A New Data Transmission Paradigm for Visual Analysis in Edge-Cloud Collaboration

Chen Zhuo

Interdisciplinary Graduate School

Rapid-Rich Object Search (ROSE) Lab

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

28-Sep-2020

Date

Chen Zhuo

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

28-Sep-2020

.....

Date

Lin Weisi

.....

Prof. Lin Weisi

Authorship Attribution Statement

This thesis contains material from 6 paper(s) published or submitted in the following peer-reviewed journals abd conferences in which I am the first author.

Chapters 3 and 4 are published or submitted as Zhuo Chen, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Toward intelligent sensing: Intermediate deep feature compression. IEEE Transactions on Image Processing 29 (2019): 2230-2243.
Zhuo Chen, Kui Fan, Shiqi Wang, Ling-Yu Duan, Weisi Lin, and Alex Kot. Lossy intermediate deep learning feature compression and evaluation. In Proceedings of the 27th ACM International Conference on Multimedia, pp. 2414-2422. 2019.
Zhuo Chen, Ling-Yu Duan, Shiqi Wang, Weisi Lin, and Alex Kot. Data Representation in Hybrid Coding Framework for Feature Maps Compression. In 2020 IEEE International Conference on Image Processing (ICIP). (accepted)
Zhuo Chen, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Video Codec Based Framework for Intermediate Deep Feature Compression. Under submission.

The contributions of the co-authors are as follows:

- Prof. Weisi Lin and Prof. Lingyu Duan conceived the presented idea of the new data transmitting strategy together with me.
- Dr. Kui Fan helped to conduct the experiments together with me.
- Assist Prof. Shiqi Wang helped to write the papers together with me.
- Prof. Alex Chichung Kot helped to revise the manuscripts and provided the hardware support.

Chapter 5 is published as Zhuo Chen, Weisi Lin, Shiqi Wang, Long Xu, and Leida Li. Image quality assessment based label smoothing in deep neural network learning. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6742-6746. IEEE, 2018.

The contributions of the co-authors are as follows:

- Prof. Weisi Lin conceived the presented idea of IQA label smoothing together with me.
- Assist Prof. Shiqi Wang helped to write the papers together with me.
- Dr. Long Xu provided the hardware support.
- Prof. Leida Li helped to revise the manuscript.

Chapter 6 is published as Zhuo Chen, Jie Lin, Zhe Wang, Vijay Chandrasekhar, and Weisi Lin. Beyond Ranking Loss: Deep Holographic Networks for Multi-Label Video Search. In 2019 IEEE International Conference on Image Processing (ICIP), pp. 879-883. IEEE, 2019.

The contributions of the co-authors are as follows:

- Dr. Jie Lin conceived the presented idea of the holographic network together with me.
- Zhe Wang helped to conduct the experiments together with me.
- Dr. Vijay Chandrasekhar and Prof. Weisi Lin helped to revise the manuscript.

28-Sep-2020 ..... Date

Chen Zhuo ..... Chen Zhuo

Acknowledgements

In the beginning, I would say thanks to my supervisor Prof. Lin Weisi to guide me throughout my research work from title selection to finding the results. Not a single line in this thesis would exist without him. I am also grateful to my co-supervisors and mentor: Prof. Alex Kot, Prof. Lap-Pui Chau and Prof. Yap Kim Hui.

Apart from my thesis advisory committee members, I won't forget to express the gratitude to rest of the team in Prof. Lin's group: Dr. Yuan Yuan, Dr. Li Qiaohong, Dr. Zhang Yabin, Dr. Chen Wentao, Dr. Yang Sheng, Hou Jingwen, Cheng Yupeng, Dr. Jiang Qiuping, Dr. Jin Jian, Dr. Zhao Baoquan, Dr. Jong-Uk Hou, Dr. Gu Ke, Dr. Cong Runming, Zhang Huan, and other members, for giving the encouragement and sharing insightful suggestions.

I am also pleased to say thank you to my lovely lab mates in ROSE Lab, including Dr. Wang Shiqi, Dr. Zhang Xinfeng, Dr. Fan Kui, Dr. Li Leida, Dr. Zhang Tianyi, Tao Qingyi, Liu Weide, Wu Zhonghua, Shi Xiangxi, Dr. Gu Jiuxiang, Wang Sicheng, Dr. Zha Zhiyuan, and the management team Dr. Dennis Sng and Wang Qian. I would always remember you guys for the fun-time we spent together, sleepless nights that gave us the courage to complete tasks before deadlines and for stimulating the discussions.

I am grateful to collaborators in I2R of A*STAR: Dr. Lin Jie, Dr. Vijay Chandrasekhar, Wang Zhe, and Dr. Huang Dongyan; and collaborators in Panasonic: Dr. Pongsak Lasang, Teerawat Piriyatharawet, and Dr. Shen Shengmei. It is my honor to work with you.

I am indebted to Prof. Duan Lingyu of PKU for taking me into the world of standardization. In AVS and MPEG, I met so many good friends, including Zhang Wenhan, Chen Jie, Dai Yongxing, Guan Tongfan, Lou Yihang, Baiyan, Wang Ce, and so on. So happy to have delicious food with you guys every time during standard meetings in different cities around the world.

A very special gratitude goes to Dr. Xu Long and Dr. Ma Lin for introducing me to NTU to pursue a PhD degree. It is life-changing and was the best decision I have made in my twenties. It not only led me to a fantastic 4-year doctorate experience, but also brought me my biggest cheerleader, Wan Jing. Thanks Wan Jing for being a companion by my side throughout my PhD journey, collating research data for me and always cheering me up. Wish I could put her into the subsequent "last but not least" parargraph soon.

Last but not least, I would like to say I love you to my family for everything I can imagine.

Abstract

Edge-cloud collaboration, where sensor data is acquired at edge end while analyses finish at cloud end, has become a new fashion for deep learning based visual analysis applications. The data communication which serves as the fundamental infrastructure is playing an important role in edge-cloud collaboration. To enable better balance among computing load, bandwidth usage and generalization ability, I propose a new paradigm of transmitting intermediate deep learning features instead of visual signals or ultimately utilized features, which inspires research and standardization of compression techniques for intermediate deep learning features.

To improve the data transmission efficiency, I develop a video-codec-based coding framework for intermediate deep learning feature compression. Besides, I also provide an overview and propose new coding tools for PreQuantization and Repack modules in the coding framework, with extensive comparative experiments analyzing their pros and cons. The optimal combination of the proposed modes can achieve over 50x compression ratio with less than 1% task performance drop, where the bitstream of intermediate deep learning features can be much smaller than that of corresponding visual signals. It is also worth mentioning that the proposed coding framework and coding tools have been partially adopted into the ongoing AVS (Audio Video Coding Standard Workgroup) - Visual Feature Coding Standard, and provided evidences for MPEG Video Coding for Machine (VCM) standard.

Moreover, to train more robust and generic backbone neural networks for feature extraction at edge end, I present an image quality assessment (IQA) based label smoothing method to tune the objective functions in neural network training. To provide better task-specific models on top of the intermediate deep features for the cloud end, I also propose a deep holographic network with a holographic composition operator to improve task performance with less memory costs. Extensive evaluations demonstrate the efficiency of the proposed methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, I firstly introduce the background and motivations for the new paradigm of intermediate deep learning feature transmission in Section 1.1. Next, I indicate possible research problems in context of the new data transmission paradigm in Section 1.2. Then, the main contributions of my studies on such research problems are summerized in Section 1.3. Finally, the organization of this thesis is provided in Section 1.4.

