A General Framework for Model Adaptation to Meet Practical Constraints in Computer Vision

Shiyuan Huang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2024

Abstract

A General Framework for Model Adaptation to Meet Practical Constraints in Computer Vision

Shiyuan Huang

Recent advances in deep learning models have shown impressive capabilities in various computer vision tasks, which encourages the integration of these models into real-world vision systems such as smart devices. This integration presents new challenges as models need to meet complex real-world requirements. This thesis is dedicated to building practical deep learning models, where we focus on two main challenges in vision systems: data efficiency and variability. We address these issues by providing a general model adaptation framework that extends models with practical capabilities.

In the first part of the thesis, we explore model adaptation approaches for efficient representation. We illustrate the benefits of different types of efficient data representations, including compressed video modalities from video codecs, low-bit features and sparsified frames and texts. By using such efficient representation, the system complexity such as data storage, processing and computation can be greatly reduced. We systematically study various methods to extract, learn and utilize these representations, presenting new methods to adapt machine learning models for them. The proposed methods include a compressed-domain video recognition model with coarse-to-fine distillation training strategy, a task-specific feature compression framework for low-bit video-and-language understanding, and a learnable token sparsification approach for sparsifying human-interpretable video inputs. We demonstrate new perspectives of representing vision data in a more practical and efficient way in various applications.

The second part of the thesis focuses on open environment challenges, where we explore model adaptation for new, unseen classes and domains. We examine the practical limitations in current recognition models, and introduce various methods to empower models in addressing open recognition scenarios. This includes a negative envisioning framework for managing new classes and outliers, and a multi-domain translation approach for dealing with unseen domain data. Our study shows a promising trajectory towards models exhibiting the capability to navigate through diverse data environments in real-world applications.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I extend my heartfelt gratitude to Professor Shih-Fu Chang for his invaluable guidance and unwavering support throughout the completion of this thesis. I deeply admire not only his outstanding leadership of the Digital Video and Multimedia Lab (DVMM) but also his years of widespread success and tireless dedication to advancing the future of science and education. The accomplishments detailed in this thesis stem from collaborative endeavors. During this time, I've had the privilege of interacting with brilliant researchers. Special thanks go to Gunnar Sigurdsson for his valuable contributions in Chapters 3 and 4. I'm also grateful to Jiawei Ma, Guangxing Han, Xudong Lin and many others who have enriched my work with their insightful discussions. Finally, I want to extend my appreciation to my family for their unwavering support and to all the close individuals in my life who provided assistance during the challenging COVID-19 period, which encompassed a significant portion of my Ph.D. study. Their presence and support have been invaluable during these pandemic years, contributing significantly to my academic journey.

Chapter 1: Introduction

1.1 Background and Motivation

Visual information is an indispensable source of media of people's daily lives. Over the past decades, deep learning models have exhibited remarkable capabilities in understanding visual data and successfully executing diverse computer vision tasks, including visual recognition in images and videos, object detection and tracking, visual question answering, etc. This progress takes a significant step towards vision-capable intelligence. It also encourages a growing integration of deep learning models into practical vision systems for real-world production, such as smart devices, which poses new challenges as models need to adhere to more complex requirements imposed by real-world scenarios.

Recent advancements in deep learning have demonstrated the ability to learn powerful models using large-scale data and strong computational resources. Different pre-training strategies have been proposed to accommodate variations in data availability, quality, and properties. The resulted models, which are referred as pre-trained models, can then be used for various downstream applications. However, when applied to real-world vision systems, visual data by its nature imposes specific challenges for deployment. One practical concern is that visual data is huge in volume and presents difficulties in processing and storage within the constraints of hardware. For example, a real-time home monitoring system demands efficient management of large recorded videos for a timely and accurate monitoring. Another practical concern is that vision systems encounter diverse and dynamic circumstances, leading to highly variable visual data. For example, a surveillance system should be able to handle new objects and individuals while also adapting to different environments such as outdoor and indoor scenes. Additional considerations include fairness, privacy, ethics and more. Addressing these concerns necessitates research efforts focused on building practical deep learning models and designing new capabilities.

In this thesis, we investigate and propose methods to extend pre-trained deep learning models towards practical capabilities. We focus on the aforementioned two important data challenges in real world: data efficiency and variability, and design model adaptation techniques to meet with different constraints exhibited by visual data in computer vision applications. The thesis is hence divided into two parts: Part I delves into model adaptation towards efficient representations; Part II focuses on model adaptation to open environments, encompassing new, unseen classes and domains. Next we will give an overview of the relevant literature, and connect our work to the existing research endeavors.

1.2 Overview of Model Adaptation

Model adaptation is a broad area covering different contexts, directions and applications in computer vision. An important field is model fine-tuning, where the pre-trained model is continued training for new downstream applications on a small set of task-specific data. Research works such as [1] and [2] underscore the importance of fine-tuning multi-modal models for specific vision-language tasks such as visual question answering and video retrieval. However, in dynamic open environments where the quality and quantity of data are not well guaranteed, standard fine-tuning approaches can be sub-optimal. To enhance adaptability in changing environments, several techniques are proposed to make models more generalizable. For example, meta learning [3] trains model on a variety of pseudo tasks to improve generalization; feature disentanglement [4] decomposes features to extract environment-invariant parts. Our works in Chapter 5 and 6 address similar challenges of adapting models to unseen classes or new domains. Chapter 5 is built upon meta learning by designing task-level training scheme. Chapter 6 explores feature debiasing techniques akin to feature disentanglement, in order to identify domain-invariant components.

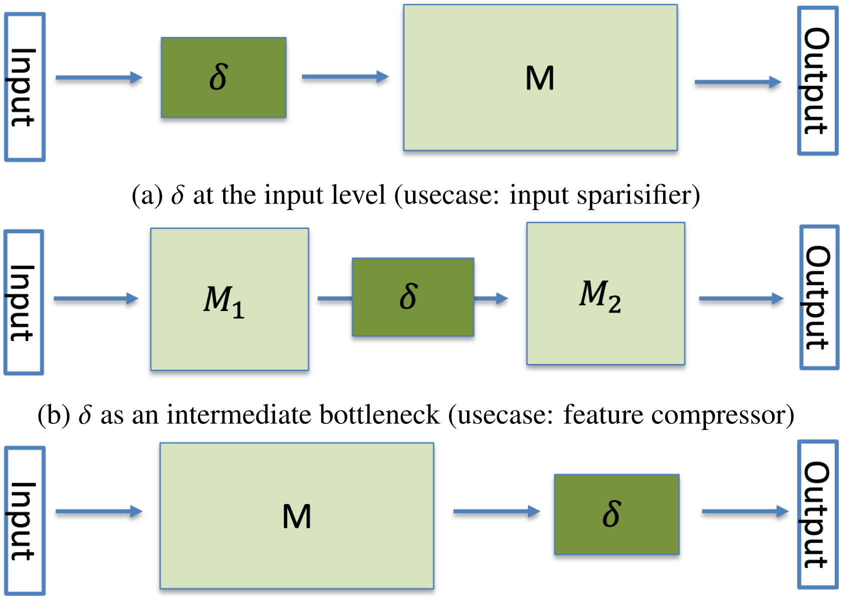
In a different context of model adaptation, the focus is on computational efficiency. For example, model distillation [5] distills a large powerful model into a smaller-sized network, in order to reduce memory and energy consumption while retaining the original capabilities. Additionally, there is also increasing attention to data efficiency, where people seek to reduce data size for both computation and performance advantages. For example, feature selection [6] selects the important set of features for better model performance and processing speed. We start with Chapter 2 that examines model distillation under the context of compressed-domain video understanding. Then in Chapter 3 and 4, we introduce new perspectives of adapting model for efficient data representations such as low-bit feature and sparsified inputs.

1.3 Our General Framework for Model Adaptation

In this thesis, we present a general model adaptation framework where our works follow. We highlight the critical need for efficiency in practical deployment, hence our primary objective is to minimize the changes to the original model architecture and parameters. We introduce our general framework as $\mathcal{M} + \delta$, where $\mathcal{M}$ refers to the pre-trained model, and δ refers a small architecture added to the original model. Throughout the thesis, we design and instantiate δ in different ways, serving for different goals and purposes, coupled with different learning strategies. Regarding the design and incorporation of δ, we discuss a special case where $\delta = 0$ (Chapter 2), and then explore δ as an additional input module (Chapter 4), as an intermediate bottleneck (Chapter 3) and as a final module on top of the model (Chapter 5 and 6). In terms of purposes and goals, we examine δ as an efficient representation adapter, where it functions as a feature compressor (Chapter 3) and a input sparsifier (Chapter 4); and δ as an open environment adapter, where it serves as a negative prototype generator (Chapter 5) and a domain translator (Chapter 6). Figure 1.1 summarizes the instantiations and usecases of our $\mathcal{M} + \delta$ framework covered in the thesis. The versatility of our approach allows for adaptable model enhancements tailored to specific objectives and deployment scenarios.

1.4 Thesis Outline and Contributions

Putting together, we here provide a full outline of the chapters and summarize our contributions briefly. In a nutshell, we design model adaptation methods towards practical data challenges in



(a) δ at the input level (usecase: input sparisifier)

(b) δ as an intermediate bottleneck (usecase: feature compressor)

(c) δ at the output level (usecase: prototype generator, domain translator)

Figure 1.1: Overview of $\mathcal{M} + \delta$ framework with δ at different positions serving as different purposes.

vision applications. The following chapters can be divided into two parts according to the type of data challenges they focus on:

Part I. Model adaptation for efficient data representations, which consists

- Chapter 2: We develop a two-stream framework for compressed-domain video recognition, bypassing the need for decoding. We examine the properties of compressed video formats in modern video codecs, and group the compressed modalities into two branches, representing spatial and temporal signals respectively. In transitioning from conventional two-stream networks, we propose a fine-to-coarse distillation approach. This approach enhances the low-resolution and noisy signals with the more robust decoded representations. We show that our network substantially narrows the performance gap from decoded video recognition, outperforming other compress-domain methods. Moreover, it accelerates the inference speed by two orders of magnitudes compared to methods with decoded formats. In this chapter, we demonstrate how to transform a conventional two-stream network M into a compressed-domain network with $\delta = 0$.

- Chapter 3: We introduce a novel Few-Bit VideoQA problem, where only limited number of bits of video information are permitted for accomplishing VideoQA tasks. To address this, we propose a task-specific feature compression approach that learns to extract task-specific tiny features of very few bits. Integrating a light-weight feature compressor into a pre-trained video-and-language model enables its adaptation into a low-bit compressor, facilitating the extraction of tiny features along with task accomplishment. Our experiments demonstrate that with as low as 10-bit of information, we can successfully perform VideoQA tasks with only a minor decrease in accuracy across VideoQA benchmarks. We provide detailed analysis on the learned low-bit feature, point out its storage efficiency and the privacy advantage and discuss how it opens up new possibilities for on-device or cloud-based data storage and processing. In this chapter, we adapt a video-and-language model for low-bit representation with δ being an intermediate bottleneck module serving as a feature compressor.

- Chapter 4: We demonstrate that videos can be efficiently represented as sparse frames or texts, which are human-interpretable, to accomplish VideoQA tasks. We introduce the concept of sparsified VideoQA problem, and propose a token sparsification approach that learns to drop input tokens, applicable to both visual and textual modalities. By incorporating an input sparsifier at the model's outset, the sparsifier is trained along with model towards task accomplishment. We also propose a multi-gumbel estimator to tackle the non-differentiability of the sampling process. Our approach is evaluated on VideoQA benchmarks, showing a minor drop of performance with very limited number of frames and/or words. We also provide analysis on the sparsified frames and words, highlighting their effectiveness in capturing salient information. In this chapter, we adapt a video-and-language model for sparsifying inputs with δ being an input sparsifier.

Part II. Model adaptation for open environment, which consists

- Chapter 5: We propose a negative envision framework to enhance prototype-based recognition models, enabling it to recognize new classes given few examples and reject outliers. Addressing limitations of threshold-based open recognition approaches, we offer a threshold-free solution that generates negative prototypes to dynamically adjust rejection power. Coupled with this, we propose a new conjugate training strategy that modifies the conventional meta-learning scheme with conjugate sampling and loss. Our approach achieves state-of-the-art results on few-shot recognition benchmarks, in both closed-set and open-set evaluations. We also formulate a more realistic evaluation of recognition models, showing our approach excels at handling new and unseen classes without compromising original class capabilities. In this chapter, we adapt a recognition model for new and unseen classes with δ being a negative prototype generator added to output class prototypes.

- Chapter 6: We develop a language-guided domain generalization approach that adapts a model to new domains by leveraging their textual descriptions. Our approach includes a domain translator module to transforms visual embeddings, and a training scheme with both domain-invariant and domain-specific losses. Furthermore, we introduce the one-to-many domain generalization problem which requires adaptation of a model to a combination of new domains. To address this challenge, we propose a multi-domain generalization learning scheme which learns an additional domain selector to locate the correct target domain. In experiments, we validate our approach on domain generalization benchmarks, evaluating both one-to-one and one-to-many scenarios, and also provide analysis of the quality of translated features. In this chapter, we adapt a recognition model to unseen domains with δ being a domain translator on top of visual embeddings.

Chapter 2: Adapting Model for Compressed Video Domain

Video processing could be a substantial computational bottleneck. A common practice is to first decode a video into a sequence of image frames, then apply other techniques on top of image sequences. In this chapter, we take a different perspective and inspect compressed-domain video formats as an efficient video modality. In this chapter, we examine the properties of these compressed data and seek to seamlessly transform a traditional decoded-domain model $\mathcal{M}$ to compressed-domain, with the special case $\delta = 0$.

2.1 Introduction

Video is nowadays the dominating visual data source in all kinds of application scenarios. The temporal context embedded in videos captures additional information compared to still images. However, the temporal aspect of videos also induces information redundancy and computational burden. Early works [7, 8] on video recognition apply convolutional neural networks (CNNs) directly on RGB frame sequences decoded from video but get limited success, due to the difficulty of extracting discriminative spatiotemporal information from the frames that highly resemble each other. Recent works [9, 10] have demonstrated the effectiveness of two-stream networks where appearance and motion are modeled by separate CNNs and then fused together to obtain the final prediction. Most state-of-the-art works keep RGB frame sequences as the appearance modality, and use optical flow (*OF*) [11] calculated from consecutive RGB frames as the motion modality. Though adding an optical flow stream improves the performance, it is known to be computationally heavy. Some works [12, 13] try to estimate *OF* using CNNs. But to make pixel-level accuracy, most of those networks are heavy. *OF* estimation remains a huge computational bottleneck for two stream frameworks.

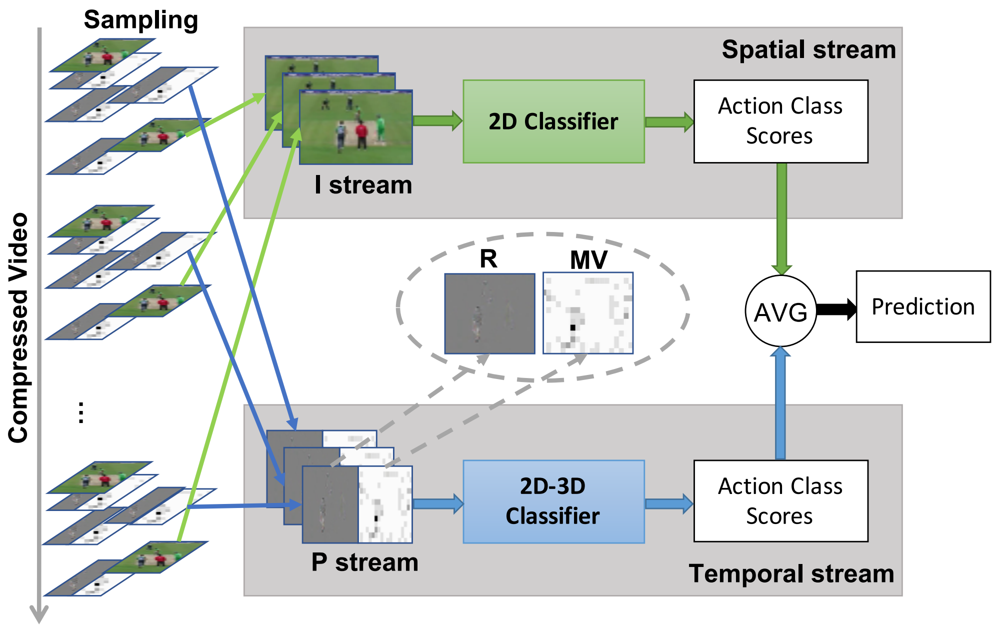


Figure 2.1: Overview of our IP Two-Stream Network (**IP TSN**). We propose a novel compressed domain two-stream network, which exploits the two frame types readily available in compressed videos. The I-stream samples RGB I-frames and process them with a 2D-Net to get still scene information. The P-stream samples the weaker P-frame modality, motion vectors (*MV*) and residual errors (*R*), processed by a 2D-3D Net for effective temporal modeling. The two streams outputs are fused to obtain the final prediction. Our model achieves significant performance gains in both accuracy and efficiency.

To improve video recognition efficiency, some recent works have started to look into modern compressed video encoding formats (MPEG4, H.264, etc) as input. CoViAR [14] and DMC-Net [15] consider MPEG4 [16] videos that contain I-frame (intra-coded frames as RGB images) and P-frame (predictive frames) represented by both Motion Vectors (*MV*) and Residuals (*R*). Both of them follow a multi-stream setting (CoViAR TSN) where *I*, *MV* and *R* are fed into separate CNNs together as the spatial stream, and then fused with another temporal stream (Fig. 2.2a,2.2b). While CoViAR (Fig. 2.2a) still uses high-cost *OF* for the temporal stream, DMC-Net (Fig. 2.2b) resolves the latency by replacing *OF* with fast "Discriminative Motion Cue" (*DMC*) generated from *MV* and *R*. However, CoViAR TSN ignores the encoding structure as well as the shared information between P-frame modalities. The dense P-frames offer stronger temporal information that the current network design hardly captures. Also separate networks for *MV*, *R* and *DMC* could be redundant as *MV* and *R* already share motion information, and yet insufficient as *MV* and *R* are both weak modalities for classification.

In addition, DMC-Net reconstructs full-resolution *OF*-like *DMC* to enrich discriminative information. But several works [17, 18] have demonstrated that the End-Point-Error (EPE) accuracy of *OF* does not guarantee recognition accuracy, which implies that pixel-wise reconstruction of flow does not necessarily benefit the performance. Our insight is that we can directly guide *MV* and *R* to mimic *OF* without any explicit flow reconstruction. We find through experiment in Sec. 2.4.4 that feature level supervision is stronger in enhancing the weak P-frame modalities.

Based on the above observations of existing methods, we propose a new compressed-domain two-stream network, **IP TSN**, depicted in Fig. 2.1, where the spatial stream (I-stream) is modeled by *I* only and the temporal stream (P-stream) is modeled by a unified network for *MV* and *R*. Based on the nature of video encoding format which consists of detailed RGB I-frame and weaker P-frame, we use a strong 2D CNN in the spatial stream to extract still scene information from sparse I-frames, and a light 2D-3D CNN in the temporal stream for better spatio-temporal modeling from P-frames. We also propose to exploit high-level feature supervision from *OF*, during training time only, to better extract motion information from *MV* and *R*.

We compare with other compressed-domain methods on two video understanding tasks. For action recognition, we are able to reduce the total inference GFLOPS by 20% while increasing the accuracy by $\sim$ 7% on HMDB-51. For action detection, our flow distillation achieves a 3.6% gain on VIRAT.

In a nutshell, in this chapter, we consider $\mathcal{M}$ being a decoded-domain two-stream video recognition model and seeks to adapt it to compressed-domain. We accomplish this with $\delta = 0$ and uses generalized distillation techniques to distill $\mathcal{M}$ into a new compressed-domain model. With the supervision of high-level optical flow (*OF*) features during training time only, we are able to replace *OF* with *MV* and *R*, and speed up the temporal stream by more than 60 times compared to using optical flow while achieving similar or better accuracy. Our approach is able to achieve higher accuracy than other compressed-domain methods by large margins while improving the efficiency significantlyon both action recognition and action detection benchmarks. We are able to make large improvements over current compressed-domain methods and get significantly closer to the upper-bound performance using conventional decoded videos.

2.2 Related Works

Two Stream Networks for Video Action Recognition. Two stream networks are first proposed by [10, 19] where separate 2D CNNs are used for RGB frames and *OF* frames, and then fused for final prediction. Later, 3D CNNs [20, 8, 21] are shown to perform better in spatiotemporal modeling. With many large-scale video datasets [22, 23, 24, 25] coming out, 3D CNNs are able to get high accuracies when incorporated into a two-stream framework. However, 3D CNNs are computationally heavy and there are some efforts like [26, 27], trying to alleviate 3D computations and get comparable or even better performances. [28] and [26] partially insert 3D layers into the network arguing that temporal modeling is only needed at certain stages.

All the two-stream networks discuss above in decoded video requires additional calculation of *OF* and takes the temporally-aligned RGB and *OF* as inputs. In contrast, our IP TSN simply use two types of encoded frames as inputs that are readily available in the video and not temporally aligned. Also, given the nature of encoding structure, we use non-symmetric networks for I- and P-streams.

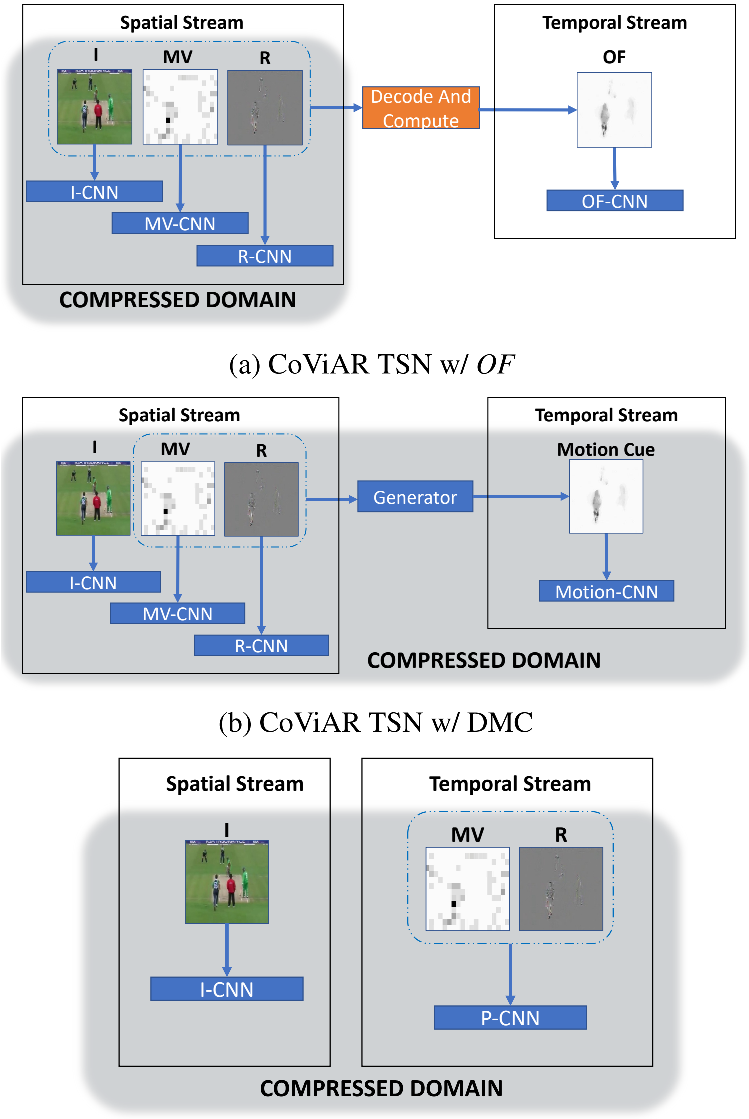
Compressed Video Action Recognition. Recent works have started to look into compressed video domain and showed that compressed video modalities contain rich information that can be exploited quickly using light-weight networks. [29] utilizes motion vector as a cheap alternative to optical flow, but it still needs the full decoded RGB stream in the traditional two stream setting. CoViAR [14] proposes to exploit the compressed video modalities as alternatives to decoded RGB frames. As shown in Fig. 2.2a, it uses three separate networks for *I*, *MV* and *R* as the spatial streams. But to get comparable performance with decoded-domain methods, the full CoViAR TSN framework uses the expensive *OF* stream computed in the decoded space as the temporal stream, which diminishes its speed superiority. To solve this issue, DMC-Net [15] proposes to generate fast *OF*-like discriminative motion cues (DMC) directly from *MV* and *R* to replace the *OF* stream in the CoViAR TSN, as illustrated in Fig. 2.2b.

However, the current CoViAR TSN setting fails to consider the embedded encoding structure and the inter-relations among these modalities. For example, shared motion boundaries in *MV* and *R*; different I- and P-frame densities. The brute way of applying similar networks for each modality separately hence could be suboptimal and redundant, especially for the weak P-frame modalities. From the above observations, we want to maximize the efficiency by searching for the best utilization of the compressed-domain modalities. We propose a novel compressed-domain framework that is designed towards the encoding structure to better exploit compressed-domain modalities.

Generalized Distillation. Knowledge distillation was first proposed by [30] as a concept of transferring knowledge from a high-performance complex model (teacher) to a simple model (student) through the supervision of complex model soft predictions. Recent works [31, 32, 33] apply the concept along with privileged information [34] in cross-modality transfer learning and show promising results. [35, 36] propose to use knowledge distilled from the optical flow to get motion representations directly from RGB inputs. [29, 37] applies similar techniques but on the motion vectors. All these works either consider only one student modality, or multiple student modalities independently. Our work takes optical flow as the teacher to enhance the P-stream in a similar spirit. However, we consider a new student modality, P-frame, which by itself is multi-modal since it consists of both *MV* and *R*. Our insight is that *MV* and *R* contain complementary motion information that are well aligned, and hence can be merged into one stream and supervised by *OF*.

2.3 Approach

In this section we first define the compressed video notations. Then we present our IP TSN framework for compressed video recognition. Finally, we introduce the distillation technique we use to enhance P-stream.



(a) CoViAR TSN w/ *OF*

(b) CoViAR TSN w/ DMC

(c) **IP TSN [ours]**

Figure 2.2: Compressed-domain video recognition frameworks. (a)(b) Existing compressed-domain methods follow multi-stream CoViAR TSN framework, where (a) CoViAR[14] requires decoding and computing *OF* for the temporal stream; and (b) DMC-Net[15] generates Discriminative Motion Cue (DMC) to replace *OF*. (c) Our **IP TSN** instead takes directly *I* and *MR* (*MV* +*R*) as the spatial and temporal stream inputs respectively without any form of flow computation.

2.3.1 Compressed Video Formats and Notations

Following CoViAR [14] and DMC-Net [15], we consider MPEG-4 Part2 [16] encoded videos with Group of Structure (GOP) size 12. Starting with the first frame, I-frames appear every 12 frames, with 11 P-frames filling in between. Each I-frame is a full resolution RGB image (*I*). Each

P-frame is represented by a Motion Vector (*MV*) computed from 16×16 macroblock displacement from the previous frame and a Residual (*R*) computed from the RGB difference between the original image and motion compensated image. Both *I* and *R* are three channel (RGB) images of the same resolution as in the original video, while *MV* is two channel of horizontal and vertical displacements that has 16× smaller resolution in effect. In visualization, *MV* are blocky images where each 16×16 block is filled with the same value of its displacement. Additionally, we denote the optical flow as *OF*, which is the pixel displacement calculated from RGB frames. In our experiment, we use TV-L1 [11] to extract optical flows. We use the FFmpeg [38] based implementation provided by CoViAR to extract *I*, *MV* and *R* from MPEG4 videos.

2.3.2 IP TSN for Compressed Video Recognition

Existing compressed domain methods (Fig. 2.2a,2.2b) create multi-streams for compressed-domain modalities to replace RGB and fuse with another temporal stream. As *MV* is computed from block displacement and hence of low resolution, and both *MV* and *R* contain a lot of noises, prior works use accumulation or pixel-level *OF* reconstruction to enrich the discriminative power for these weak modalities. The limitation with their framework is that they ignore the encoding structure and the different information embedded in different frame types. While the sparse I-frames offers rich still scene information, the dense P-frames are stronger for temporal modeling. Also *MV* and *R* are well-aligned and share motion information, which implies separate modeling is not optimal.

To better utilize the compressed-domain modalities, we propose a new non-symmetric framework, **IP TSN**, illustrated in Fig. 2.2c, which is composed of an I-stream which uses *I* for appearance modeling, and a P-stream which fuses *MV* and *R* for motion modeling. Final prediction is computed from late fusion of these two streams. We split a video into different frame types and feed them into separate networks for the spatial and temporal modeling purposes. Following TSN style [9] sampling as in DMC-Net and CoViAR, we split the video into N segments sample one I-frame and one P-frame each segment, as is shown in Fig. 2.1.

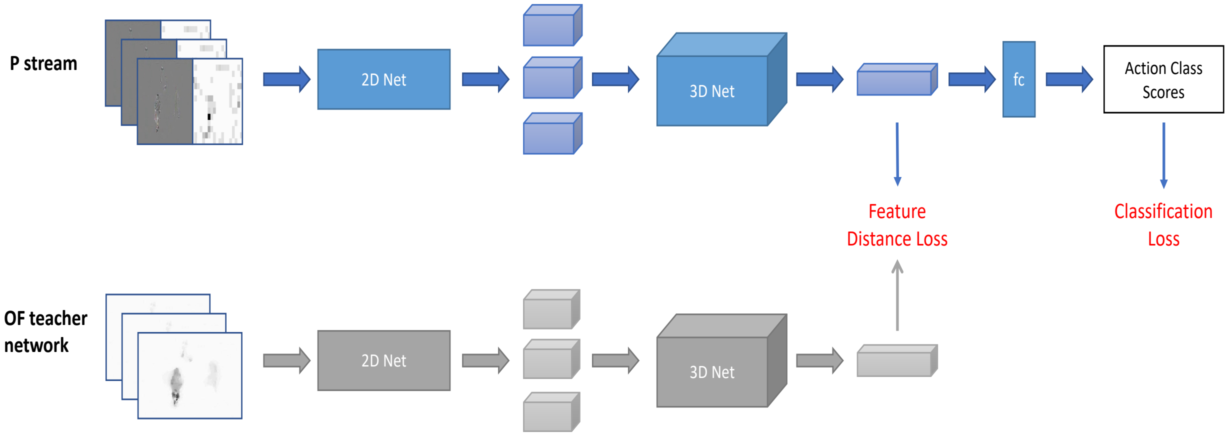


Figure 2.3: Details of our P-stream network. We use a 2D-3D Res18 that partially inflates a ResNet18 [39]. The P-stream takes as input a clip of stacked *MV* and *R* under supervision of ground truth action labels. Furthermore, during training, we exploit a pre-trained *OF* net as teacher to distill its *OF* based features. At inference time, only the P-stream (blue parts) remains.

Based on the properties of the compressed video, we propose to use different network architectures for the I-stream and P-stream. Since I-frames have fine details, we use a heavy 2D CNN, ResNet152, as is used in [14] to extract still scene information. On the other hand, *MV* and *R* of P-frame are weaker and noisy modalities that contain rich motion information. Recent works [28, 26] have shown 3D CNNs at the top layers help extract high level motion features. Hence we use a combine 2D-3D Net for efficient motion modeling. Fig. 2.3 illustrates the details of P-stream. For better comparison and efficiency, we use a 2D-3D ResNet18 where the later half of the layers are inflated into 3D. Details are given in Sec. 2.4, where we also study how the architectures of I- and P-streams affect the performance.

2.3.3 P-frame Enhancement with Flow Feature

OF is computed from pixel movement between consecutive frames and hence is a full-resolution motion representation of fine details compared to *MV* and *R*. It is widely used as the input modality of temporal stream and is proved to be effective [9, 20]. DMC-Net proposes to reconstruct *OF*-like motion cues at full resolution from *MV* and *R*. Recent works [40, 41] demonstrate that the discriminative information embedded in *OF* is more important than the pixel-level accuracy. As a result, we propose to use high-level *OF* feature as supervision to better guide *MV* and *R*.

As shown in Fig. 2.3, at training time, the P-stream network is trained with a standard cross-entropy loss using the ground truth class action label as well as the the feature output of a pre-trained *OF* network as supervision for knowledge distillation. *OF* network is pretrained on the same dataset with classification loss. Then the *OF* network weights are frozen when training the P-stream network. At test time, the *OF* network is discarded and only the P-stream network (blue part) remains. Note that compared to DMC-Net, we do not need extra layers besides the motion stream feature extractor, so our method is more efficient. Also the flow distillation method is generic and can be applied to any stronger classifiers to boost the performance.

