(\mathbf{I'_{camera,k}}), f_{audio}(\mathbf{I_{audio}}))) \tag{13}$$

We observe that when the delay configuration $k(k \in [1, d])$ matches the corresponding generator G_k, the output is close to $\mathbf{F}^*_{\mathbf{camera,0}}$. Specifically, when the delay configuration k matches the corresponding generator G_k, the corresponding synthetic feature map $\mathbf{F}^*_{\mathbf{camera,k}}$ should be close to the real feature map $\mathbf{F_{camera}}$. In this case, the distance between $\mathbf{F}^*_{\mathbf{camera,0}}$ and $\mathbf{F}^*_{\mathbf{camera,k}}$ is at a minimum. Therefore, we search for the minimum $L1$ loss of the two feature maps $\mathbf{F}^*_{\mathbf{camera,0}}$ and $\mathbf{F}^*_{\mathbf{camera,k}}$ among all possible delay configurations $k \in [1, d]$ to find the best alignment k^* and align multi-modal data streams, which can be derived in Eq. 14. We use this observation to design a new method for aligning multi-modal streams without using timestamps.

$$k^* = \underset{k \in [1,d]}{\arg\min} ||\mathbf{F}^*_{\mathbf{camera,0}} - \mathbf{F}^*_{\mathbf{camera,k}}||_1 \tag{14}$$

Figure 5 shows some examples of data alignment in our system. For all examples, the L1 loss is minimized when the delay configuration matches the corresponding generator. We also observe that all the curves are concave downward. When the two data streams are not well aligned, the intermediate feature map tends to be erroneous, which increases the L1 distance. And the error increases with more misalignment between the two streams, leading to a larger L1 distance. To further improve the efficiency of data alignment, we apply a binary search algorithm to find the minimum L1 loss. Each alignment is an independent operation, so the alignment error will not accumulate. Our module assumes that the delay between two streams is less than a full window, e.g., the system needs to receive at least one frame from both streams within the same inference window. If the delay is larger than a full window, the system will wait until it receives one frame from the slow stream or discards the alignment task after the timeout.

One-time computational cost Executing a GAN for each possible delay configuration has a high cost, but we note that this cost is only incurred the first time that streams need to be fused. Once the synchronization offset has been determined, only minor adjustments are needed to mitigate the effects of clock drift over time. For example, a 15ppm clock crystal will drift by roughly 50 ms/hr, so infrequent re-synchronization can be done by looking for a few delay configurations around the current minima to obtain a new delay configuration.

4 IMPLEMENTATION

Overview We implement our system on a server utilizing two GTX-1080Ti GPUs, 32 GB RAM, and a Xeon E5-2620 v3 2.40 GHz CPU, which serves as a stand-in for the **cloud**. We use two Raspberry Pi 4Bs as **IoT devices**, one to send fast-stream data (e.g., audio) and the other to send slow-stream data (e.g., video). Execution of the GAN and inference models takes place on the cloud; the IoT devices are only used for data pre-processing and transmission. Thus, during the experiment we profile and benchmark the latency for GAN models and inference models on the cloud side.

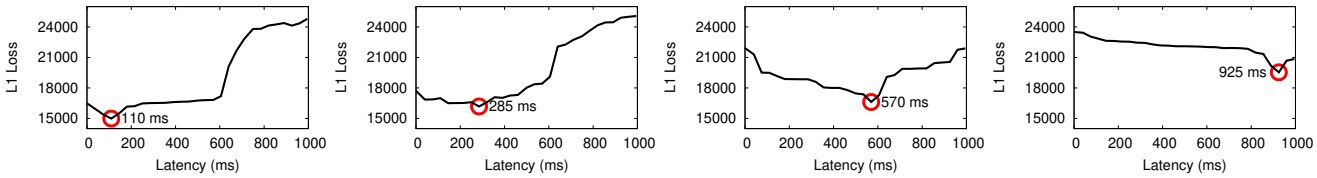
IoT devices In our setup, both of the IoT devices connect to the cloud via either Wi-Fi or Bluetooth Low Energy(BLE) for data and control-signal transmission. We use *Linux* command *wondershaper* to directly limit upload bandwidths based on real-world traces of network dynamics to run evaluations under different bandwidth settings. We emulate the video streaming process by buffering raw camera sensor data on the IoT devices and periodically extracting and transmitting a new camera frame.

Video Encoding and Decoding We use the standard *libav** library from *ffmpeg* for video encoding on IoT devices and decoding in the cloud. The rollback module requires additional, lower-resolution frames and a specific range of frames. The specification of the resolutions and number of frames is sent from the cloud and received by the IoT devices. After receiving the specifications, the IoT devices re-initialize the encoder codec *AVCodecContext* and discard all the encoded frame packets in the buffer. We maintain another buffer of raw input data so that the discarded encoded frame can be re-encoded at a lower resolution.

Test-bed Execution We duty-cycle the IoT nodes to better emulate a distributed sensor scenario to which speculative inference and rollback are well-suited. During cycle-off periods, the audio node sleeps, awaiting the execution of the rollback module if necessary. During cycle-on periods, both the audio and video nodes stream continuously to the cloud. The duration of the cycle-on period varies across tasks: the cycle-on period for speech recognition is 1 second, but for event detection, the active period is 10 seconds. Due to asymmetric data rate requirements, audio data is fully transmitted while only the partial video data is transmitted at the end of each active period. On the cloud side, we align the partial video data and full audio data to generate inference results, which are compared against a statistical accuracy threshold to determine whether to perform a rollback. Following the determination of the rollback module, the cloud sends the specifications and number of additional video frames back to the video node, then waits until the cloud receives the additional frames. Finally, we compute the final output based on both the initial and additional data.