1.1 Background and Motivations

In the recent decade, deep neural networks (DNNs) have demonstrated the incomparable performance in various computer vision tasks, e.g., image classification [2–5], image object detection [6, 7], visual tracking [8] and visual retrieval [9]. Different from handcrafted features, like Histogram of Oriented Gradient (HOG) [10] and Scale-Invariant Feature Transform (SIFT) [11], deep learning features are directly learned from masses of data. For image classification, which is the fundamental task of computer vision, AlexNet [2] has achieved 9% better classification accuracy than the previous handcrafted methods in the 2012 ImageNet competition [12], which provides a large scale training dataset with 1.2 million images and one thousand categories. Inspired by the fantastic achievement of AlexNet, DNN models continue to be the undisputed leaders in the competition of ImageNet. In particular, both VGGNet [3] and GoogLeNet [13] announced promising performance in the ILSVRC 2014 classification challenge, which demonstrated that deeper and wider

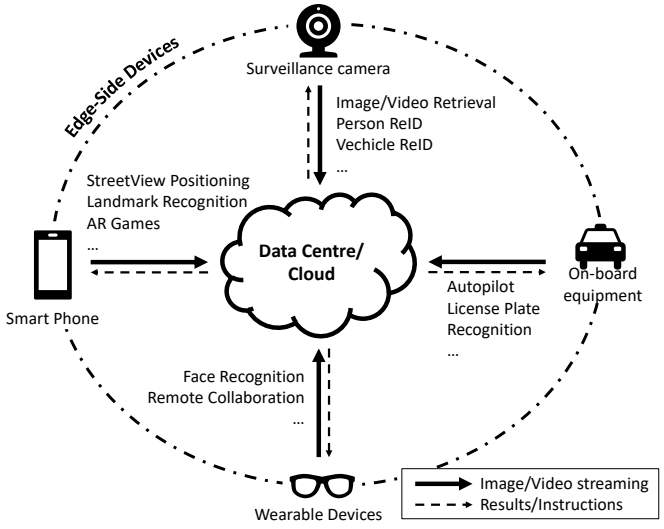


FIGURE 1.1: Visual analysis applications in the edge-cloud collaboration manner. Images and videos are acquired at the edge end and the analysis is performed at the cloud end. The two sides collaborate together through data transmission.

architectures can bring great benefits in learning better representations via large scale datasets. In 2016, He *et al.* also proposed residual blocks to enable very deep learning structure [4].

With the advances of network infrastructure, edge-cloud collaboration is springing up in recent years. In particular, the edge-side devices (a.k.a., front-end devices) acquire information from users or the physical world, which are subsequently transmitted to the cloud end (i.e., data center) for further process and analysis. In particular, for visual analysis, the edge-side devices deployed in the real world, such as surveillance cameras and wearable devices, acquire massive visual data which are transmitted to the cloud side for analyses, as shown in Figure 1.1. Many computer vision models powered by deep learning can be applied in such edge-cloud collaboration manner, such as pedestrian detection [14], person [15] and vehicle re-identification [16] in surveillance systems; autopilot [17] and license plate recognition [18] with on-board devices; face recognition [19, 20], landmark retrieval [21] and object detection [6, 7] in portable device (e.g., mobile, smart glasses) applications.

For data communication between edge-side devices and cloud sever, video compression and transmission serve as the foundation infrastructure in the traditional "compress-then-analyse" paradigm. In other words, the edge-side devices capture and compress the visual data at signal level, such that the coded bitstream can be transmitted to the cloud server for analyses. After the decoding process at the

cloud side, the feature extraction and visual analysis are subsequently performed. However, the vast amount of edge-side devices produce thousands-of-thousands bitstreams simultaneously, especially in the scenarios of video surveillance and Internet-of-Things (IoT). The signal level visual compression imposes high transmission burden, which is usually unaffordable in practical applications. Moreover, the computational load of the numerous deep learning models executed simultaneously for feature extraction also becomes a significant bottleneck for scaling up at the cloud end.

An alternative strategy "analyze-then-compress" [22], the rational of which lies in compressing and transmitting the features extracted at the front-end to the cloud center, provides a feasible solution as features instead of the visual signals are ultimately used for analysis. For hand-crafted features, the standards from MPEG including MPEG CDVS [23] and MPEG CDVA [24] specify the feature extraction and compression processes. For deep learning features, top-layer features of the deep learning models are usually transmitted to the cloud side, since the top-layer features of deep models are compact and can be straightforwardly utilized for analyses. For instance, in the face recognition task, the deep feature of a human face is only with dimension of 4K in Facebook DeepFace [20], 128 in Google FaceNet [25], and 300 in SenseTime DeepID3 [26]. In such scenarios, only the lightweight operations such as feature comparison are required to be performed at the cloud servers, while the heavy workloads of feature extraction are distributed to the front-end. Moreover, transmitting features is also favorable for privacy protection. In particular, instead of directly conveying the visual signal which may easily expose privacy, feature communication can largely avoid the disclosing of the visible information. However, one obstacle that potentially hinders the applications of deep learning feature compression is that deep learning models are normally designed and trained for specific tasks, and the top-layer features are extraordinary abstract and task-specific, making such compressed features difficult to generalize. When facing multi-task scenarios, multiple deep learning models need to be deployed on the edge, which results in the increase of design complexity and costs for the edge-end equipment. This also prevents the applications of the future standardization of the deep feature coding, as the standardized compact deep features shall be well generalized to enable the interoperability in different application scenarios.

In view of pros and cons of aforementioned two paradigms, a new paradigm of

transmitting intermediate deep learning feature instead of sensor data or ultimately utilized features (it will be denoted as "ultimate feature" in the following for convenience) is proposed in this thesis. With such paradigm, deep neural networks are split into two parts deployed on edge and cloud respectively. The inference of neural networks will be performed in a edge-cloud collaborative way. Particularly, generic backbone networks taking most of computing burden in visual analyses are located on the edge, while the task-specific components are assigned to the cloud, as shown in Figure 1.2. As such, the new paradigm reduces construction and operation costs of cloud servers by distributing the computing load. By applying generic deep learning architecture, it reduces the cost of edge-end equipment, cutting down energy consumption as a whole. The presented paradigm can be regarded as a compromise between the two extremes "analysis-then-compression" and "compression-then-analysis", and provides a good balance among the computational load, communication cost and the generalization ability.

1.2 Objective and Scope

To improve the performance and economize the costs for visual analysis in edge-cloud collaboration, we explore the following problems in this thesis:

- to verify the feasibility of transmission and compression of intermediate deep learning features;
- to improve coding efficiency of intermediate deep learning feature;
- to train more robust and generic backbone neural networks for feature extraction at edge side;
- to design better task-specific models on top of the intermediate deep features for the cloud side.

Relations among these research problems are demonstrated in Figure 1.2.