We train our P-stream using the combination of the classification loss and the loss between *OF* and P-stream features:

$$\mathcal{L}_p = \mathcal{L}_{cls} + \lambda \mathcal{L}_d(f_p, f_{OF}) \tag{2.1}$$

where $\mathcal{L}_{cls}$ is the cross-entropy loss of P-stream, and $\mathcal{L}_d$ is the Euclidean distance between P-stream feature f_p and *OF* feature f_{OF}. λ is the feature loss weight. We experiment on different choices of L_d function and its weight λ, and finally choose $L1$ loss with $\lambda = 50$ which gives the best balance between two losses. Note that in general, we could have

$$\mathcal{L}_p = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_d(f_p, f_{OF}) + \lambda_2 \mathcal{L}_s(logit_p, logit_{OF}) \tag{2.2}$$

where $\mathcal{L}_s$ the soft label cross-entropy loss as is widely used in knowledge distillation. But we find in Section 2.4.4 that it actually downgrades the performance. Our insight to this is that *MV* and *R* can capture different properties of the video stream from *OF*. Also, *R* has some object boundary information that *OF* does not capture but are useful for recognition. Hence we do not want to simply reproduce the classification output of *OF*. Instead, we only want *OF* to help extract more useful motion information from *MV* and *R*.

2.4 Experiments: Action Recognition

In this section we describe the implementation details and present the performance of our framework for the action recognition task. We will show that our flow-distilled IP TSN achieves both high accuracy and high efficiency.

2.4.1 Datasets and Metrics

UCF-101[42], which contains 13320 videos from 101 action categories. 3 training/testing splits are offered. The performance result is averaged over splits.
HMDB-51[43], which contains 6766 videos from 51 action categories. 3 training/testing splits are offered. The performance result is averaged over splits.
Kinetics [20], which contains about 246k training videos and 20k validation videos from 400 action categories.
Metrics: We report top-1 class prediction accuracy. For efficiency measurement, we report GFLOPs and Videos Per Second [28] using 16-frame clip unless specified.

2.4.2 Implementation

We here detail our training and testing parameters and procedures.

Training

We follow the same setting as CoViAR and DMC-Net where videos are resized to 340×256 and we use 224×224 random cropping and random flipping for data augmentation. For I-stream, we employ a ResNet152 classifier and train it with cross-entropy classification loss exactly as CoViAR and DMC-Net. The P-stream network is a combination of 2D and 3D CNNs. CoViAR and DMC-Net use ResNet18 [39] for P-frame modalities. For better comparison and efficiency, we use the same backbone for P-stream. We take the layers of ResNet18 up to conv3x layer for 2D Net, where the output feature has 28×28 spatial dimension. For 3D Net, we take 3D-ResNet18 [28, 8] from conv4x until the end. In effect, we replace the later half of the ResNet18 layers with its 3D version. Hence our P-stream is a 2D-3D network.

In practice, we first train a 2D ResNet18 taking the stacked *MV* and *R* (MR) as input with classification loss only. We similarly train our *OF* teacher network as a 2D ResNet18 using *OF* as input with cross-entropy loss. Then, we perform a first round of distillation of the *OF* features. Then, we initialize the 2D-3D Res18 with the MR 2D-ResNet18 weights, where the 2D part simply copies the weights and the 3D part inflate the weights from the corresponding layers and we initialize similarly the *OF* teacher 2D-3D Res18 with copy and inflation from the *OF* 2D ResNet18 network weights. We then train both 2D-3D networks with cross-entropy loss. Finally, the P-stream network is initialized with the MR 2D-3D Res18 and trained with both cross-entropy loss and L1-loss with the average-pooled features of the frozen teacher network as discussed in Sec. 2.3.3.

Inference

During inference, we uniformly sample 16 I-frames and P-frames with a center crop of size 224×224 from each modality, which is less than CoViAR and DMC-Net that use 25-frame inputs. I-stream prediction score is averaged over 16 I-frames. The 16 P-frames form one clip input and is fed into the network to get one P-stream class scores. The final prediction is calculated through late fusion of the I-stream and P-stream scores.

2.4.3 Flow-Distilled IP TSN Performance

Fig. 2.4 shows the accuracy speed trade-off with our method compared to using *OF*-stream instead of P-stream and DMC-Net and CoViAR. All the numbers are reported on 16-frame input each with one central crop. Our method clearly exhibits the best trade-off with high speed and high accuracy. We also additionally compare with decoded video methods on the left side of Tab. 2.1 and give detailed timings. Since our P-stream network requires clip input, we measure the inference time per video [28]. Speed measurements are performed using one NVIDIA TITAN

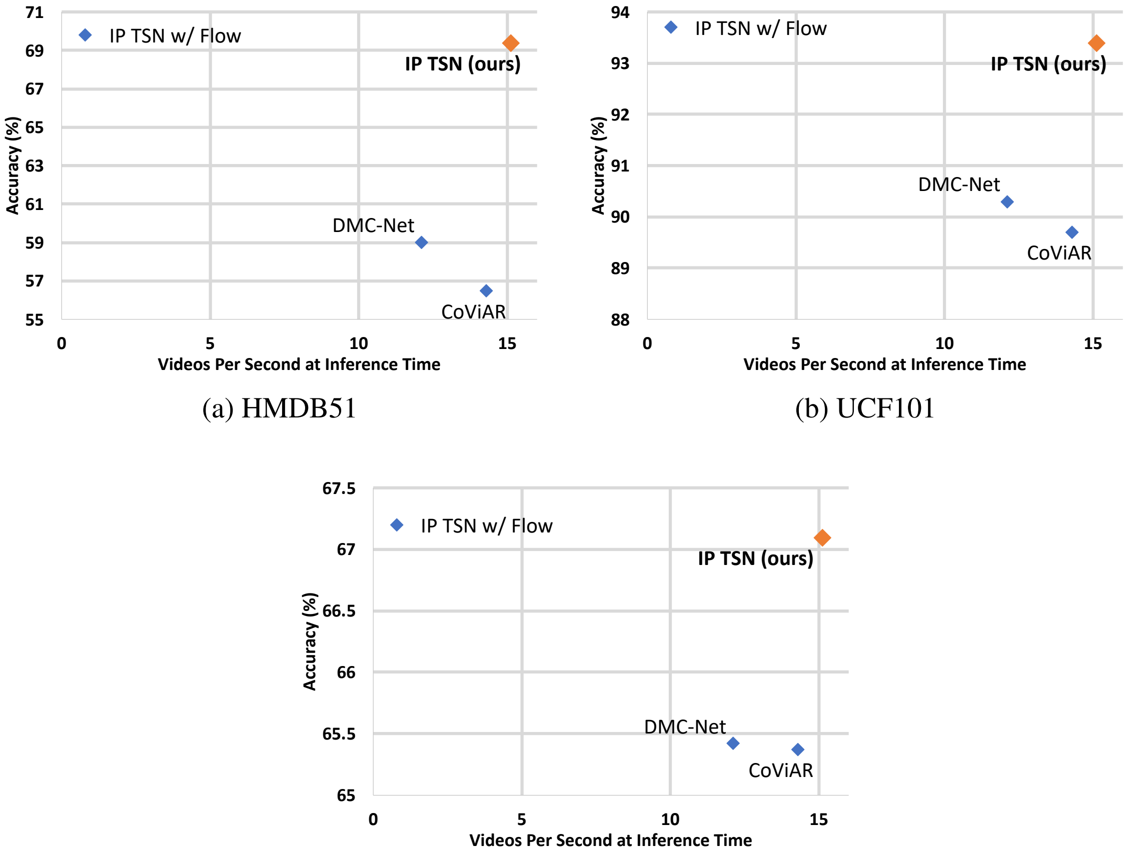
Table 2.1: Comparison of efficiency and accuracy among two stream methods. All compressed-domain networks take 16-frame input with one crop. TSN [9] and I3D [20] are test with their default settings.

		Decoded Video (RGB+Flow)		Compressed Video (I, *MV*, R)					*OF*	
		TSN [9]	I3D [20]	CoViAR [14]	DMC-Net [15]	**I-stream** (ours)	**P-stream** (ours)	**IP TSN** (ours)	*OF*	I + *OF*
Time (ms)	Preprocess	1200	1200	14.7	14.7	3.3	11.4	14.7	1200	1200
	Spatial	12.2	45.5	55.3	55.3	43.1	-	43.1	-	43.1
	Temporal	12.2	33.7	-	12.8	-	8.5	8.5	8.5	8.5
	Total	1224.4	1279.2	70	82.8	46.4	8.5	66.3	1208.5	1251.6
VPS		0.8	0.8	14.3	12.1	21.5	**50.2**	**15.1**	0.8	0.8
GFLOPs		33 + 33	436 + 401	243.4	275.6	185.6	**33.9**	**219.5**	185.6	33.9
Acc. (%)	HMDB-51	68.5	80.7	56.5	59.0	51.5	61.8	**69.1**	60.0	69.2
	UCF-101	94.0	98.0	89.7	90.3	86.7	87.1	**93.4**	84.9	93.7
	Kinetics	-	71.6	65.4	65.4	62.4	33.8	**67.1**	28.1	67.2

RTX GPU with CUDA 10 and cuDNN 7.5.0. To measure the inference speed for CoViAR and DMC-Net, as well as I-stream, we take a batch of 16 frames and feed into the 2D Nets. To measure the inference speed for P-stream, we take one clip of 16 frames as one batch. Each frame is centered-cropped without any flipping. The reported results of TSN [9] and I3D [20] are based on their default settings, respectively. As Tab. 2.1 shows, our P-stream (50.2VPS) runs more than 60x faster than *OF* (0.8VPS). The full IP TSN runs about 18x faster than using *OF* instead. Compared to the state-of-the-art compressed-domain methods, our IP TSN significantly improves the accuracy while decreasing the GLOPs and increasing VPS both by roughly 20%.

2.4.4 Ablation Study

Contributions from IP TSN and Flow Distillation. Since we propose both a new framework and a new compressed-domain modality enhancement method, we evaluate how each component contributes to the accuracy and efficiency in Tab. 2.2. We compare our proposed framework, IP TSN(Fig. 2.2c), against the existing compress-domain framework, CoViAR TSN(Fig. 2.2a,2.2b). Under each framework, we ablate our flow distillation method and compare with DMC generator. Specifically, **MR(raw)** refers to training *MV-R* with cross-entropy loss only. **MR w/ DMC** uses DMC generator to reconstruct *OF*-like motion cues from MR. **Distilled MR** refers to our work that uses flow feature as extra supervision in addition to ground truth labels. (1) We can observe the



(a) HMDB51

(b) UCF101

(c) Kinetics

Figure 2.4: Accuracy vs Speed at inference time. Input of each network is 16-frames with one crop. Results on HMDB51 and UCF101 are averaged over 3 splits. Our IP TSN achieves similar accuracy compared to IP TSN w/ Flow while being about 20× faster, and as fast as other compressed-domain video recognition methods that have much lower accuracy.

benefit of IP TSN framework by comparing each row under CoViAR TSN and IP TSN respectively. Our IP TSN significantly improves the accuracy with slight speedups. This validates that dense P-frame needs strong temporal modeling; (2) We can see the benefit of flow feature distillation by comparing the rows within each framework. Flow distillation significantly enhance the weak P-frame modalities (row 1. vs 3., 4. vs 6.). Compared to DMC, flow distillation brings substantial speed improvement and performance gains especially under our IP TSN setting.

Flow Distillation Loss. We study commonly used variants in knowledge distillation: L1 loss on last averaged pooled feature (the adopted choice), soft label cross-entropy loss with temperature = 8, and the combination of both which can be exploited jointly in the general loss function intro-

Table 2.2: Ablation on *OF* feature supervision and IP TSN framework. Result is averaged over three splits for HMDB51 and UCF101. Both our IP TSN framework and *OF* distillation brings substantial improvement on accuracy and speed.

	Motion(MR) branch	Distill?	2D-3D?	VPS	HMDB51	UCF101
CoViAR TSN (I+*MV* +R+MR)	1.MR(raw)	×	×	14.3	59.1	90.2
	2.MR w/DMC	×	×	12.1	61.8	90.9
	3.**Distilled MR[ours]**	✓	×	14.3	**62.1**	**91.1**
IP TSN[ours] (I+MR)	4.MR(raw)	×	✓	15.1	64.9	92.5
	5.MR w/DMC	×	✓	12.9	68.0	92.9
	6.**Distilled MR[ours]**	✓	✓	**15.1**	**69.1**	**93.4**

Table 2.3: Top-1 Accuracy on HMDB51 with different training losses and type of teacher flow. Feature distance gives the best result, as it probably induces a better complementarity with the cross-entropy loss with labels. The result is averaged over splits.

Flow	L1 loss	Soft label loss	IP TSN	*I* +*OF*
Type	weight λ_2	weight λ_2	Acc.	Acc.
TV-L1	0	50	66.0	69.2
	25	50	67.1	
	25	25	68	
	50	50	68	
	50	**0**	**69.1**	
PWC	50	0	67.0	66.3

duced in eq. (2.2). As shown in Tab. 2.3, the soft label cross-entropy alone performs worst and actually hurts the performance of the L1 loss when combined with it. This may indicate that the L1 loss on features induces a better complementarity with the cross-entropy loss on the ground truth labels than the soft-label cross-entropy loss. Hence we use only the last pooled feature as supervision with L1 loss in all other experiments. We also experiment on different types of teacher flow in Tab. 2.3. TV-L1 flow is stronger than PWC flow [12]. But P-stream can always get similar or even better performance with our distillation technique, no matter what kind of teacher flow is used.

Number of Streams. We argue that CoViAR TSN requires multi-streams mainly because it doesn't fully exploit P-frame modalities. In Tab. 2.4, with our 2D3D architecture, adding extra *MV* and *R* like CoViAR TSN doesn't necessarily help. This shows our framework is fully utilizing the P-frame modalities.

Non-symmetric Architecture Search. Based on the encoding structure, we propose a non-symmetric IP TSN. For P-stream architecture search, we report in Tab. 2.5 how the inflation position affects both the performance and computational cost. Note that 'conv1.x' is equivalent to 3D Res18 and 'conv5.x' is equivalent to 2D Res18 (same as CoViAR TSN with Distilled MR in Tab. 2.2). We choose to inflate at 'conv3.x' as it gives the best accuracy-efficiency trade-off. For I-stream architecture search, Tab. 2.6 gives the result on different I-stream architecture and input length, where we fix the P-stream architecture as 2D-3D Net we have proposed. We can see that 2D-3D Res18 is even worse than 2D Res18 on I-stream. One reason for this is that I-frame are rare in short videos hence hardly provide any useful spatiotemporal information. However, increasing the clip length improves the performance substantially. We get our best results with ResNet152. The architecture search results for I- and P-stream validates our non-symmetric design concept. We also report in Fig. 2.5 the accuracy as a function of GFLOPs obtained by our method and the DMC-Net approach using different I-stream network architectures: ResNet18, ResNet34, and ResNet152, and how performance changes with either 8 or 16 frames as input for each video. We see that at a same number of GFLOPs our method always outperforms DMC-Net.

***OF* vs P-stream** We report on the right side of Tab. 2.1 the performance of the spatial I-stream, and of the temporal stream either *OF* or our P-stream or separately. We also report the performance of two-streams network with either P-stream or *OF* as the motion stream. P-stream is roughly *60*× faster than *OF* while enjoying a higher accuracy. When combined with I-stream, P-stream is $<$ 0.5% lower than *OF*-stream. We can observe that the *OF* stream pre-processing time is prohibitive and does not come at any accuracy benefit compared to our P-stream.

Table 2.4: Top-1 Accuracy on HMDB51 with different number of streams. CoViAR TSN (2D Res18 based) requires multiple streams to get the best performance, which does not necessarily help in our setting (2D3D Res18 based). Our IP TSN already fully utilizes the P-frame modalities.

Stream Backbone	*MV*	*R*	Distilled MR	Acc.
2D Res18	✓	✓	✓	62.1
			✓	60.9
2D3D Res18	✓	✓	✓	68.2
			✓	**69.1[ours]**

Table 2.5: Accuracy on HMDB51 split3 and GLOPs at different inflation layer positions. We choose to inflate at conv3.x as it gives the best accuracy-complexity tradeoff.

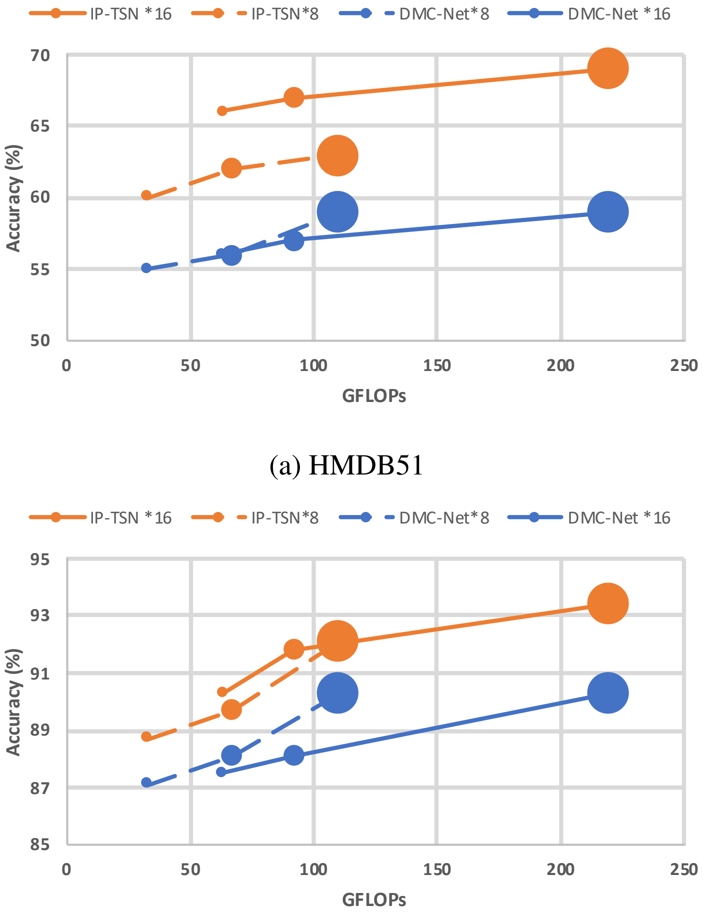
	Top 1 Acc	GFLOPs
conv1.x	61.4	86
conv2.x	70.2	50
conv3.x	**69.6**	**34**
conv4.x	66.5	35
conv5.x	62.1	29

2.4.5 Comparison with State-of-the-Art

We finally compare our IP two-stream network with current state-of-the-art methods in Tab. 2.7. Our method outperforms all other compressed-domain methods significantly by 7% on HMDB51, 2.5% on UCF101 and 1.7% on Kinetics. We have brought the compressed-domain methods performance significantly closer to the decoded video methods. Without pre-training on large-scale video datasets, we are competitive with many decoded methods, apart from the most expensive two-stream methods using heavy architectures such as I3D. For example, our proposed IP TSN achieves comparable or better performance compared to ECO and ECO Lite [28], a recently proposed decoded-video based efficient video understanding model, under the same amount of input frames (16). Note that as is shown in Tab. 2.1, all the decoded two stream based networks need to compute optical flow, which is often the main bottleneck of real time processing. However, we are able to greatly speed up the process while maintaining a comparable performance.

Table 2.6: Accuracy on HMDB51 using different I-frame backbones and input length. The result is averaged across 3 splits.

I frame	seg	I-stream	P-stream	VPS	HMDB51
backbone		Acc.	Acc.		Acc.
2D-3D Res18	16	40.5	61.9	25.1	66.8
ResNet 18	8	44.6	53.1	57.7	60.1
	16	45	61.9	52.1	66.2
ResNet 152	8	51.4	53.1	24	63.6
	16	51.8	61.9	15.1	**69.1**



(a) HMDB51

(b) UCF101

Figure 2.5: Accuracy vs GFLOPs at inference time. Results on are averaged over 3 splits. This shows how testing segments and I-stream backbone affects the result. Three node sizes corresponds to ResNet18, 34, 152 respectively. Solid and dashed line refer to testing with 16-frames and 8-frames respectively. Orange nodes are result on IP-TSN, and blue nodes are on DMC-Net.

2.4.6 Additional Ablation Study

Modalities Both CoViAR and DMC-Net include separate networks for *I*,*MV*,*R* as spatial stream, and DMC-Net further includes a DMC network as temporal stream. Our ***IP TSN*** instead only use I frames as the spatial stream, and use stacked *MV* and *R* as temporal stream under the guidance of OF, which achieves better accuracy as well as efficiency. We also evaluate the result of adding individual streams of *MV* and *R* to the spatial stream, like what is proposed in CoViAR and DMC-Net. As is demonstrated in Table 2.8 . $CoViAR + P$ refers to fusing ResNet-18 based *MV* and

Table 2.7: Accuracy on HMDB-51, UCF-101 and Kinetics for both state-of-the-art compressed-domain and decoded-domain methods.

	HMDB-51	UCF-101	Kinetics
Compressed video based methods			
E*MV*-CNN [29]	51.2 (split1)	86.4	-
DT*MV*-CNN [37]	55.3	87.5	-
CoViAR [14]	59.1	90.4	65.4
DMC-Net [15]	61.8	90.9	65.4
CoViAR TSN w/ distilled MR [ours]	**62.1**	**91.1**	-
IP TSN [ours]	**69.1**	**93.4**	**67.1**
Decoded video based methods *(RGB only)*			
Frame-level classification			
ResNet-50 [44]	48.9	82.3	61.3
ResNet-152 [44]	46.7	83.4	63.0
Motion representation learning			
ActionFlowNet [18]	56.4	83.9	-
PWC-Net (ResNet-18) + CoViAR [12]	62.2	90.6	-
TVNet [45]	71.0	94.5	-
Efficient spatio-temporal modeling			
ECO Lite$_{16f}$ [28]	68.2	91.6	64.4
ECO$_{16f}$ [28]	68.5	92.8	69.0
ECO$_{En}$ [28]	72.4	94.8	70.0
Decoded video based Two Stream methods *(RGB + Flow)*			
Two-stream [19]	59.4	88.0	-
Two-Stream fusion [10]	65.4	92.5	-
I3D [20]	80.7	98.0	74.2
R(2+1)D [26]	78.7	97.3	75.4

R. This is equivalent to fusing CoViAR with our P-stream. $IP + (MV + R)_{2d-3d}$ refers to fusing 2D-3D ResNet-18 based *MV* and *R* with our ***IP TSN***. Following DMC-Net and CoViAR, we train *MV* and *R* streams with cross-entropy loss and fuse all the scores for final prediction. We can see from the result that adding *MV* and *R* does not improve the accuracy. This validates that our ***IP TSN*** already exploits the necessary spatial and temporal information repectively. *MV* and *R* do not offer new information while possibly adding noises. ***IP TSN*** performs at the best accuracy as well as efficiency.

Alternative 3D Sub-networks Besides inflation from 2D to 3D, inserting temporal convolution after spatial convolution to construct (2+1)D networks is also a promising option for spatial-temporal modeling [46]. We also evaluate using (2+1)D network as 3D sub-network on HMDB-51. We first train the (2+1)D teacher network using optical flow as input for P stream. The average accuracy

Table 2.8: Top 1 Accuracy on HMDB-51 by adding *MV* and *R* streams to ***IP TSN***. Network architectures are specified for each modality . Additional modalities do not improve the accuracy. The result is averaged over splits.

		IP TSN(ours)	CoViAR+P	IP+(MV+R)$_{\text{2d-3d}}$
Spatial	I	Res152	Res152	Res152
	MV	-	Res18	2D-3D Res18
	R	-	Res18	2D-3D Res18
Temporal	MR	2D-3D Res18	2D-3D Res18	2D-3D Res18
Top 1 Acc.		**69.1**	66.1	68.9

on three splits is 69.6%, which is slightly lower than the performance (69.8%) of the inflated 3D version. We then train **(2+1)D IP TSN**, under the guidance of the teacher network using optical flow. **(2+1)D IP TSN** achieves an average accuracy of 69.0%, which is slightly lower than the performance (69.1%) of the inflated 3D version. Therefore, we use inflated 3D for ***IP TSN*** in all the experiments.

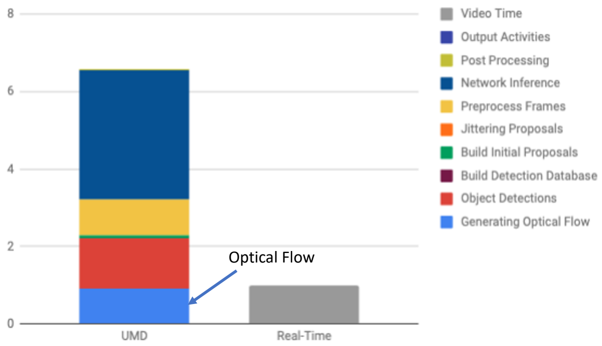
2.5 Experiments: Action Detection

As discussed in Sec. 2.3.3, flow distillation can apply to any *OF*-based classifiers to get significant speedup. In this section, we evaluate our flow distillation method on action detection task.

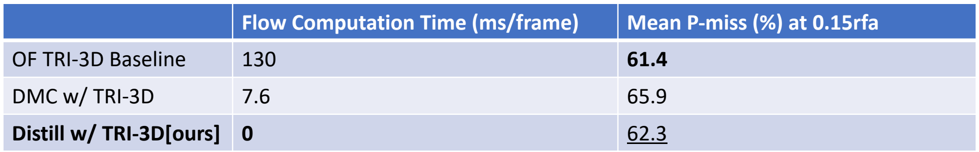
Datasets and Metrics. We test on VIRAT[47], a public action detection benchmark in long surveillance videos. It contains 64 training and 54 validation videos from 18 action categories. We report mean probability of missing detection (p-miss) at fixed rates of false alarm per minute (RFA) [48] on the validation set. The lower the mean p-miss value, the better the performance.

Implementation. We integrate our flow distillation method into an OF-based TRI-3D pipeline proposed by [49], and replace their *OF* inputs with P-frame modalities. Their original *OF* model is used only in the training stage for feature distillation. We follow their proposal generation and preprocessing steps. For each action proposal, our inputs are 64-frame *MV* and *R* of size 224*224 for both training and testing.

Main Results. Fig. 2.6a shows the speed breakdown for the original TRI-3D pipeline, where *OF* computation is one of the speed bottleneck that consumes $\sim$ 20% of the total time. We are able to completely zero-out the *OF* computation time while maintaining the performance about 1% from the original baseline, as is shown in Fig. 2.6b and Fig. 2.6c . Compared to DMC method, our method achieves much better accuracy (3.6% gain) and faster speed.

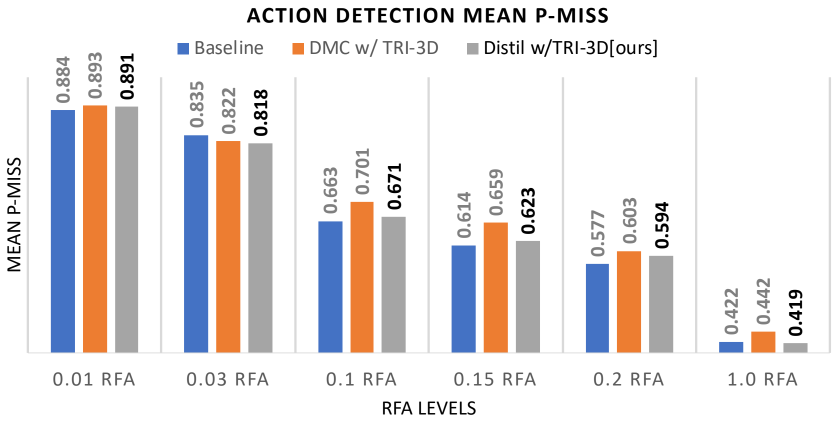


(a) TRI-3D Pipeline Speed Chart



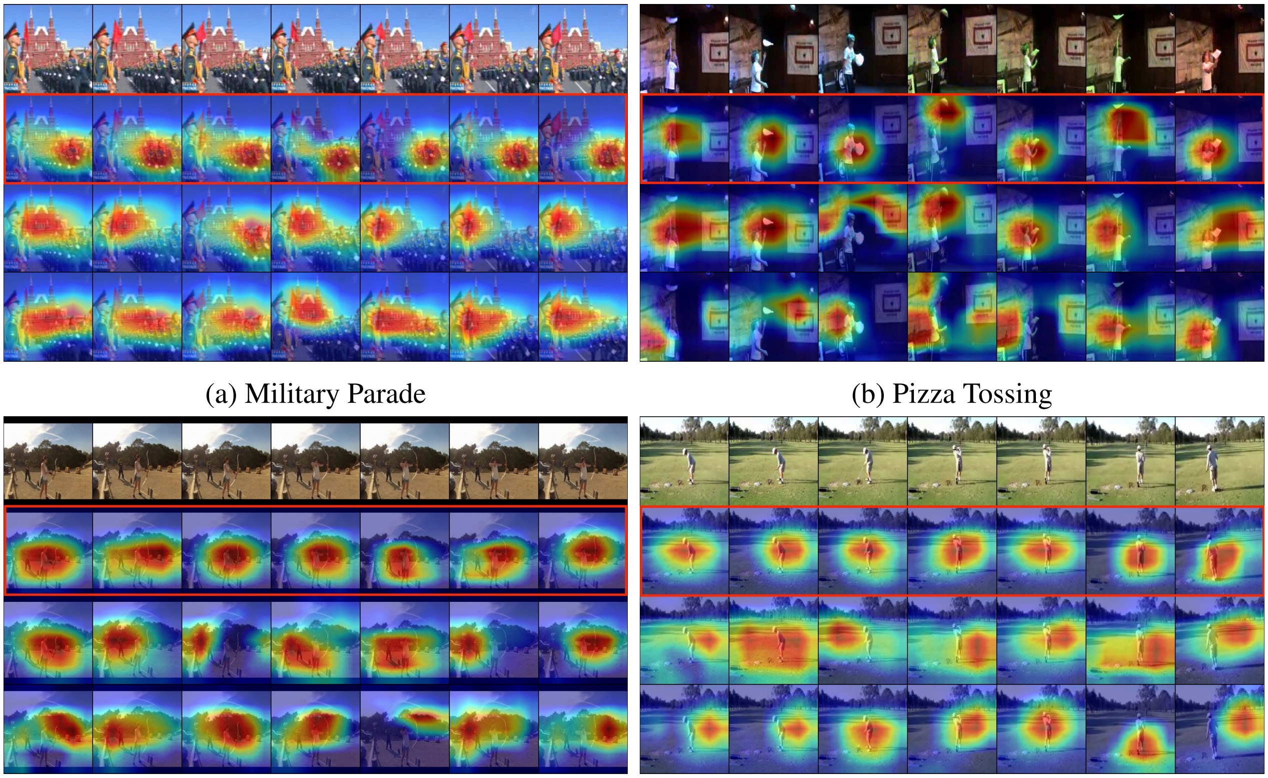
	Flow Computation Time (ms/frame)	Mean P-miss (%) at 0.15rfa
OF TRI-3D Baseline	130	**61.4**
DMC w/ TRI-3D	7.6	65.9
Distill w/ TRI-3D[ours]	**0**	62.3

(b) Speed-Accuracy Tradeoff. We reach the maximum efficiency while outperforming DMC by large margins.



(c) Action Detection Accuracy (Mean P-miss) at different RFA levels.

Figure 2.6: Action detection results on VIRAT. We are able to achieve similar performance with the *OF* baseline with zero flow computation. (light blue in the speed chart)



(a) Military Parade

(b) Pizza Tossing

(c) Archery

(d) Golf Swing

Figure 2.7: Saliency Maps generated by Grad-CAM method [50]. The first row is the original video clip. Row 2 is the map from our flow-distilled model. Row 3 is from OF teacher. Row 4 is from DMC-Net [15]. Our flow-distilled model is able to focus on the action regions precisely, while other two pay false attention which leads to incorrect class prediction.

2.6 Visualizations

We use the Grad-CAM method [50] to generate the saliency maps which localize the most discriminative video regions of the ground truth class. We apply Grad-CAM on our flow-distilled model, OF teacher model as well as DMC-Net with the same ResNet-18 backbone. The maps generated with respect to OF or P-frame modalities are then plotted on corresponding RGB frames for better visualization.

Fig. 2.7 visualizes 7-frame clips from different videos, where our flow-distilled model successfully finds the correct class while DMC-Net and OF teacher fail. Row 1 in each example gives the original video clip; Row 2 is the Class Activation Map generated from flow-distilled model; Row 3 is from OF teacher model; Row 4 is from DMC-Net. We can observe that DMC-Net and OF teacher deviates from the moving areas and pay false attention to other regions, while our flow-distilled model is able to localize action regions precisely. For example, in Military Parade, our flow-distilled method focuses on the moving soldiers but the other two look at the house in the background. This shows that P-frame modality contains rich motion information and our flow-distilled method performs well in enhancing the discriminative power of P-frame.