5 EVALUATION

We start by describing the datasets we use and baselines we compare against, followed by a comprehensive evaluation.



(a) 110 ms delay (b) 285 ms delay (c) 570 ms delay (d) 925 ms delay

Figure 5: Examples of data alignment.

5.1 Datasets and setup

We look at three application scenarios — speech recognition, event detection, and activity recognition — that require high accuracy with minimal inference latency. We run our system on three multi-modal models using public datasets [17, 67, 68]. As described in Sec. 2.2, we assume that sensor data for each modality is streamed continuously from a separate device to the cloud during active, cycle-on periods. We then describe the four baselines — naïve blocking, adaptive bitrate streaming, one-time profiling, and periodic profiling — that we compare against.

Speech Recognition Dataset We use a large Lip Reading in the Wild (LRW) dataset [17] consisting of 500 English word classes [5, 18]. Many words are either visually or verbally similar, making the words harder to recognize by a single modality only. The dataset consists of over 1,000 individual short video clips (1.16 seconds) from BBC news and talk shows for each word class; hundreds of different speakers are present in the videos. The LRW clips have been cropped to extract speakers' faces only. The native resolution of the original videos is 1080p or higher, with the resolution of the LRW video cropped down to 256×256. The video frame rate is 25 FPS. The audio sampling rate is 44.1 kHz.

Event Detection Dataset We use the Audio-Visual Event (AVE) dataset, containing over 4,000 10-second videos covering a wide range of 28 audio-visual events (e.g., church bell ringing, dog barking) and domains (e.g., human activities, musical performances) [68]. There are at least 60 videos for each event category, and over half of the videos contain audio-visual events that span the full ten seconds. Each video contains ten labels (one label per second). The native resolution of the original video varies from 360p to 720p, and the authors resized the image resolution to 224×224 to extract visual features. The video frame rate is 30 FPS. The audio sampling rate is 44.1 kHz.

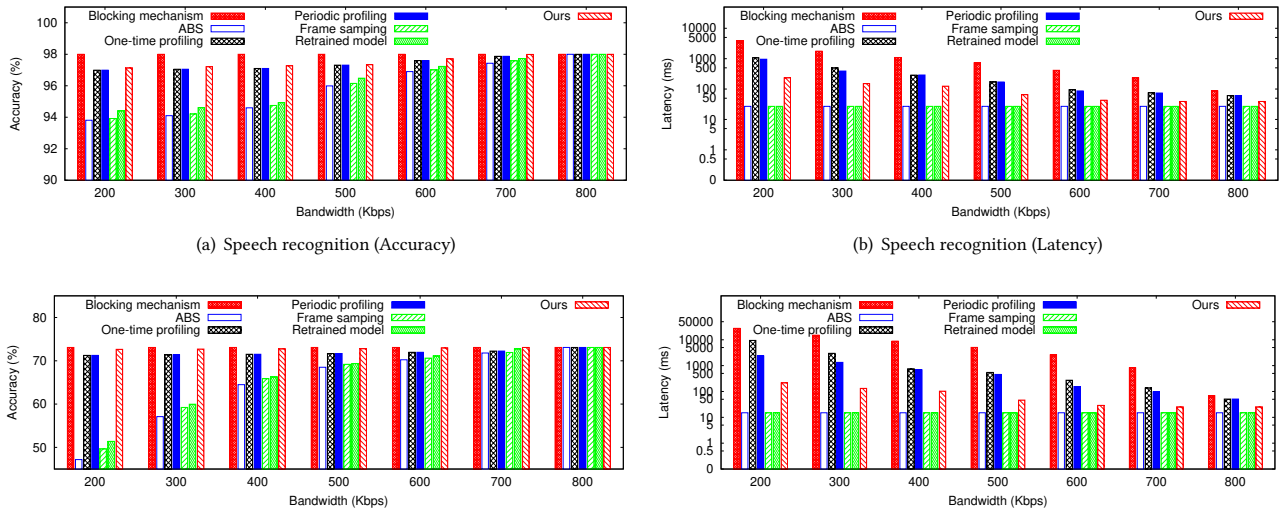
Heterogeneity Activity Recognition Dataset We use the STISEN dataset [67] comprising 43 million accelerometer and gyroscope readings from nine users performing six activities: biking, sitting, standing, walking, stair up, and stair down. Readings were recorded from four smartwatches, 31 smartphones and one tablet, representing 13 device models from four manufacturers. Note that, unlike audio-visual data, inertial data is not considered high-bandwidth; however, BLE radios are highly duty-cycled to minimize energy consumption, so delays of tens to hundreds of millisecond still occur.

Multi-modal Models We leverage three end-to-end multi-modal models for speech recognition [61], event detection [68], and activity recognition [62]. The speech recognition model leverages two separate residual networks (ResNet [32]) to extract features from raw images and audio signals, and two two-layer BGRUs to model the temporal dynamics of each modality. Two cascaded BGRU layers fuse the video and audio data by concatenating the features of each modality. Finally, a softmax activation function is used to predict the inference class. The event detection model utilizes two pre-trained CNNs to extract visual and audio features. Two LSTMs model temporal dependencies between the two modalities. A multi-modal fusion network, based on Multi-modal Residual Network [44] is used to generate a joint audiovisual representation. Finally, a fully-connected layer with a softmax activation function is used to classify audiovisual events. For activity recognition, each sensing modality has a dedicated CNN to extract preliminary features. These features are combined through fully-connected layers to infer the output.