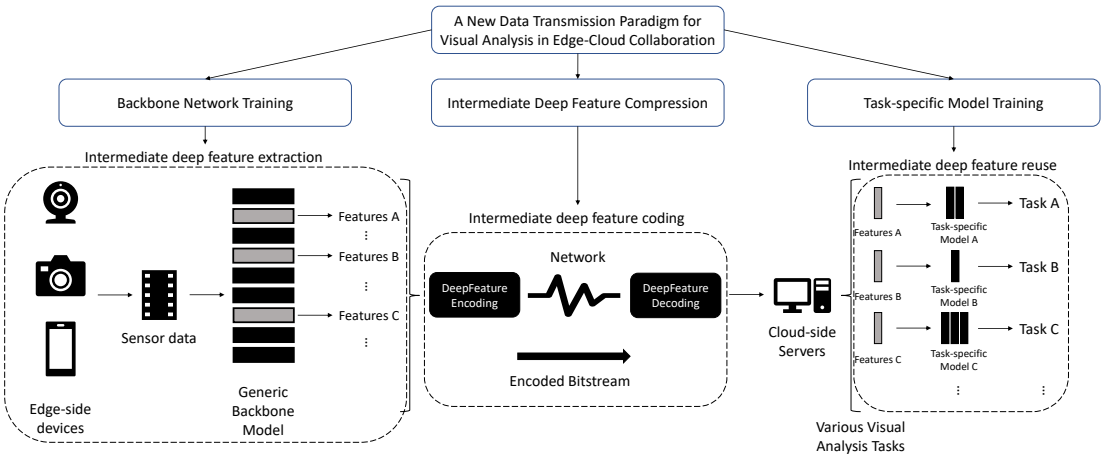


FIGURE 1.2: Diagram of the proposed paradigm of intermediate deep feature transmission, and how my research works presented in this thesis can contribute to the new paradigm.

1.3 Major Contributions

In this thesis, we have four main contributions to address the aforementioned four research problems respectively.

Firstly, we propose a new paradigm of transmitting intermediate deep learning features instead of visual signals or ultimate features to facilitate the collaborating approach between front and cloud ends. By literature research and analysis, we find the new paradigm can enable a good balance among the computational load, transmission load and the generalization ability for edge-cloud collaboration when deploying the deep neural networks for large scale cloud based visual analysis. Moreover, the presented strategy also makes the standardization of deep feature coding more feasible and promising, as a series of tasks can simultaneously benefit from the transmitted intermediate layer features. We also present the results for evaluations of lossless deep feature compression, which provide meaningful investigations and baselines for future research and standardization activities.

Secondly, we propose a video-codec-based coding framework and evaluation metrics for intermediate deep learning feature compression. Moreover, we propose several coding tools for PreQuantization and Repack modules in the coding framework. A large number of comparative tests have analyzed their pros and cons. Comprehensive experimental results show the effectiveness of my proposed coding framework and further verify the feasibility of the proposed data transmission paradigm. It is

worth mentioning that the proposed coding framework and coding tools have been partially adopted in to the ongoing AVS (Audio Video Coding Standard Workgroup) - Visual Feature Coding Standard, and provided evidences for MPEG Video Coding for Machine (VCM) standard.

Thirdly, we propose an image quality assessment (IQA) based label smoothing method to tune the objective functions in neural network training, which improves the robustness and generalizing capability of backbone deep learning models. In particular, in view of the fact that most widely deployed deep learning models are susceptible to various image distortions, we investigate in training deep models with data augmentation by both high and low quality images, and in-depth analyses of involving low quality images in the training process are also provided. Furthermore, we adopt the IQA measure for label smoothing in the deep neural network training process, which leads us to the robust neural network generalized to images with a broad range of quality levels.

Fourthly, we propose a deep holographic network (DHN) with a holographic composition operator to learn similarity metrics of deep learning features with multiple labels. The proposed method explicitly encodes distance metric at intermediate layer of the network instead of ranking loss driven deep metric learning (e.g. siamese loss and triplet loss), which could potentially enable the network to learn richer pairwise relationships. Moreover, the holographic composition is parameter-free and memory efficient, which economize the costs for cloud servers. Extensive evaluations on video retrieval task demonstrate that DHN performs significantly better than traditional deep metric learning approaches as well as other compositional networks.

1.4 Outline of the Thesis

The thesis consists of seven chapters which are summarized as follows. Chapter 1 gives a brief introduction to the background and motivations of this thesis, and highlights the major contents and contributions. Chapter 2 is the literature review to get a broad view of the current research status of related topics. Chapter 3 introduces the new paradigm for visual analysis in edge-cloud collaboration. Literature research and analysis are conducted to verify the feasibility of transmission and compression of intermediate deep learning features. Chapter 4 proposes a video-codec-based

coding framework, evaluation metrics, and coding tools for lossy intermediate deep learning feature compression. Chapter 5 presents the IQA based label smoothing technique for more robust and generic backbone network training. Chapter 6 proposes the deep holographic network which improves the performance for retrieval tasks while reduces network parameters and memory costs for implementation. Chapter 7 concludes the thesis with a summary of the aforementioned research works and provides potential research topics for the future endeavors.

Chapter 2

Literature Review

As introduced in Chapter 1, traditional data transmission paradigms for visual analysis tasks in edge-cloud collaboration can either be visual signal transmission or ultimate feature transmission. The data transmission and coding techniques for these two conventional paradigms have been studied and standardized for decades. However, only in recent years has the research on my proposed intermediate deep learning feature coding begun to emerge. There is a research gap, shown in Figure 2.1, for us to explore.

2.1 Visual signal transmission and compression

Compression and transmission for digital visual signals (i.e., digital images or videos) dates back to the middle of last century, after digital representation of the analog image and video signal was enabled by pulse code modulation (PCM) [27] in 1951. Earliest methods for image/video compression in 1950s were generally in the lossless way, including differential pulse-code modulation (DPCM) [28–30] and redundancy reduction methods such as variable length coding [31] and run-length coding [32]. Even with the development of transform algorithms in 1560s, such as Fourier transform [33] and Hadamard transform [34, 35], image/video coding was still inefficient and impractical for data transmission, as the bandwidth requirement was thousands times greater than the telecommunication bandwidth [36]. One milestone was the invention of discrete cosine transform (DCT) [37] which ushered in boom times for image/video coding development in 1970s. DCT largely boosted

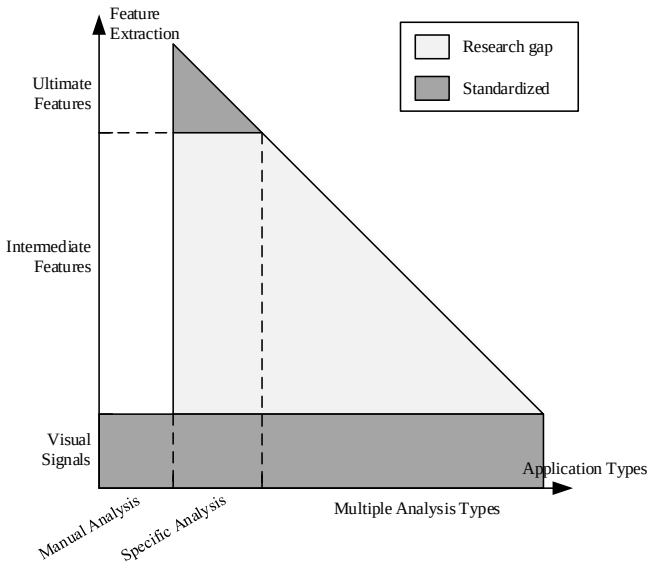


FIGURE 2.1: The compression and transmission of visual signals and ultimate features has been widely investigated and standardized, but the study on transmitting intermediate features is limited which needs further exploration.

the performance of images and intra-frame compression for videos. Combining with motion compensation techniques [38], motion-compensated DCT [39] significantly reduced inter-frame redundancy, which made video coding practical.