2.7 Summary

In this chapter, we tackle the problem of adapting decoded-domain video model cM to compressed-domain, where we provide a solution with $\delta = 0$. Specifically, we propose a new IP two-stream framework for compressed video understanding, where the different frames (I and P) of the compressed representation are exploited for different modeling purposes (spatial and motion, respectively). We also propose an efficient P-stream training strategy using an OF feature supervision to train a 2D-3D architecture for the motion stream. Our P-stream is 60× faster than an OF stream with similar accuracy. The overall framework is highly efficient as it can process 15 videos per second at a high level of accuracy, much faster than real time. Evaluation on both action recognition and detection benchmarks validate our method and shows significant improvement over prior compressed domain action recognition method and approach decoded video methods performance. We show in this chapter the potential of efficient data modality and showcase how we can adapt mode cM for the new modality with distillation techniques.

Chapter 3: Adapting Model for Low-Bit Representation

In practical vision applications, it is always a basic but crucial question to ask: "How do we store vision data?", given the substantial sizes of visual signals. In this chapter, we pick video as a typical visual data coming with storage and privacy issues. and seek to provide a machine learning solution from the perspective of data minimization. Specifically, we adapt a deep learning model to learn and extract task-specific low-bit representations that provide unique practical advantages. Figure 3.1 shows a high-level illustration of the framework we'd use, where we instantiate δ as an intermediate bottleneck, serving as a learnable feature compressor.

3.1 Introduction

Video data exemplifies various challenges with machine learning data: It is large and has privacy issues (*e.g.* faces and license plates). For example, an autonomous robot driving around may collect gigabytes of video data every hour, quickly filling up all available storage with potentially privacy-sensitive information. Yet, video is useful for a variety of computer vision applications, *e.g.* action detection [52], object tracking [53], and video question answering (VideoQA) [54, 55]. With increasing adoption of computer vision applications, and mobile devices, efficiently storing video data becomes an increasingly important problem.

VideoQA [54, 55] is a general machine learning task that requires analyzing video content to answer a general question. As such, it contains elements of multiple video problems, such as classification, retrieval, and localization. The questions in VideoQA are not all available beforehand, which means that the model needs to extract general semantic information that is still useful enough to answer a diverse set of questions, such as any question about the task "Human Activities", for example. Concretely, the system may have been trained on queries including "Is the

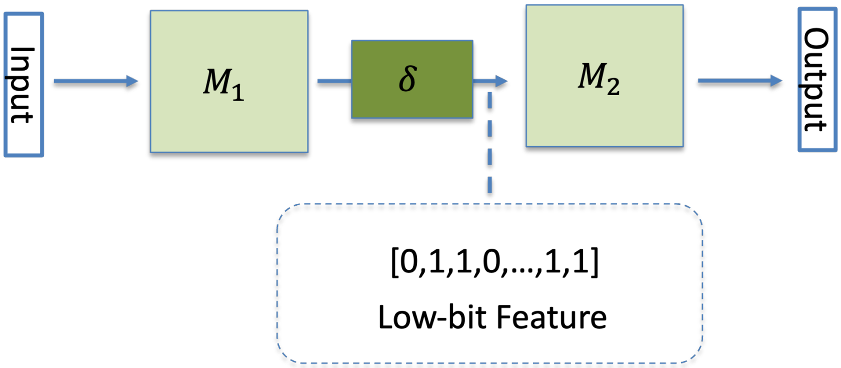


Figure 3.1: In this chapter, δ serves as an intermediate feature compressor to encode task-specific low-bit features that provides storage efficiency and privacy advantages.

person laughing?", and at inference time, the system needs to record and store a video, and may later receive the query "Do they look happy?" which was not seen at training time. In contrast with classification problems where the system could potentially store the answer to all possible queries, a VideoQA system needs to keep enough information to answer any query about the video.

While early video analysis used hand-crafted visual descriptors to represent videos, recent advances in deep neural networks are able to learn highly abstracted features [56, 57, 58]. These features still contain more information than what is actually needed in a specific task. For example, a popular video encoder network ResNet3D [51] extracts a 512-dim floating number feature, which requires 16,384 bits. If the task is to only answer questions about the happiness of a person, such as "do people look happy", we theoretically only need 1-bit of information to accomplish that task by encoding the existence of person + laughing. A single 16,384-bit feature encodes much more information than required to answer this question. Moreover, it is hard to interpret what information is captured by continuous features, and hence hard to guarantee that the features do not contain any sensitive information, which may carry privacy concerns.

Much of recent work on VideoQA focuses on learning stronger vision features, improved architectures, or designing better multi-modal interaction [59, 60, 1, 61, 62]. This chapter instead investigates how many bits are really needed from video in current VideoQA tasks. To this end, we introduce a novel "*Few-Bit VideoQA*" problem, which aims to accomplish VideoQA where only few bits of information from video are allowed. To our knowledge, the study of few-bit features is

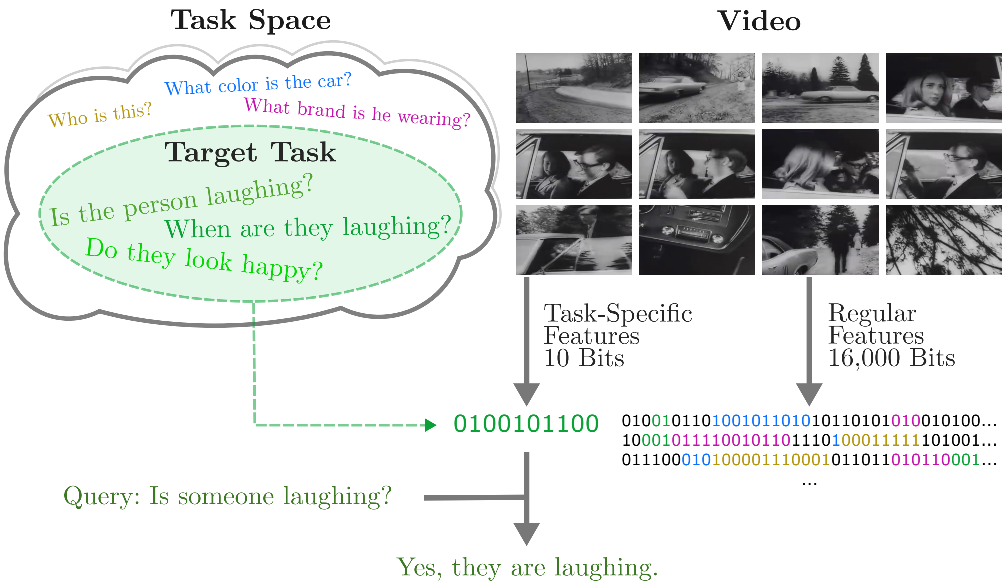


Figure 3.2: We introduce the problem of *Few-Bit VideoQA* where only few bits of video information is allowed to do VideoQA tasks. Our proposed method compresses the features inside a neural network to an extreme degree: A video input of 1 MB can be compressed down to 10 bits, and still solve common question-answering tasks with a high degree of accuracy, such as “Is someone laughing?”. This has both storage and privacy implications. Here the *Target Task* is all questions related to *laughing*, a subset of all possible tasks in the *Task Space*. *Regular Features* have additional task-irrelevant information visualized with questions and bits of corresponding color. *Regular Features* corresponds to 512 floating point features from a state-of-the-art video network, such as [51].

an understudied problem, with applications to storing and cataloging large amounts of data for use by machine learning applications.

We provide a simple yet effective task-specific compression approach towards this problem. Our method is inspired by recent learning-based image and video coding [63, 64, 65, 66, 67, 68, 69, 70, 71, 72], which learns low-level compression with the goal of optimizing visual quality. In contrast, our method looks at compressing high-level features, as shown in Figure 3.2. Given a video understanding task, we compress the deep video features into few bits (e.g., 10 bits) to accomplish the task. Specifically, we utilize a generic **Feat**ure **Comp**ression module (**FeatComp**), which can be inserted into neural networks for end-to-end training. FeatComp learns compressed binarized features that are optimized towards the target task. In this way, our task-specific feature compression can achieve a high compression ratio and also address the issue of privacy. This approach can store large amounts of features on-device or in the cloud, limiting privacy issues for

Table 3.1: Overview of different levels of feature storage and privacy limitations.

	Data Amount	Can Identify User?	Can Discriminate User?
Original Data	1,000,000 bits (e.g. MPEG4 video)	Yes	Yes
Regular Feature	16,000 bits (e.g. 512 floats)	Yes (Can be inverted)	Yes
Regular Compressed Feature	100 bits	No	Yes
Task-Compressed Feature	10 bits	No	No (k-Anonymity [73])

stored features, or transmitting only privacy-robust features from a device. Note that this work is orthogonal to improvements made by improved architectures, shows insights into analyzing how much video data is needed for a given VideoQA dataset, and provides a novel way to significantly optimize storage and privacy risks in machine learning applications.

10-bits is a concrete number we use throughout this chapter to demonstrate the advantage of tiny features. While 10-bits seems like too little to do anything useful, we surprisingly find that predicting these bits can narrow down the solution space enough such that the model can correctly pick among different answers in VideoQA [54, 55]. Different tasks require different number of bits, and we see in Section 3.4 that there are different losses of accuracy for different tasks at the same number of bits. Storing only 10 bits, we can assure that the stored data does not contain any classes of sensitive information that would require more than 10 bits to be stored. For example, we can use the threshold of 33 bits (8 billion unique values) as a rule-of-thumb threshold where the features stop being able to discriminate between people in the world. In Table 3.1 we show different levels of features and how they compare to this threshold. At the highest level, *Original Data*, we would have the full image or video with all their privacy limitations.[1] Even with feature extraction, various techniques exist to invert the model and reconstruct the original data [74, 75]. Using *Regular Compressed Features* (*i.e.* off-the-shelf compression on features in Section 3.5.1 or task-independent compression in Section 3.4.1) would still pose some privacy threats. Our *Task-Compressed Features*, provide privacy guarantees from their minimal size.

We experiment on public VideoQA datasets to analyze how many bits are needed for VideoQA tasks. In our experiments, the "**Task**" is typically question types in a specific dataset, such as

[1] In the MSRVTT-QA dataset the videos are 630KB on average, which consists of 320×240-resolution videos, 15s on average.

TGIF-Action [76], but in a practical system this could be a small set of the most common queries, or even a single type of question such as “do people look happy in the video?”. Note that while the “Task” of all questions in a single dataset might seem very diverse and difficult to compress, the task is much more narrow than any possible question about any information in the video, such as “What color is the car?”, “What actor is in the video?”, etc.

In a nutshell, in this chapter, we instantiate $\mathcal{M}$ as a pre-trained video-language model and aims to adapt it for low-bit representation. We introduce a novel Few-Bit VideoQA problem, where only few bits of video information is used for VideoQA; and we propose a simple yet effective task-specific feature compression approach that learns to extract task-specific tiny features of very few bits, where we insert a feature compressor δ as an intermediate bottleneck for $\mathcal{M}$. Extensive study of how many bits of information are needed for VideoQA. We demonstrate that we lose just 2.0%–6.6% in accuracy using only 10 bits of data, which provides a new perspective of understanding how much visual information helps in VideoQA.

The outline of the chapter is as follows. In Section 3.2, we review related work in VideoQA, image/video coding and deep video representations. In Section 3.3 we introduce the Few-Bit VideoQA problem, and our simple solution with feature compression. In Section 3.4 we discuss several experiments to analyze and validate our approach. Finally, in Section 3.5 we demonstrate applications of this novel problem, including distributing tiny versions of popular datasets and privacy.

3.2 Related Work

Video Question Answering VideoQA is a challenging task that requires the system to output answers given a video and a related question [77, 76, 54, 78, 79]. Recent approaches include multi-modal transformer models [1, 62] and graph convolutional networks [80]. We instead look at VideoQA under limited bits, which shares some philosophy with work that has looked at how much the visual content is needed for Visual Question Answering [81].

Image and Video Coding Image and video coding is a widely studied problem which aims to compress image/video data with minimum loss of human perceptual quality. In the past decades, standard video codecs like HEVC [82], AVC [83], image codes like JPEG [84] have been used for compressing image/video data. More recently, learning-based image/video compression [63, 64, 65, 66, 67, 68, 69, 85, 86] has been proposed to replace the codec components with deep neural networks that optimize the entire coding framework end-to-end, to achieve better compression ratios. Some works **??** explore compressed image/video formats and design specific networks for efficient recognition. All of these existing approaches compress with the goal of pixel-level reconstruction. Our approach is inspired by these works but is applied under a different context — we aim to solve a novel Few-bit VideoQA problem.

Deep Video Representation Deep neural networks have been shown effective to learn compact video representation, which is now a favored way to store video data for machine learning applications. With the emergence of large-scale video datasets like Kinetics [56] and HowTo100M [57], recent advances in representation learning [59, 60, 1, 61, 62, 87] extract continuous video features which contain rich semantic information. Such pre-computed video features can be successfully applied to video understanding tasks like action detection, action segmentation, video question answering, etc [1, 61, 62]. However, deep video features could still contain more information than what's actually needed by a specific task. We instead focus on learning tiny video features, where we aim to use few bits of data to accomplish the target task.

3.3 Few-Bit VideoQA

In this section, we first establish the problem of Few-Bit VideoQA; then provide a simple and generic task-specific compression solution; finally we introduce our simple implementation based on a state-of-the-art VideoQA model.

3.3.1 Problem Formulation

In a standard VideoQA framework, a feature extractor (e.g., ResNet3D [51]) is applied to a video sequence to extract the video embedding x, which is a high-level and compact representa-

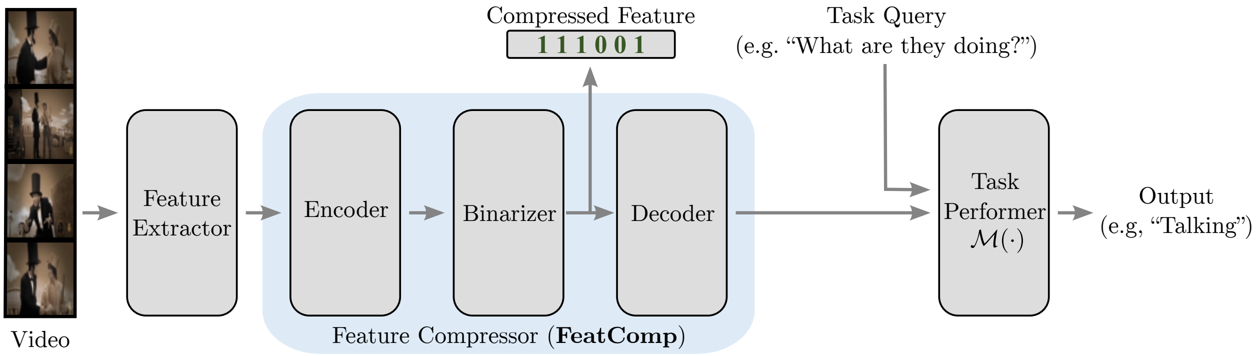


Figure 3.3: Pipeline of our generic feature compression approach (**FeatComp**) towards Few-Bit VideoQA, which follows the procedure of encoding, binarization and decoding. It has learned to only encode information relevant to the questions of interest through task-specific training.

tion of the video, normally a vector composed of floating point numbers. We write $\mathcal{M}(\cdot)$ as the VideoQA task performer, and the output from $\mathcal{M}(x, q)$ is the predicted answer in text, where q refers to the text embedding of the question. In our context, we assume $\mathcal{M}(\cdot)$ is a neural network that can be trained with an associated task objective function $\mathcal{L}_{\text{task}}$.

Though the compact feature x already has a much smaller size compared to the original pixel data, it still raises storage and privacy concerns as discussed in the introduction. To this end, we introduce a novel problem of Few-Bit VideoQA, where the goal is to accomplish VideoQA tasks with only few bits of visual information, i.e., we want to perform $\mathcal{M}(x, q)$ with the size of x less than N bits, where N is small (e.g., $N = 10$).

3.3.2 Approach: Task-Specific Feature Compression

We propose a simple yet effective approach towards the problem of Few-Bit VideoQA. As shown in Figure 3.3, we insert a feature compression bottleneck (**FeatComp**) between the video feature extractor and task performer $\mathcal{M}$ to compress x. We borrow ideas from compression approaches in image/video coding [70, 71, 72]. But rather than learning pixel-level reconstruction, we train the compression module solely from a video task loss, which has not been explored before to our knowledge. In fact, FeatComp is a generic module that could be applied to other machine learning tasks as well.

Encoding and Decoding FeatComp follows the encode-binarize-decode procedure to transform floating-point features into binary, and then decode back to floating-point that can be fed into the task performer. In encoding, we first project x to the target dimension, $x' = f_{\text{enc}}(x)$, where $x' \in \mathbb{R}^N$ and N is the predefined bit level to compress into. Binarization is applied directly on feature values; so we map the feature values to a fixed range. We use batch normalization (BN) [88] to encourage bit variance, and then a hyperbolic function $tanh(\cdot)$ to convert all the values to $[-1, 1]$. Binarization is inherently a non-differentiable operation, in order to incorporate it in the learning process, we use stochastic binarization [70, 71, 72] during training. The final equation for the encode-binarize-decode procedure is:

$$x_{\text{dec}} = \text{FeatComp}(x) = (f_{\text{dec}} \circ \text{BIN} \circ \text{ tanh } \circ \text{BN} \circ f_{\text{enc}})\,(x) \tag{3.1}$$

where $\circ$ denotes function composition. The output after the binarization step is $x_{\text{bin}} \in \{0, 1\}^N$, which is the N-bit compressed feature to be stored.

Learning Task-Specific Compression Our FeatComp is generic and can be inserted between any usual feature extractor and task performer $\mathcal{M}(\cdot)$. The task performer will instead take the decoded feature to operate the task: $\mathcal{M}(x_{\text{dec}})$ or $\mathcal{M}(x_{\text{dec}}, q)$. To compress in a task-specific way, FeatComp is trained along with $\mathcal{M}(\cdot)$ with the objective $\mathcal{L} = \mathcal{L}_{\text{task}}$, where $\mathcal{L}_{\text{task}}$ is the target task objective.

Simple Implementation Our FeatComp can be easily implemented into any VideoQA models. As an instantiation, we choose a recent state-of-the-art model ClipBERT [62] as our baseline, and add our compression module to study the number of bits required for VideoQA. Specifically, ClipBERT follows the similar pipeline as in Figure 3.3. We then insert FeatComp after feature extractor. For encoding and decoding, we use a fully connected layer where $f_{\text{enc}}\colon\ \mathbb{R}^{(T\times h\times w\times D)} \mapsto \mathbb{R}^N$ and $f_{\text{dec}}\colon \{0, 1\}^N \mapsto \mathbb{R}^{(T\times h\times w\times D)}$. x is flattened to a single vector to be encoded and binarized, and the decoded x_{dec} can be reshaped to the original size. Then the answer is predicted by $\mathcal{M}(x_{\text{dec}}, q)$. For FeatComp with N-bit compression, we write it as FeatComp-N. This implementation is simple and generic, and as we see in Section 3.4 already works surprisingly well. Various other architectures for encoding, decoding, and binarization could be explored in future research.

Intuition of FeatComp To learn a compression of the features, various methods could be explored. We could cluster the videos or the most common answers into 1024 clusters, and encode the cluster ID in 10 bits, etc. For a task such as video action classification, where only the top class prediction is needed, directly encoding the final answer may work. However, in a general video-language task such as VideoQA, the number of possible questions is much more than 1024, even for questions about a limited topic. Our method can be interpreted as learning 2^N clusters end-to-end, that are predictable, and useful to answering any questions related to the task.

3.4 Experiments

In this section, we show the experimental results on Few-Bit VideoQA, study how much visual information is needed for different VideoQA tasks, and analyze what the bits capture.

Datasets We consider two public VideoQA datasets: 1) TGIF-QA [76] consists 72K GIF videos, 3.0s on average, and 165K QA pairs. We experiment on 3 TGIF-QA tasks — Action (*e.g.* "What does the woman do 5 times?"), Transition (*e.g.* "What does the man do after talking?"), which are multiple-choice questions with 5 candidate answers; and FrameQA (*e.g.* What does an airplane drop which bursts into flames?"), which contains general questions with single-word answers. 2) MSRVTT-QA [54] consists of 10k videos of duration 10−30s each and 254K general QA pairs, *e.g.* "What are three people sitting on?".

Implementation Details We leverage the ClipBERT model pre-trained on COCO Captions [89] and Visual Genome Captions [90], and train on each VideoQA dataset separately, following [62]. During training, we randomly initialize FeatComp, and finetune the rest of the network; we randomly sample T=1 clip for TGIF-QA and T=4 clips for MSRVTT-QA, where in each clip we only sample the middle frame. We fix the ResNet backbone and set the learning rate to 5×10^{-5} for FeatComp, and 10^{-6} for $\mathcal{M}$. We use the same VideoQA objective and AdamW [91] optimizer as in [62]. During inference, we uniformly sample T_{test} clips to predict answer, where T_{test}=T, unless noted otherwise. We set N=$1, 2, 4, 10, 100, 1000$ for different bit levels. Code will be made available.

Evaluation Metrics QA accuracy at different bit levels.

Baselines To our knowledge, there is no prior work directly comparable; hence we define the following baselines:

- *Floats*: the original floating-point-based ClipBERT; it provides performance upper bound using enough bits of information .[2]
- *Q-only*: answer prediction solely from question.
- *Random Guess*: randomly choose a candidate/English word for multiple-choice/single-word-answer QA.

Additionally, since our approach follows an autoencoder-style design, we also study whether an objective of feature reconstruction helps with Few-Bit VideoQA. We add the following two approaches for comparison:

- *R-only*: learn FeatComp solely with a reconstruction loss, $\mathcal{L}_R$=MSE(x, x_{dec}), then finetune $\mathcal{M}$ using learned compressed features with task objective.
- *FeatComp+R*: learn FeatComp from both task and reconstruction objectives, i.e., $\mathcal{L}$=$\mathcal{L}_{task}$+$\mathcal{L}_R$.

We also study how test-time sampling affects the results with:

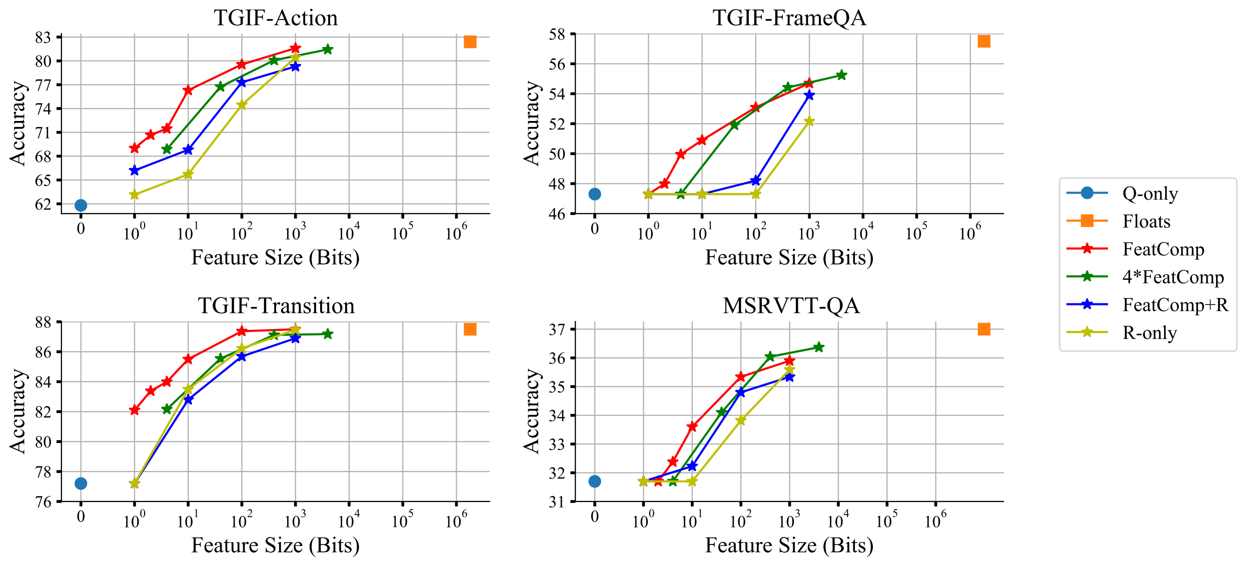
- *4×FeatComp*: test-time sampling T_{test}=4T.

3.4.1 Few-Bit VideoQA Results

We demonstrate our Few-Bit VideoQA results in Figure 3.4 and compare to the baselines. As expected, more bits yields accuracy improvements. Notably, for our *FeatComp*, even a 1-bit video encoding yields a 7.2% improvement over *Q-only* on TGIF-Action. On all datasets, at 1000-bit, we maintain a performance drop <2.8% while compressing more than >1000 times. At 10-bit, the drop is within 2.0−6.6% while compressing over >100,000 times, demonstrating 10-bit visual information already provides significant aid in VideoQA tasks.

We also compute *supervision size* as a naive upper bound of bits required to accomplish the

[2] ClipBERT uses 16-bit precision, we use this for calculation.



Method	Bits	TGIF-Action	TGIF-FrameQA	TGIF-Transition	MSRVTT-QA
Random Guess	0	20.0	0.05	20.0	0.07
Q-only	0	61.8	47.3	77.2	31.7
R-only	10	63.7	47.3	83.5	31.7
FeatComp+R	10	75.8	47.4	82.8	32.2
FeatComp	10	76.3	50.9	85.5	33.6
Floats	1.8M - 9.8M	82.4 (82.9)	57.5 (59.4)	87.5 (87.5)	37.0 (37.0)
Supervision Size		30.9 Bits	59.2 Bits	57.7 Bits	59.5 Bits

Figure 3.4: Analysis of how bit size affects VideoQA accuracy on MSRVTT-QA and TGIF-QA. *Floats* refers to the original ClipBERT model that serves as an upper bound of the performance without bit constraints. We both report the number we reproduced and cite paper results in parenthesis. With our simple approach *FeatComp*, we can reach high performance using only a few bits. Notably, at 10-bit level (see table), we get only 2.0–6.6% absolute loss in accuracy.

task. Theoretically, the compressed features should be able to differentiate the texts in all QA pairs. Hence, for each task, we use bzip2[3] to compress the text file for the training QA pairs, whose size is used as the upper bound. We observe correspondence between supervision size and compression difficulty. TGIF-Action and TGIF-Transition are easier to compress, requiring less bits to get substantial performance gains; and they also appear to require less bits in supervision size. Instead TGIF-FrameQA and MSRVTT-QA are harder to compress, also reflected by their relatively larger supervision size.

Role of Reconstruction Loss Our approach FeatComp is learned in a task-specific way in order

[3] https://www.sourceware.org/bzip2

to remove any unnecessary information. On the other hand, it is also natural to compress with the goal of recovering full data information. Here we investigate whether direct feature supervision helps learn better compressed features from *FeatComp+R* and *R-only*. We can see that integrating feature reconstruction harms the accuracy at every bit level. In fact, R-only can be considered as a traditional lossy data compression that tries to recover full data values. It implies that recovering feature values requires more information than performing VideoQA task; hence feature reconstruction loss may bring task-irrelevant information that hurts the task performance.

Role of Temporal Context For longer videos, frame sampling is often crucial for task accuracy. We study the impact in Figure 3.4 (*FeatComp* vs. *4×FeatComp*). We can see denser sampling benefits compression ratio especially at higher bits on longer videos like MSRVTT-QA. For example, 400-bit compression with T_{test}=16 outperforms 1000-bit compression with T_{test}=4. However, on short video dataset TGIF-QA, it does not yield improvements, which implies TGIF-QA videos can be well understood with single frames.

How Task-Specific are the Features? Here we evaluate the task-specificity by analyzing how much information is removed by the compression. We use FeatComp-10 learned on a source TGIF-QA task to extract the compressed features x_{bin}; then train a new VideoQA model for different target tasks on top of x_{bin}. For fair comparison, we use the same $\mathcal{M}$ and decoding layers and follow the previous practice to initialize $\mathcal{M}$ from the model pre-trained on COCO Captions and Visual Genome, and randomly initialize the decoding layers. Then we train the network with learning rate 5×10^{-5} for 20 epochs. Table 3.2 shows the results, where we also cite *Q-only* from Figure 3.4. Compression from the same task gives the best result as expected. Note that *Action* and *Transition* are similar tasks in that they query for similar actions but *Transition* asks change of actions over time. *FrameQA* instead queries for objects, which has minimal similarity with *Action/Transition*. Compression from a highly relevant source task (*Action* vs. *Transition*) gives pretty high performance. However, compression from an irrelevant source task (*Action/Transition* vs. *FrameQA*) yields similar performance as *Q-only* (0-bits), which again implies that the learned compression discards any information unnecessary for its source task.

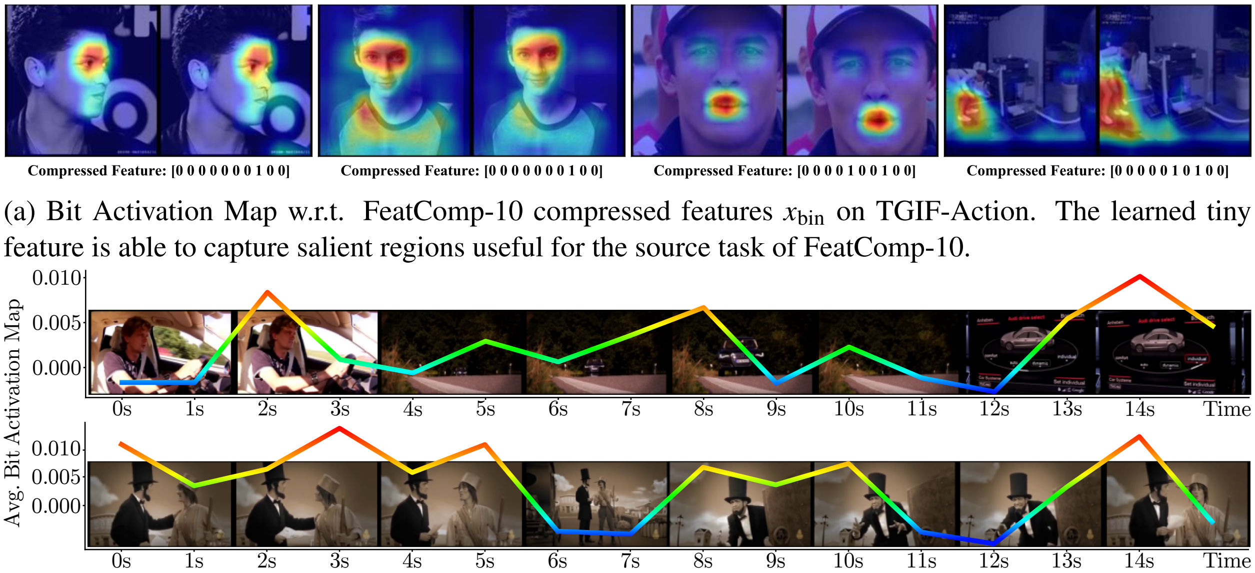
Table 3.2: Task-specificity of FeatComp-10. Compression learned from an irrelevant source task is less helpful for a target task, while compression from the same task gives the best performance.

	Target Task		
Source Task	Action	FrameQA	Transition
Action	**76.9**	47.6	83.8
FrameQA	62.3	**51.7**	77.1
Transition	76.0	47.3	**85.0**
Q-Only	61.8	47.3	77.2

3.4.2 Qualitative Analysis

Here we study qualitatively what the learned bits are capturing. We apply Grad-CAM [92] directly on the compressed features x_{bin} to find the salient regions over frames. Grad-CAM is originally a tool for localizing the regions sensitive to class prediction. It computes the weighted average of feature maps based on its gradients from the target class, which localizes the regions that contribute most to that class prediction. Here we instead construct *Bit Activation Map* (BAM) by treating each binary bit x_{bin}^{i} as a "class": 1 is a positive class while 0 is negative. We calculate BAM at the last convolutional layer of ResNet. We average BAM for all bits where for x_{bin}^{i}=0 we multiply them with -1. Note that no class annotations, labels or predictions are used for BAM. Figure 3.5a shows heat map visualization results on TGIF-Action for FeatComp-10. The learned compression is capturing salient regions (e.g., eyes, lips, legs) which align with human perception. We also study in Figure 3.5b how important temporal signals are captured, where we average the BAM over frames. The frames with high BAM scores tend to capture more important semantic information across time that is relevant to the task. Additional examples are provided in Figure S1 and Figure S2.