Evaluation Metrics Performance is evaluated by two metrics: accuracy and latency. The best accuracy Acc_{best} is defined as the accuracy achieved using all data from both modalities within a given time window. End-to-end latency is computed by subtracting the timestamp of the final output from the timestamp recorded when incoming data from the faster steam (e.g., audio) reached the cloud for processing.

Baselines We compare our system with six baselines. The first baseline is naive blocking, in which clients transmit the highest data quality possible (full resolution and maximum frame-rate), and the inference pipeline simply blocks until all inputs from all data streams have arrived. The second baseline is adaptive bitrate streaming (ABS), widely used in live video streaming [10, 54]. This experiment requires the client to measure bandwidth in real time and vary the streaming video resolution to minimize latency.

We also compare our system against three existing video analytics methods. One is based on frame sampling, and the other two are profiling methods [39, 84]. Frame sampling minimizes the bandwidth required to transmit data by filtering out redundant information (e.g., images with fewer objects) [35, 85]. For a fair comparison, we choose a key-frame sampler that streams data without buffering or batch processing. Specifically, we sample key frames within an inference window using optical flow [79]. We then apply linear imputation to fit within dynamic bandwidth constraints. Profiling optimizes video streaming configurations to minimize the inference latency while achieving a desired accuracy. In general, state-of-the-art profiling methods can be further classified into two categories: *one-time profiling* [29, 84] in which the processing pipeline configuration is profiled once when video streaming begins, after which configuration is fixed, and *periodic profiling* [39, 80] in which the configuration is profiled periodically to find an optimal



(a) Speech recognition (Accuracy)

(b) Speech recognition (Latency)

(c) Event detection (Accuracy)

(d) Event detection (Latency)

Figure 6: Synthetic trace-driven evaluation. Our system consistently outperforms the baselines in both application scenarios. Among all of the methods that achieve high accuracy, our system has the lowest end-to-end latency.

accuracy-latency trade-off at the cost of profiling overhead. In our experiment, we do not take the profiling overhead into account.

Finally, we compare our system with an approach that retrains the multi-modal models themselves with varying amounts of stream lag. The number of the retrained models is equal to the number of delay configurations. During the test phase, the approach will measure the delay and select a retrained model that leverages the partial data to get the outputs without blocking the inference pipeline.

Configurations Vision algorithms often contain various configurable parameters or "knobs." We focus on two knobs: frame rate and image resolution. For the LRW dataset, the frame rates are 25 FPS or 15 FPS, and the resolutions are 256×256, 128×128, 64×64, or 32×32. For the AVE dataset, the frame rates are 30 FPS or 15 FPS, and the resolutions are 224×224, 112×112, 56×56, or 28×28.

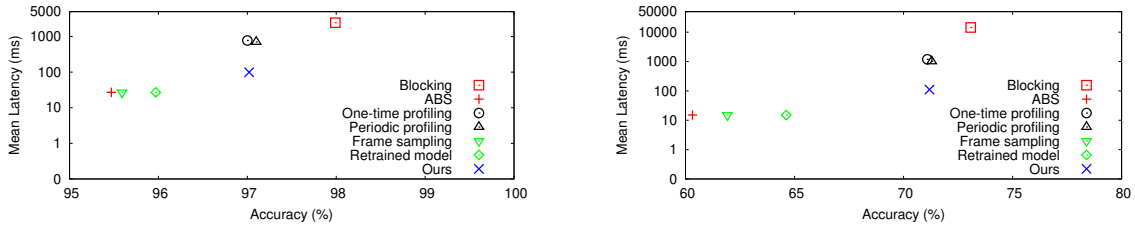
5.2 End-to-end Improvement

We start with the end-to-end improvement of our system over the six baselines. In this experiment, we do not consider the impact of encoding and compression at the IoT devices, which will be shown in Sec. 5.7. For our method, we leverage the data alignment module in Sec. 3.3 to align the data streams. For the baselines, we assume the data streams are perfectly aligned, as other techniques cannot align data streams without using timestamps or injecting synchronization pulses.

Synthetic trace-driven evaluation To demonstrate our system's ability to adapt to asymmetric bandwidths across different modalities, we artificially vary bandwidths to simulate fluctuating network conditions using synthetic network traces generated with Pensieve [53], which uses a Markovian model in which each state represents an average throughput in the range from 200–800 Kbps. Each bandwidth value was drawn from a Gaussian distribution centered around the mean bandwidth with variance uniformly distributed between 0.05–0.5. The granularity of the network traces is one second.

Figure 6 shows that our system incurs significantly reduced latency with minimal loss of accuracy under fluctuating network dynamics and consistently outperforms the baselines. Our accuracy drops by only 1% compared to naïve blocking but achieves 2–128x latency speedup over this method since it can speculate using imputed data. We perform better than both one-time and periodic profiling methods in terms of latency (2–43x, and 1.5–10x speedup) while achieving the same accuracy. The profiling methods cannot adapt to rapid bandwidth dynamics – if actual bandwidth is less than profiled bandwidth, they block until sufficient data arrives. Our system is more adaptive and can speculate using imputed data. ABS, frame sampling, and model retraining have lower latency (15% to 30%), but our method achieves 1% – 25% higher accuracy. The advantage of our method is particularly evident under low bandwidth conditions, in which the advantages of a rollback-based method over changing video resolutions become apparent. We also repeat the experiment without network fluctuations, and the results are similar to Fig. 6. We observe that when the network is fast (e.g., > 800 Kbps bandwidth), our system offers only marginal improvements because both streams can transmit full-resolution data to the cloud. However, IoT nodes often leverage low-power wireless communication technologies like BLE or Zigbee, which only achieve a maximum of sub-Mbps throughput under practical conditions [55], to transmit data due to strict energy constraints. For example, [6, 7, 21] reported BLE throughputs of 221Kbps, 341Kbps and 724Kbps for BLE 4.2, BLE 5.0 indoor environment, and BLE 5.0 body area network, respectively. Even for cellular networks using 4G LTE, the average throughput can drop to 300 Kbps [36]. Therefore, applications using modern wireless technologies can still see significant gains from our system.