With the convergence of techniques, the standardization of image/video compression began from 1980s. Based on adaptive temporal DPCM method [40], the first digital video coding standard H.120 was launched in 1984. The inefficiency of H.120, due to lossless compression of DPCM, made this first standard hard to be accepted by the industry. Although been revised twice in 1988 and 1993 to enhance the performance, H.120 was overshadowed by the late-coming standard H.261. Ratified in 1988, H.261 [41] was the first practical standard for video coding by adopting motion-compensated DCT techniques. Also powered by DCT tools, JPEG [42] has became the most widely used image coding standard since 1992. The standardization of video coding does not slow down its pace in the 21st century. MPEG-4 Advanced Video Coding (AVC) / H.264 [43] finalized in 2003 is the most popular video codec in current world. It is widely used in Blu-ray Disc, HD DVD, HDTV and online video streaming. Beyond H.264, High Efficiency Video Coding (HEVC) / H.265 [44] was ratified in 2013, which offers up to 50% better compression than H.264 at the same visual quality level. As the successor to H.265, Versatile Video Coding (VVC) / H.266 [45] was finalized in 2020. This state-of-the-art video coding standard is

designed to achieve 50% lower data requirements relative to HEVC. It is also with much superior performance for 4K and 8K ultra HD videos than H.264 or H.265.

2.2 Ultimate feature transmission and compression

Ultimate feature coding, also known as description coding, is developed to generate compact image/video descriptions for efficient feature transmission, matching and storage. Compression of handcrafted features emerges after the innovation of SIFT descriptor [46] in 1999, which remains the most popular handcrafted feature in visual tasks. As SIFT descriptor is usually of larger size than the JPEG compressed image, feature compression is necessary to make the technique practical in applications such as mobile visual-search systems. Compression methods for handcrafted features can typically be categorized into hashing based [47, 48], vector quantization based [49–51], and transform coding based [52]. Inspired by the success of AlexNet [2] in 2012 ImageNet competition [12], deep learning methods quickly dominated the computer vision community. The handcrafted feature was overshadowed by the deep learning feature. To generate compact deep learning feature, existing feature compression methods either attempt to design deep learning models with small embedding layers [25, 53] or apply dimension-reduction / binarization (e.g. PCA and hashing) operations on top of the embedding layers [54–57].

Different from video coding standards, the standardization of ultimate feature coding should specifies both feature extraction and compression process to ensure the interoperability. It results in that ultimate feature coding standards are usually highly task-oriented. Designed for visual search tasks, including image matching, image retrieval and object instance recognition, Compact Descriptors for Visual Search (CDVS) [23] was released in 2015 under MPEG-7. On top of CDVS, Compact Descriptors for Video Analysis (CDVA) [24] was standardized to enable search, retrieval and instance recognition applications in video domain. It is worth noting that the CDVA standard explored the combination of handcrafted feature and deep learning feature [9].

2.3 Intermediate deep learning feature transmission and compression

Intermediate deep learning feature transmission and compression began to emerge since 2018. My works [58, 59] provided good motivation for this new research field. They also presents lossless evaluation results indicating the lossless methods are impractical for intermediate deep feature coding. As feature coding is to serve machine vision instead of human vision, rather than task performance metrics (e.g., mAP) and human perceptual metrics (e.g., SSIM [60]), our work [1] proposed new evaluation metrics for the lossy compression. Regarding lossy intermediate deep feature compression techniques, most works employed a video-codec-based coding framework which integrates the traditional video codecs to perform feature compression. [61, 62] performed lossy intermediate deep feature compression for object detection task utilizing traditional image/video codecs including PNG, JPEG, JPEG2000, VP9 and HEVC. My works [1, 59] proposed to have three modules (i.e., PreQuantization, Repack, and VideoEncoder) to encode the intermediate deep features and reported the compression performance on three computer vision tasks (i.e., image classification, image retrieval, and image object detection). In addition, there are some other works investigating the compression-aware intermediate feature generation by designing the neural network architecture [63] and improving the learning methods [64], and [63, 65] explored which types of features in a neural network should be transmitted.

In terms of standardization, AVS (Audio Video Coding Standard Workgroup of China) visual feature coding working group has made the first attempt to standardize intermediate deep learning feature compression since December 2017 [66]. The video-codec-based coding framework for intermediate deep feature compression has been adopted in this standard [67]. MPEG also launched adhoc group on Video Coding for Machines (VCM) on July 2019 [68], where intermediate deep learning feature coding serves an important part in the pipeline of this standard [69]. Compression results with the video-codec-based coding framework provided good evidences for VCM [70].

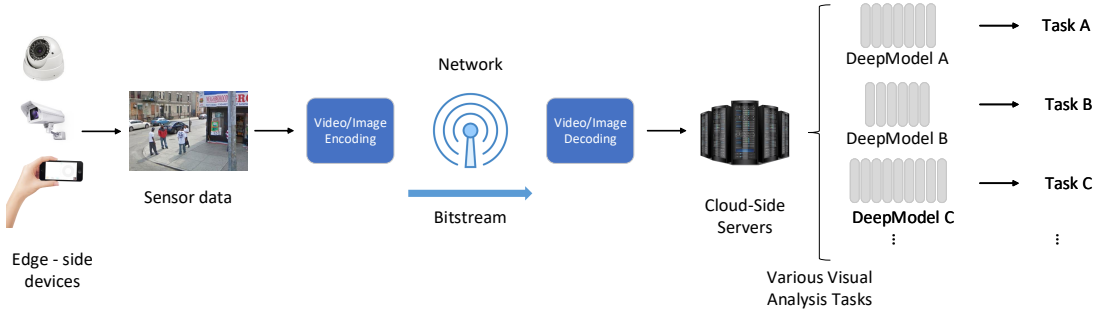
Chapter 3

Toward Transmitting and Compressing Intermediate Deep Learning Features

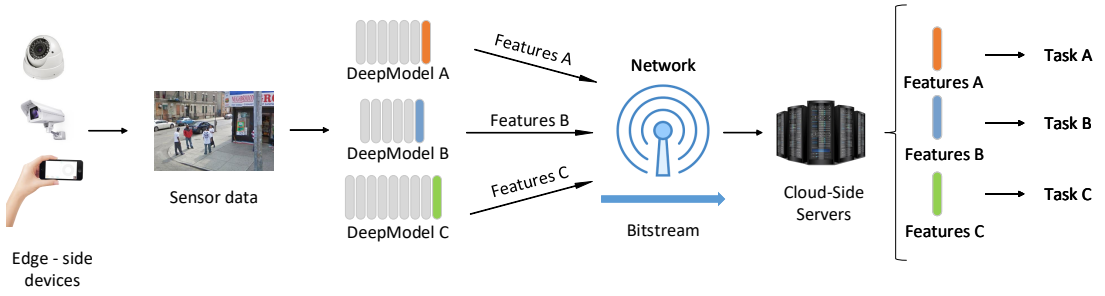
In this chapter, Section 3.1 introduces the proposed paradigm of intermediate deep learning feature transmission. In-depth analyses demonstrate the superiority of the proposed paradigm to the conventional approaches. By looking into the characteristics of intermediate deep learning features, the feasibility of feature compression with the uniform method is discussed in Section 3.2. Moreover, future standardization of intermediate deep learning feature coding is envisioned. Section 3.3 presents the preliminary evaluation results of lossless intermediate deep feature compression.