Additionally, we investigated whether the bits capture task-specific information by analyzing the correspondences between VideoQA vocabularies and compressed features. We calculate a word feature as the averaged FeatComp-10 bit features over the videos whose associated QAs contain this word, and find its top-7 closest words based on Euclidean distance. Table 3.3 shows



(a) Bit Activation Map w.r.t. FeatComp-10 compressed features x_{bin} on TGIF-Action. The learned tiny feature is able to capture salient regions useful for the source task of FeatComp-10.

(b) Averaged Bit Activation Map over video frames on MSRVTT-QA. We can see that learned tiny feature can capture scene changes — the bit activation peaks appear when there is a different scene useful for the source task of FeatComp-10.

Figure 3.5: Qualitative visualization examples of bit activation map w.r.t FeatComp-10 compressed features x_{bin}. Note that no question, answer, prediction, or class label was used for these visualizations—This is visualizing what the network used to generically compress the video.

some sample results on TGIF-Action. Neighbor words tend to demonstrate semantic associations. E.g., ‘eyes’ is closely related to ‘blink’, ‘smile’ and ‘laugh’; ‘step’ is associated with its similar actions like ‘walk’ and ‘jump’, implying that the compression captures task-relevant semantic information.

Finally, in Figure S3, we provide some sample predictions made by different FeatComp at 1, 10, and 1000 bits, the question-only model (*Q-only*), as well as the original ClipBERT model (*Floats*). As expected, more bits are typically more likely to lead to the correct answer. Often N=10 bits are sufficient to get the correct answer among the candidates. However, VideoQA is a challenging task, and even the original ClipBERT model (*Floats*) sometimes guesses the wrong answer while a lower bit model guesses correctly.

Table 3.3: Top-7 closest words from FeatComp-10 on TGIF-Action.

Word	Most Similar Words
head	bob neck stroke fingers raise cigarette open
wave	touch man bob hands hand stroke raise
spin	around ice pants jump foot kick knee
step	walk spin around ice pants jump kick
guitar	object who keys black a strum bang
eyes	blink smile laugh cigarette lip tilt sleeve

3.5 Applications of Few-Bit VideoQA

The problem of Few-Bit VideoQA has practical applications to data storage efficiency and privacy. Below we discuss how our approach can compress a video dataset into a tiny dataset and use that to achieve the same task (Section 3.5.1); then we demonstrate the privacy advantages of few-bit features (Section 3.5.2); finally we show how we can create video summaries from the learned bits and perform quantitative evaluation (Section 3.5.3).

3.5.1 Tiny Datasets

With our task-specific compression approach, we can represent a video using only a few bits, which allows extreme compression of gigabytes of video datasets into almost nothing. For example, TGIF-QA contains 72K videos whose MPEG4-encoded format takes up about 125GB of storage; with FeatComp-10, we end up with a 90KB dataset. We follow Section 3.4.1 to train a model using stored compressed features x_{bin}, with source and target tasks being the same, to evaluate the feature quality. Table 3.4 compares the compression size and testing accuracy on TGIF-QA and MSRVTT-QA tasks. Our tiny datasets achieve similar performance with Figure 3.4 at all bit levels. In addition to MPEG4 video and uncompressed Floats, we also report the size of Floats compressed with the off-the-shelf lossless compression standard ZIP.[4] The result implies that the original features indeed contain a lot of information not easily compressible, and we are learning meaningful compression.

[4] `numpy.savez_compressed`

Table 3.4: VideoQA accuracies with our compressed datasets at different bit levels. Looking at 10-bit, we can get 100,000-fold storage efficiency, while maintaining good performance.

Datasets	TGIF-QA				MSRVTT-QA			
	Size	Task Accs.			Size	Task Accs.		
		Action	FrameQA	Transition		What	Who	All
1-bit set	9KB	68.3	47.3	82.4	1.3KB	24.8	37.7	31.8
10-bit set	90KB	76.9	51.7	85.0	13KB	27.4	44.1	33.7
1000-bit set	9MB	80.8	54.4	87.2	1.3MB	28.3	46.0	34.9
Floats set	16.2GB	82.4	57.5	87.5	12.3GB	31.7	45.6	37.0
Comp. Floats set	14.0GB	82.4	57.5	87.5	9.5GB	31.7	45.6	37.0
MPEG4 set	125GB	82.4	57.5	87.5	6.3GB	31.7	45.6	37.0

3.5.2 Privacy Advantages from Tiny Features

Here we demonstrate how our tiny features offer privacy advantages.

Advantages of Data Minimization.

Intuitively, we expect 10 bits of information to not contain very much sensitive information. The principle of Shannon Information[5], or the pigeonhole principle, tells us that 10 bits of information cannot contain a full image (approximately 10KB). Using 10 bits as an example, we can divide sensitive information into two groups based on if it can be captured by 10 bits or not in Table 3.5.

Note we assume the data was not in the training set as otherwise the model could be used to extract that potentially sensitive information (training networks with differential privacy can relax this assumption [93]). By capturing only 10 bits, we can assure a user that the stored data does not contain any classes of sensitive information that require more bits to be stored.

Furthermore, considering that there are 8B unique people in the world, the identity of a person in the world can be captured in 33 bits, so we cannot reconstruct the identity of a person from 10 bits. The identity can be identifying information, photographic identity, biometrics, etc. This is consistent with the following experiment where feature-inversion techniques do not seem to work,

[5] `en.wikipedia.org/wiki/Information_content`

whereas they work on regular compressed 16,384 bit features (as in Table 3.1). This effectively de-identifies the data.

Table 3.5: What sensitive information can be stored in 10 bits?

Impossible (bits)	Possible (bits)
Credit Card Num. (≈53)	Gender (≈2)
Social Security Num. (≈30)	Skin Color (≈3)
Street Address (≈28)	Social Class (≈3)
License Plate (≈36)	...
Personal Image ($\approx 100,000$)	
Phone Number (≈33)	
...	

Robustness to Feature Inversion

Storing only a few bits also has applications to preventing reconstruction of input from stored features (*i.e*. Feature Inversion). Various methods exist for inverting features [75, 74] typically by optimizing an input to match the feature output, with an optional regularization. To evaluate this, we started with a re-implementation[6] of the Frederikson *et al*. attack [74] for model inversion, but instead of class probability, we used a MSE loss between the target feature and the model feature output before the last linear layer, or after the binarizer. We used a two layer neural network trained on the AT&T Faces Dataset [94]. In Figure 3.6 we demonstrate two examples on how the inversion deteriorates as the model uses fewer bits. The 1,280 bits result has no compression (40 32-bit numbers) whereas the rest of the results have increasing compression. All models were trained until at least 97.5% accuracy on the training set. More similar visualization examples are also provided in Figure S3, where we can see that when reducing the number of stored bits, it is increasingly difficult to reconstruct the original image.

Feature Quantization In Frederikson *et al*. [74] the authors note that rounding the floating point numbers at the 0.01 level seems to make reconstruction difficult and offer that as mitigation strategy. This suggests that compression of features will likely help defend against model inversion

[6] `github.com/Koukyosyumei/secure_ml`

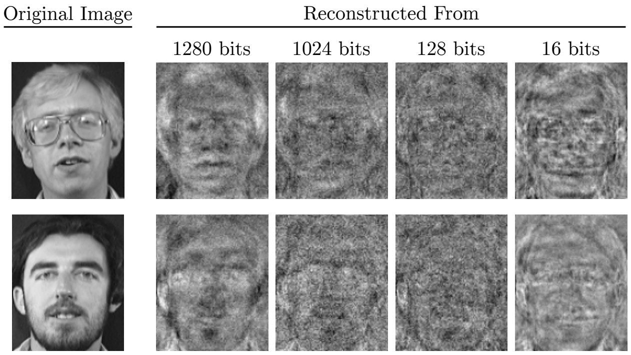


Figure 3.6: Feature inversion with increasingly compressed features.

and we verify this here. For comparison, rounding a 32-bit float (less than 1) at the 0.01 level corresponds to 200 different numbers which can be encoded in 8 bits, a 4-fold compression. Our method can compress 512 32-bit floats (16,384 bits) into 10 bits, a 1,638-fold compression, while having a defined performance on the original task.

k-Anonymity of Tiny Features

For a system that has N users and stores 10 bits (1,024 unique values), we can assure the user that:

> Any data that is stored is indistinguishable from the data from approximately $N/1{,}024$ other users.

This ensures privacy by implementing k-anonymity [73] assuming the 10 bits are uniformly distributed, for example if N=10^7, k≈10,000. Finally, we can use a variable number of stored bits to ensure any stored bits are sufficiently non-unique, similar to hash-based k-anonymity for password checking [95].

3.5.3 Video Summarization

With the compressed representation, we can create a video summary by finding the temporal segments that contribute to most of the bits. We use Grad-CAM to calculate the bit activation maps (detailed in Section 3.4.2) and compute the averaged bit activation value over frames.Then we select the 3 frames with top prominent peak values as the key frames.The summary is then the combination of 1-sec clips centered around the key frames. To evaluate the summary, we calculate its VideoQA performance on the original floating-point ClipBERT model.

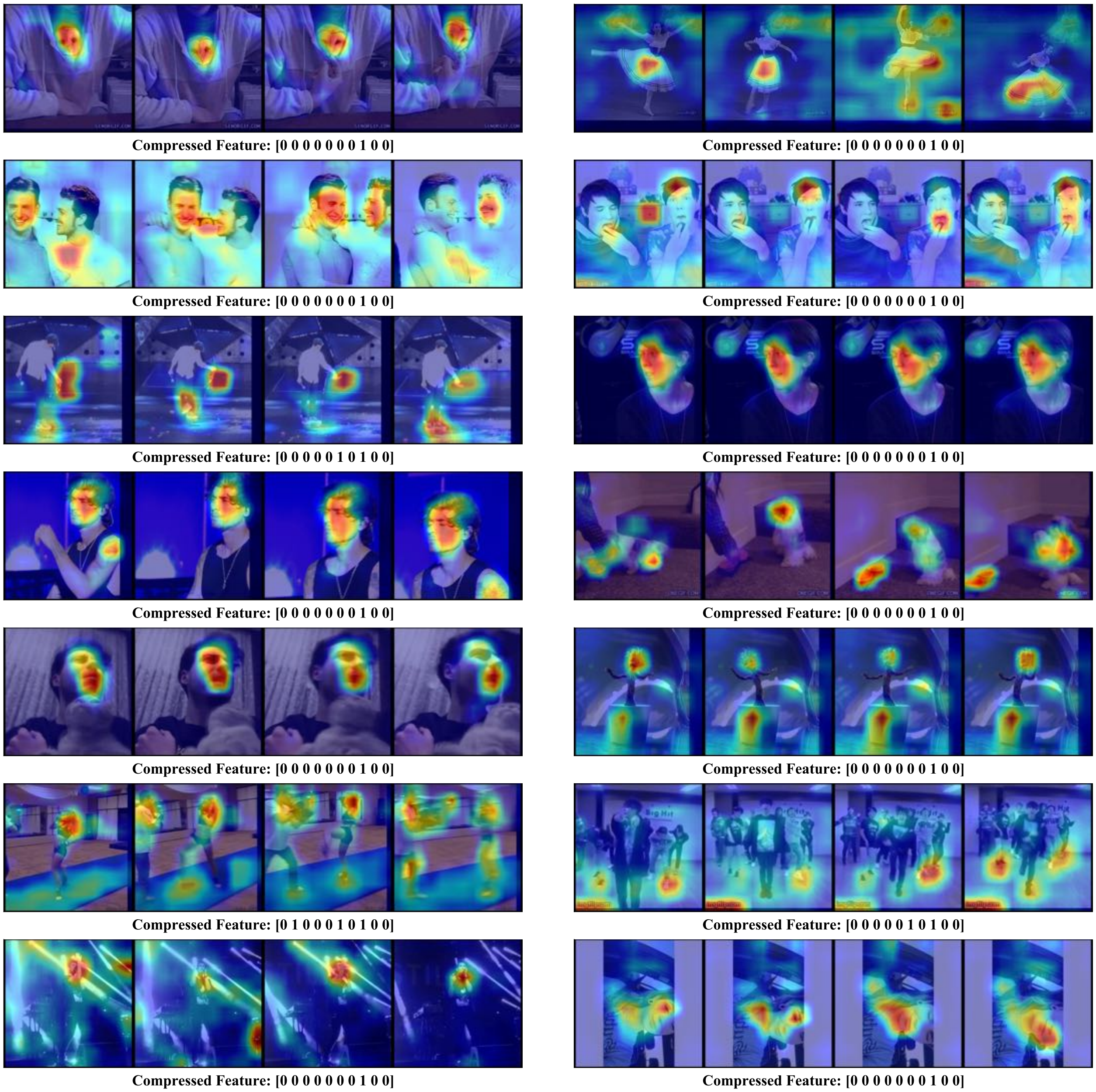
Table. 3.6 quantitatively shows the MSRVTT-QA video summary performance, where we test on videos ≥20 seconds. We compare to random summary from three 1-sec clips and summary from floating-point feature (i.e., Grad-CAM applied over floating-point features). We also report the ClipBERT T_{test}=16 result on full-length videos (No-Summary). We can see 10-bit summary yields the highest quality. Floats-summary are close to 1-bit and random, as it contains more noise. We observe little room between random-summary and no-summary, probably because MSRVTT-QA videos are not long enough to include many scene dynamics.

Table 3.6: VideoQA using 3-sec summaries extracted from MSRVTT-QA videos.

	Random	Floats	1-bit	10-bit	1000-bit	No-Summary
Acc.	35.5 ± 0.2	35.6	35.6	36.0	35.7	37.0

3.6 Summary

In this chapter, we showcase how to adapt a video-language model $\mathcal{M}$ for low-bit representation by learning a task-specific feature compressor δ. To this end, we introduce a novel Few-Bit VideoQA problem, which aims to do VideoQA with only few bits of video information used; we propose a simple and generic FeatComp approach that can be used to learn task-specific tiny features. We experiment over VideoQA benchmarks and demonstrate a surprising finding that a video can be effectively compressed using as little as 10 bits towards accomplishing a task, providing a new perspective of understanding how much visual information helps in VideoQA. Furthermore, we demonstrate the storage, efficiency, and privacy advantages of task-specific tiny features — tiny features ensure no sensitive information is contained while still offering good task performance. We hope these results will influence the community by offering insight into how much visual information is used by question answering systems, and opening up new applications for storing large amounts of features on-device or in the cloud, limiting privacy issues for stored features, or transmitting only privacy-robust features from a device.



Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 1 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 1 0 0 0 1 0 1 0 0]

Compressed Feature: [0 0 0 0 0 1 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Compressed Feature: [0 0 0 0 0 0 0 1 0 0]

Figure S1: Bit activation map visualization over frames w.r.t. FeatComp-10 compressed features x_{bin} on TGIF-QA. Our learned compression captures salient regions.

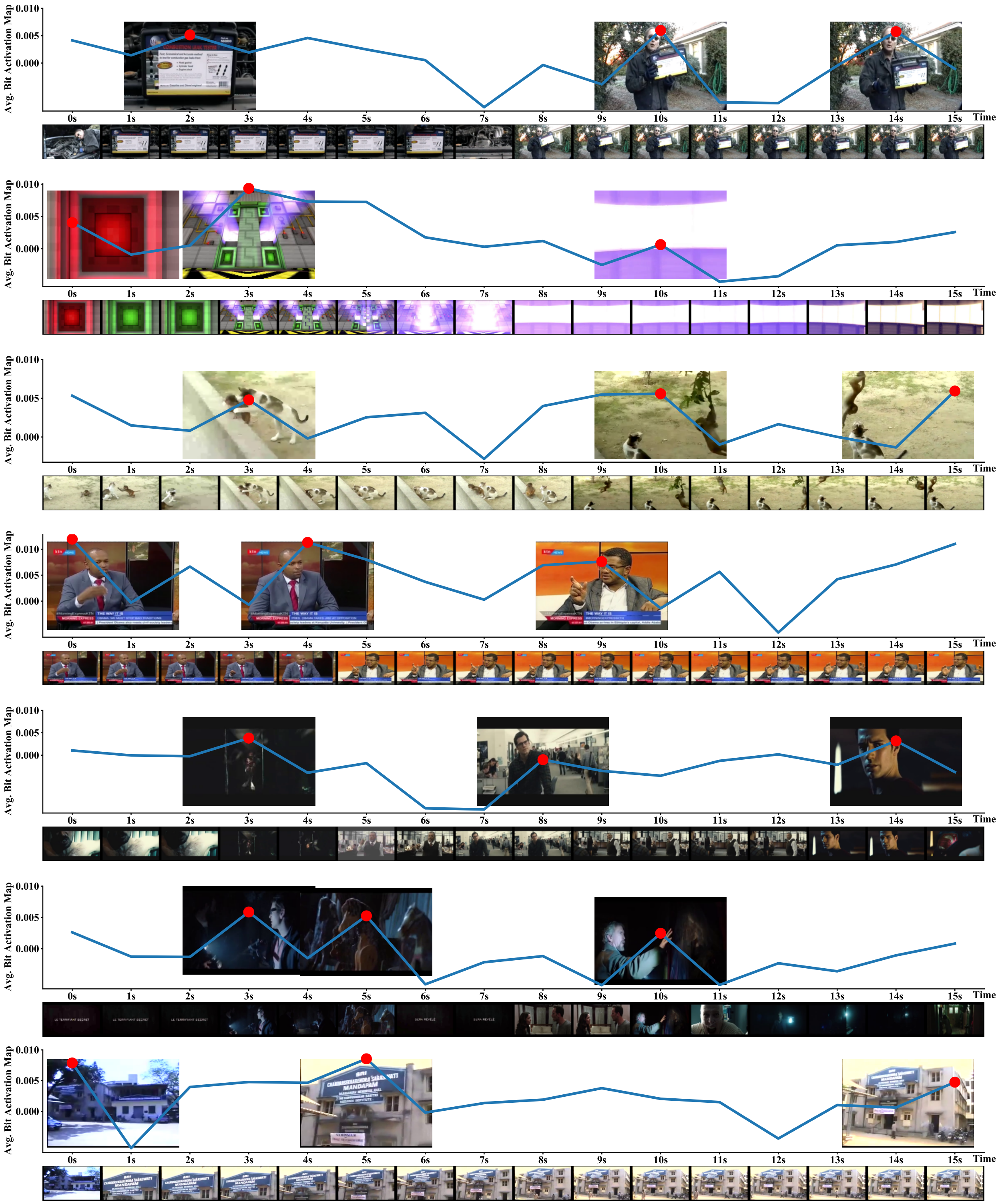


Figure S2: Averaged bit activation maps over time w.r.t FeatComp-10 compressed features x_{bin} on MSRVTT-QA. Our learned compression captures important scene changes.

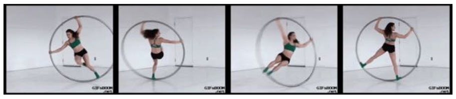
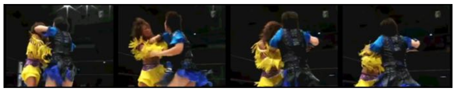
	Q-only	FeatComp-N			Floats	
		N=1	N=10	N=1000		
	B	B	**C**	**C**	D	Q: What does the singer do 3 times ? A. roll B. turn head **C. touch hair** D. tap E. shake ball
	D	D	**E**	**E**	A	Q: What does the man do 2 times ? A. step B. bounce paw C. shake left paw D. touch woman **E. flip on the air**
	E	E	E	D	**A**	Q: What does the man do 4 times ? **A. bounce egg** B. slide C. point D. jump over blocks E. bob shoulders
	D	A	D	**B**	**B**	Q: What does the guy on right do 2 times ? A. lower eyebrows **B. extend left hand** C. return ping pong ball D. point E. wag legs
	**B**	E	D	**B**	**B**	Q: What does the girl do 5 times ? A. bounce a ball **B. spin around** C. slip D. circle right arm E. wave a pen
	B	B	**E**	**E**	**E**	Q: What does the woman with a blue shirt do 10 or more than 10 times ? A. fluff blanket B. blow smoke ring C. beat hand drum D. roll **E. hit opponent**
	E	B	**A**	**A**	**A**	Q: What does the woman do 5 times ? **A. chuckle** B. cough into hand C. put palms up D. blink turnlight E. spit ball
	B	**E**	**E**	**E**	**E**	Q: What do the women do 2 times ? A. stride B. jump C. splash puddle D. move steer wheel **E. shake fingers**
	E	D	D	C	D	Q: What does the woman do 2 times ? A. leap paws B. bounce paw C. shake flower bouquet D. sway **E. spin on herself**
	E	B	B	D	D	Q: What does the man in top hat do 2 times ? A. shake both feet B. blink **C. kiss baby** D. jerk head E. glance

Figure S3: Sample predictions made by *Q-only*, *Floats* and *FeatComp* at different bit levels. Correct predictions and answers are bold in green. In general, more bits are typically more likely to lead to the correct answer.

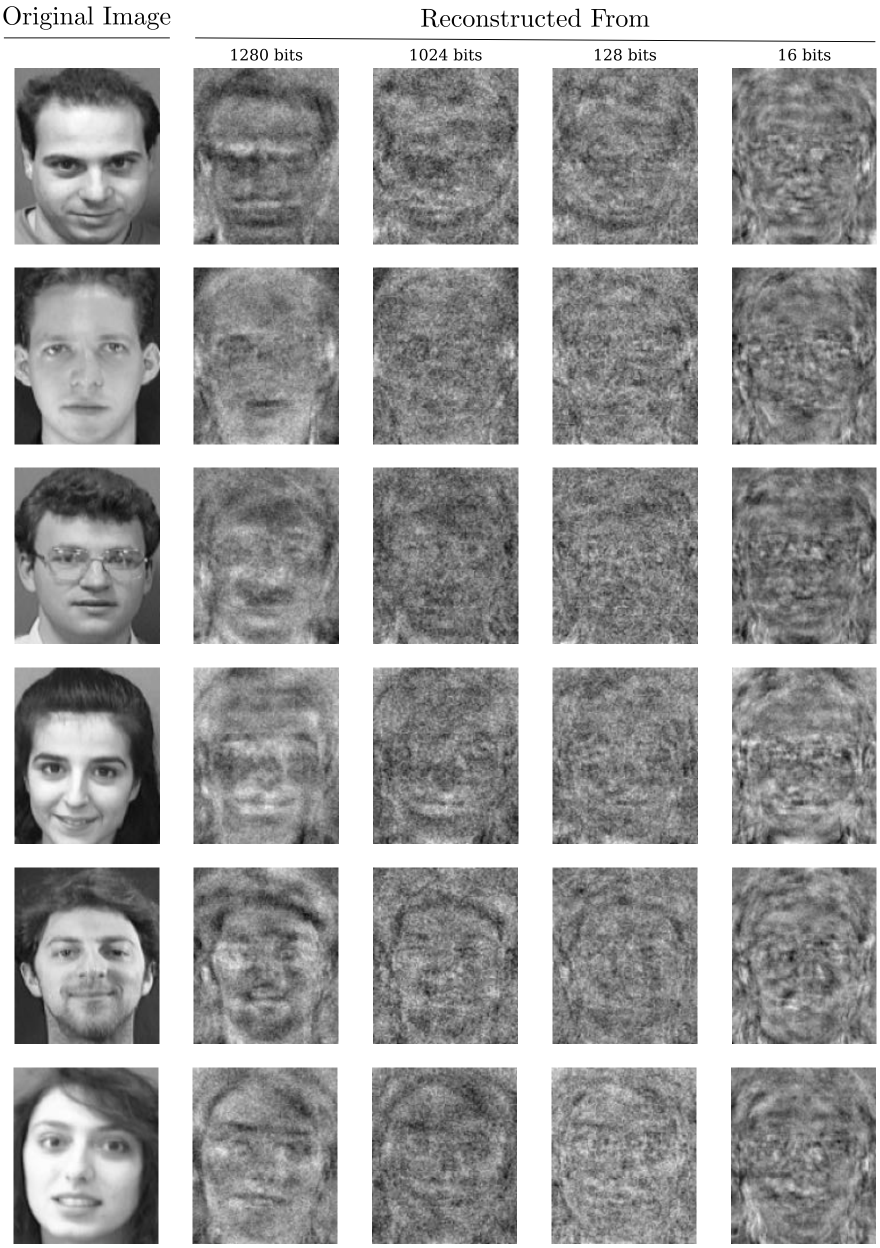


Figure S3: Feature inversion with increasingly compressed features.

Chapter 4: Adapting Model for Sparsified Inputs

In many real-world scenarios, we'd want to keep videos in a human-interpretable format so that not only machine systems, but also users can analyze the information by themselves. In this chapter, we consider data efficiency challenges from long videos and seek to adapt deep learning models for reducing video sequence into sparsified and interpretable formats. Figure 4.1 shows a high-level illustration of the framework, where δ serves as a learnable input sparsifier that generates sparse frames and/or texts for machine and human understanding.

4.1 Introduction

Watching long videos is time-consuming and easily loses user attention. How to efficiently present videos to users is an important and practical problem in various video applications. For example, for home surveillance videos which are usually recorded continuously throughout the day, it is hard for users to capture a moment of package delivery from an hour-long video. More generally speaking, for videos that are not carefully edited (e.g., Youtube videos), they often contain purposeless parts and need pre-processing of content so that users can quickly get meaningful information.

Videos often come from different modalities. Commonly, they are composed of image frame sequences. With the advances of recording devices and editing tools, videos often contain speech (e.g., Youtube videos recorded from user phones) and subtitles (e.g., in movies and TV-shows). It has been shown that leveraging different modalities benefits various video tasks [2, 1]. However, it is worthwhile noticing that the various modalities in video could be quite noisy and redundant — meaningless utterances, repeating frames, etc. — causing computational inefficiency and distracting model learning. Furthermore, the problem of modality imbalance [96] has been studied, where

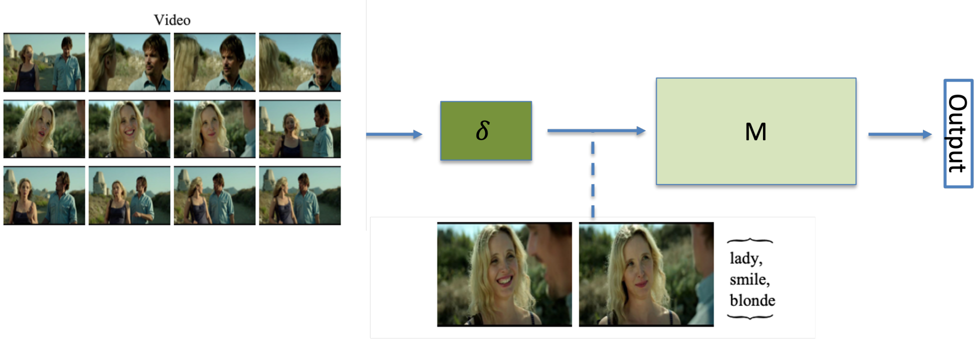


Figure 4.1: In this chapter, δ serves as an input sparsifier, which learns to sparsify video inputs temporally for a specific task.

the unbalanced information across modalities could result in significant bias towards one modality. For example, prior works [97] have shown that in TV-related videos, the major contribution for the various video-language tasks comes from subtitles while the video frames play a negligible role.

In this chapter, we characterize the VideoQA problem from the perspective of input sparsity. As illustrated in Figure 4.2, we aim to answer the question: "How much visual/textual signals are sufficient for a task?" For VideoQA specifically, different questions require different amount of video information to give the answer. For example, if the question asks about people, then theoretically the system only needs to look at the moments where people are present. In the literature, there is evidence showing that video action classification can be accomplished with single frame [98, 99]. Recently there have also been works that imply sparse uniform sampling of the video is sufficient for video and language tasks [62], and an analysis tool which shows that video and language tasks could be achieved by picking one optimal frame [100]. In this chapter, we instead move beyond single frame input, and try to characterize the role of videos by learning to select an optimal set of video inputs. We propose a generic framework which learns to drop video inputs while training for the video-and-language task. This framework can be applied to different kind of video modalities, and in our experiments we provide analysis on visual-only (i.e., video frames), text-only (i.e., video subtitles or key words), and visual-textual inputs.

We instantiate our underlying machine learning model $\mathcal{M}$ as a transformer-based video-language model and adapt it for sparsified representation by designing a learnable sparsifier δ that can be

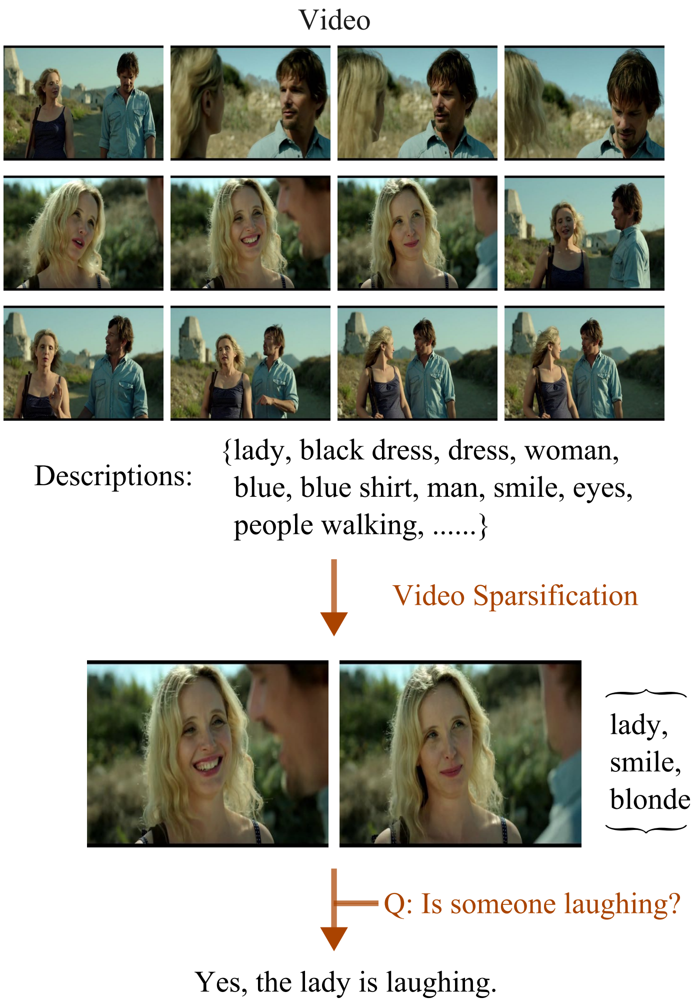


Figure 4.2: We study the Sparsified VideoQA problem, where we learn to sparsify the original long video into very few inputs for QA. We design a video sparsification process to deal with video of multiple modalities (frames, word and phrase descriptions, etc).

plugged at the input layer. From our experiments, we demonstrate that with very sparse inputs, the task can still be accomplished pretty well. Specifically, we are able to achieve 5.2%−5.8% loss of accuracy with only 10% length of the video, which corresponds to only 2−4 selected frames. Meanwhile, we also observe complimentary behaviour between modalities even with sparsified multi-modal inputs. Our finding suggests the potential of improving data efficiency under either single or multi-modal settings for video-and-language tasks.

4.2 Related Works

Video Question Answering VideoQA is a video understanding task about predicting answers given a video and a related question [77, 76, 54, 78, 79]. Recent works usually use multi-modal transformer models [1, 62] and feed in a combined sequence of frame and question word tokens, and the model processes the multi-modal inputs with attention mechanisms. To ensure full utilization from a video, prior works usually sample dense frames and feed them altogether into the model, trying to maintain as much information from the video as possible, and then perform analysis based on the assumption of minimal information loss from the video at the feature level. We instead want to characterize the video behaviour of VideoQA task from the perspective of limited information.

Vision vs. Language in Multi-modal Learning It has been pointed out by several works [62, 101] that there exists information imbalance between different modalities. Especially, [96] shows the language bias in image-language tasks and provides a solution to overcome it. Recently, [101] shows that video-language tasks can be accomplished pretty well with as little as 10-bit of information, again implying the strong bias towards language in video-language tasks. Recently, [100] provides a detailed analysis showing video-language tasks weakly rely on the temporal information from video and that video-language tasks can be comparably accomplished by a strong image-language baseline. Our work is also built upon these interesting observations and aim to characterize the behaviour of multi-channel video inputs in different downstream tasks.

Efficient Video Understanding Our work is related to some recent works on improving video data efficiency in video understanding applications. [62] shows that sparsely sampled video frames can be trained end-to-end to perform video-language tasks well. [98] study how to reduce video frames for fast video recognition tasks. There are some evidence [100, 101] showing video understanding tasks can be accomplished even with reduced inputs. Prior works [100, 98] often focus on extracting single key frames to represent video. More recently, [102] proposes efficient vision transformer for video action recognition by dropping spatial-temporal tokens. We offer a way to analyze video-language tasks, which typically rely on transformer models, with inputs at user-controlled sparsity levels. Given the multi-channel nature of videos, we can similarly study textual-based sparse representation of videos as well.