Real-world trace-driven evaluation We now evaluate our system against several real-world bandwidth traces. Specifically, we look at multi-path TCP traces that were collected in a complex city environment [20]. The dataset covers the traffic produced by more than ten users using Nexus 5 smartphones. The Nexus 5



(a) Speech recognition (b) Event detection

Figure 7: Real-world trace-driven evaluation. Our system (blue cross) consistently outperforms the rest of the baselines in both application scenarios. Among the models that achieve good accuracy, our system achieves the least end-to-end latency.

Methods	Speech recognition			
	Inference Latency (ms)	Alignment Latency (ms)	Rollback Latency (ms)	End-to-end Latency (ms)
Naive Blocking	27	-	-	2,448
ABS	27	-	-	27
One-time profiling	27	-	-	774
Periodic profiling	27	-	-	702
Frame sampling	27	-	-	27
Retrained model	27	-	-	27
Ours	27	<1	62	<90

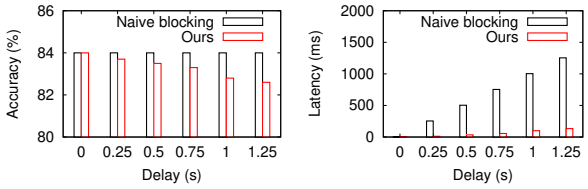
Table 1: Latency breakdown.

smartphones ran a modified Linux kernel (includes Multi-path TCP v0.89.5) to collect both Wi-Fi and cellular traces on Android 4.4. Since we assume that data from each modality is continuously transmitted from a separate device to the cloud, we randomly select two traces for the microphone node and the camera node. The network bandwidth fluctuates from 100 Kbps to 1 Mbps with a state-transition granularity of one second.

Figure 7 shows the comparison between our system and the four baselines on real-world network traces, and Table 1 shows the latency breakdown across different modules. The results are similar to those attained with synthetic traces. ABS, frame sampling, retrained model achieve the minimum end-to-end latency (roughly 60ms lower than our method) but at significant accuracy loss (up to 12% accuracy loss). Also, the training overhead of the retrained model method (100–250 hours) is about ten times higher than our system (8–20 hours), as our system only imputes intermediate features rather than the raw sensor data and it does not modify the existing inference model. We show accuracy comparable to one-time and periodic profiling but offer 7.5–10x, and 7–8x latency speedup. We have 1% lower accuracy than naïve blocking but 24–120x latency speedup. Since our system does not optimize data-transmission configurations before the latency is detected on the cloud, it is also possible to integrate profiling and frame sampling methods into our system to further improve inference efficiency.

Generalizability Across Modalities While most of our results show performance on audiovisual (LRW [17] and AVE [68]) datasets, we briefly demonstrate that our technique generalizes to other sensor modalities. In this experiment, we evaluate our system on a multi-modal model for activity recognition with accelerometer and gyroscope data [62].

Figure 8 compares our approach against naïve blocking. (Results against other baselines are similar to the above observations.) We introduce various delays (0–1.25s) to one of the sensor streams corresponding to duty-cycled radio transfer delays. Similar to previous results, our system achieves 7–24× latency speedup over naïve blocking, with accuracy drops of only 1.4%, even when a 1.25s delay



(a) Accuracy (b) Latency

Figure 8: End-to-end improvement for activity recognition.

Methods	Speech recognition		Event detection	
	Accuracy (%)	Latency (ms)	Accuracy (%)	Latency (ms)
BicycleGAN [89]	96.01	121.66	69.95	133.47
Pix2PixHD [77]	96.13	108.54	70.38	119.89
Ours	97.02	89.31	71.19	115.49

Table 2: The comparison of different GAN architectures.

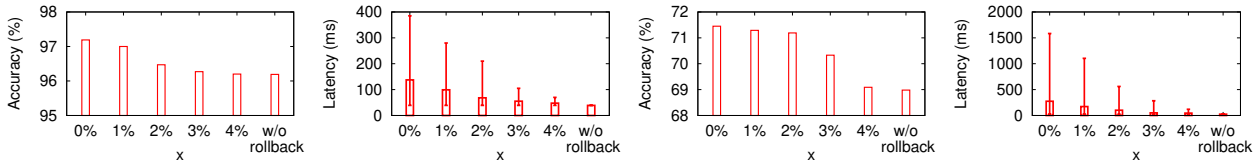
to one modality is introduced. Therefore, our system is quite robust and can be generalized to various sensing modalities.

5.3 Impact of Data Imputation

Next, we microbenchmark the impact of individual components in our system. We start with our data imputation module and investigate the impact of different GAN networks. Our system applies an image-to-image translation framework Pix2Pix [38] to impute intermediate features. We also implement our system on two commonly used generative networks: BicycleGAN [89] and Pix2PixHD [77]. BicycleGAN is a popular image-to-image multimodal translation architecture that combines Conditional Variational Autoencoder (CVAE) and Conditional Latent Regressor (CLR) models to enforce the connection between latent encoding and output in both directions jointly and achieve improved performance [89]. Pix2PixHD also synthesizes images from semantic labels, and it addresses two main issues of the existing work, namely, the difficulty of generating high-resolution images with GANs and the lack of details and realistic textures in previous high-resolution results [77]. Table 2 shows the performance comparison of different GAN networks. Among the GAN candidates, we did not find an advantage in using a variational method. This implies that the random samples from a learned distribution may counteract the benefits of uncertainty modeling, which introduces higher noise to the output [51]. Our method outperforms Pix2PixHD in both application scenarios with 0.8% improvement in accuracy and 4–22 *ms* less inference latency.