3.1 Toward transmission of deep learning features

In cloud-based visual analysis scenarios, visual signal acquisition and analysis are processed in distributed devices. In particular, images/videos are usually acquired in front-end devices (e.g. mobile phones, surveillance cameras) while the analysis is completed in the cloud side. As such, the data transmission between the edge and cloud sides is inevitable. Typically, the data to be transmitted can be either visual signals or features, as shown in Figure 3.1.



(a) Visual signal transmission. By transmitting the visual signal, a series of visual analysis tasks can be performed in the cloud. As such, the computing load including feature extraction and analysis is imposed on the cloud side.



(b) Ultimate feature transmission. Computing load can be distributed to front-end devices. Only specific types of analysis can be performed at the server-end, depending on the deep models used at the front-end.

FIGURE 3.1: Two commonly used strategies for cloud-based visual analysis.

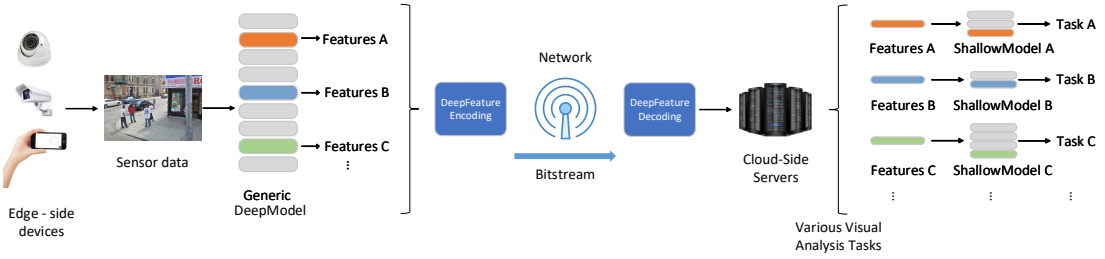


FIGURE 3.2: Diagram of the proposed approach. The intermediate-layer features of a generic deep model can be applied to a broad range of tasks. The features of specific layers will be transmitted based on the analysis requirements on the cloud side. On top of these transmitted features, shallow task-specific models will be applied for visual analysis.

As the most conventional paradigm, the visual signal compression and transmission methods have been well developed and standardized. As shown in Figure 3.1(a), visual signals (i.e., images and videos) are captured and encoded in the front-end for transmission, and decoded and analyzed in the cloud-end after receiving the

bit-stream. More specifically, various analysis tasks can be performed in the cloud-end, since the original visual signals are available. However, it is questionable that whether such visual signal level transmission can efficiently handle the visual big data. Moreover, although the state-of-the-art coding standards such as HEVC and VCC have dramatically improved the coding efficiency, all the computing load for analysis tasks remain on the cloud side. It is almost impossible for the cloud-side servers to timely analyze all the visual signals sent from the front-end devices in the context of visual big data, as the deep learning models are characterized with high computational complexity. For instance, the processing speed of a CNN-based object detection model can reach around 50 FPS with a single Titian X GPU in the best case [7], which is the best performance ever reported to my best knowledge. It implies that one state-of-art GPU card can only process two video signal inputs for one single task in real time. As the edge-side cameras can easily proliferate to a larger population, e.g., a smart city can have over one million surveillance cameras installed, a comparable amount of GPUs should be allocated in the cloud side to timely perform the visual analysis, which is unbearable in terms of economic cost and power consuming.

Benefiting from the development of the low-power AI processors [71–73], deep learning models are able to be implemented on front-end devices. It enables the intelligent analysis to be performed directly after the sensor data captured in the front-end devices, where this new fashion is named as "intelligent sensing". To reduce the computing load on cloud side, an alternative approach is to transmit the features instead of the visual signals, as shown in Figure 3.1(b). In this case, features are extracted right after the visual signals being captured in the front-end devices. Then, after the feature transmission, visual analyses based on the received features instead of the visual signal can be applied in the cloud-end servers. As the feature extraction usually takes the majority of computing load in a visual analysis application, the cloud-side server only needs to handle light computing loads, such as feature comparison, making visual big data analysis feasible. For handcrafted features, there are quite a few standards defining the feature extraction, compression and transmission, such as the previously mentioned MPEG CDVS and CDVA [23, 24]. As the feature extraction substantially performs dimensionality reduction on the original visual signals, the features are usually featured with less generalization ability than the visual signals, such that the transmitted features can only be applied to very specific types of tasks. For example, the features defined by

TABLE 3.1: Comparison of three transmission strategies

	Transmit Video Signal	Transmit Intermediate Feature	Transmit Ultimate Feature
Load allocation	All the computing load is on cloud end	The computing load can be largely shifted to the front end	
Usability	The transmitted signal can be applied to various tasks		The transmitted feature can be only applied to few specific tasks
Privacy	Visual signal can easily expose privacy of users	It is difficult to unscramble features, especially when without corresponding models	
Research Status	Well explored and standardized (AVC, HEVC, VVC, etc.)	Research gap	Well explored and standardized (MPEG-CDVA, CDVS, etc.)

CDVS are more suitable for image retrieval and matching tasks. The deep learning models, which are learned in a data-driven manner, are usually task-specific and the generalization ability is highly concerned in this scenario. Considering the deep learning model as one feature extractor, the top layer feature of a deep model is usually extracted as the visual embedding. Comparing with handcrafted features, although the deep leaning features are more expressive and powerful, they still cannot generalize to all the visual analysis tasks. In summary, transmitting the deep learning features can facilitate the shifting of the computing load from the cloud side to the front side which makes visual big data analysis possible. However, the supported analysis tasks that can be achieved on the cloud side are quite limited. In other words, the availability of visual analysis applications on the cloud side is constrained by the models employed in the front-end devices.

In view of the pros and cons of aforementioned two paradigms (summarized in Table 3.1), an approach which can ideally balance the computing load between the front and cloud sides without the limitation of the analysis capability in the cloud side is highly demanded. As shown in Figure 3.2, we aim to transmit the intermediate layer features instead of original visual signals and ultimate features. Deep learning models are usually characterized by hierarchical structures, which implies that a deep model shall be considered as a combination of stacked feature extractor rather than a single straightforward feature extractor. As such, higher layer features are characterized with large global receptive field which makes them more abstract and task-specific, while lower layer features are characterized with smaller receptive field and with more location information encoded in the 2D feature maps, enabling them to generalize to a broader range of analysis tasks. This provides the flexibility for the cloud side to request appropriate features from the front-end depending on the requirement of analysis task.