4.3 Approach

In this section, we introduce our token sparsification approach. We first provide a brief preliminary on multi-modal transformers. Then we explain how the learnable token sparsification works. Finally, we explain how we can extend it to multi-modal setting.

4.3.1 Preliminaries — Multi-modal Transformers

To solve for video tasks that involve multi-modal inputs (e.g., VideoQA), current literature usually converts both visual and textual signals into sequence of embeddings, projects them into a common embedding space and applies a multi-modal transformer which takes the sequence concatenation to generate the final output. i.e.,

$$y = \mathcal{M}([v_1, \dots, v_n; w_1, \dots, w_m]) \,, \tag{4.1}$$

where $\mathcal{M}$ is the multi-modal transformer, y is the task output (e.g., predicted answer in text), $\{v_i\}_{i=1}^n$ is the sequence of video frame/segment features, and $\{w_j\}_{j=1}^m$ is the sequence of word embeddings for the text inputs.

To get $\{v_i\}$, the common practice is to span the entire video, and use an off-the-shelf feature extractor to get frame or segment level features (depending on whether the feature extractor is image-based or segment-based). In this way, all the video information is included for the task. However, videos contain a lot of redundant information which is not necessarily useful for the task. For example, if the task is asking about "What the person is wearing", intuitively it only needs one video frame that contains the target person. From this perspective, we aim to design a framework that learns to select only the key information for the task.

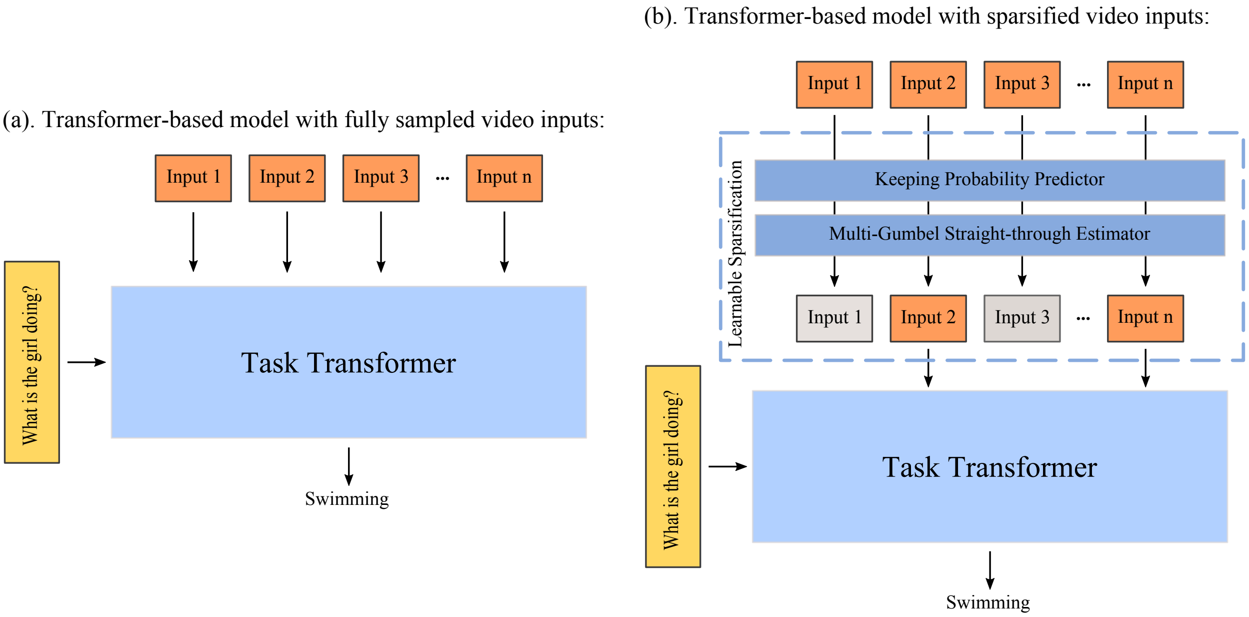


Figure 4.3: The pipeline of our method. We insert a learnable selection module between the inputs and the task transformer to sparsify the inputs. Our method can be generally applied to single or multi- modal inputs from videos. The entire network is trained end-to-end using the task objective as well as the input sparsity constraints.

4.3.2 Token Sparsification

Observing the architecture of transformers which treat visual and/or textual embeddings as sequence tokens, we propose to learn to sparsify the input tokens during training. Starting with single modality inputs $\{v_i\}_{i=1}^n$, we first generate the keeping probability s_i for each token v_i to estimate how likely v_i should be kept. To ensure s_i is a valid probability value within the range of $[0, 1]$, we place a predictor $f(\cdot) : \mathcal{R}^d \to \mathcal{R}^2$ to map each token v_i to 2-dim, and apply a Softmax normalization to re-scale the 2-dim vector into $[0, 1]$, and take the first entry as the keeping probability s_i. Specifically,

$$s_i =< \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathrm{Softmax}(f(v_i)) >,$$

where $< \cdot >$ is the inner product function. At inference time, we can treat sparsification as a ranking procedure, whereas we rank tokens according to their keeping probabilities and select top-K tokens.

During training, to facilitate the learnability of the network, we treat sparsification as a top-K sampling procedure. That is, we draw K samples from $\{v_i\}$ from their keeping probabilities as the sparsified inputs to the rest of the model. By default, the sampling process is not differentiable. In order to overcome this non-differentiability, we refer to and extend the Gumbel-Softmax [103] trick.

Gumbel-Softmax Straight-through Estimator [103] offers a differentiable way for *single* discrete sampling. It first re-parameterizes the sampling distribution by adding Gumbel noise followed by a Softmax normalization, i.e.,

$$s_i^g = \frac{exp((\log s_i + g_i)/\tau)}{\sum_{j=1}^{n} exp((\log s_j + g_j)/\tau)},$$

Then it follows the design of straight-through estimator [70] to get the sample index from $k = argmax_i\{s_i^g\}_{i=1}^n$ and select the corresponding v_k in the forward pass while zeroing-out the rest. In the backward propagation, the gradients of s_i^g are kept for use. We encourage referring to [103] for more details and proofs.

Multi-Gumbel Straight-through Estimator. The standard Gumbel-Softmax STE trick only supports single sampling, as it performs along the sequence and only selects one most likely token. In our context, would like to extend it to multi-sampling of most likely tokens. [104] introduces a trick to perform sampling without replacement. In our implementation and experiment, we instead modified [104] to two variants:

1. **Gumbel-TopK Selection**: select the top-K tokens with higher s_i^g values. And then use straight-through estimator after selection, i.e., we only keep the selected tokens and zero-out the rest, but in the backward pass, we still use the gradients of all s_i^g.

2. **Ratio-controlled Gumbel**: instead of hard selection of K samples, we allow sampling with arbitrary number of tokens while keeping a sparsity ratio constraint in the loss during train-

ing. Specifically, we add Gumbel disturbance to s_i:

$$s_i^g = \frac{exp((\log s_i + g_{i_0})/\tau)}{exp((\log s_i + g_{i_0})/\tau) + exp((\log(1 - s_i) + g_{i_1})/\tau)}$$

where $g_{i_0}, g_{i_1} \sim Gumbel(0, 1)$ are Gumbel noises. We then keep all the tokens whose perturbed keeping probability $s_i^g > 0.5$. For a human-selected target keeping ratio p, we add a loss constraint on the overall keeping ratio over the batch:

$$\mathcal{L}_{select} = \frac{1}{B}\sum_{b=1}^{B}(p - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[v_i^b \text{ is kept}])^2,$$

where B is the batch size, and v_i^b is denoted as the tokens within a batch.

The overall training loss generically is the weighted balance between the task loss and the selection loss:

$$\mathcal{L} = \mathcal{L}_{task} + \lambda\mathcal{L}_{select}, \tag{4.2}$$

where λ is the balancing weight between two loss components, and $\mathcal{L}_{task}$ is the task loss. Note that $\mathcal{L}_{select}$ is not necessary required for the first variant and we can simply set $\lambda = 0$ in that case. The method is generic and task-agnostic. For VideoQA, we refer to [1] for $\mathcal{L}_{task}$, which essentially is a cross entropy loss between the predicted answer and the ground-truth answer word. During inference, we directly rank $\{s_0\}$ and select the inputs associated with the top-K scores. In this way, we are able to get the sparsified video inputs for the target task.

4.3.3 Sparsified Positional Encoding

Positional encoding is a typical mechanism designed for transformers to encode the order of tokens within a sequence. Typical designs for positional encoding are either fixed, such as sinusoidal function with selected frequencies [105], or learnable positional embeddings like [106]. In our context, we make use of learnable positional embeddings following [1, 2]. For a full-length sequence, each token v_i is added to a unique learnable embedding vector p_i that sticks to this position i. After sparsification, tokens are dropped and kept by probability, so the remaining tokens could be at various positions.

In our preliminary experiments, we found that inappropriate positional embedding could harm training convergence, especially when we just leave the positional embeddings as they are in the full sequence scenarios. As a result, in our implementation, we arrange the sparsified tokens $\{v_{k_i}\}_i^K$ into a new sequence $\{v'_i\}_i^K$ (where $v'_i = v_{k_i}$), and assign p_i to v'_i accordingly. We observe a good convergence behaviour and performance from this simple rearrangement.

4.3.4 Multi-modal Token Sparsification

We generalize the standard single modality token sparsification to multi-modal scenarios where videos come with both visual and textual signals. Specifically, our method allows the inputs being any kind of tokenized inputs. For videos that come with multi-channel inputs, e.g., subtitles, we can directly concatenate time-stamped subtitles along with the frames as multi-modal inputs. Then we can perform similar token sparsification on both of them. Specifically, we denote the multi-modal inputs as $\{v_i, w_i\}_{i=1}^n$ (note that we can assume same number of v_i's and w_i's by interpolating or padding, etc). We first get the unified representation from multi-modal inputs by

$$u_1, \ldots, u_n = C_m([v_1, \ldots v_n; w_1, \ldots, w_n]) \tag{4.3}$$

where C_m is a context model (e.g., cross-modal transformer) that exploits both visual v_i and subtitle w_i information. Then we apply the uni-modal sparsification technique introduced in Sec. 4.3.2 on top of $\{u_i\}$ sequence to compute the keeping scores, and sparsify the original multi-modal input sequence $\{v_i, w_i\}$ according to the score, to get the sparsified multi-modal sequence $\{v_{k_i}, w_{k_i}\}_{i=1}^K$. Then this sparsified sequence is fed to the task performer $\mathcal{M}$ to compute the outputs (eq. 4.1).

We also consider another approach where we ease the restriction on time-matched input pairs and operate on each modality regardless of their timestamps. In this setting, we apply different context model on each modality sequence, and then use separate selection loss to constrain the sparsity. We will provide more details on this setting in the Experiment details, where we demonstrate our idea with key-word selection that summarizes the videos.

4.4 Experiments

In this section, we provide our experiment results. First we detail our implementation and VideoQA datasets; then we provide VideoQA results under different input sparsities, followed by multi-modal results. Finally, we offer some qualitative visualization to analyze our approach.

4.4.1 Implementation Details

To verify our idea, we experimented on two state-of-the-art video-and-language models VQA-T [2] and HERO [1]. HERO considers multi-channel videos where videos come with subtitles as additional channel of inputs. HERO follows a hierarchical transformer architecture to first exploit the information within video modalities and contexts, and then has another task head to operate the task. VQA-T simply consists of two Distill-BERT models to deal with video+question inputs and answer candidates, and computes the answer based on embedding similarity. For extracting the video features, we follow [2] to use the S3D model pre-trained on Howto100M dataset. For extracting the key word candidates, we use the model offered by [2] and the vocabulary from the training split of the dataset to extract the words/phrases based on feature similarity.

4.4.2 Datasets and Metrics

We evaluate our idea on public VideoQA benchmarks including VLEP [107], VIOLIN [108] and iVQA [2]. For VLEP and VIOLIN, we follow [1] to build our method on top of HERO. VLEP and VIOLIN provide both raw videos and subtitles as inputs. Our selection is then based on the multi-modal inputs. For iVQA, we follow [2] to build our method on top of VQA-T. We report VideoQA accuracies across different input sparsity level: 10%, 30%, 50%, 70% and full (100%) inputs.

Table 4.1: Effect of the temperature τ. Smaller τ leads to more exploitation while higher leads to more exploration. We observe that more explorative selection is beneficial for denser inputs.

	VLEP			VIOLIN		
Input Percentage	$\tau = 0.01$	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.01$	$\tau = 0.1$	$\tau = 0.5$
10%	**60.25**	56.01	58.94	56.25	**60.57**	58.80
30%	60.95	**63.52**	59.13	61.72	57.64	**62.34**
50%	63.05	63.64	**64.30**	65.57	64.48	**66.06**
70%	63.73	65.14	**65.32**	65.94	66.52	**67.06**

4.4.3 VideoQA Experiments

We present our single modality sparsified VideoQA results here. First, we study the design choices of our two multi-gumbel estimator variants, followed by the comparison between our approach and other token sparsification baselines.

Effect of temperature τ. In our experiments, we found that varying τ could result in very different performance. We elaborate the result in Table 4.1 with our Gumbel-TopK selection variant, where we choose $\tau = (0.01, 0.1, 0.5)$ for each sparsity level, and fix $\lambda = 1.0$. A smaller τ means the model focuses more on exploitation, while a larger τ makes the model focus more on exploration. We can observe that on both datasets, the model that is more explorative with denser inputs gives a better results; but on sparser inputs, the model tends to stick to exploitation.

Effect of loss balancing weight λ. We also study how the balancing weight λ affects the performance with our ratio-controlled Gumbel estimator. In Table 4.2, we choose $\lambda = 0.01, 0.1, 1.0$ and 10.0, and then report the results at highly sparsified (10%) and lowerly sparsified (70%) levels on VLEP dataset. We fix $\tau = 0.01$ for 10% level and $\tau = 0.5$ for 70% level. λ has slightly larger impact on highly sparsified setting. We observe that $\lambda = 1.0$ yields the best balance and hence choose it for all the other performance.

Comparison of two multi-gumbel variants. We compare the two variants of Gumbel estimator for token sparsification. In Table 4.3, we compare these two variants at different sparsity levels on VLEP. Our Gumbel-TopK selection variant is better than ratio-controlled Gumbel overall. Ratio-controlled Gumbel is superior at highly sparsified level (10%) as it adds more flexibility in

Table 4.2: Effect of the balancing weight λ. λ balances the selection loss and task loss as specificed in eq. 4.2. We report results on VLEP dataset and observe that $\lambda = 1.0$ yields the best balance. Higher λ might lead to distraction of task, while lower λ might lead to insufficient sparsification. We pick $\lambda = 1.0$ based on the following ablation.

Input Percentage	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 10.0$
10%	59.12	59.85	60.25	59.90
70%	65.23	65.32	65.32	65.11

Table 4.3: Comparison of our two Gumbel variants. Overall the first variants perform slightly better. The second variant is superior at highly sparsified level (10%) as it adds more flexibility in individual sparsity levels across different videos.

Input Percentage	10%	30%	50%	70%
Gumbel-TopK Selection	60.25	63.52	64.30	65.32
Ratio-controlled Gumbel	61.43	63.42	63.49	65.01

individual sparsity levels across different videos.

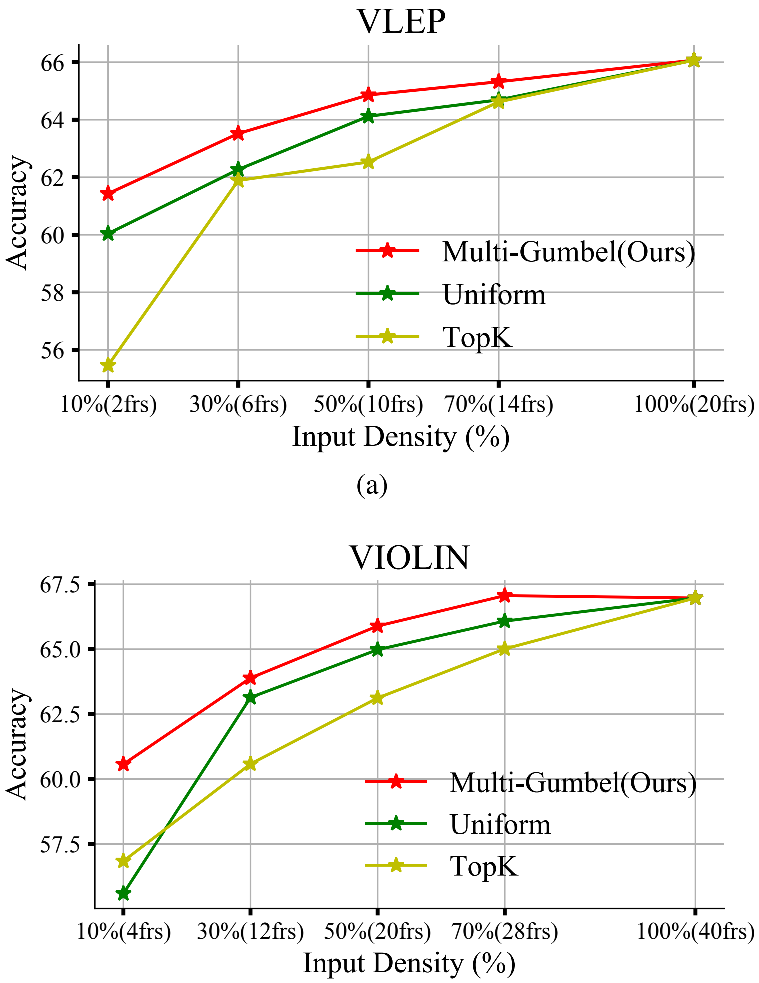
Comparison with other sparsification approaches. To our knowledge, no prior work has studied the same topic on VideoQA before, so there is no direct comparison. To validate our approach of Multi-Gumbel Estimator, we define the baselines on our own:

1. Uniform(Fixed): Fixed uniform sampling of inputs w.r.t. different sparsity levels.

2. TopK: During training, directly select inputs with higher keeping probability s_i after softmax step, without noise perturbing.

3. Multi-Gumbel(Ours): Our approach which stochastically sparsifies tokens with Gumbel perturbing, we plot the better result from the two variants we introduced.

We show the accuracy vs. density curve in Figure 4.4. We can see that our Multi-Gumbel approach module is able to achieve the best performance across different sparsity levels. Compared to learnable selection, fixed uniform sampling is weaker as it does not contain any form of task adaptive selection. A direct TopK selection training performs weaker than training with stochastic sampling, as we observe that the deterministic selection tends to a local optimal choice, while our



(a)

(b)

Figure 4.4: Sparsified VideoQA results on VLEP and VIOLIN datasets. Accuracy at the 100% level refers to the original full input baseline result. We can conclude that learnable sparsification is better than fixed sampling (Uniform), and that stochasitic sampling is better than deterministic selection (TopK). Our Multi-Gumbel estimator achieves the best result overall.

stochastic Multi-Gumbel approach gives more flexibility of by adding noises while learning. One noticeable observation is that, at 10% level, which corresponds to very few frames (2 frames for VLEP and 4 frames for VIOLIN), the performance is still quite good. It implies the potential of accomplishing the task with very few inputs.

Table 4.4: VideoQA results on iVQA. We apply our approach on the state-of-the-art method [2]. We consider multi-modal sparsification where we sparsify both visual (i.e., frames) and textual (i.e., words) inputs. Compared to single-modality, multi-modal performance is stronger at different sparsification levels. With additional extracted words, we also outperform the state-of-the-art result on iVQA (last column).

		Visual (Snippets)				
		0	1 snippet	2 snippets	5 snippets	20 snippets
Textual (Words)	0	14.6 (Q-only)	28.65	30.24	31.26	35.43 [2]
	5 words	17.5	28.68	30.31	31.70	35.43
	10 words	18.22	29.87	31.43	31.88	36.01
	25 words	20.14	30.16	31.59	32.03	36.09
	100 words	26.75	31.47	32.11	33.21	**36.42**

4.4.4 Multi-modal Sparsification Results on iVQA

In the multi-modal experiments, we would like to study the relation between visual and textual modalities under a controlled input setting. In order to do that, we extend our learnable selection module to the multi-modal setting following Section 4.3.4 to generate key frames and key words from the original video inputs. We first get a pool of candidate inputs from the raw video. The candidate frames are directly sampled from the videos, while the candidate key words are extracted using CLIP-based model, which finds the closest words or phrases using nearest embedding matching. We use all the phrases and words from the iVQA training set as the vocabulary dictionary to choose words from. To better demonstrate the results, we use the format of few-word or few-frame inputs. For visual frame inputs, we process with the same method as before. For textual inputs, we treat 5-word as one unit. 5-word/sec is the average reading speed for adults, which consumes similar attention from watching a frame. So 5-word and one single frame could be thought of as equivalent in consuming user attention. We combine the units into a sequence, then apply the same selection method for word selection. For multi-modal setting, we concatenate the frames and word units as a multi-modal sequence and select from both. We fix $\tau = 0.1$ for training the models.

Our results are shown in Table 4.4. For single-modality inputs, we similarly observe an increasing performance trend with increasing number of inputs. Even with very few inputs, the VideoQA performance is very close to the upper bound from dense inputs. We can also observe a boost of performance from increasing density of inputs on both modalities at sparsified levels, which validates the effectiveness of our sparsification techniques. In general, the visual inputs perform stronger than textual inputs, which is mainly due to the fact that visual signals are much more informative. On the other hand, we can still observe an increase of performance from adding even very few multi-modal inputs. For example, adding only 5-word to the visual snippet could still get some performance gains. This implies the complimentary manner from different modalities from the perspective from strictly controlled inputs. Noticeably, as an intermediate output from our learnable selection, we can get a few-frame and few-word summarization of the original video, which is human-interpretable. We provide more examples and analysis in the following section to demonstrate this advantage.

4.4.5 Qualitative Analysis

Here we provide visualizations on the selected frame and/or key words from iVQA dataset. For illustration purposes, we present the result for single frame selection and 10-word extraction in Figure 4.5, along with their associated questions and the predicted answer. Answer in green color means the system correctly predicts the answer, while red color means the system predicts the wrong answer and the ground truth is in parenthesis. In the successful cases, the sparsified output is able to capture an appropriate figure for the topic, and the texts also contain words related to the answer, which leads to the correct answer. In the first failure case, even though the selected frame contains information related to the answer "shirt", the textual component is a distraction, and the system generates an answer more related to the key words which are closely describing pipes. In the second failure case, the generated key frame and words are both irrelevant to the question. This is probably because the question itself is asking something minor (since most of the video contents are about the architectures and surroundings) while the model is trained to get information that is of major interest for the overall dataset and task.

Additionally, we analyze the token importance using the tool provided by [109] which cal-

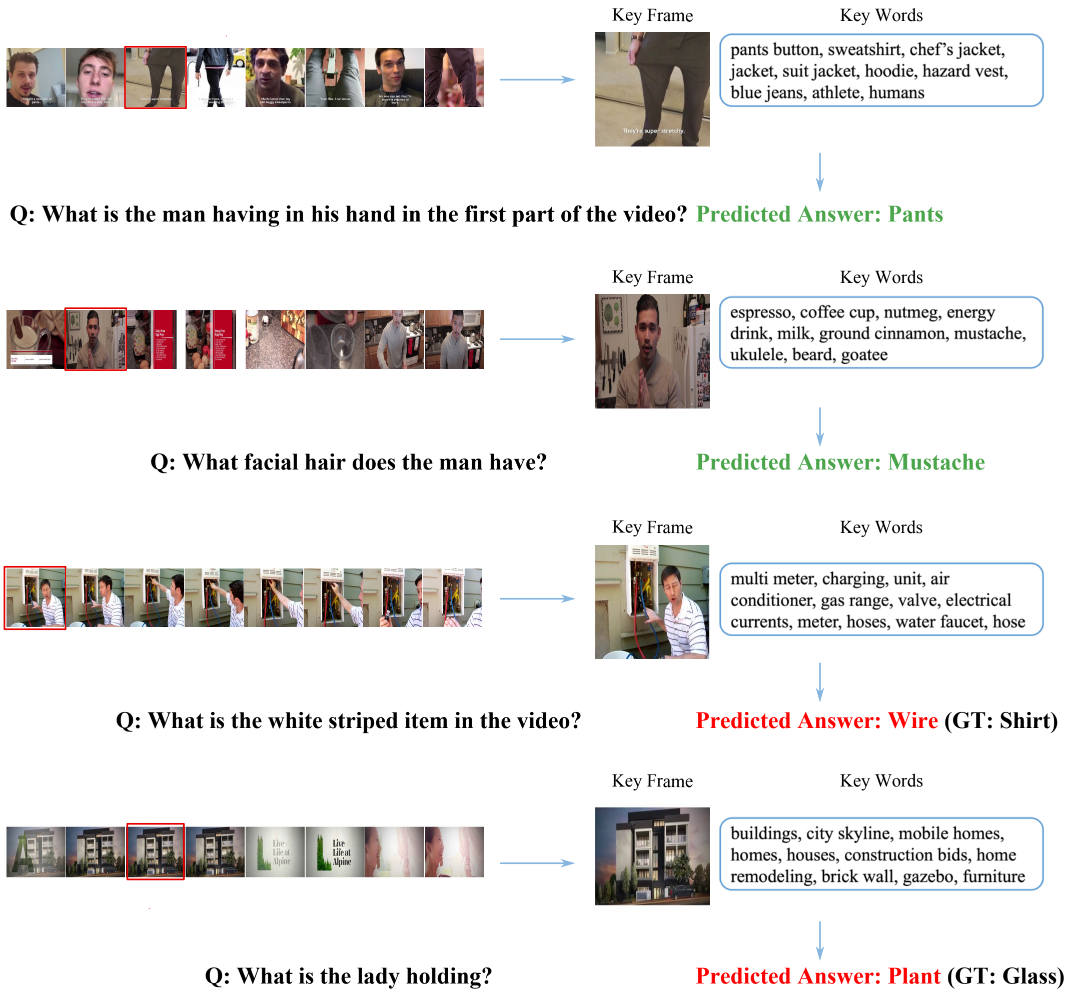

Key Frame
Key Words
pants button, sweatshirt, chef's jacket, jacket, suit jacket, hoodie, hazard vest, blue jeans, athlete, humans
They're super stretchy.
Q: What is the man having in his hand in the first part of the video?
Predicted Answer: Pants
Key Frame
Key Words
espresso, coffee cup, nutmeg, energy drink, milk, ground cinnamon, mustache, ukulele, beard, goatee
Q: What facial hair does the man have?
Predicted Answer: Mustache
Key Frame
Key Words
multi meter, charging, unit, air conditioner, gas range, valve, electrical currents, meter, hoses, water faucet, hose
Q: What is the white striped item in the video?
Predicted Answer: Wire (GT: Shirt)
Key Frame
Key Words
buildings, city skyline, mobile homes, homes, houses, construction bids, home remodeling, brick wall, gazebo, furniture
Q: What is the lady holding?
Predicted Answer: Plant (GT: Glass)

Figure 4.5: Qualitative examples on iVQA videos. Single frame and ten-word summary is generated from the original video for Video Question Answering task. First two examples demonstrate successful cases where both visual and textual signals signals are able to capture the question-relevant information. The last two examples show some failure cases where visual and/or textual signals are distracted from the question.

culates the importance score of each input token w.r.t. to the task prediction. In Figure 4.6, we provide some visualization examples where question words and video frame inputs are highlighted according to their importance scores. For illustration purposes, we only sample 10 frames in each sample. Words or frames that are of darker green color means they contribute more to the prediction. From the given examples, we can see that not every video frame is of significant importance. The model is able to discard frames that do not contain any useful information for the question (e.g., in the second example, only the frames showing the fingers are contributing). On the other

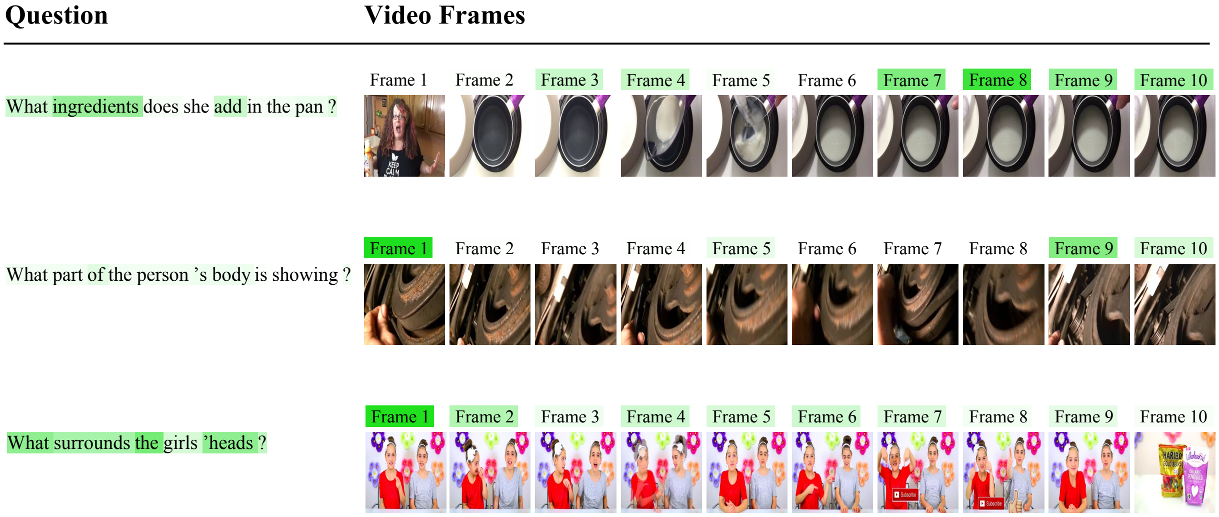


Figure 4.6: Frame importance visualization. Darker color means the corresponding word/frame is of more importance to predict the answer. We can see that the model is able to discard some repetitive frames or frames that are not relevant.

hand, in the example where the scene is relatively stable, we can also observe that the model focuses mostly on one of these similar frames (as in the third example), while the rest seems to be diverging. These observations show the potential of dropping unnecessary video inputs to improve the efficiency, which validates our motivation.

4.5 Summary

In this chapter, we show how we can adapt a transformer-based video-language model $\mathcal{M}$ towards sparsified representation by designing a learnable sparsifier δ for the inputs. We propose to use a learnable selection module to adaptively select the best and representative input. This allows us to get multi-length input on different types of modalities. In our experiments, we characterize video question answering from the perspective of sparsified inputs. We analyze the current Video Question Answering benchmarks, where we can preserve a fair performance with a small input budget. We also observe a complimentary behaviour between modalities, i.e., adding one modality always helps with the other, meaningfully shows the potential of improving data efficiency under various video representation types.

Chapter 5: Model Adaptation for Open Recognition

A practical vision system would face a diverse open environment, where there could be new objects of interest as well as disturbance information. It is important to build a robust model that deals with both signals. In this chapter, we extend a recognition model with the capabilities of new object recognition and outlier detection. Figure 5.1 shows a high-level illustration of our idea where we instantiate δ as a negative prototype generator that models outliers towards robust recognition.

5.1 Introduction

With the emergence of large-scale image datasets [110, 5, 111], deep learning has achieved great success in various vision tasks [112, 113, 114, 115, 116, 117]. Current recognition systems usually assume a predefined set of classes with sufficient number of labeled data. Each testing sample is supposed to belong to these predefined classes so that the systems only need to perform closed-set classification.

In real-world applications, we face more challenging recognition scenarios. First, sufficient labeled training data are hardly guaranteed due to high cost of data collection and possibly limited access to sensitive or rare data. Few-shot learning [118, 119, 120, 121] (FSL) typically tackles data insufficiency scenario by fast adaptation of recognition system to new classes with access to very few (e.g., only one) labeled instances. But FSL still holds a closed-set assumption.

In addition, there are existing efforts that aim to provide a recognition system with the ability to handle out-of-distribution testing samples. Open-set recognition (OR) [122, 123, 124, 125, 126] considers the case where testing samples could come from other unknown source under a large-scale training setting. Real-world systems also need to detect queries from unknown classes (i.e.,

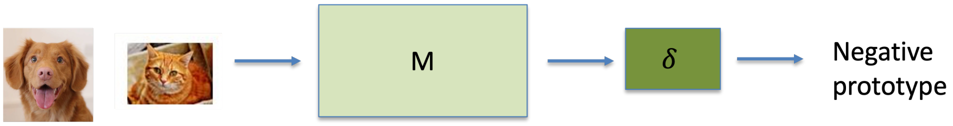


Figure 5.1: In this chapter, δ serves as the negative prototype generator, which estimates the detection boundary for outlier signals.

negative queries). Current OR methods typically learns an open-set classifier by either calibrating prediction scores or synthesizing negative queries. They rely on large amount of data, including those for the unseen classes, to avoid overfitting and estimate distribution properly. But with only a few labeled instances, it becomes hard to do so. Hence direct application of OR methods under few-shot setting degrades the performance significantly[127, 128].