5.4 Impact of Rollback

A key component of our system is the rollback module. Figure 9 summarizes the impact of the rollback module for different values of the user-defined parameter x, which controls how much accuracy



(a) Speech recognition (Accuracy) (b) Speech recognition (Latency) (c) Event detection (Accuracy) (d) Event detection (Latency)

Figure 9: Impact of rollback module. x is a user-defined parameter to control the granularity of rollback. We plot error bars covering the 90% confidence interval.

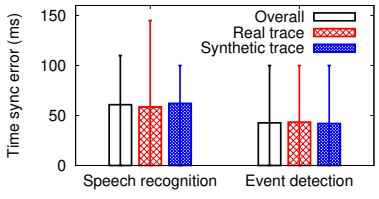
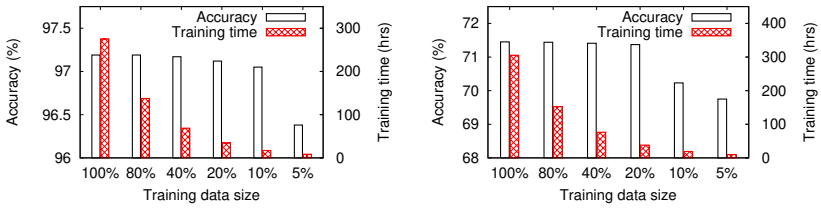


Figure 10: Data alignment deviation.



(a) Wavelengths (b) Intensity levels

Figure 11: The impact of training overhead.

loss can be tolerated. The trend is as expected, i.e., accuracy degrades as x increases, but we see an inflection point around $x > 1\%$ in speech recognition and $x > 2\%$ in event detection at which point the accuracy degrades more rapidly. Therefore, we select $x = 1\%$ as the best trade-off between accuracy and latency for these datasets. We note that x does not need to be pre-determined and can be set adaptively depending on network conditions. For example, if the bandwidth is low (e.g., < 200 Kbps), we can set a high x to further reduce the latency, whereas if the bandwidth is high (e.g., > 800 Kbps), we can set a low x to improve the accuracy.

5.5 Impact of Data Alignment

We now investigate the impact of our data alignment method on accuracy and latency. Figure 10 shows the data alignment deviation between our method and ground truth. Overall, the mean deviation is 45 *ms* and 42 *ms* for speech recognition and event detection, respectively. Since we search for the minimal L1 distance between feature maps to align data streams, the granularity is one frame (e.g., 33 ms for 30 FPS or 40 ms for 25 FPS). Thus, our data alignment module achieves an average deviation of 1 to 1.5 frames. We observe that the data alignment deviation for speech recognition is higher than that for event detection. Due to the nature of lipreading images, the difference between adjacent images is smaller than that for events like car racing, which makes it hard for the data alignment module to search for the minimum L1 loss. We also investigate the accuracy and the latency differences between our system and one with perfect data alignment. We observe that the differences are marginal, and the latency added by data alignment is less than 10 ms, and the accuracy difference is nearly zero.

5.6 Training and Resource Overhead

Our system introduces training overhead since we train over several configurations (data resolution and delays). However, even though we train several GAN models, we do not need to use the entire training dataset since we are imputing intermediate features rather than raw sensor data. Figure 11 shows the impact of training data size on the performance of our system. By leveraging a fifth or even a tenth of the original training data, we can significantly reduce the entire training overhead down to about 8–20 hours while still retaining high accuracy. The runtime resource overhead for our system is also small (about 100 MB of memory in the GPU) because we only need to load two GANs (data imputation and rollback) into memory at one time.