In this case, a generic deep model, the features of which can be applied to a broad range of tasks in visual analysis, is anticipated to be applied in front-end devices. Contemporarily, commonly-used pre-trained deep neural networks, such as VGGNet

and ResNet, which are trained on ImageNet dataset consisting of 1.2 million images of 1000 classes, in general can be regarded as generic. Features of these deep learning models are widely adopted in many applications as visual feature extractors, as shown in Figure 3.3. For instances, in image captioning tasks, the work [74] leverages the *conv*5 features (the output feature maps of the fifth convolutional block, we use the shorthands for convenience in the rest of this thesis, more details can be found in Table 3.5) of VGGNet to represent given images. The authors in [75] encoded the full image with the ResNet to extract both spatial and semantic information from its *conv*4 layer. In visual tracking tasks, *pool*4 and *pool*5 features of VGGNet were employed in [8]. In image object detection tasks, the work in [6] used the $fc2$ features and *pool*5 feature of VGGNet was employed in [76, 77]. In visual retrieval, the *pool*5 features of VGGNet were modified to introduce translation, scale and rotation invariances for image retrieval [9]. Handcrafted features and $fc1$ features of VGGNet were combined to achieve better retrieval performance [78]. In image QA tasks, the *conv*4 feature of ResNet was leveraged as the visual representation [79], and *pool*5 features of VGGNet and *conv*5 features of ResNet were used in [80]. In view of this, a plenty of visual analysis problems can be solved by applying task-specific neural networks on top of the features extracted by a generic deep model. As the generic model can provide the task-specific neural network with strong representations of the visual signals, a shallow architecture is usually adequate to handle the rest of the visual analysis task which is favorable in terms of the computing costs. Furthermore, we observe that most of task-specific networks prefer to take high level features (*conv*4 or higher) as their input. As the computing load are mainly laid on low layers in neural networks (as shown in Table 3.2), it can help saving great computing cost for the server-end with my proposed strategy. Thus, the deployment of my proposed data transmission approach can minimize the computing load on the cloud side while maximizing the availability of various analysis types. Furthermore, it is envisioned that in the future the deep learning models will be developed to more and more generic. At that stage, my proposed approach will have more advantages over the former ones.

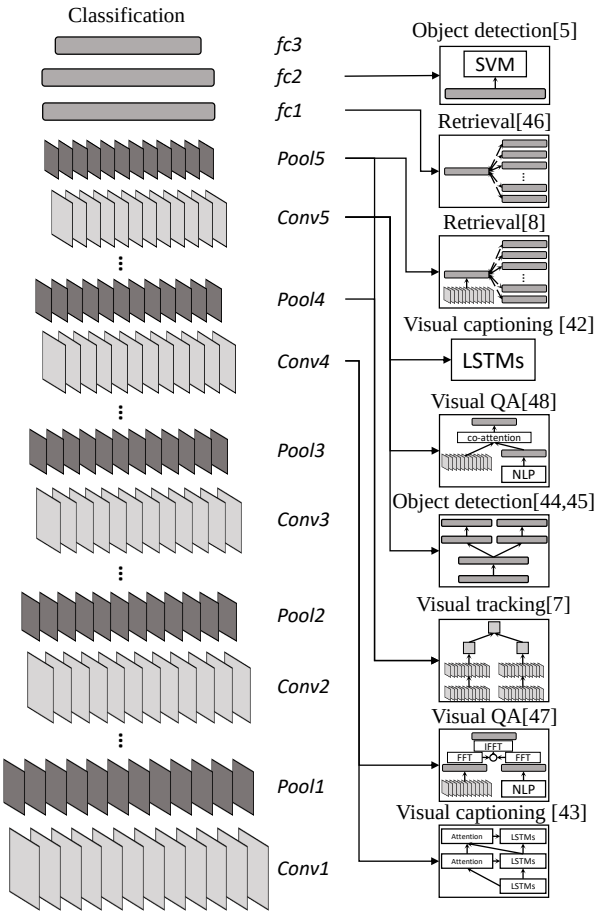


FIGURE 3.3: Generic deep models (e.g. VGGNet and ResNet trained on ImageNet classification task) are widely employed as backbone networks in many computer vision tasks. Task-specific networks are designed on top of the intermediate layers of generic models.

TABLE 3.2: The computational complexity of VGGNet and ResNet. The computing cost of nueral networks are usually laid on lower layers.

	FLOPs	
	VGGNet-16	ResNet-50
$conv1$	$1.94G(12.5\%)$	$0.12G(3.1\%)$
$conv2$	$2.77G(30.5\%)$	$0.67G(20.4\%)$
$conv3$	$4.62G(60.3\%)$	$0.95G(45.0\%)$
$conv4$	$4.62G(90.2\%)$	$1.39G(81.0\%)$
$conv5$	$1.39G(99.2\%)$	$0.73G(99.9\%)$
fc	$0.12G(100\%)$	$2.05M(100\%)$
Total	$15.47G$	$3.86G$

3.1.1 System comparisons of the three paradigms

To better elaborate pros and cons of the three data transmission paradigms in practical implementation, comparisons in terms of computation resources, bandwidth usage, power consumption, construction cost, security level of the edge-cloud systems are provided in Table 3.3 and Table 3.4. In particular, Table 3.3 gives the qualitative comparisons while Table 3.4 compares typical values of some examples of existing solutions.

In conventional visual signal transmission paradigm, also known as cloud computing, videos or images are captured at edge devices while analyzed by deep learning models fully deployed at cloud. To enable the execution of deep learning model, cloud servers are usually with heterogeneous computing architecture, such as CPU+GPU configuration. In the inference process of a neural network (process management such as memory control is not considered here), GPU can be used to carry heavy parallel computing operations such as convolution and matrix multiplication, while CPU can easily handle other operations such as NMS in objected detection tasks. To simultaneously execute two tasks (at 24 FPS), saying image classification and image object detection, it usually requires computing power of a half GPU and a CPU [7]. Taking Nvidia Tesla GPU [81] as an example, the system will cost around 150-Watt power and S$1000 construction fees at the cloud side, as shown in Table 3.4. As to the edge side, only a camera (sensor) is required to capture images or videos. Data transmitted in between is the image or video stream, which shall typically take over 5 Mbps at 1080P resolution. As fully meaningful images and videos carrying private and confidential information is conveyed in this paradigm, security level of such system is not high comparing with the other two feature transmission paradigms.

In ultimate feature transmission paradigm, also known as edge computing, deep learning models are deployed at the edge end to analyze the captured images or videos, while only lightweight computing tasks (e.g., feature matching, database retrieval) are allocated on the cloud side. To fully enable the inference of neural networks, computing units with heterogeneous architecture is also necessary for the edge devices. One common solution is to deploy a terminal box (edge server) connecting multiple camera devices to process the captured images or videos. The terminal box (edge server) are usually equipped with desktop CPUs and GPUs to afford heavy computation. With Nvidia GeForce GPUs [82] at edge, to process

two tasks simultaneously, the power consumption shall be similar to the cloud computing paradigm, but the construction cost will be much lower, as shown in Table 3.4. Another emerging solution is to use an embedded system [83] integrating CPU and AI accelerator to enable image capture and analysis in a compact design, such as smart cameras. Such solution can greatly reduce the power consumption and construction cost for single device. However, limited computing power make it requires multiple edge devices for multi-task scenario. As to the cloud side, computation resources is not necessary since the feature extraction has been fully processed in edge devices. Bandwidth usage between edge and cloud is also small, as the ultimate feature or final analytic result is usually compact.