We aim to adapt a recognition model $\mathcal{M}$ pre-trained on many-shot classes for solving both challenges, i.e., few-shot and open-set recognition (FSOR). The goal of FSOR is to both 1) accept & recognize *positive queries* from few-shot classes with very few labeled samples and 2) detect *negative queries* from undisclosed (*negative*) classes. The number of negative classes and negative queries might vary, and we study the impact of the size of negative classes and queries in our empirical analysis. Previous FSOR methods [127, 128] provide meta-learning-based solutions for learning threshold-based negative detector. They calibrate few-shot close-set classifier and output a rejection score for each testing sample. A sample is rejected if the rejection score is above a certain rejection threshold, which has to be manually defined. However, as shown in Fig. 5.2, a good recognition performance relies heavily on a good choice of threshold: (a) a few-shot classifier may have similar detection score for a negative query and a positive query, where different thresholds need to be set separately; (b) to reject a negative query, a threshold works properly for one task may fail in other tasks. In summary, threshold tuning could be a challenging process as different FSOR tasks contain different few-shot classes that may need very different rejection powers to determine outliers.

Instead, we propose to solve this issue following our general $\mathcal{M} + \delta$ framework. We propose to integrate threshold tuning into the learning process for FSOR, yielding a threshold-free solu-

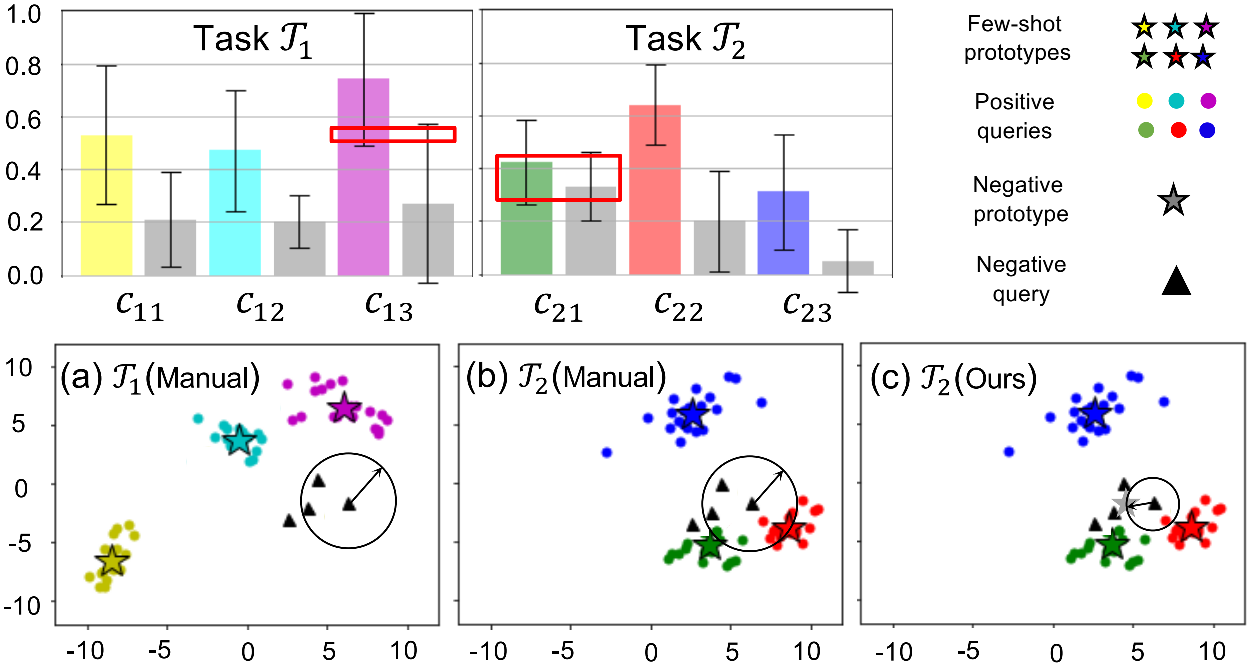


Figure 5.2: (Up): For each few-shot class in few-shot open-set recognition (FSOR) task $\mathcal{T}_1$ and $\mathcal{T}_2$, we calculate the detection scores (similarity) of both negative and positive queries and find their mean and standard deviation. (We use the same negative queries for both tasks.) However, a negative query may have similar detection score to the positive queries (highlighted by red box). (Down): In addition, to reject a negative query, existing FSOR method relies on a manually selected threshold. However, a rejection threshold working properly for $\mathcal{T}_1$ may fail in $\mathcal{T}_2$. Instead, we propose to learn a negative prototype that automatically estimates a task-adaptive threshold for negative detection.

tion. We extend the few-shot classifier with additional prototypes that represent the negative class. Specifically, a negative generator δ is applied on few-shot class prototypes and learns negative prototypes across tasks via meta-learning, so that negative prototypes can serve as task-adaptive rejection boundaries for different FSOR tasks. A testing query is then rejected if the prediction scores on all few-shot classes are lower than that on the negative prototype. We study the design of negative generator and experimentally demonstrate an optimal solution that involves task-level information into the negative prototype envision. We also introduce the concept of conjugate task of FSOR where two FSOR tasks are considered conjugate if the few-shot class in one task can be used to simulate unknown sources in the other. To this end, we propose a *conjugate training* strategy to facilitate the learning process. Moreover, we consider a new but more challenging problem, generalized FSOR (GFSOR), where the recognition system needs to classify on both many-shot and few-shot classes as well as reject negative samples. In this case, negative prototypes are generated from both many-shot and few-shot classes. We name our method of learning negative prototypes as *task-adaptive negative class envision*. Our method is validated by extensive experiments on public benchmarks for both FSOR and GFSOR problems, where our approach is able to achieve SOTA performance on FSOR benchmarkes, and also on shown effective on our newly formulated GFSOR problem.

In the following sections, we will discuss related literature in FSL, OR and FSOR (Sec. 5.2); In Sec. 5.3 we formally define FSOR and GFSOR tasks and go over existing threshold-based meta-learning solutions. In Sec. 5.4 we present our approach of task-adaptive negative envision. Finally in Sec. 5.5 we demonstrate the experimental analysis and results of our approach.

5.2 Related Works

Few-Shot Learning. FSL aims for fast adaptation to new recognition task with very few labeled examples. Meta-learning is widely used to learn transferable knowledge upon a set of tasks using episodic training. There are mainly two types of meta-learning approaches: 1). Optimization-based method [118, 129, 130, 131] modifies the gradient back-propagation so that the parameter updates can be more sensitive to the few training examples; 2). Metric-based methods [120, 121, 119, 132, 133, 134, 135, 136] learns to obtain an optimal metric space so that a class with the highest similarity is assigned to the query. As an extension on FSL, generalized FSL (GFSL) [137] learns to expand many-shot classifier with novel classes using a few training data. Both (G)FSL hold a closed-set assumption where testing queries belong to novel classes (or many-shot classes in GFSL). Our work instead extends (G)FSL to open-set setting.

Large-Scale Open-Set Recognition. OR aims to learn a classifier sensitive to negative queries that come from unknown classes. OR methods typically include class probability re-calibration [123, 124, 138] and negative sample synthesis with generative methods [139, 140]. Those methods typically assume large number of training data. A most relevant work to us [141] also proposes to augment classifier to learn adaptive rejection thresholds. But it relies on large-scale data to train the augmented classifier from scratch, while ours generates negative prototypes based on few-shot classes. Direct application of OR methods to few-shot setting fails or degrades the performance [127, 128] mainly due to over-fitting. Our work instead provides a few-shot-specific OR solution to deal with limited data.

Few-Shot Open-Set Recognition. To bridge FSL and OR, recently [127] provides a meta-learning-based solution for FSOR that introduces an open-set loss in the meta-training process to calibrate few-shot prototype-based classifier. [128] improves the limitation of negative sampling in [127] by imposing a transformation consistency regularization on few-shot samples. However, their methods are threshold-based, which require careful selection of thresholds to perform good recognition. Instead, we propose a threshold-free solution to overcome the challenge.

5.3 Problem Formulation

With only a few labeled training samples, few-shot open-set recognition (FSOR) aims to 1) detect negative queries that come from *unknown* sources and 2) correctly classify positive queries. Formally, a FSOR task can be denoted as $\mathcal{T} = (\mathcal{S}, Q^f, Q^n | C^f)$ where C^f refer to the few-shot classes that have few labeled training samples (also called supports): $\mathcal{S} = \cup_{c \in C^f} \mathcal{S}_c$. The goal is to learn a recognition model with the supports so that during testing time, it can successfully classify positive queries Q^f and detect negative queries Q^n. We denote $Q = Q^f \cup Q^n$ as the entire query set. We call a FSOR task *N-way K-shot* if we have $|C^f| = N$ and $|\mathcal{S}_c| = K$ for all $c \in C^f$. Briefly speaking, the only difference between FSOR and conventional FSL tasks is that FSOR has additional negative queries that need to be rejected.

Existing FSOR approaches [127, 128] are built upon the popular metric-based FSL method ProtoNet[121], and our approach also follows the same fashion. Below we provide more context on ProtoNet.

ProtoNet[121] learns a prototype-based few-shot classifier. In detail, each few-shot class $c \in C^f$ is represented by a prototype $\mathbf{p}_c$, computed by the average of K support features: $\mathbf{p}_c = \frac{1}{K}\sum_{s\in\mathcal{S}_c} f(s)$, where f is a feature extractor and $f(s) \in \mathcal{R}^d$. Then, all prototypes $\mathbf{P}^f = \{\mathbf{p}_c\}_{c\in C^f}$ build up a closed-set classifier where a positive query $q \in Q^f$ can be classified by nearest neighborhood search, *i.e.*,

$$\text{argmax}_c\big(\{f_s(f(q), \mathbf{p}_c)\}_{c\in C^f}\big), \tag{5.1}$$

where $f_s(\cdot,\cdot)$ is a function to measure the closeness between two inputs, *e.g.*, cosine similarity.

In order to learn an open-set classifier, existing FSOR approaches [127, 128] calibrate the few-shot close-set classifier to get per-class detection scores and reject via thresholding. As illustrated in Fig. 5.3(a), for threshold-based FSOR methods, a threshold θ_m needs to be manually set and a negative query $q \in Q^n$ will be rejected if all of the detection scores are below θ_m, *i.e.*, $\max\big(\{f_s(f(q), \mathbf{p}_c)\}_{c\in C^f}\big) < \theta_m$.

In addition to FSOR tasks, we further consider a more realistic situation where both few-shot classes C^f and many-shot classes C^* (i.e., containing large amount of labeled data) exist, resulting an imbalanced distribution. To this end, we formulate the generalized few-shot open-set recognition (GFSOR) task $\mathcal{T}^* = (\mathcal{S}, Q^*, Q^f, Q^n|C^*, C^f)$ where Q^* are queries from C^*. And the goal is to correctly classify both $Q^* \cup Q^f$ and reject negative query Q^n. Similarly, we call a GFSOR task *N-way K-shot* if we have $|C^f| = N$ and $|\mathcal{S}_c| = K$ for all $c \in C^f$.

We also provide Table 5.7 summarizing the symbols for clearer reference.

5.4 Approach

Here we present our threshold-free approach towards (G)FSOR. We first provide an overview of how to use negative envision to estimate task-adaptive rejection boundaries; then we provide a list of negative generators used in practice; finally we introduce conjugate training which encourages the learning process from task mutual supervision.

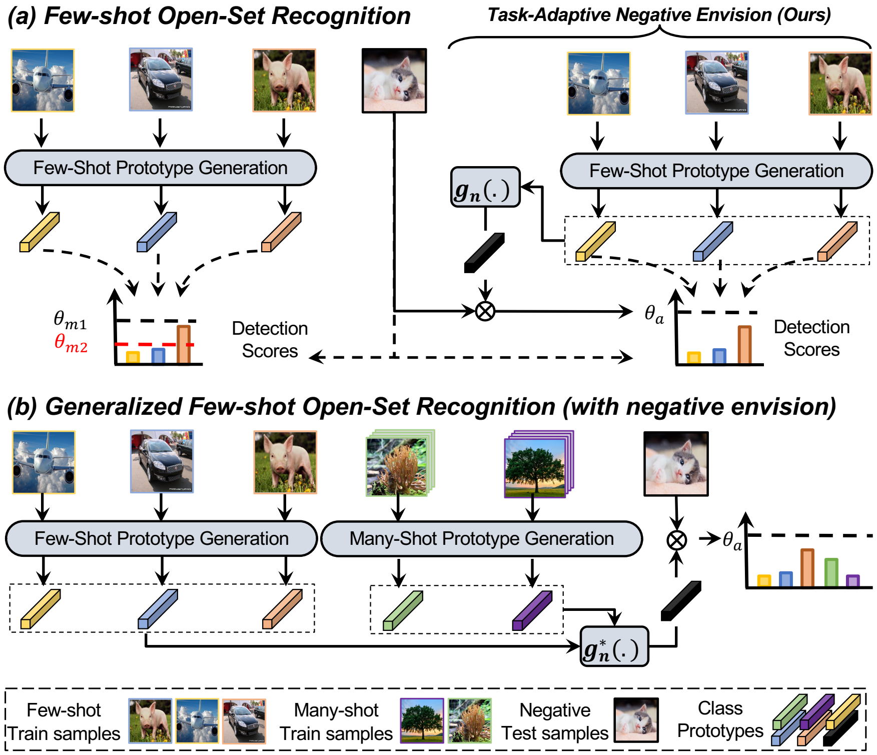

(a) Few-shot Open-Set Recognition
Task-Adaptive Negative Envision (Ours)
Few-Shot Prototype Generation
Few-Shot Prototype Generation
$g_n(.)$
θ_{m1}
θ_{m2}
Detection Scores
θ_a
Detection Scores
(b) Generalized Few-shot Open-Set Recognition (with negative envision)
Few-Shot Prototype Generation
Many-Shot Prototype Generation
θ_a
$g_n^*(.)$
Few-shot Train samples
Many-shot Train samples
Negative Test samples
Class Prototypes

Figure 5.3: (a) Comparison between conventional threshold-based FSOR methods and ours with negative envision. Conventional methods reject a sample if the detection scores of all classes are below a carefully selected threshold (θ_{m1}). An improperly selected threshold (θ_{m2}), on the other hand, would result in recognition failure. Instead, we propose to envision a negative prototype to *learn to* estimate threshold (θ_a) for each instance dynamically within the task. (b) For a GFSOR task, the negative prototype is generated from both many-shot and few-shot prototypes.

5.4.1 Overview

Fig. 5.3 provides an overview of our *Task-Adaptive Negative Envision* approach and how it compares to threshold-based methods. Threshold-based methods [127, 128] calculate per-class detection scores and manually define a threshold for rejection. Without a carefully cherry-picked a threshold for each task, it's hard to successfully detect $q \in Q^n$ across different tasks (Fig. 5.3(a)). Instead, we expand classifier with negative prototype $\mathbf{p}^-$ that are computed from few-shot class prototypes $\mathbf{P}^f$ via a negative generator $g_n(\cdot)$. When a query comes in, it's able to automatically calculate a task-specific threshold from the negative prototype:

$$\theta_a = f_s(f(q), \mathbf{p}^-). \tag{5.2}$$

Then, a negative query $q \in \mathcal{Q}^n$ will be rejected if $\max\big(\{f_s(f(q), \mathbf{p}_c)\}_{c \in C^f}\big) < \theta_a$. As such, rejection boundaries are dynamically estimated with respect to few-shot classes C^f and support instances $\mathcal{S}$. Our approach can be also applied to GFSOR tasks where the negative prototype are generated from both few-shot and many-shot class prototypes to get task-adaptive threshold with respect to both few-shot and many-shot classes.

5.4.2 Negative Generator

To find the best negative generator $g_n(\cdot)$, we explore different choices which we describe below in detail.

MLP

We start with a simple generator that consists of a single MLP layer applied on averaged class prototypes, i.e.,

$$\mathbf{p}^- = f_n(\mathbf{p}^-_{avg}), \;\; \mathbf{p}^-_{avg} = \frac{1}{N} \sum\nolimits_{c \in C^f} \mathbf{p}_c \tag{5.3}$$

where f_n is a MLP that takes $\mathbf{p}^-_{avg}$, the mean of $\mathbf{P}^f$, as input so that $\mathbf{p}^-$ is independent from the prototype order. Meanwhile, we set $\mathbf{p}^-_{avg}$ as a naive baseline (**AVG**) as $\mathbf{p}^-_{avg}$ is also order-independent.

ATT

Transformers[105] are proved effective in exploiting relations, which is also independent from the input order (without positional encoding). We apply a standard Transformer attention block over few-shot class prototypes to generate the negative prototype. Specifically, we calculate the self-attention weight between class prototypes, i.e.,

$$\mathbf{A}_{(\mathbf{P}^f, \mathbf{P}^f)} = \frac{1}{\sqrt{d}} (\mathbf{P}^f \mathbf{K}_q (\mathbf{P}^f \mathbf{K}_k)^T),$$

where $\mathbf{A}_{(\mathbf{P}^f,\mathbf{P}^f)} \in \mathcal{R}^{|C^f|\times|C^f|}$ is the attention weights matrix, and $\mathbf{K}_q, \mathbf{K}_k \in \mathcal{R}^{d\times d}$ are trainable linear projection kernels. Then, we normalize the weights and output

$$\mathbf{P}^{'} = \mathbf{P}^f + \sigma(\mathbf{A}_{(\mathbf{P}^f,\mathbf{P}^f)})(\mathbf{P}^f\mathbf{K}_v),$$

where $\sigma(\cdot)$ is a softmax function for each row in $\mathbf{A}_{(\mathbf{P}^f,\mathbf{P}^f)}$ and $\mathbf{K}_v \in \mathcal{R}^{d\times d}$ is another trainable linear projection kernel. Then, we feed the average of $\mathbf{P}^{'}$ to a MLP function f_n to get the negative prototype $\mathbf{p}^-$.

ATT-G

The above generators is suitable for FSOR problem. Now we consider the more challenging GFSOR task. Directly employing the above methods may introduce bias towards C^* as C^* has plenty of training samples and the prototype $\mathbf{P}^*$ can be better-estimated compared against the few-shot prototypes $\mathbf{P}^f$. Hence we need another negative generator compatible with GFSOR , which should take care of both C^* and C^f. We build our ATT-G generator on top of a popular GFSL method [137], which uses an attention mechanism to calibrate few-shot prototypes $\mathbf{P}^f$ with $\mathbf{P}^*$. Specifically, we follow [137, 142, 143] to first train a network under large-scale classification task using the labeled samples of C^* (*i.e.*, pre-training) and use the weight in the last linear layer as many-shot class prototypes $\mathbf{P}^*$. Then we apply the attention block between $\mathbf{P}^f$ and $\mathbf{P}^*$ to generate the negative prototype $\mathbf{p}^-$, i.e.,

$$\mathbf{A}_{(\mathbf{P}^f,\mathbf{P}^*)} = \frac{1}{\sqrt{d}}(\mathbf{P}^f\mathbf{K}_q(\mathbf{P}^*\mathbf{K}_k)^T), \tag{5.4}$$

$$\mathbf{P}^{'} = \mathbf{P}^f + \sigma(\mathbf{A}_{(\mathbf{P}^f,\mathbf{P}^*)})(\mathbf{P}^*\mathbf{K}_v), \tag{5.5}$$

and $\mathbf{p}^-$ is similarly computed by feeding the average of $\mathbf{P}^{'}$ into a MLP f_n. Furthermore, we'd like to filter out task-irrelevant information by applying a channel-wise gating mechanism on top of

$\mathbf{P}^f$:

$$\mathbf{p}'_c = \mathbf{p}_c \odot \phi(f_g(\frac{1}{N-1} \sum\nolimits_{i \in C^f \backslash \{c\}} \mathbf{p}_i)), \tag{5.6}$$

for $c \in C^f$ where $\odot$ and ϕ denote element-wise multiplication and sigmoid operation, and f_g is a fully-connected layer. Then, we use the updated $\mathbf{P}^{f'}$ to replace the input $\mathbf{P}^f$ in Eq. 5.4. Finally, we follow the order of many-shot prototypes $\mathbf{P}^*$, few-shot prototypes $\mathbf{P}^f$, and negative prototype $\mathbf{p}^-$ to build the open-set classifier for a GFSOR task, where $\mathbf{P}^*$ are the weights in the last linear layer after pre-training.

SEMAN-G

Inspired by recent cross-modal FSL works [144, 145], we further explore how class semantics could help model negative class. Specifically, we use a cross-modal attention mechanism on top of **ATT-G**. For each class $c \in C^f \cup C^*$, we concatenate $\mathbf{p}_c$ with its word embedding $\mathbf{e}_c \in \mathcal{R}^w$ along the channel to have $\mathbf{z}_c = [\mathbf{p}_c, \mathbf{e}_c] \in \mathcal{R}^{w+d}$. Then we use $\mathbf{Z}^f$ and $\mathbf{Z}^*$ instead of $\mathbf{P}^f, \mathbf{P}^*$ in Eq. 5.4 to calculate the attention. And stick to Eq. 5.5 to take $\mathbf{P}^*$ as input since we are still comparing visual features for recognition.

Multiple Negative Prototypes

In addition, we can easily extend from single negative generation to multiple negative generation. Specifically, we can learn a set of generators $\{g_{n,i}\}_{i=1}^M$ to generate multiple negative prototypes for each task. For ATT, ATT-G, and SEMAN-G, to reduce the number of trainable parameters, we choose to share the linear projection kernels in attention mechanism used to calculate $\mathbf{P}^{'}$, but just train separate MLPs $\{f_{n,i}\}_{i=1}^M$ to synthesis multiple negative prototypes. In this way, we get multiple thresholds $\{\theta_{a,i}\}_{i=1}^M$. Then the maximum threshold $\theta_a = \max\big(\{\theta_{a,i}\}_{i=1}^M\big)$ is used as the final threshold for open-set recognition.

5.4.3 Conjugate Training

Here we present our conjugate training strategy towards (G)FSOR. Conjugate training is built upon the standard FSOR meta-training approach [127, 121]. We first go over the standard FSOR meta-training then introduce our method.

Standard FSOR Meta-Training

Standard FSOR meta-training strategy [127, 121] trains the model by simulating FSOR tasks from the given *base* dataset $\mathcal{D}^B$. Specifically, it trains on a set of tasks sampled from the *base* dataset $\mathcal{D}^B$ where the images are from *base* classes C^B. Within an N-way K-shot FSOR task $\mathcal{T}$, unknown sources are simulated using a different set of N classes C^n, *i.e.*, $C^n \subset C^B - C^f$ where $|C^n| = N$, and then randomly sample images belonging to C^n from $\mathcal{D}^B$ for Q^n. Then the model is trained using an objective, typically an open recognition loss within the sampled $\mathcal{T}$. The standard FSOR meta-training can be generalized to GFSOR. For a GFSOR task $\mathcal{T}^*$, we can sample C^f and C^* from C^B and then simulate unknown sources as $C^n \subset C^B - (C^f \cup C^*)$ where $|C^n| = N$. And a GFSOR training objective may be specified to learn the model for GFSOR. Note that, during inference time (i.e., meta-testing), tasks are sampled from *novel* dataset $\mathcal{D}^N$ where the images are from classes C^N and no sample from C^N is seen during meta-training, *i.e.*, $C^B \cup C^N = \varnothing$.

Conjugate Tasks

The idea of conjugate training is to sample task pairs whose few-shot examples of one task are used as the negative source of the other. Formally, we define two tasks $\mathcal{T}_1 = (\mathcal{S}_1, Q_1^f, Q_1^n | C_1^f)$ and $\mathcal{T}_2 = (\mathcal{S}_2, Q_2^f, Q_2^n | C_2^f)$ as a *conjugate task pair* when $Q_1^n = Q_2^f$ and $Q_1^f = Q_2^n$, *i.e.*, the few-shot classes C_1^f (C_2^f) in $\mathcal{T}_1(\mathcal{T}_2)$ is used as the negative source in task $\mathcal{T}_2(\mathcal{T}_1)$. For a conjugate GFSOR task pair $(\mathcal{T}_1^*, \mathcal{T}_2^*)$, in addition, $\mathcal{T}_1^*$ and $\mathcal{T}_2^*$ share the same many-shot class C^* and its queries Q^*.

Conjugate Training Loss

We use a standard cross-entropy loss $\mathcal{L}_{CE}(\cdot,\cdot)$ [146]. For an FSOR task $\mathcal{T}$, we use cosine similarity as f_s and use $Q^f \cup Q^n$ to perform $(N+1)$-way classification. For each positive query $q \in Q^f$, we learn to maximize the class score of its label category by minimizing $\mathcal{L}_{CE}(y_q, q)$ where $y_q \in \{1, ..., N\}$ is the class label of q. For each negative query $q \in Q^n$, we set its ground truth label as $N+1$ and maximize the threshold θ_a by minimizing $\mathcal{L}_{CE}(N+1, q)$. During conjugate training, we consider the dependency of negative sampling mentioned in [128]. Without loss of generality, for a positive query $q \in Q_2^f$ belonging to class $c_n \in C_2^f$ in $\mathcal{T}_2$, it is used as the negative query and is trained to have high similarity with the negative prototype in $\mathcal{T}_1$.

With a simple classification loss, the negative prototypes are optimized to learn a tight rejection boundary for a specific task. Besides, for the attention-based generators, we also regularize the intermediate variables $\mathbf{P}'$ as class-specific negative prototypes. For each prototype $\mathbf{p}'_c \in \mathbf{P}'$ generated from a positive prototype $\mathbf{p}_c$, we can think of $\mathbf{p}'_c$ as the negative prototype for class c. Then, for each $\mathbf{p}'_c$, we minimize its similarity with queries of class c and maximize its similarity of negative queries with a binary cross-entropy loss $\mathcal{L}_{BCE}$

$$\begin{aligned}\mathcal{L}_{neg}(c) = &\frac{1}{|Q_{c,1}^f|} \sum\nolimits_{q \in Q_{c,1}^f} \mathcal{L}_{BCE}(0, f_s(f(q), \mathbf{p}'_c)) \\ &+ \frac{1}{|Q_1^n|} \sum\nolimits_{q \in Q_1^n} \mathcal{L}_{BCE}(1, f_s(f(q), \mathbf{p}'_c))\end{aligned}$$

where $Q_{c,1}^f = \{q | q \in Q_1^f, y_q = c\}$ and y_q denotes the class label of q. Finally, without loss of generality, for $\mathcal{T}_1$ in the conjugate task pair $(\mathcal{T}_1, \mathcal{T}_2)$, we have

$$\mathcal{L}_{\mathcal{T}_1} = \mathcal{L}_{CE}(Q_1^f \cup Q_1^n) + \frac{1}{|C_1^f|} \sum\nolimits_{c \in C_1^f} \mathcal{L}_{neg}(c), \tag{5.7}$$

and the total conjugate training loss is $\mathcal{L} = \mathcal{L}_{\mathcal{T}_1} + \mathcal{L}_{\mathcal{T}_2}$.

Similarly, for the network trained on GFSOR tasks, given a conjugate task pair $(\mathcal{T}_1^*, \mathcal{T}_2^*)$, we have

$$\mathcal{L}^*_{\mathcal{T}^*_1} = \mathcal{L}_{CE}(Q^*_1 \cup Q^f_1 \cup Q^n_1) + \frac{1}{|C^f_1|} \sum\nolimits_{c \in C^f_1} \mathcal{L}_{neg}(c),$$

where the class label for a negative query is $N + 1 + |C^*|$, and the total loss being $\mathcal{L}^* = \mathcal{L}^*_{\mathcal{T}^*_1} + \mathcal{L}^*_{\mathcal{T}^*_2}$. In this way, our *conjugate training* involves the class-correlation during network training.

5.5 Experiments and Analysis

Datasets

For FSOR tasks, we evaluate on two widely used public benchmarks: MiniImageNet [120], TieredImageNet [147]. MiniImageNet [120] contains 100 classes and the class split for (meta-training, meta validation,meta-testing) is (64,16,20). Each class has 600 images. TieredImageNet [147] contains 608 classes and the class split is (351, 97, 160) while the *base* dataset contains around 450K images. We evaluate GFSOR on MiniImageNet [120] and set the base classes during meta-training as the many-shot classes during meta-testing. We follow [137] and use another 300 images for each *base* class for the GFSOR simulation. All images for the two datasets are sized to 84×84. For **SEMAN-G**, we extract word embedding using GloVe [148]. More details of the datasets can be found in the supp. material.

Implementation Details

We use ResNet12 [149] network as the feature backbone. Following [142, 143], we pre-train the ResNet12 and a classifier (a linear layer) with cross-entropy loss and a self-supervised rotation loss on the base set under fully-supervised classification task for 90 epochs using a SGD optimizer with learning rate 0.05 decayed by 10 at epoch 60. The weights in the linear layer are used as base-class many-shot prototypes $\mathbf{P}^*$ for **ATT-G** and **SEMAN-G**. Through the experiment, we interchangeably use the term *base* and *many-shot*. During meta-training, the learning rate is set to 0.0001 for the ResNet12 feature extractor, and 0.05 for all other layers in the negative prototype generator. The entire network is trained for $18k$ tasks with a SGD optimizer, where the learning rate is decayed when the validation accuracy saturates. During meta-testing, we follow [128, 127] to randomly sample 600 tasks, and report the average value with 95% confidence interval for all the metrics. We use cosine similarity [137] as the similarity function to compute per-class prediction scores. For FSOR evaluation, we follow [127] to sample training and testing tasks, where we set $N = 5$ and $K = 1, 5$. For each task, we sample 15 positive queries from each few-shot class. For negative detection, we sample 5 negative classes with each containing 15 negative queries. For each GFSOR task, in addition to query samples from few-shot and negative classes, we select 75 query samples for the *base* classes (each class has at least one sample). Following the setup in [120], we randomly sample 1000 5-way GFSOR tasks to learn to generate an open-set classifier for the union of 64 base classes and 5 novel classes.

Metrics

To measure the standard closed-set classification performance, we report top-1 accuracy for FSOR tasks over few-shot classes. For GFSOR, we follow the protocol defined in [150, 151], and report both arithmetic mean and harmonic mean between mean accuracy of base samples and mean accuracy of novel samples. In addition, we report Δ-value to measure the accuracy drop between prediction among specific classes (base or novel classes) and prediction among all combined classes, where a better classifier is supposed to balance the prediction and have low Δ-value. To measure the negative detection performance, we follow the protocol in [127, 128] to report AUROC (area Under ROC Curve). To measure the overall open-set recognition performance, we follow the protocol in [141] to report *macro-averaged F1-scores* on all many-shot/few-shot and negative classes.

5.5.1 FSOR Results

Comparison of Negative Generator

We first compare different choices of negative generators on FSOR tasks in Tab. 5.1. Note that **ATT-G** and **SEMAN-G** can also be applied for FSOR and compared to other approaches since all

Table 5.1: F1-score comparison on 5-way 1-shot FSOR tasks on Mini-ImageNet. *: our implementation.

Neg. Gen.	0.3	0.5	0.7	0.9
PEELER	34.01	40.71±0.59	41.44	35.30
Dynamic+PEELER	38.19	45.34±0.64	44.10	30.95
Neg. Gen.	Single Neg.		Multi Neg.	
AVG	45.6±0.71		-	
MLP	46.12±0.74		47.21±0.72	
ATT	46.38±0.73		47.29±0.70	
ATT-G	47.03±0.74		48.19±0.71	
SEMAN-G	47.95±0.72		50.10±0.69	

models are trained using base set only (including base prototype) and doesn't use any extra data. We can see that attention-based methods are effective in negative generation as they are good at modeling inter-class relations. Adding class semantic information is also beneficial for discrimination. Meanwhile, by enabling multiple negative prototypes, *i.e.*, $M = 5$, we can automatically estimate the threshold θ_a with more flexibility, which then achieve consistent performance gain in F1-score, when compared generating a single negative prototype $M = 1$. For the following experiment results, we set $M = 5$ for our methods.

Comparison with Threshold-based Classifier

For threshold-based methods, threshold-tuning is crucial to get good recognition performance. To evaluate the overall open-set recognition, we compare macro-weighted F1-score. For threshold-based approaches, we define different thresholds and compute the corresponding F1-score. We illustrate our result in both Tab. 5.1 and Fig. 5.4(a), where we consider two threshold-based classifiers PEELER[127] and a combination of PEELER and our ATT-G method baseline, Dynamic[137], which calibrates novel prototype with base-class prototypes. In detail, we apply PEELER's open-set training strategy on top of Dynamic.