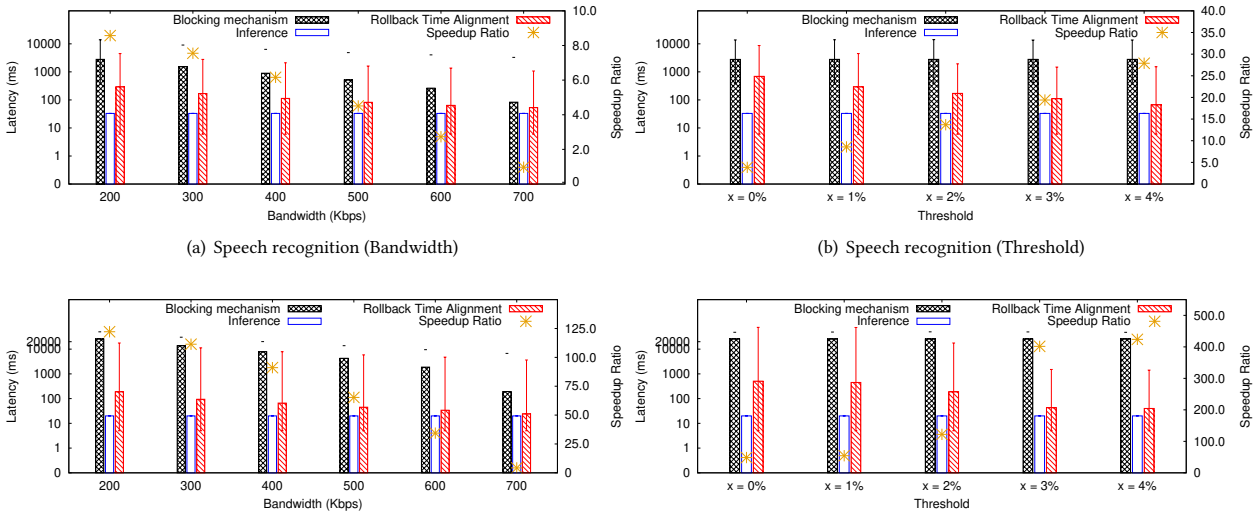
5.7 Impact of IoT nodes

We now look at end-to-end improvement using real IoT devices and Wi-Fi radio. So far, we have assumed that the sensor data can be instantly processed on IoT devices, but in practice, there are many real-world delays, including encoding and compression on the IoT device and processing delays in the cloud. In this experiment, IoT nodes process video in real time. Figures 12(a) and 12(c) illustrate our system's performance compared to the naïve blocking method. Overall, our method still outperforms the blocking baseline under all bandwidth conditions and achieves up to 9× and 122× latency speedup on the LRW and the AVE dataset, which is consistent with our previous results. The mean computation overhead of video and audio data streaming is 4ms per frame and 3ms per second, respectively. Figures 12(b) and 12(d) show the speedup ratio for different user-defined parameters. When $x = 4\%$, our system achieves 28× and 424× latency speedup on the LRW and AVE datasets. The results validate our previous evaluation and demonstrate that our approach offers significant benefits in real-world systems.

6 RELATED WORK

Minimizing inference lag Existing work to minimize inference lag in deep learning has focused largely on server-side constraints, with proposed solutions seeking to optimize the utilization of available compute resources. General-purpose job schedulers [33, 71] are not well suited to lag-sensitive applications because they make scheduling decisions based on available cluster resources, prioritizing job fairness, rather than on the characteristics of jobs.

Highly specialized schedulers [29, 30, 39, 84] use pipeline profiling to determine how best to allocate resources to video analytics processes whose accuracy and lag requirements vary and whose demands may change with time. Adaptive schedulers [2, 11, 19, 25,



(a) Speech recognition (Bandwidth)

(b) Speech recognition (Threshold)

(c) Event detection (Bandwidth)

(d) Event detection (Threshold)

Figure 12: Impact of IoT nodes. 12(a) and 12(c) are under the threshold of 1% amd 2% respectively. 12(b) and 12(d) are under the bandwidth of 200kbps. We plot error bars covering the 90% confidence interval.

42, 72] are able to make scheduling decisions with some knowledge of the jobs being scheduled and are able to better distribute jobs both across nodes and temporally within nodes, but are not specialized enough to develop individual pipeline profiles for each query type. Splitting computation between cloud resources and edge nodes in order to minimize server-side computational burden and conform to bandwidth constraints is another recent approach. In these cases, lightweight image-detection models operating at the edge identify frames of interest or coarse candidate results for object queries [35, 85]. These frames, rather than the entire video stream, are then forwarded to the cloud for more computationally expensive high-resolution object recognition.

Although related, these approaches generally address limitations in single-modality scenarios (e.g., object detection in video streams). Our work specifically addresses inference lag arising from missing or misaligned sensor data in multi-modal scenarios, rather than from competitive compute environments with single-modality streams. Because deep learning applications over real-time multi-modal IoT sensor data is a relatively new problem, our comparisons draw from related work in adjacent fields.

Compensating for misaligned sensor streams The need to align multi-modal sensor data prior to use in machine learning or deep learning applications is not new, but most existing work assumes either that 1) the sensor readings are already temporally aligned, or 2) that the raw signals need to be temporally aligned before being input to the model.

Commonly, the sensors used to record multi-modal streams are co-located on the same device (e.g., camera and microphone, or cameras and LIDAR on an autonomous vehicle [12, 15, 47, 48, 82]) and share a common clock. In other cases the sensor may be recorded by separate devices but still share a common clock that is either internal to one or external to both, or the devices will synchronize via a shared synchronization protocol [49]. In either of these scenarios, the first assumption is sound and no further processing to align the signals is required.

The second assumption applies when sensors are located on devices that are not tightly synced to a shared clock. Prior to the rise of deep learning, analytical techniques like dynamic time warping [1, 41, 86], manifold warping [26, 73] and canonical correlation analysis [31, 65, 87, 88] were used to align signals with a shared dimensionality. Deep learning techniques have been developed to compensate for the inherent dimensionality limitations of these techniques [4, 34, 69, 70], but in order to work, raw data from all sensor modalities must be received prior to alignment.

Our technique is robust to completely missing or poorly aligned data, without requiring explicit alignment of the raw signals prior to fusion. By imputing the features of delayed signals at multiple temporal offsets and finding the best imputed alignment, we can bypass explicit alignment of raw signals and proceed with inference despite poor temporal alignment between streams.

7 CONCLUSION

We present speculative inference on multi-modal data streams to adapt to asymmetric bandwidth. The system adaptively and speculatively generates inference from multi-modal models without waiting for all data streams to be available and aligned. We implement the system on three multi-modal scenarios. Experiments on publically available datasets show that our system achieves 7 – 128× latency speedup with the same accuracy as six existing solutions.

ACKNOWLEDGEMENTS

We sincerely thank reviewers for their insightful feedback. The research reported in this paper was sponsored in part by the CCDC Army Research Laboratory (ARL) under Cooperative Agreement W911NF-17-2-0196 (ARL IoBT CRA) and by National Science Foundation under Grant No. 1815347. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL, NSF or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] John Aach and George M. Church. Aligning gene expression time series with time warping algorithms . *Bioinformatics*, 17(6):495–508, 06 2001.