In our proposed intermediate deep feature transmission paradigm, deep neural networks (i.e., AI models) are split into two parts deployed on edge and cloud respectively, where the inference of neural networks will be performed in an edge-cloud collaborative way. Particularly, generic backbone networks taking most of computing burden in visual analyses are located on the edge, while the task-specific components are assigned to the cloud. As the generic backbone networks usually consist of unified operators, such as convolution and matrix multiplication, ultra-low-power AI accelerator mainly consisted of multiplier–accumulator (MAC unit) is efficient for intermediate deep feature extraction at the edge. For example, Gyrfalcon Lightspeeur chip can process generic backbone networks with 2.8 TOPs at 224mW [84]. Task-specific networks with unique operators, such as NMS of detection task, can be applied with CPUs at cloud servers. As such, construction cost and power consumption of the proposed paradigm can be lower then the conventional data communication paradigms. Intermediate deep feature stream conveyed between the edge and cloud can be of much smaller volume than the full resolution image/video stream with the compression methods introduced in the following chapters.

TABLE 3.3: Qualitative system comparisons of the three paradigms.

	Cloud resources	Edge devices	Bandwidth usage	System
Cloud computing (transmit images / videos)	Heterogeneous computing architecture (e.g., CPU+GPU)	Sensor	Image/ video bitstream	Low privacy level AI models are open for customization Almost none engineering work*
Edge computing (transmit ultimate features/ results)	CPU (data management only)	Sensor + Computing units with heterogeneous architecture (e.g., CPU+GPU, CPU+AI accelerator)	Ultimate features/ results	High privacy level AI models are usually associated with edge devices Considerable engineering work*
Our proposed paradigm (transmit intermediate features)	CPU (+GPU) for lightweight computing	Sensor + AI accelerator	Intermediate features	High privacy level AI models are open for customization Little engineering work*

* Engineering work of hardware implementation for deep learning model on light-weighted platforms

TABLE 3.4: Quantitative system comparisons of the three paradigms.

	Cloud resources	Edge devices	Bandwidth usage	System
Cloud computing (transmit images/ videos)	100% computing load	/ (computing load)	>5Mbps @ 1080P	>5Mbps bandwidth usage
	~150W (0.5GPU*+CPU)	/ (power consumption)		150W power consumption
	~S$1000 (0.5GPU*)	/ (construction cost)		S$1000 construction cost
Edge computing (transmit ultimate features/ results)	/ (computing load)	100% computing load	~1-5 Kb/s	50Kbps- bandwidth usage
	/ (power consumption)	~15W×2 (2 smart cam.)** ~150W (0.5×terminal box)***		30-100W power consumption
	/ (construction cost)	~S$300×2 (2 smart cam.)** ~S$500 (0.5×terminal box)***		S$500-600 construction cost
Our proposed paradigm (transmit intermediate features)	1-10% computing load	90%+ computing load	~10-30 Kb/s	<250Kbps bandwidth usage
	~5W (0.1CPU)	~10W		15W power consumption
	~S$20 (0.1CPU)	~S$200		S$220 construction cost

* With Nvidia Tesla GPU solution [81]
** With Nvidia Jetson embedded solution [83]
*** With Nvidia GeForce GPU [82]

3.2 Deep learning feature compression

Transmitting intermediate-layer features instead of ultimate features and visual signals is superior at easing the computing load of the cloud end and maintaining the availability of various analysis tasks. However, the transmission bandwidth may limit the deployment of such approach, as the data volume of the intermediate-layer features is non-negligible. In deep learning models, the feature volume of first few layers can be even larger than the input visual signals, as shown in Table 3.5. As such, to optimize the bandwidth, compression for deep features is necessary.

3.2.1 Features of Deep Neural Networks

As the deep neural networks are characterized with a hierarchical structure of multiple layers, a group of features can be extracted, where the outputs of each layer of the deep model can be considered as features. In the rest of this section, features of convolutional neural networks (CNNs), which are the dominant deep model type in the visual computing task, will be investigated.

Typically, CNN consists of convolutional layers, normalization layers, pooling layers and fully connected layers as its hidden layers. The convolutional layer is the core building block of a CNN that accounts for most of the computational heavy lifting. It applies convolutional filtering to the input, and generates a 3-D matrix with appointed depth. For convenience, the feature of the last convolutional layer in $\mathbf{i}-$th block is recorded as *conv*$\mathbf{i}$ in the rest of this thesis. In general, pooling layers are periodically inserted in-between successive convolutional layers to progressively reduce the spatial size of the representations. The pooling layer operates independently on each slice of the convolutional feature and resizes it

spatially by combining multiple outputs of previous convolutional layer into a single value. The output of last pooling layer in **i**−th block is denoted as *pool***i** feature. It is also worth noting that some architectures use convolutional layers, instead of pooling layers, to down-sample the input matrix by modifying the stride factors of convolutional layers. Fully connected layers are stacked in the top of a CNN to extract high-level semantic information. Such layer applies connections to all outputs of the previous layer with a matrix multiplication followed by a bias offset. The output of a fully connected layer is a 1-D matrix (i.e. a vector) with fixed size. We call the feature of **i**−th fully connected layer as $fc\mathbf{i}$. Various normalization layers, such as local response normalization (LRN), batch normalization (BN), can also be adopted in a CNN. They regularize the network for better performance. Normalization layers are always parameter-free, and they will not change the shapes of input matrices. Such layers cannot bring the features with new semantic meanings. As such, the outputs of normalization layers will not be discussed in the rest of this thesis.

Although various CNN architectures have been proposed in recent years, we find that they share common characteristics in terms of hierarchical structures and feature sizes. Table 3.5 lists four milestone CNN architectures in image classification tasks, including AlexNet [2], VGGNet [3], ResNet [4] and DenseNet [5]. With the same input size, these state-of-the-art CNNs extract the features in a hierarchical manner. In the convolutional part, the sizes of feature maps gradually get reduced along with the inference process. It is found to be regular that the feature map size will be halved after one certain block. Such block can be composed of either one single convolutional layer such as in AlexNet, few stacked convolutional layers such as in VggNet, or some more advanced structures like residual or dense connections of several convolutional layers. Along with the size reduction, feature maps usually can represent more high level semantic information in higher layers. When the feature map size becomes small enough, fully connected layers will be followed to convert the visual characteristics into the task-related semantic space, which will largely reduce the spatial information in the feature. It can be easily observed that the CNNs share similar feature map size for each block, only the number of feature maps varies. In addition, the fully connected layer features are with similar volume. Such observations imply that CNNs are with analogical hierarchical structures which can provide semblable features. It is also worth mentioning that most activation functions of CNNs constrains the numerical distribution of deep

TABLE 3.5: Architectures of four benchmark deep convolutional neural networks. 'op. unit' stands for operation unit, and it can be either a single layer or a combination of multiple layers. 'feat. symbol' is the symbol of feature which indicates the specific type of feature. 'feat. size' contains the shape and bit size of the feature.