We implement PEELER and Dynamic+PEELER using the same feature extractor (ResNet12) as ours, which is first pre-trained on the base set. (1) PEELER: Our training and sampling strategy are built upon PEELER [127], and hence pick it as one threshold-based method for comparison in Fig. 3 and Tab. 1. In detail, we implement PEELER using our pre-trained feature extractor, and then perform meta-training with PEELER's open-set loss. Then rejection score is calculated as $\text{argmax}_{y\in\ \mathcal{C}^f}\, p(x|y)$. (2) Dynamic+PEELER: To compare to ATT-G more fairly, we combine PEELER with Dynamic as another threshold-based approach in Fig. 3 and Tab. 1. Specifically, we use PEELER's loss and training strategy on top of the calibrated few-shot prototypes. Then the rejection score is calculated as $\text{argmax}_{y\in\ \mathcal{C}^f}\, p(x|y)$.

In Fig. 5.4(a), we simulate 45k FSOR tasks and find their optimal rejection thresholds θ_m for the threshold-based approach Dynamic+PEELER. We plot the distribution of θ_m, which shows that it covers a wide range between 0 and 1. It demonstrates that different FSOR tasks may need very different rejection threshold in practice with current threshold-based approach. And the overall recognition performance largely depends on threshold selection, as is shown in Tab. 5.1. Our method instead automatically learn a task-adaptive rejection boundary, and we can see from Tab. 5.1 that all our negative envision instantiations outperform threshold-based methods. Fig. 5.4(b) further analyze the recognition behaviour under different openness [152]:

$$\text{openness} = 1 - \sqrt{\frac{2|C^f|}{2|C^f| + |C^n|}}$$

where we fix $|C^f| = 5$ and vary $|C^n|$ from 5 to 15. Similarly, we test for 600 randomly selected FSOR tasks and take the average. As is validated by Fig. 5.4, our method clearly outperforms threshold-based methods at all openness levels.

Comparison with SOTA Methods

We compare our method with other SOTA methods. The baselines we compare to include standard FSL methods (ProtoNet, FEAT), large-scale OR methods (OpenMax, CounterFactual), and existing FSOR methods (PEELER, SnaTCHer). We cite most of the baseline results from [128], and additionally compare to CounterFactual, a generative OR method which synthesize fake negative images and then train a $N + 1$ classifier. To apply in our FSOR setting, we first train its GAN network on base set and use the support set to synthesize fake images. The averaged fake

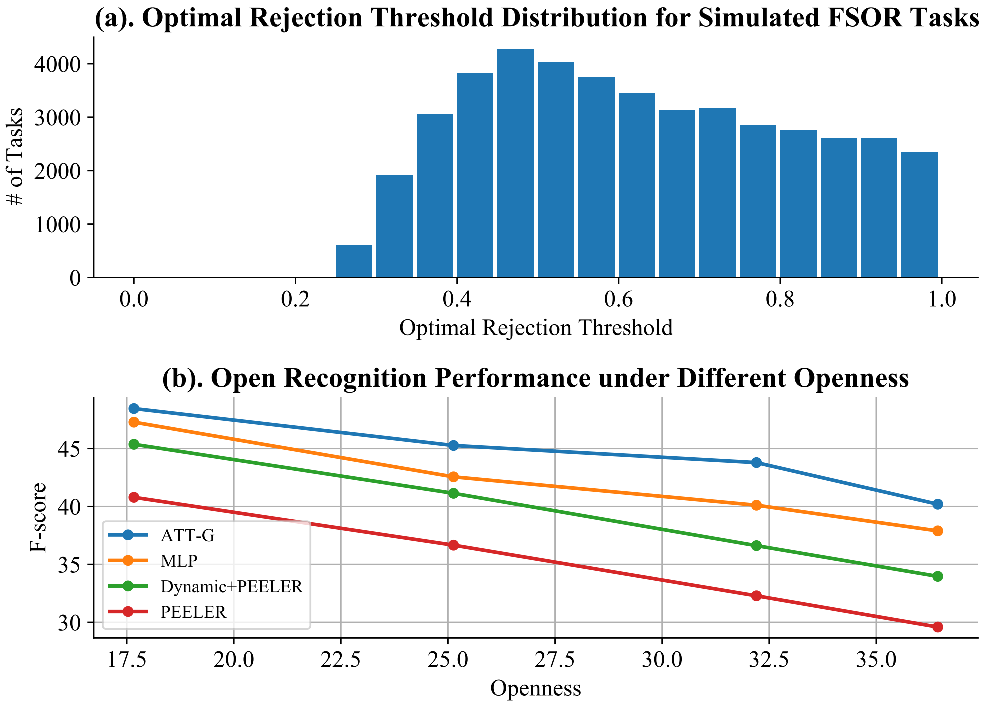


Figure 5.4: (a) Distribution of optimal rejection thresholds with Dynamic+PEELER on Mini-ImageNet. (b) Comparison of open recognition performance under different openness.

image feature is used as the negative prototype for FSOR.

Tab. 5.2 demonstrates the results. Standard FSL methods perform poorly in negative detection due to its closed-set nature. Large-scale OR methods yields unsatisfactory performance especially on 1-shot classification. Interestingly, CounterFactual gives a relatively fair performance on negative detection, which also validates our concept of negative envision. But it's still much worse than our few-shot-specific negative generation strategy, which validates that our approach better suits for the limited data scenario. Both **ATT-G** and **SEMAN-G** outperform other methods on Mini-ImageNet and get comparable result on Tiered-ImageNet.

Ablation Study on Conjugate Training

Tab. 5.3 shows the impact of conjugate training. We observe consistent improvement on all metrics and datasets, validating that conjugate training efficiently boosts the learning process by enabling mutual supervision from two tasks.

Table 5.2: 5-way 1-shot and 5-shot FSOR results. *: our implementation.

Algorithm	MiniImageNet,5-way				TieredImageNet,5-way			
	1-shot		5-shot		1-shot		5-shot	
	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC
ProtoNet [121]	64.01±0.88	51.81±0.93	80.09	60.39	68.26	60.73	83.40	64.96
FEAT [153]	67.02±0.85	57.01±0.84	82.02	63.18	70.52	63.54	84.74	70.74
OpenMax [123]	63.69±0.84	62.64±0.80	80.56	62.27	68.28	60.13	83.48	65.51
CounterFactual* [140]	63.7±0.83	64.17±0.88	81.44±0.54	71.58±0.76	70.08±0.94	71.04±0.80	85.36±0.60	78.66±0.62
PEELER [127]	65.86±0.85	60.57±0.83	80.61	67.35	69.51	65.20	84.10	73.27
SnaTCHer-T [128]	66.60±0.80	70.17±0.88	81.77	76.66	70.45	74.84	84.42	82.03
SnaTCHer-L [128]	67.60±0.83	69.40±0.92	82.36	76.15	70.85	**74.95**	85.23	80.81
ATT (ours)	67.64±0.81	71.35±0.68	82.31±0.49	79.85±0.58	69.34±0.95	72.74±0.78	83.82±0.63	78.66±0.65
ATT-G (ours)	68.11±0.81	72.41±0.72	83.12±0.48	79.85±0.57	70.58±0.93	73.43±0.78	85.38±0.61	81.64±0.63
SEMAN-G (ours)	**68.24±0.82**	**72.85±0.69**	**83.48±0.48**	**82.07±0.58**	**71.06±0.92**	74.27±0.77	**86.02±0.58**	**82.59±0.57**

Table 5.3: Ablate study on conjugate training. We report 5-way-1-shot FSOR result using ATT-G over three metrics on both datasets.

Dataset	Metric	w/o Conjugate	w/ Conjugate
Mini ImageNet	ACC	66.28±0.84	68.11±0.81
	AUROC	71.80±0.77	72.41±0.72
	F1-score	46.94±0.68	48.19±0.71
Tiered ImageNet	ACC	70.08±0.94	70.58±0.93
	AUROC	71.84±0.82	73.43±0.78
	F1-score	50.23±0.77	51.56±0.81

Table 5.4: 5-way generalized few-shot open-set recognition results on Mini-ImageNet.†: Implemented by CASTLE [154]. *: our implementation.

Algorithm	1-shot		5-shots		1-shot	5-shots	1-shot	5-shots
	Arith. Mean	Δ	Arith. Mean	Δ	Harmonic Mean		AUROC	
IFSL [150]	54.95±0.3	11.84	63.04±0.3	10.66	-	-	-	-
L2ML [155]	46.25±0.04	27.49	45.81±0.03	35.53	2.98±0.06	1.12±0.04	-	-
ProtoNet2 †	53.93±0.08	22.09	72.64±0.08	11.41	27.73±0.19	68.99±0.11	-	-
CASTLE [154]	66.48±0.11	9.94	76.25±0.09	8.14	64.29±0.14	**75.79±0.1**	-	-
DynamicFSL* [137]	60.85±0.14	12.97	73.1±0.13	10.92	60.13±0.13	69.8±0.09	67.56±0.17	72.86±0.11
ATT-G(ours)	65.49±0.13	11.25	75.51±0.10	10.97	63.94±0.12	73.89±0.12	73.12±0.16	77.22±0.13
SEMAN-G(ours)	**66.83±0.11**	10.24	**77.02±0.08**	9.78	**64.77±0.12**	75.62±0.09	**73.55±0.14**	**78.22±0.11**

5.5.2 GFSOR Results

In Tab. 5.4, we compare **ATT-G** and **SEMAN-G** with other standard methods on GFSOR tasks.

We implement some of the baseline methods, which are marked with * in the main paper. For fair comparison, all the re-implemented methods use the same feature extractor (ResNet12) as ours, which is first pre-trained on the base set.

We implement DynamicFSL [137] on our own using ResNet12 backbone, using the standard Transformer attention block. We calculate its accuracy in the normal way, while for AUROC, we take $\text{argmax}_{y \in C^f \cup C^*} p(x|y)$ as the rejection score.

Under the more challenging GFSOR setting, we achieve comparable GFSL classification accuracy with SOTA method and significantly improve the AUROC score which measures negative query detection. In addition, as GFSL methods are not trained to envision negative prototype but has more classes to recognize during evaluation, it will be challenging to manually set a threshold to reject negative queries while maintaining high classification accuracy. Thus, it is necessary to learn to dynamically generate threshold for each query for GFSOR.

5.5.3 More Experiments

We further conduct FSOR experiments on two few-shot benchmark datasets: CIFAR-FS[156], FC100 [132]. CIFAR-FS [156] contains 100 classes with the class split for (64,16,20). FC100 [132] contains 100 classes with the class split (60,20,20). Each class has 600 images and all images for the two sets are of size is 32×32. As shown in Tab. 5.5 and 5.6, we compare our methods with the threshold-based methods and direct application of large-scale open-set recognition methods. Consistent with Tab. 5.2, for low-resolution dataset, our method achieves the best performance on both classification accuracy and negative query rejection, which again demonstrates the effectiveness of our approach. For OpenMax results, we follow [128] to fit Weibull models over training tasks using the predicted class scores (i.e., logits). Then the mean activation vectors are used for negative detection.

5.6 Summary

In this chapter, we consider the case where $\mathcal{M}$ is a standard recognition model trained with many-shot data, and we adapt it to better perform few-shot open-set recognition tasks. We show the limitation of threshold-based approaches for few-shot open-set recognition where different tasks may need very different rejection threshold and hence the tuning process could be challenging.

Table 5.5: 5-way FSOR results CIFAR-FS.*: our implementation.

Algorithm	1-shot		5-shot	
	Acc	AUROC	Acc	AUROC
OpenMax[123]*	71.65±0.65	50.21±0.07	85.66±0.48	75.78±0.47
CounterFactural[140]*	71.71±0.65	72.57±0.61	85.71±0.45	80.44±0.37
PEELER[127]*	71.47±0.67	71.28±0.57	85.46±0.47	75.97±0.33
Dynamic[137]*	71.56±0.67	66.89±0.52	85.78±0.49	76.03±0.37
ATT-G (ours)	72.43±0.65	76.72±0.59	86.52±0.49	84.64±0.38
SEMAN-G (ours)	74.55±0.65	78.10±0.58	86.71±0.47	86.47±0.37

Table 5.6: 5-way FSOR results on FC100. *: our implementation.

Algorithm	1-shot		5-shot	
	Acc	AUROC	Acc	AUROC
OpenMax[123]*	44.70±0.60	50.10±0.11	60.11±0.62	57.78±0.44
CounterFactural[140]*	44.53±0.60	57.20±0.47	61.12±0.60	62.35±0.45
PEELER[127]*	44.45±0.57	55.86±0.44	60.86±0.59	61.07±0.40
Dynamic[137]*	44.88±0.59	55.62±0.54	60.45±0.61	59.01±0.52
ATT-G (ours)	45.11±0.60	59.55±0.57	61.18±0.61	63.34±0.50
SEMAN-G (ours)	46.01±0.60	59.73±0.53	62.18±0.57	64.46±0.50

To this end, we propose our task-adaptive negative envision approach towards (G)FSOR. Given a standard recognition model $\mathcal{M}$, we propose to add a negative generator δ to compute negative prototypes from few/many-shot class examples. We study the different design of negative generator δ, and find attention-based generator works the best; adding class semantics further improves the performance. We also introduce a new conjugate class training strategy to better facilitate the learning process. Extensive experiments demonstrate the effectiveness of our approach. We note the limitation where we assume the negative source only being the images from other categories. Other possible negative sources include, e.g., data from different domains, adversarial data, etc. We will leave those as future work and study they affect our approach.

Table 5.7: Summary of Symbol Notation

Variable	Definition	Note
$\mathcal{T}$	a few-shot open-set recognition task	
$\mathcal{T}^*$	a generalized few-shot open-set recognition task	
$\mathcal{C}^f$	the set of few-shot classes	
$\mathcal{C}^*$	the set of many-shot classes	
$\mathcal{C}^n$	the set of negative classes (simulate unknown sources)	
$\mathcal{C}^B/\mathcal{C}^N$	base/novel class set	class split in experiment
$\mathcal{D}^B/\mathcal{D}^N$	base/novel class dataset	dataset split in experiment
$\mathcal{S}$	training samples (support) set of few-shot classes	
$\mathcal{S}_c$	training samples (support) set of few-shot class c	(character sub-index)
$\mathcal{Q}^f/\mathcal{Q}^*/\mathcal{Q}^n$	testing samples (query) set of few-shot/many-shot/negative classes	
$\mathbf{P}^f$	few-shot prototype vectors (average the support feature vectors)	
$\mathbf{P}^*$	many-shot prototype vectors (classifier weights after pretraining)	
$\mathbf{Z}^f/\mathbf{Z}^*$	semantic-visual vectors for few/many-shot classes	
$\mathbf{p}_c$	a prototype feature of class c	
$\mathbf{p}^-$	a negative prototype feature	
$\mathbf{e}_c$	word embedding (semantic representation) of class c	
$\mathbf{z}_c$	semantic-visual vector for class c	concatenate $\mathbf{p}_c$ and $\mathbf{e}_c$
$\mathbf{K}_q, \mathbf{K}_k, \mathbf{K}_v$	learnable kernels for attention calculation	
$\mathbf{A}_{((x,y))}$	the unnormalized attention weights between x and y	
s	a support sample	
q	a query sample	its label is y_q
f	feature extractor function, extract one feature vector for each image	
f_s	distance function, calculate similarity between two feature vectors	
f_n	an MLP for negative prototype generation	
f_g	gating function used in ATT-G and SEMAN-G	
g_n	general representation for the negative prototype generation function	$g_{n,i}$ the i-th function
θ_m	manual threshold for negative query detection	
θ_a	automatic/task-adaptive threshold	$\theta_{a,i}$ the i-th threshold
τ	a margin between rejection score and positive detection scores	the margin value is positive
$\odot$	element-wise multiplication	
ϕ	element-wise sigmoid operation	
σ	row-wise softmax operation	

Note: the numerical sub-index used in the paper is for the definition of conjugate tasks and their related calculation

Chapter 6: Adapting Models for Unseen Domains

Vision data from different source, environments and scenes would have different visual properties. A vision system should be capable of adapting to various domains towards a more universal utility. In this chapter, we design efficient model adaptation approaches from one domain to other domains with only domain-level information. Figure 6.1 shows a high-level illustration of the idea and framework, where δ is instantiated as a domain translator to bridge the gap between visual domains.

6.1 Introduction

In recent years, the success of large-scale pre-trained models has been evident across a wide range of computer vision applications such as object recognition and detection [149, 157, 158, 159, 160, 161, 162], which brings steps further into vision-capable AI systems. Despite their effectiveness, challenges arise when deploying these models into real scenarios. One typical challenge is that, in real open world, data are diverse and not necessarily seen by the model before. For a vision system to operate in practice, one important capability is to deal with data from new unseen domains.

Researchers have formulated this challenge as domain generalization problem, which aims to enhance model robustness by transferring learned representations from a source domain to target domains. Many research endeavors have been put into adapting pre-trained models to new domains, since the pre-trained models might fail to perform well due to the divergence between the training and test domains. These approaches [163, 164, 165, 166, 167] usually design specific training strategies for handling out-of-distribution data [168]. However, traditional domain generalization methods typically require access to target domain data, even if unlabeled, posing

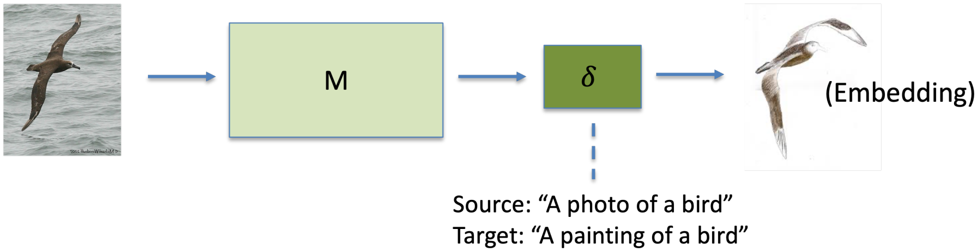


Figure 6.1: In this chapter, δ serves as a domain translator that translates a source domain visual embedding to another target domain.

challenges in scenarios where data availability is limited, as is often the case in situations such as rare diseases or when data are sensitive. Recognizing this limitation, there is a growing need for domain generalization techniques that can effectively adapt pre-trained models to new domains without relying on extensive labeled data from the target domain.

In this chapter, we investigate the problem of domain generalization without target data. We believe a desired vision system should be both robust to unseen domains but also not rely on additional target domain data. We are inspired from the recent vision-and-language model CLIP [158] that exhibits strong transferrable ability on vision applications, and propose to translate visual embeddings from source to target domain using the semantic information (e.g., domain descriptions) that is bridged using common space projection. Our insight is that the domain-level knowledge encoded by large multi-modal models like CLIP [158] is able to address domain shifts in an efficient manner, i.e., with rapid training process.

Though we may directly extract the rich visual embeddings from pre-trained models and apply to new domains, we notice that they might show strong bias towards the source domain, especially when the source domain data are well seen by the pre-trained models during their pre-training stages. For example, CLIP performs well on natural images or photos as it may have encountered majority of photo images when it was initially trained. Thus, to improve the generalization on target domains, it is essential to disentangle the domain invariant representation from the domain specific counterpart.

To this end, we consider a machine learning model $\mathcal{M}$ trained with source domain data, and

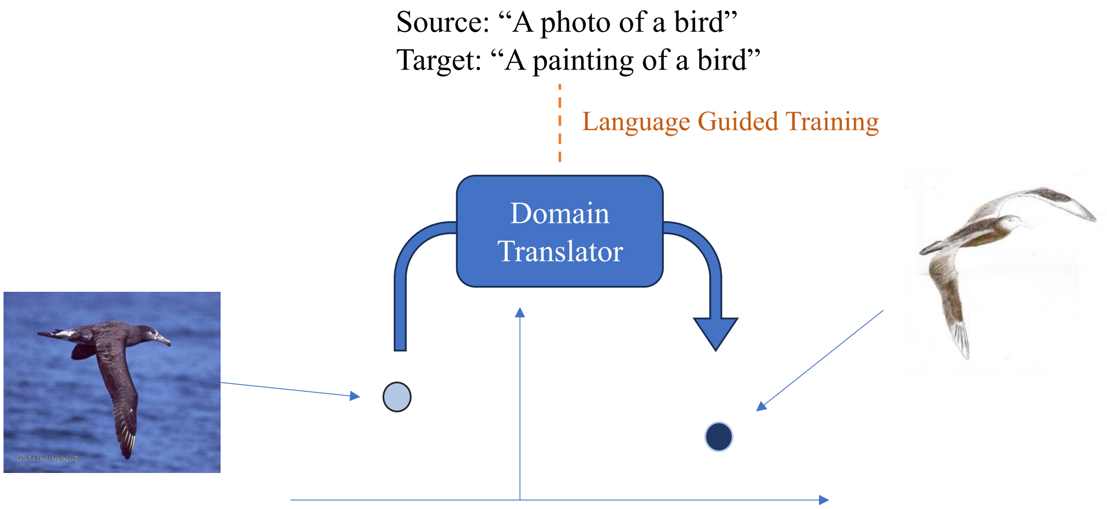


Figure 6.2: In this chapter, we investigate the problem of domain generalization, where the model is learned to be adapted to unseen domain without any access to the target domain images. To this end, we propose a language-guided domain translator framework to effectively transform source domain data into target domain.

seek to adapt it for target domain by adding a small domain translator δ to it. Specifically, we develop a language-guided domain translator framework that transform visual embeddings to the target domain via language guidance. Fig. 6.2 illustrates a high-level concept of our goal. Specifically, our domain translator contains two parts: feature disentangler and feature hallucinator. Feature disentangler is used to disentangle the domain-invariant component from source-domain visual embedding, and then fed into feature hallucinator to get target-domain visual embedding. As we do not have access to any target domain images, we propose to design language-guided losses to learn the domain translator. We construct text prompts describing the source domain as well as the class names, and use them to guide the disentanglement process. The translated target domain embedding is then supervised by the text prompt combing domain and class names. In this way, we are able to generate pseudo target domain visual features that can be used for learning a robust classifier.

We further identify the limitation of current domain generalization mechanisms where they are usually designed for one-to-one generalization, where we assume all the samples coming from this single target domain. Towards a more realistic setting, we propose to extend our method to deal with one-to-many scenario, where during test time, the samples might come from a mixture of target domain. Specifically, we design a domain scorer that predicts scores for estimating which domain it belongs to. We validate our framework on four major domain generalization benchmarks which show promising performance gains under different evaluation settings.

6.2 Related Work

Domain Adaptation & Generalization This problem has been broadly studied in the literature, which aims to adapt models to a new target domain while still performing well. Earlier works [163, 164, 165, 166] assume access to target domain images, and design methods based on different availability of the target image annotations. To ease the efforts on annotation, various works [169, 170] are centered around the setting where target domain images are unlabeled, proposing to learning a classifier invariant to source and target domains. More recently, research efforts turn to an even more challenging setting where there lacks the target domain images as well. Works such as [171, 172] investigate domain adaptation with very few number (for example only one) of target domain images. The most relevant work to ours is LADS [173] which requires zero image from target domain but only language descriptions of the new domains. Specifically, it trains a data augmentation network that transforms source domain embeddings to the target domain. However, its approach only works on one-to-one domain generalization, that is, we assume the prior knowledge of which specific target domain the test-time sample belongs to. In our work, we ease this strong assumption by providing a one-to-many domain translator that handles the situation where test-time samples could come from multiple domains.

Zero-shot Model Adaptation Our work in general is related to Zero-shot model adaptation, where we are not provided task-specific training data. The emergence of large pre-trained models like CLIP [158] has proven powerful in various downstream applications under zero-shot settings [174, 175], showing the strong transferrable ability of these pre-trained embeddings. When new labeled data come in, they are also quickly tuned towards task-specific data. One insight for zero-shot transfer is to use semantic embeddings to transfer concepts across modalities and domains, as described in [176, 177]. For example, [176] propose to utilize semantic relations between categories to perform zero-shot transfer. We similarly follow this insight and propose to translate visual embeddings across domains using semantic description, under the context of domain generalization.

Multi-modal Learning The field studies how to better exploit and utilize information from different modalities, typically from visual signals (e.g., images and videos) and textual signals (e.g., sentences, phrases and paragraphs). Earlier works [105, 178, 179, 180] focus on designing specific architectures for modelling the interaction and fusion of different modalities. More recently, CLIP [158] and its variants come out, which provides powerful visual and textual encoders that are trained on huge number of image-text pairs, which are designed to project visual and textual embeddings into a common space. In this way, if offers strong visual and textual embeddings that can be directly applied for different tasks. Our model is related to multi-modal learning as well since we learn to map textual embedding domain descriptions to same space as for visual embeddings, so that it can be easily transferred to unseen domain even without seeing new domain images.

6.3 Preliminary

Here we go over our formulation of domain generalization problem. Overall all our approach is based on pre-trained CLIP-based model, which consists of a visual encoder M_V and textual encoder M_T. We fix these two encoders throughout the process. Given an image, we can get its original visual embedding $x = M_V(im) \in \mathcal{R}^d$; given a text or phrase, we can get its textual embedding $e = M_T(im) \in \mathcal{R}^d$. Since our approach is built upon these pre-extracted visual and textual features, we will use x and e to represent the inputs. We denote $\mathcal{D}_s^{train} = \{(c, \mathbf{x})\}$ as the training data from source domain s, where $c \in C$ is the class name of image $\mathbf{x}$. Domain generalization aims to adapt the model to unseen target domain t while still retain the capability on source domain. We denote $\mathcal{D}_t^{test}$ as the target domain query at inference time, and $\mathcal{D}_s^{test}$ as the source domain query at inference time. The model is expected to perform well on both $\mathcal{D}_t^{test}$ and $\mathcal{D}_s^{test}$.

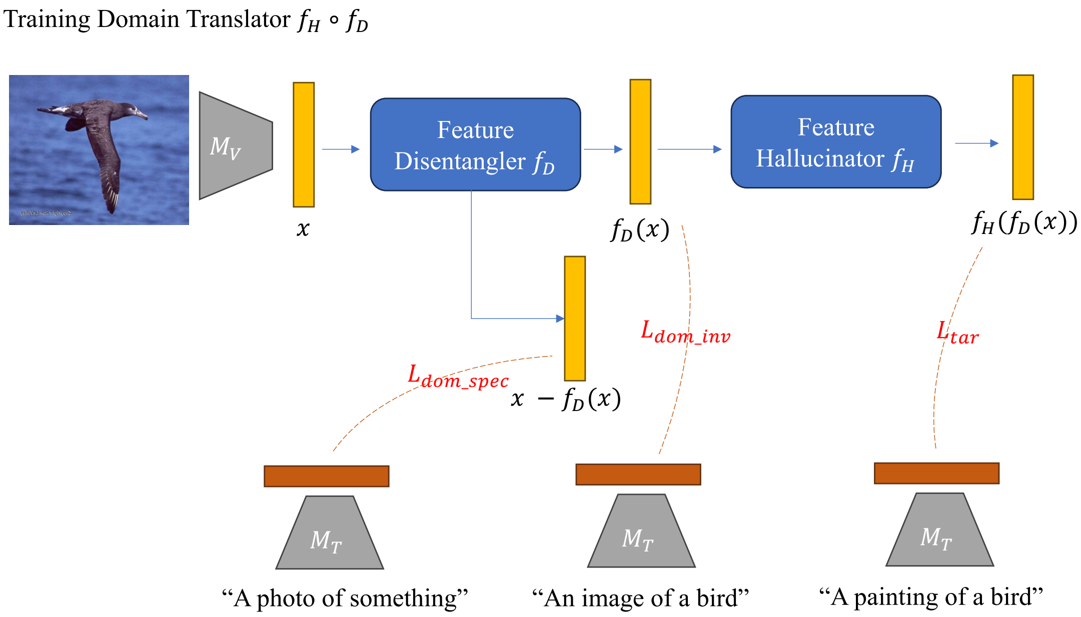


Figure 6.3: Our training pipeline for domain translator. Our framework is built on top on CLIP-based multi-modal model, where M_V refers to the visual encoder, and M_T refers to the textual encoder. The domain translator is composed of two parts: feature disentangler f_D and feature hallucinator f_H. For a source domain visual embedding x, we first disentangle the domain invariant component $f_D(x)$, as well as its domain-specific component $x - f_D(x)$. Then the domain invariant part is transformed to the target domain as $f_H(f_D(x))$. We use three textual prompts to put constraints, where a domain-only description is used for guiding the domain-specific $x - f_D(x)$, a class-only description is used for guiding the domain-invariant $f_D(x)$ and finally we use target description to guide the target visual embedding. When the domain translator is done training, we use all the translated visual embeddings to fit a linear classifier on top of the original CLIP visual encoder M_V that is used for recognizing new-domain images.

6.4 Approach

Here we talk about our language-guided domain translation framework, as illustrate in Fig 6.3.

6.4.1 Language guided domain translator

We propose to use a domain translator to transform source-domain visual embedding to new target domain. Our approach is inspired by the recent study [173] which implies that a visual feature could be represented by the sum of a domain-invariant and a domain-specific component embeddings. Hence our domain translator consists two parts: feature disentangler f_D and feature hallucinator f_H. Feature disentangler f_D aims to disentangle the domain invariant part of the original visual feature x, denoted as $f_D(x)$. A good domain invariant component should contain information irrelevant to the domain. We model its domain-specific counterparts as $x - f_D(x)$ which is supposed to only contain domain-specific information. Then we build feature hallucinator on top of the domain-invariant feature $f_D(x)$ to get the translated target domain embedding :

$$\hat{x} = f_H(f_D(x))$$

Since we do not have any target domain images, we propose to use domain and class descriptions to guide the generation of each component. Specifically, we use three textual prompts to put constraints, where a domain-only description is used for guiding the domain-specific $x - f_D(x)$, a class-only description is used for guiding the domain-invariant $f_D(x)$ and finally we use target description to guide the target visual embedding. We use the following text templates to construct our three prompts:

- *class prompts*: we use the template [`A image of a {class}`];
- *domain prompts*: we use the template [`A {domain} of something`];
- *domain-class prompts*: we use the template [`A {domain} of a {class}`].

We extract their embeddings, denoted as e_c,e_d,e_x, using M_T. And use them to guide the training of f_D and f_H. We propose the combination of three losses. First, we put a constraint on domain-invariant component. Our insight is that this component should retain class information. As such, we use a cross entropy loss on its cosine similarity with all the class prompts:

$$\mathcal{L}_{dom-inv} = \text{Cross-entropy}(f_D(x), e_{c_i}; c)$$

Then we would want the domain-specific part to only contain domain information, so we maximize

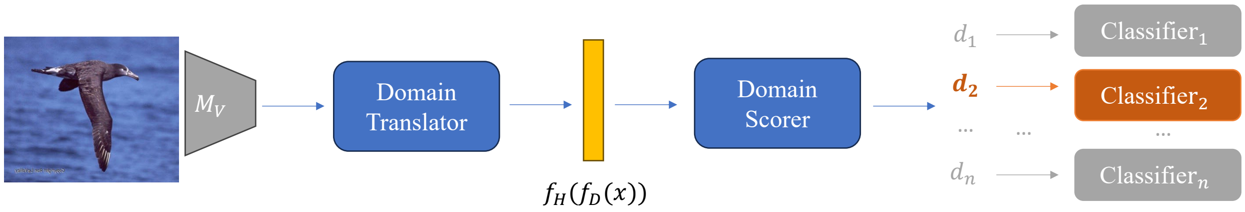


Figure 6.4: We can extend our approach to multi-domain generalization where during test time the samples come from mixture of target domains. We propose to train a domain scorer to identify the most probable domain and use its corresponding classifier to perform the task.

its similarity with domain prompts

$$\mathcal{L}_{dom-spec} = 1 - \text{Cosine}(x - f_D(x), e_d)$$

Finally we put constraint on the hallucinated target domain feature, forcing it to perform well with domain-class based classifier.

$$\mathcal{L}_{tar} = \text{Cross-entropy}(\hat{x}, e_x; c)$$

Then our total loss is the weighted sum of these three parts.

$$\mathcal{L} = \mathcal{L}_{dom-inv} + \alpha_1 \mathcal{L}_{dom-spec} + \alpha_2 \mathcal{L}_{tar}$$

6.4.2 Training strategy

Our full training process contains two steps. First, we train the domain translator as described above, where for each source domain visual embedding x from $\mathcal{D}_s^{train}$, we can get its hallucinated feature $\hat{x} = f_H(f_D(x))$ for the target domain t. When we are done translating the source domain data, we will get a pseudo target domain set $\hat{\mathcal{D}_s^{train}}\mathcal{D}_s^{train} = \{(c, \hat{x})\}$. Finally we fit a linear classifier C_t using $\hat{\mathcal{D}_s^{train}} \cup \mathcal{D}_s^{train}$. During inference time, when testing query $x^{'} \in \mathcal{D}_t^{test}$ comes in, we extract its CLIP feature $M_V(x^{'})$ and use C_t for predicting the class label.