[2] L. Amini, N. Jain, A. Sehgal, J. Silber, and O. Verscheure. Adaptive control of extreme-scale stream processing systems. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 71–71, 2006.

[3] Mr José M Anca Jr. *Multimodal Safety Management and Human Factors: Crossing the Borders of Medical, Aviation, Road and Rail Industries*. Ashgate Publishing, Ltd., 2012.

[4] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. volume 28 of *Proceedings of Machine Learning Research*, page 1247–1255, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[5] Iryna Anina, Ziheng Zhou, Guoying Zhao, and Matti Pietikäinen. Ouluvs2: A multi-view audiovisual database for non-rigid mouth motion analysis. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–5. IEEE, 2015.

[6] Michael A. Ayoub and Ahmed M. Eltawil. Throughput characterization for bluetooth low energy with applications in body area networks. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2020.

[7] Behnam Badihi, Muhammad Usman Sheikh, Kalle Ruttik, and Riku Jäntti. On performance evaluation of ble 5 in indoor environment: An experimental study. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, 2020.

[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] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE international conference on computer vision*, pages 2745–2754, 2017.

[10] Divyashri Bhat, Amr Rizk, Michael Zink, and Ralf Steinmetz. Network assisted content distribution for adaptive bitrate video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 62–75, 2017.

[11] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, page 285–300, Broomfield, CO, October 2014. USENIX Association.

[12] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[13] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[14] Moitreya Chatterjee and Anoop Cherian. Sound2sight: Generating visual dynamics from sound and context.

[15] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving, 2016.

[16] Max Chu, Annette Patton, Josh Roering, Cora Siebert, John Selker, Cara Walter, and Chet Udell. Sitkanet: A low-cost, distributed sensor network for landslide monitoring and study. *HardwareX*, 9:e00191, 2021.

[17] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In *Asian Conference on Computer Vision*, pages 87–103. Springer, 2016.

[18] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America*, 120(5):2421–2424, 2006.

[19] Carlo Curino, Djellel E Difallah, Chris Douglas, Subru Krishnan, Rahgu Ramakrishnan, and Sriram Rao. Reservation-based scheduling: If you're late don't blame us! Technical Report MSR-TR-2013-108, October 2013. Published in SoCC 2014.

[20] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. A first analysis of multipath tcp on smartphones. In *International Conference on Passive and Active Network Measurement*, pages 57–69. Springer, 2016.

[21] F. John Dian, Amirhossein Yousefi, and Sungjoon Lim. A practical study on bluetooth low energy (ble) throughput. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 768–771, 2018.

[22] Royston Dmello, Sai Yerremreddy, Samriddha Basu, Tejas Bhitle, Yash Kokate, and Prachi Gharpure. Automated facial recognition attendance system leveraging iot cameras. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 556–561. IEEE, 2019.

[23] A Gómez Eguíluz, Iñaki Rañó, Sonya A Coleman, and T Martin McGinnity. Multimodal material identification through recursive tactile sensing. *Robotics and Autonomous Systems*, 106:130–139, 2018.

[24] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *arXiv preprint arXiv:1808.03841*, 2018.

[25] Andrew Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. Jockey: Guaranteed job latency in data parallel clusters. 04 2012.

[26] D. Gong and G. Medioni. Dynamic manifold warping for view invariant action recognition. In *2011 International Conference on Computer Vision*, pages 571–578, 2011.

[27] MTA Cooperative Group et al. National institute of mental health multimodal treatment study of adhd follow-up: 24-month outcomes of treatment strategies for attention-deficit/hyperactivity disorder. *Pediatrics*, 113(4):754–761, 2004.

[28] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836, 2016.

[29] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136, 2016.

[30] Mark Hardiman, Ying Ou, Ryan Frazier, Zeyi Lee, and Longxiang Cui. Project noscope. Master's thesis, EECS Department, University of California, Berkeley, May 2015.

[31] M. A. Hasan. On multi-set canonical correlation analysis. In *2009 International Joint Conference on Neural Networks*, pages 1128–1133, 2009.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*, volume 11, pages 22–22, 2011.

[34] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[35] 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.

[36] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An in-depth study of lte: Effect of network protocol and application behavior on performance. *ACM SIGCOMM Computer Communication Review*, 43(4):363–374, 2013.

[37] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5077–5086, 2017.

[38] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[39] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266, 2018.

[40] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[41] B. . Juang. On the hidden markov model and dynamic time warping for speech recognition — a unified view. *AT T Bell Laboratories Technical Journal*, 63(7):1213–1243, 1984.

[42] Sangeetha Abdu Jyothi, Carlo Curino, Ishai Menache, Shravan Matthur Narayanamurthy, Alexey Tumanov, Jonathan Yaniv, Ruslan Mavlyutov, Inigo Goiri, Subru Krishnan, Janardhan Kulkarni, and Sriram Rao. Morpheus: Towards automated slos for enterprise clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, page 117–134, Savannah, GA, November 2016. USENIX Association.

[43] Hyeon-Woo Kang and Hang-Bong Kang. Prediction of crime occurrence from multi-modal data using deep learning. *PloS one*, 12(4):e0176244, 2017.

[44] Jin-Hwa Kim, Sang-Woo Lee, Donghyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Multimodal residual learning for visual qa. In *Advances in neural information processing systems*, pages 361–369, 2016.

[45] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2019.