6.4.3 Multi-domain Generalization

We further extend our framework to the multi-domain generalization setting, where during test-time the queries might come from n different target domains, and the model should be able to identify the correct domain and the task. To do so, we propose to additionally train a domain scorer f_S on top of our domain translated features. It outputs n scores representing how likely the sample goes into this domain. Then it will select the classifier for this domain and predict the class label. During training, we first fit multiple domain-specific classifier $C_{t_1}, \ldots, C_{t_n}$ using our one-to-one approach. Then we train f_S using the combination of target domain translated features using a cross-entropy loss between the domain scorer and their domain labels. During inference time, we rank the scores and select the domain with the highest score $i^* = argmax_{i=1,\ldots,n}(d_i)$ and use $C_{t_{i^*}}$ for label prediction. We illustrate our multi-domain generalization pipeline in Fig. 6.4.

6.5 Experiments

In this section, we present and analyze the experimental results for domain adaptation. We first introduct the evaluation setups, implementation details, followed by concrete results across four public domain adaptation benchmarks as well as ablation study.

6.5.1 Implementation and Evaluation Setups

Datasets. We validate our approach on four public benchmarks:

- CUB-Painting: 200 finegrained classes of bird species from natural and painting domains respectively. It is constructed as subsets selection from CUB-200-2011 [181] and CUB-200-Paintings [182], where the first are real photo images and the latter are paintings collected from the web. The number of training, validation and testing samples are (5994, 2897, 3047).
- DomainNetMini [183]: a subset of DomainNet [184] containing 40 classes from four different domains: sketch, real, clipart and painting. We use this dataset for both one-to-one and one-to-many evaluation. For one-to-one, we follow the commonly used evaluation

setup [173] where the model is trained on “sketch” domain (source) while leaving the rest for test-time evaluation. The number of training, validation and testing samples are (5537, 1200, 11468).

- Colored MNIST [185]: it is constructed upon the MNIST Digits [186], where numbers are colored as red or blue. The model is trained with even numbers red and odd numbers being blue, while during test-time, the colors are random. The recognition task is to classify the digits from 0 to 9. The number of training, validation and testing samples are (30000, 5000, 5000).

- Waterbirds [187]: It is a synthetic dataset composed of landbird and waterbird images taken from CUB-200 dataset. The bird images are synthesized with either forest or water background using Places [188] dataset. The model is trained with landbirds in forest and waterbirds in water, while during test-time, there is a randome mixture of birds and backgrounds. The task is to recognize from two types of bird species. The number of training, validation and testing samples are (4795, 600, 2897).

Evaluation Metrics. We report the classification accuracy for both source and target domains. For one-to-one domain generalization results on DomainNetMini, we report the average of the accuracies across different target domains.

Baselines. We mainly compare our approach to the following baselines methods:

- CLIP-ZS-G [158]: zero-shot recognition using CLIP embeddings where the classifier uses the class name (e.g., "bicycle") along as the prompt.

- CLIP-ZS-G [158]: zero-shot recognition using CLIP embeddings where the classifier uses the class name along with the domain as the prompt (e.g., "a photo of a bicycle").

- CLIP-LP: extracts the CLIP visual embeddings and fits a linear classifier using source domain images.

- CLIP-LP-ZS: similar to CLIP-LP but initialize the classifier with CLIP textual embeddings.

- WiSE-LP [189]: similar to CLIP-LP but using a weighted ensemble of linear classifiers consisting of the zero-shot and the fine-tuned linear classifier.
- VQGAN+CLIP [190]: a generative method that generates target-domain images using the text prompt and source-domain images. Then a linear probe is trained on top of generated image embeddings. We cite the results from [191].

Implementation Details. We fix the parameters of CLIP encoders, and train our domain translator on top. We instantiate it with two light-weight architectures: two-layer MLPs, or one standard transformer block. We will report results for both of them. We train the network on one TiTAN RTX GPU with batch size being 64. We tune and select different the learning rate for each dataset, with learning rate being 0.001 for CUB-Paintings, Colored MNIST and Waterbirds and 0.0003 for DomainNetMini.

6.5.2 Results

One-to-one domain generalization We first present in Tab. 6.1 and 6.2 the standard one-to-one domain generalization results. We are able to outperform our baselines on different benchmarks. We can see from the CLIP-ZS results that directly application of CLIP features do not transfer well on new unseen domain, since CLIP is not trained for it. This validates the motivation of domain generalization problem in general. Compared to other techniques, we still get performance gains, which validates our idea of modeling the problem as a domain translation. We can observe a more significant boost of performance on CUB dataset compared to others. Our insight is that CLIP does not specifically tackle fine-grained bird specifics, hence making it more challenging problem on identifying fine-grained classes. Our approach distills class information during the tranlation process, which could well mitigate this issue.

One-to-many domain generalization Here we present in Tab. 6.3 the one-to-many domain generalization results on a subset of DomainNetMini. For comparison, we extend CLIP-ZS and CLIP-LP to multi-domain setting, where we build the classifier with all the possible combination between domain and class, i.e., [`A {domain} of a {class}`] for all domains and classes, then select

Table 6.1: One-to-one domain generalization results on CUB-Painting and DomainNetMini.

Approach	CUB-Painting			DomainNetMini		
	Source	Target	Average	Source	Target	Average
CLIP Zero-Shot [158]	60.34	52.84	56.59	93.49	95.94	94.72
CLIP Zero-Shot* [158]	61.93	54.38	58.16	93.24	96.01	94.62
CLIP LP	85.91	64.33	75.12	95.03	93.75	94.39
WiSE-LP [189]	81.74	64.80	73.27	95.19	93.68	94.44
DomTrans (Ours)	**86.27**	**66.76**	**76.52**	**95.34**	**96.04**	**95.69**

Table 6.2: One-to-one domain generalization results on Colored MNIST and Waterbirds.

Approach	Colored MNIST			Waterbirds		
	ID	OOD	Extended	ID	OOD	Extended
CLIP ZS-G [158]	57.88	55.44	56.66	83.04	65.43	74.23
CLIP ZS-A [158]	79.96	**77.44**	78.70	83.86	72.58	78.22
CLIP LP	99.45	39.57	69.51	97.79	61.12	79.66
WiSE-LP [189]	99.38	58.50	78.94	90.06	68.45	79.25
DomTrans (Ours)	**99.48**	75.10	**87.29**	**98.01**	**72.93**	**85.47**

the one with highest probability. We can see from the results also reflect the effectiveness of our approach. We report the domain only, class only and overall accuracy defined as

- domain accuracy = $\frac{\text{\#correct domain predictions}}{N}$
- class accuracy = $\frac{\text{\#correct class predictions}}{N}$
- oevrall accuracy = $\frac{\text{\#correct <domain,class> pair predictions}}{N}$

where N is the total number of test samples. We can see from the table that our approach works better than other alternatives.

Domain generalization with additional unlabeled data Previous results assume model only trained with source domain images. Here we study an interesting setting where we assume access to additional task-relevant data, which are unlabeled. This is a practical assumption since collecting data are cheap, rather than annotating data. Inspired by the emergence of large-scale image-text dataset constructed from web, we investigate the impact of training from additional unlabeled image-text pairs. Specifically, we use ImageNet-1K [192], which is a subset of LAION database [193] consisting of 10M image-text pairs as our additional data. We train our domain

Table 6.3: One-to-many domain generalization results on DomainNetMini Subset.

Approach	Domain Accuracy	Class Accuracy	Overall Accuracy
CLIP ZS-A [158]	80.31	96.01	77.48
CLIP LP	79.12	93.75	73.31
DomTrans (Ours)	82.53	96.04	78.79

translator module using contrastive loss on them, and then finetune with our domain generalization losses. We present in Tab. 6.4 the results on CUB-Painting and DomainNetMini. We can observe that additional unlabeled data is beneficial for domain generalization.

Table 6.4: Study of the effects of additional image-text pair data on domain generalization.

Training Data	CUB-Painting			DomainNetMini		
	Source	Target	Average	Source	Target	Average
Source Images	86.27	66.76	76.52	95.34	96.04	95.69
Source Images + Image-text pairs	88.13	67.62	77.89	95.79	96.41	96.1

6.5.3 Ablation Study and Analysis

Here we provide a few ablation study analyzing our approach. We ablate the component of our training loss in Tab. 6.5 on CUB dataset, where we can see all components contribute to the performance, validating the effectiveness of our domain-invariant vs domain-specific design.

6.6 Summary

In this chapter, we present a language-guided domain translator framework for adapting a source-domain model $\mathcal{M}$ to target domain using a light-weight domain translator δ. The domain

Table 6.5: Ablation study of our loss components on CUB-paintings.

Loss Components			Accuracies		
$\mathcal{L}_{tar}$	$\mathcal{L}_{dom-spec}$	$\mathcal{L}_{dom-inv}$	Source	Target	Average
✓			83.23	64.18	73.71
✓	✓		84.02	65.89	74.96
✓	✓	✓	86.27	66.76	76.52

translator module δ is light-weight and could be generically applied on various vision-language common space models. Specifically, we use the language embeddings of domain descriptions to guide the visual embeddings to disentangle its domain invariant component, which can be used for recognizing images in unknown domains. Coupled with this framework, we propose a training loss consisting domain invariance and specific objectives. The approach is further extended to one-to-many domain generalization, where the test-time samples could come from multiple unknown domains. Towards this challenge, we learn a domain scorer that differentiate between different domains. We validate our approach on public benchmarks, showing the potential for a more robust and adaptable recognition system in real-world scenarios.

Chapter 7: Conclusion

7.1 Summary of Contributions

This thesis is dedicated to developing practical deep learning models for real-world challenges. Towards this goal, we offer a general model adaptation framework, $M + \delta$, to adapt deep learning models for practical data challenges in computer vision applications. We focus on two main challenges: model adaptation for efficient data representations (Chapter 2, 3, 4) and model adaptation for open environment (Chapter 5, 6).

For efficient representation challenges, we start with Chapter 2 exploring efficient compressed modalities from modern video codecs, and propose a fine-to-coarse knowledge distillation approach to accomplish the transition from decoded-domain two-stream network to compressed domain, which obtains a significant speedup of the network inference and reduce the performance gap between decoded and compressed domain video recognition. Inspired by the advantages from efficient data modalities, we then propose different modal adaptation techniques for learning efficient video representations, following our general $M + \delta$ framework. In Chapter 3, we formulate a novel few-bit VideoQA task and propose to train a task-specific feature compressor that learns low-bit representations. We instantiate δ as a light-weight intermediate module that compresses data, with a straight-through estimator to facilitate end-to-end training. Our approach is able to get minor drop of VideoQA performance using as small as 10-bit features, which captures compact task-specific information. We provide an insight about the advantages of tiny features, including its storage and privacy advantages for on-device and cloud computing. Then we investigate the challenge from another perspective, where we'd like to find efficient video representation that is human-interpretable. In Chapter 4, we introduce the problem of sparsified VideoQA, where videos are represented as few frames or text descriptions that are useful for the question answering. We propose a learnable token sparsification approach that extracts sparse frames or word phrases towards task completion, where we instantiate our δ as a token sparsifier. We demonstrate that the proposed method effectively learn summaries, in either visual or textual or both formats, of the videos, which provide the advantages for both machine interpretation and human perception.

For open environment challenges, we inspect the scenarios where the model encounters unknown classes and unknown domains respectively. In Chapter 5 we aim to adapt a model for open recognition, where it should be able to adapt to new classes and be robust to outliers. We propose a negative envision framework for modeling the negative classes as an additional negative prototype. Then we propose a new conjugate training strategy to learn to adapt model for new classes even with limited labeled samples. Our approach gets significant performance improvement under multiple practical evaluation settings, demonstrating the advantage of negative modeling which is handled by the negative generator δ. On the other hand, in Chapter 6, we tackle the problem of adapting model for unseen domains, where we develop a language-guided domain generalization framework. We design δ as the domain translator that is trained with our domain-invariant and domain-specific losses. We also propose a learnable domain scorer that adapts model to a more realistic multi-domain scenario. We demonstrate the effectiveness of our approach on public benchmarks under different evaluation settings.

7.2 Future Work and Open Issues

Here we discuss a few future directions and open issues beyond this thesis.

Generalizable Efficient Representation Data efficiency is a broad topic in practical computer vision systems. This thesis mainly investigates this issue by controlling the information flow, particularly in the context of videos as a complex media characterized by large sizes and challenging processing requirements. We show that task knowledge is useful in discarding non-essential information in features for enhanced data efficiency. Future endeavors may delve into simplifying task priors. For example, for tasks concerned with understanding complex activities composed of a sequence of actions, how to learn efficient representation to capture the action components and their underlying temporal order with only activity-level knowledge; for the efficient representation learned from indoor activities, how can it be generalize to tasks involving understanding outdoor activities, without knowing what specific outdoor environment looks like in advance. This could involve designing mechanisms that facilitate the creation of universally applicable and efficient representations, establishing precise controls over the encoded information necessary for generalizing across specific scenarios.

Adapting Model to Complex Environment The second part of the thesis focuses on model adaptation under two typical open environment scenario:dealing with unknown classes and unfamiliar domains. In real-world applications, systems often encounter a significantly more intricate environment where out-of-distribution data can manifest in various forms. For instance, the system might be required to process images featuring both unknown objects and scenes that are entirely new. It would be interesting direction to investigate the feasibility of model designs capable of effectively managing such complex environmental changes.

Extend to Other Modalities The thesis primarily centers on the analysis of visual data and proposes design approaches aligned with the characteristics of visual signals. However, there is an opportunity to broaden the scope of the thesis to encompass additional data modalities, including languages, remote-sensing data, medical images, and combinations of multiple data types. We posit that the practical challenges addressed in the thesis are prevalent across various data applications. Exploring the development of modality-specific model adaptation approaches represents an intriguing direction for further investigation.

Practical Deployment of the Framework The thesis provides a general $\mathcal{M} + \delta$ framework and discusses its concrete usecases for different applications. To ensure the practical implementation of the proposed framework, a more thorough examination is needed regarding the specific deployment of different modules and components within the framework. For example, to facilitate the adoption of the framework in edge applications, it is essential to pinpoint the optimal deployment locations for the original model $\mathcal{M}$ and the adaptor module δ. This necessitates further research into the communication dynamics between the cloud and local device, as well as the precise design of data-model interactions.

References

[1] L. Li, Y.-C. Chen, Y. Cheng, Z. Gan, L. Yu, and J. Liu, "Hero: Hierarchical encoder for video+ language omni-representation pre-training," in *EMNLP*, 2020.

[2] A. Yang, A. Miech, J. Sivic, I. Laptev, and C. Schmid, "Just ask: Learning to answer questions from millions of narrated videos," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1686–1697.

[3] A. Afrasiyabi, J.-F. Lalonde, and C. Gagné, "Associative alignment for few-shot image classification," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 18–35.

[4] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, "Learning optical flow," in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*, Springer, 2008, pp. 83–97.

[5] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, A. Fitzgibbon et al. (Eds.), Ed., ser. Part IV, LNCS 7577, Springer-Verlag, 2012, pp. 611–625.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[8] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *arXiv preprint arXiv:1708.05038*, 2017.

[9] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, Springer, 2016, pp. 20–36.

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[11] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Joint Pattern Recognition Symposium*, 2007.

[12] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *CVPR*, 2018.

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, 2017, p. 6.

[14] C.-Y. Wu, M. Zaheer, H. Hu, R Manmatha, A. J. Smola, and P. Krähenbühl, “Compressed video action recognition,” in *CVPR*, 2018.

[15] Z. Shou *et al.*, “Dmc-net: Generating discriminative motion cues for fast compressed video action recognition,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1268–1277.

[16] D. Le Gall, “Mpeg: A video compression standard for multimedia applications,” *Communications of the ACM*, 1991.

[17] L. Sevilla-Lara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black, “On the integration of optical flow and action recognition,” in *German Conference on Pattern Recognition (GCPR)*, 2018.

[18] J. Y.-H. Ng, J. Choi, J. Neumann, and L. S. Davis, “Actionflownet: Learning motion representation for action recognition,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1616–1624.

[19] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *NIPS*, 2014.

[20] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *CVPR*, 2017.

[21] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, “Rethinking spatiotemporal feature learning for video understanding,” in *ECCV*, 2018.

[22] C. Gu *et al.*, “Ava: A video dataset of spatio-temporally localized atomic visual actions,” *arXiv preprint arXiv:1705.08421*, 2017.

[23] D. Damen *et al.*, “Scaling egocentric vision: The epic-kitchens dataset,” in *European Conference on Computer Vision (ECCV)*, 2018.

[24] *Activitynet challenge 2016*, `http://activity-net.org/challenges/2016/`, 2016.

[25] A. Gorban *et al.*, *THUMOS challenge: Action recognition with a large number of classes*, `http://www.thumos.info/`, 2015.

[26] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[27] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5552–5561.

[28] M. Zolfaghari, K. Singh, and T. Brox, "Eco: Efficient convolutional network for online video understanding," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 695–712.

[29] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," in *CVPR*, 2016.

[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[31] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2827–2836.

[32] Z. Luo, J.-T. Hsieh, L. Jiang, J. Carlos Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged modalities," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 166–183.

[33] N. C. Garcia, P. Morerio, and V. Murino, "Learning with privileged information via adversarial discriminative modality distillation," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[34] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," *arXiv preprint arXiv:1511.03643*, 2015.

[35] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid, "Mars: Motion-augmented rgb stream for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7882–7891.

[36] J. C. Stroud, D. A. Ross, C. Sun, J. Deng, and R. Sukthankar, "D3d: Distilled 3d networks for video action recognition," *arXiv preprint arXiv:1812.08249*, 2018.

[37] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with deeply transferred motion vector cnns," *IEEE Transactions on Image Processing*, vol. 27, pp. 2326–2339, 2018.

[38] *Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video.* `https://www.ffmpeg.org/`.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[40] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2349–2358.

[41] A. Piergiovanni and M. S. Ryoo, "Representation flow for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[42] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[43] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2556–2563.

[44] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[45] L. Fan, W. Huang, S. E. Chuang Gan, B. Gong, and J. Huang, "End-to-end learning of motion representation for video understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6016–6025.

[46] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 5534–5542.

[47] S. Oh *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*, IEEE, 2011, pp. 3153–3160.

[48] NIST, *Trecvid 2017 evaluation for surveillance event detection*, 2017.

[49] J. Gleason, R. Ranjan, S. Schwarcz, C. Castillo, J.-C. Chen, and R. Chellappa, "A proposal-based solution to spatio-temporal action detection in untrimmed videos," in *2019 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2019, pp. 141–150.

[50] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[51] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *CVPR*, 2018.

[52] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen, “Bmn: Boundary-matching network for temporal action proposal generation,” in *ECCV*, 2019.

[53] M. Minderer, C. Sun, R. Villegas, F. Cole, K. Murphy, and H. Lee, “Unsupervised learning of object structure and dynamics from videos,” *arXiv preprint arXiv:1906.07889*, 2019.

[54] D. Xu *et al.*, “Video question answering via gradually refined attention over appearance and motion,” in *ACM MM*, 2017.

[55] T. M. Le, V. Le, S. Venkatesh, and T. Tran, “Hierarchical conditional relation networks for video question answering,” in *CVPR*, 2020.

[56] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *CVPR*, 2017.

[57] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, “HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips,” in *ICCV*, 2019.

[58] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *IJCV*, 2015.

[59] C. Feichtenhofer, H. Fan, J. Malik, and K. He, “Slowfast networks for video recognition,” in *ICCV*, 2019.

[60] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, “Videobert: A joint model for video and language representation learning,” in *ICCV*, 2019.

[61] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, “End-to-end learning of visual representations from uncurated instructional videos,” in *CVPR*, 2020.

[62] J. Lei *et al.*, “Less is more: Clipbert for video-and-language learning via sparse sampling,” in *CVPR*, 2021.

[63] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, “DVC: An end-to-end deep video compression framework,” in *CVPR*, 2019.

[64] J. Lin, D. Liu, H. Li, and F. Wu, “M-LVC: Multiple frames prediction for learned video compression,” in *CVPR*, 2020.

[65] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, “Improving deep video compression by resolution-adaptive flow coding,” *ECCV*, 2020.

[66] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, “Neural inter-frame compression for video coding,” in *ICCV*, 2019.

[67] R. Feng, Y. Wu, Z. Guo, Z. Zhang, and Z. Chen, “Learned video compression with feature-level residuals,” in *CVPR Workshops*, 2020.

[68] Z. Hu, G. Lu, and D. Xu, “Fvc: A new framework towards deep video compression in feature space,” in *CVPR*, 2021.

[69] A. Chadha and Y. Andreopoulos, “Deep perceptual preprocessing for video coding,” in *CVPR*, 2021.

[70] G. Toderici *et al.*, “Variable rate image compression with recurrent neural networks,” *arXiv preprint arXiv:1511.06085*, 2015.

[71] G. Toderici *et al.*, “Full resolution image compression with recurrent neural networks,” in *CVPR*, 2017.

[72] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, “Video compression through image interpolation,” in *ECCV*, 2018.

[73] P. Samarati and L. Sweeney, “Protecting privacy when disclosing information: K-anonymity and its enforcement through generalization and suppression,” 1998.

[74] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015.

[75] A. Mahendran and A. Vedaldi, “Understanding deep image representations by inverting them,” in *CVPR*, 2015.

[76] Y. Jang, Y. Song, Y. Yu, Y. Kim, and G. Kim, “Tgif-qa: Toward spatio-temporal reasoning in visual question answering,” in *CVPR*, 2017.

[77] L. Zhu, Z. Xu, Y. Yang, and A. Hauptmann, “Uncovering the temporal context for video question answering,” *IJCV*, 2017.

[78] J. Lei, L. Yu, M. Bansal, and T. L. Berg, “Tvqa: Localized, compositional video question answering,” *arXiv preprint arXiv:1809.01696*, 2018.

[79] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, “MovieQA: Understanding Stories in Movies through Question-Answering,” in *CVPR*, 2016.

[80] D. Huang, P. Chen, R. Zeng, Q. Du, M. Tan, and C. Gan, “Location-aware graph convolutional networks for video question answering,” in *AAAI*, 2020.

[81] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6904–6913.

[82] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on circuits and systems for video technology*, 2012.

[83] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on circuits and systems for video technology*, 2003.

[84] G. K. Wallace, "The JPEG still picture compression standard," *IEEE transactions on consumer electronics*, 1992.

[85] J. Choi and B. Han, "Task-aware quantization network for jpeg image compression," in *ECCV*, 2020.

[86] T. He, S. Sun, Z. Guo, and Z. Chen, "Beyond coding: Detection-driven image compression with semantically structured bit-stream," in *PCS*, 2019.

[87] Y. Yang *et al.*, "Tempclr: Temporal alignment representation with contrastive learning," *arXiv preprint arXiv:2212.13738*, 2022.

[88] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[89] X. Chen *et al.*, "Microsoft coco captions: Data collection and evaluation server," *arXiv*, 2015.

[90] R. Krishna *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *IJCV*, 2017.

[91] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[92] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[93] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016.

[94] A. L. Cambridge, *The orl database*, `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.

[95] L. Li, B. Pal, J. Ali, N. Sullivan, R. Chatterjee, and T. Ristenpart, “Protocols for checking compromised credentials,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[96] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, “Making the v in vqa matter: Elevating the role of image understanding in visual question answering,” in *CVPR*, 2017.

[97] J. Lei, L. Yu, T. L. Berg, and M. Bansal, “Tvqa+: Spatio-temporal grounding for video question answering,” in *ACL*, 2020.

[98] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, “Adaframe: Adaptive frame selection for fast video recognition,” in *CVPR*, 2019.

[99] D.-A. Huang *et al.*, “What makes a video a video: Analyzing temporal information in video understanding models and datasets,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7366–7375.

[100] S. Buch, C. Eyzaguirre, A. Gaidon, J. Wu, L. Fei-Fei, and J. C. Niebles, “Revisiting the “Video” in Video-Language Understanding,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[101] S. Huang, R. Piramuthu, S.-F. Chang, and G. A. Sigurdsson, “Video in 10 bits: Few-bit videoqa for efficiency and privacy,” in *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, Springer, 2023, pp. 738–754.

[102] J. Wang, X. Yang, H. Li, L. Liu, Z. Wu, and Y.-G. Jiang, “Efficient video transformers with spatial-temporal token selection,” in *European Conference on Computer Vision*, Springer, 2022, pp. 69–86.

[103] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.

[104] W. Kool, H. Van Hoof, and M. Welling, “Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 3499–3508.

[105] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[106] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[107] J. Lei, L. Yu, T. L. Berg, and M. Bansal, “What is more likely to happen next? video-and-language future event prediction,” in *EMNLP*, 2020.

[108] J. Liu *et al.*, “Violin: A large-scale dataset for video-and-language inference,” in *CVPR*, 2020.

[109] H. Chefer, S. Gur, and L. Wolf, “Transformer interpretability beyond attention visualization,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 782–791.

[110] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[111] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[112] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[113] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *CVPR*, 2016.

[114] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” in *Advances in neural information processing systems*, 2013, pp. 2292–2300.

[115] G. Han, X. Zhang, and C. Li, “Revisiting faster r-cnn: A deeper look at region proposal network,” in *International Conference on Neural Information Processing*, 2017, pp. 14–24.

[116] G. Han, X. Zhang, and C. Li, “Semi-supervised dff: Decoupling detection and feature flow for video object detectors,” in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1811–1819.

[117] G. Han, X. Zhang, and C. Li, in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3360–3364.

[118] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[119] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[120] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, “Matching networks for one shot learning,” in *Adv. Neural Inform. Process. Syst.*, 2016.

[121] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Adv. Neural Inform. Process. Syst.*, 2017.

[122] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," in *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013, pp. 1757–1772.

[123] A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[124] P. Schlachter, Y. Liao, and B. Yang, "Open-set recognition using intra-class splitting," in *2019 27th European Signal Processing Conference (EUSIPCO)*, IEEE, 2019, pp. 1–5.

[125] I. Goodfellow *et al.*, "Generative adversarial nets," in *NIPS*, 2014.

[126] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[127] B. Liu, H. Kang, H. Li, G. Hua, and N. Vasconcelos, "Few-shot open-set recognition using meta-learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] M. Jeong, S. Choi, and C. Kim, "Few-shot open-set recognition by transformation consistency," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 566–12 575.

[129] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[130] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," *arXiv preprint arXiv:1909.00025*, 2019.

[131] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 403–412.

[132] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018, pp. 721–731.

[133] G. Han, S. Huang, J. Ma, Y. He, and S.-F. Chang, "Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[134] G. Han, J. Ma, S. Huang, L. Chen, and S.-F. Chang, "Few-shot object detection with fully cross-transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[135] G. Han, Y. He, S. Huang, J. Ma, and S.-F. Chang, "Query adaptive few-shot object detection with heterogeneous graph convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 3263–3272.

[136] N.-A. Ypsilantis, N. Garcia, G. Han, S. Ibrahimi, N. Van Noord, and G. Tolias, "The met dataset: Instance-level recognition for artworks," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[137] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[138] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *arXiv preprint arXiv:1711.09325*, 2017.

[139] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative openmax for multi-class open set classification," *arXiv preprint arXiv:1707.07418*, 2017.

[140] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[141] D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, "Learning placeholders for open-set recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4401–4410.

[142] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8059–8068.

[143] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need?" *arXiv preprint arXiv:2003.11539*, 2020.

[144] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang, "Boosting few-shot learning with adaptive margin loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 576–12 584.

[145] C. Xing, N. Rostamzadeh, B. N. Oreshkin, and P. O. Pinheiro, "Adaptive cross-modal few-shot learning," *arXiv preprint arXiv:1902.07104*, 2019.

[146] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[147] M. Ren *et al.*, "Meta-learning for semi-supervised few-shot classification," in *iclr*, 2018.

[148] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[149] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[150] M. Ren, R. Liao, E. Fetaya, and R. S. Zemel, "Incremental few-shot learning with attention attractor networks," *arXiv preprint arXiv:1810.07218*, 2018.

[151] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Learning classifier synthesis for generalized few-shot learning," 2019.

[152] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng, "Conditional gaussian distribution learning for open set recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 480–13 489.

[153] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8808–8817.

[154] D.-C. Z. Han-Jia Ye Hexiang Hu and F. Sha, "Learning adaptive classifiers synthesis for generalized few-shot learning," *arXiv preprint arXiv:1906.02944*, 2019.

[155] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 7032–7042.

[156] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *Int. Conf. Learn. Represent.*, 2019.

[157] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[158] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[159] J. Ma, G. Han, S. Huang, Y. Yang, and S.-F. Chang, "Few-shot end-to-end object detection via constantly concentrated encoding across heads," in *European Conference on Computer Vision*, Springer, 2022, pp. 57–73.

[160] J. Ma, Y. Niu, J. Xu, S. Huang, G. Han, and S.-F. Chang, "Digeo: Discriminative geometry-aware learning for generalized few-shot object detection," *arXiv preprint arXiv:2303.09674*, 2023.

[161] H. Lin, G. Han, J. Ma, S. Huang, X. Lin, and S.-F. Chang, “Supervised masked knowledge distillation for few-shot transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 649–19 659.

[162] S. Huang, R. Piramuthu, V. Ordonez, S.-F. Chang, and G. A. Sigurdsson, “Characterizing video question answering with sparsified inputs,” *arXiv preprint arXiv:2311.16311*, 2023.

[163] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, “Unified deep supervised domain adaptation and generalization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5715–5725.

[164] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, “Semi-supervised domain adaptation via minimax entropy,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8050–8058.

[165] S. Baek, K. I. Kim, and T.-K. Kim, “Weakly-supervised domain adaptation via gan and mesh model for estimating 3d hand poses interacting objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6121–6131.

[166] J. Xu, L. Xiao, and A. M. López, “Self-supervised domain adaptation for computer vision tasks,” *IEEE Access*, vol. 7, pp. 156 694–156 706, 2019.

[167] C. Yang and S.-N. Lim, “One-shot domain adaptation for face generation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5921–5930.

[168] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[169] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, “Maximum classifier discrepancy for unsupervised domain adaptation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3723–3732.

[170] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *International conference on machine learning*, PMLR, 2015, pp. 1180–1189.

[171] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto, “Few-shot adversarial domain adaptation,” *Advances in neural information processing systems*, vol. 30, 2017.

[172] X. Yue, Z. Zheng, H. P. Das, K. Keutzer, and A. S. Vincentelli, “Multi-source few-shot domain adaptation,” *arXiv preprint arXiv:2109.12391*, 2021.

[173] L. Dunlap *et al.*, “Using language to entend to unseen domains,” *International Conference on Learning Representations (ICLR)*, 2023.

[174] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[175] R. Zhang *et al.*, "Tip-adapter: Training-free clip-adapter for better vision-language modeling," *arXiv preprint arXiv:2111.03930*, 2021.

[176] Y. Annadani and S. Biswas, "Preserving semantic relations for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7603–7612.

[177] S. Huang, J. Ma, G. Han, and S.-F. Chang, "Task-adaptive negative envision for few-shot open-set recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 7171–7180.

[178] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[179] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[180] X. Lin *et al.*, "Towards fast adaptation of pretrained contrastive models for multi-channel video-language retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 846–14 855.

[181] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[182] S. Wang, X. Chen, Y. Wang, M. Long, and J. Wang, "Progressive adversarial networks for fine-grained domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9213–9222.

[183] S. Tan, X. Peng, and K. Saenko, "Class-imbalanced domain adaptation: An empirical odyssey," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, Springer, 2020, pp. 585–602.

[184] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415.

[185] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[186] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[187] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *arXiv preprint arXiv:1911.08731*, 2019.

[188] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[189] M. Wortsman *et al.*, "Robust fine-tuning of zero-shot models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7959–7971.

[190] K. Crowson *et al.*, "Vqgan-clip: Open domain image generation and editing with natural language guidance," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*, Springer, 2022, pp. 88–105.

[191] L. Dunlap *et al.*, "Using language to extend to unseen domains," *arXiv preprint arXiv:2210.09520*, 2022.

[192] H. Liu *et al.*, "Learning customized visual models with retrieval-augmented knowledge," 2023.

[193] C. Schuhmann *et al.*, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," *arXiv preprint arXiv:2111.02114*, 2021.