[46] KN Krishnanand and Debasish Ghose. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent and Grid Systems*, 2(3):209–222, 2006.

[47] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation, 2017.

[48] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *IROS*,

2018.

[49] M. Kushwaha, S. Oh, I. Amundson, X. Koutsoukos, and A. Ledeczi. Target tracking in heterogeneous sensor networks using audio and video sensor fusion. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 14–19, 2008.

[50] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101, 2018.

[51] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. See through smoke: robust indoor mapping with low-cost mmwave radar. In *MobiSys*, pages 14–27, 2020.

[52] Sathiya Kumaran Mani, Ramakrishnan Durairajan, Paul Barford, and Joel Sommers. A system for clock synchronization in an internet of things. *arXiv preprint arXiv:1806.02474*, 2018.

[53] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210, 2017.

[54] Hyunwoo Nam, Kyung-Hwa Kim, Doru Calin, and Henning Schulzrinne. Youslow: a performance analysis tool for adaptive bitrate video streaming. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 111–112, 2014.

[55] PrithviRaj Narendra, Simon Duquennoy, and Thiemo Voigt. Ble and ieee 802.15. 4 in the iot: Evaluation and interoperability considerations. In *International Internet of Things Summit*, pages 427–438. Springer, 2015.

[56] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[57] Edmund B Nightingale, Peter M Chen, and Jason Flinn. Speculative execution in a distributed file system. *ACM SIGOPS operating systems review*, 39(5):191–205, 2005.

[58] Steven M Nowick and Montek Singh. Asynchronous design—part 2: Systems and methodologies. *IEEE Design & Test*, 32(3):19–28, 2015.

[59] Alex Olwal, Hrvoje Benko, and Steven Feiner. Senseshapes: Using statistical geometry for object selection in a multimodal augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 300–301. IEEE, 2003.

[60] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[61] Stavros Petridis, Themos Stafylakis, Pingehuan Ma, Feipeng Cai, Georgios Tzimiropoulos, and Maja Pantic. End-to-end audiovisual speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6548–6552. IEEE, 2018.

[62] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. Multimodal deep learning for activity and context recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–27, 2018.

[63] Habib F Rashvand and Jose M Alcaraz Calero. *Distributed sensor systems: practice and applications*. John Wiley & Sons, 2012.

[64] Leonardo Ricaurte. The array of things, chicago. *Urban Planning for Transitions*, pages 171–182, 2021.

[65] S. Shariat and V. Pavlovic. Isotonic cca for sequence alignment and activity recognition. In *2011 International Conference on Computer Vision*, pages 2572–2578, 2011.

[66] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[67] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015.

[68] Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. Audio-visual event localization in unconstrained videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 247–263, 2018.

[69] G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. Zafeiriou. Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1128–1138, 2018.

[70] G. Trigeorgis, M. A. Nicolaou, S. Zafeiriou, and B. W. Schuller. Deep canonical time warping. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5110–5118, 2016.

[71] Vinod Vavilapalli, Arun Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache hadoop yarn: yet another resource negotiator. 10 2013.

[72] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Aria: Automatic resource inference and allocation for mapreduce environments. In *Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC 2011 and Co-located Workshops*, Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC 2011 and Co-located Workshops, page 235–244, July 2011. 8th ACM International Conference on Autonomic Computing, ICAC 2011 and Co-located Workshops ; Conference date: 14-06-2011 Through 18-06-2011.

[73] Hoa Trong Vu, Clifton Carey, and Sridhar Mahadevan. Manifold warping: Manifold alignment over time. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.

[74] Chia-Hung Wan, Shun-Po Chuang, and Hung-Yi Lee. Towards audio to scene image synthesis using generative adversarial network. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 496–500. IEEE, 2019.

[75] Jiayu Wang, Wengang Zhou, Guo-Jun Qi, Zhongqian Fu, Qi Tian, and Houqiang Li. Transformation gan for unsupervised image synthesis and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 472–481, 2020.

[76] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T Campbell. Studentlife: Using smartphones to assess mental health and academic performance of college students. In *Mobile health*, pages 7–33. Springer, 2017.

[77] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[78] Wei Wang, Dan Wang, and Yu Jiang. Energy efficient distributed compressed data gathering for sensor networks. *Ad Hoc Networks*, 58:112–117, 2017.

[79] Wayne Wolf. Key frame selection by motion analysis. In *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings*, volume 2, pages 1228–1231. IEEE, 1996.

[80] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. Vstore: A data store for analytics on large videos. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–17, 2019.

[81] Bhavaniprasad Yalagala, Shivam Khandelwal, J Deepika, and Sushmee Badhulika. Wirelessly destructible mgo-pvp-graphene composite based flexible transient memristor for security applications. *Materials Science in Semiconductor Processing*, 104:104673, 2019.

[82] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18:3337, 10 2018.

[83] Wenpeng Yu, Wenxuan Yao, Xianda Deng, Yinfeng Zhao, and Yilu Liu. Timestamp shift detection for synchrophasor data based on similarity analysis between relative phase angle and frequency. *IEEE Transactions on Power Delivery*, 2019.

[84] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live video analytics at scale with approximation and delay-tolerance. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 377–392, 2017.

[85] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 426–438, 2015.

[86] F. Zhou and F. De la Torre. Generalized time warping for multi-modal alignment of human motion. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289, 2012.

[87] F. Zhou and F. De la Torre. Generalized canonical time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):279–294, 2016.

[88] Feng Zhou and Fernando Torre. Canonical time warping for alignment of human behavior. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2286–2294. Curran Associates, Inc., 2009.

[89] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in neural information processing systems*, pages 465–476, 2017.