Blocks	AlexNet			VGGNet			ResNet			DenseNet		
	op. unit	feat. symbol	feat. size	op. unit	feat. symbol	feat. size	op. unit	feat. symbol	feat. size	op. unit	feat. symbol	feat. size
Input	$224 \times 224 \times 3$ RGB image											
Conv. Block 1	single conv	$conv1$	$56 \times 56 \times 96$	stacked conv	$conv1$	$224 \times 224 \times 64$	single conv	$conv1$	$112 \times 112 \times 64$	single conv	$conv1$	$112 \times 112 \times 64$
	max pool	$pool1$	$28 \times 28 \times 96$	max pool	$pool1$	$112 \times 112 \times 64$						
Conv. Block 2	single conv	$conv2$	$28 \times 28 \times 256$	stacked conv	$conv2$	$112 \times 112 \times 128$	max pool	$pool2$	$56 \times 56 \times 64$	max pool	$pool2$	$56 \times 56 \times 64$
	max pool	$pool2$	$14 \times 14 \times 256$	max pool	$pool2$	$56 \times 56 \times 128$	residual blk.	$conv2$	$56 \times 56 \times 256$	dense + trans.	$conv2$	$56 \times 56 \times 64$
Conv. Block 3	single conv	$conv3$	$14 \times 14 \times 384$	stacked conv	$conv3$	$56 \times 56 \times 256$	residual blk.	$conv3$	$28 \times 28 \times 512$	ave. pool	$pool3$	$28 \times 28 \times 64$
				max pool	$pool3$	$28 \times 28 \times 256$				dense + trans.	$conv3$	$28 \times 28 \times 64$
Conv. Block 4	single conv	$conv4$	$14 \times 14 \times 384$	stacked conv	$conv4$	$28 \times 28 \times 512$	residual blk.	$conv4$	$14 \times 14 \times 1024$	ave. pool	$pool4$	$14 \times 14 \times 64$
				max pool	$pool1$	$14 \times 14 \times 512$				dense + trans.	$conv4$	$14 \times 14 \times 64$
Conv. Block 5	single conv	$conv5$	$14 \times 14 \times 256$	stacked conv	$conv5$	$14 \times 14 \times 512$	residual blk.	$conv5$	$7 \times 7 \times 2048$	ave. pool	$pool5$	$7 \times 7 \times 64$
	max pool	$pool5$	$7 \times 7 \times 256$	max pool	$pool5$	$7 \times 7 \times 512$				dense	$conv5$	$7 \times 7 \times ?$
FC Block	single fc	$fc1$	4096	single fc	$fc1$	4096	ave. pool	$pool6$	$1 \times 1 \times 2048$	ave. pool	$pool6$	$1 \times 1 \times ?$
	single fc	$fc2$	4096	single fc	$fc2$	4096						
	single fc	$fc3$	1000	single fc	$fc3$	1000	single fc	$fc1$	1000	single fc	$fc1$	1000

features in certain ranges (e.g., range of outputs for ReLU is $[0, \infty)$, sigmoid is $(0, 1)$, tanh is $(-1, 1)$). This property is useful for the deep feature compression.

3.2.2 Toward Standardization of Deep Feature Compression

To ensure compatibility and facilitate interoperability, a series of standards have been established for transmitting visual signal and handcrafted ultimate features. It is envisioned that my proposed approach of transmitting intermediate deep features can also be standardized in the future.

Conventionally, to fully ensure interoperability, feature coding standards usually specify both feature extraction and compression processes [23, 24]. It is because, in feature coding, the features from different extractors can be diverse from each other in terms of shape, distribution, numerical type, etc. [85]. In view of this, feature extractors should be carefully designed and specified in feature coding standards including CDVS and CDVA. Such standardization strategy obtains interoperability by sacrificing the compatibility for different feature extractors and the generality for different tasks. Regarding intermediate deep feature coding, benefiting from the characteristics of the deep features, we believe the interoperability can be ensured together with the compatibility and generality. Although the deep learning models are kaleidoscopic, the deep features share similar shapes and distributions in specific layers as discussed in Section 3.2.1. Such observation provides possibility to ensure the interoperability by only standardize the feature compression process. In this manner, the choice of feature extractor (i.e. deep learning model) will be left open

for system customization, which is conducive for the standard to keep long-lasting vitality, since any emerging deep learning models in the future can be compatible with the standard seamlessly. Moreover, since intermediate features are with better generalization ability than the ultimate features to apply to various tasks, the generality of the standard can be further ensured.

Concretely, regarding to the compression process, it is expected to remove the redundancy of deep learning features in both single images and video sequences. Also, deep feature compression methods should be either lossless or lossy. This is very similar to video coding standards such as HEVC which supports both image/video compression and lossless/lossy methods. Instead of being uniform as the image or video signals, the characteristics of deep features are more diverging. For example, features of convolutional layers are in the form of feature maps which is very different from features of fully-connected layers that are in terms of vectors. In view of this, there should be different compression strategies for distinct feature categories (i.e. *conv*, *pool*, fc). For the *conv* and *pool* feature which is a combination of spatial 2D signals, many video coding techniques can be transferred to compress deep features, such as inter / intra prediction and rate distortion optimization. For the fc feature which is a vector, general data compression methods can be referred, such as entropy coding. As the dynamic range of deep feature values is commonly smaller compared with the numerical range of its data type, quantization methods should be efficient to remove the redundancy. As such, how the redundancies of deep features can be removed and how to minimize the performance drop of deep feature while maximizing the redundancy reduction should be further investigated during the standardization explorations.

3.3 Evaluations on Lossless Compression of Intermediate Deep Learning Features

In this section, we present the evaluation results of the lossless compression of intermediate deep learning features. By evaluating the benchmark lossless data compression methods on deep learning features extracted with several widely-used networks, we aim to provide the baselines for further research and standardization activities.

3.3.1 Experiment Setup

To provide the meaningful baseline evaluations, we carefully selected the generic deep learning models and data compression methods. In particular, the deep learning models are chosen based on the principle that the extracted intermediate features should be generic enough to be applied to a wide range of tasks in visual analysis. Then four conventional and widely used compression algorithms are selected to perform deep feature compression.

3.3.1.1 Deep Learning Models and Datasets

In this thesis, we adopt official models of VGGNets and ResNets to perform feature extraction. These commonly used pre-trained CNN models are the winners of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 and 2015, which are trained on ImageNet dataset consisting of 1.2 million images of 1000 classes. The bellwethers of ILSVRC become the common choice for image feature extraction as their features can be regarded as generic. As mentioned in Section 3.1, for many computer vision applications, the task-specific models are designed on top of the features of VGGNets and ResNets, such as image captioning [74, 75], visual tracking [8], image object detection [6, 7, 76, 77], visual retrieval [9, 78], image QA [79, 80].

VGGNet: Simonyan and Zisserman developped VGGNet at the ILSVRC 2014. VGG-16 stands out from the six variants of VGGNet for its good balance among performance and computational complexity. VGG-16 is very appealing thanks its neat architecture consisting of 16 convolutional layers which only performs 3×3 convolution and 2×2 pooling all the way through. Currently it is the most preferred choice to extract features from images in computer vision community.

ResNet: At the ILSVRC 2015, He *et al.* introduced Residual Neural Network (ResNet) which contains a novel technique called "skip connections". Thanks to this new structure, the networks are able to go into very deep (152 layers in He *et al.*'s work) with lower complexity than VGGNet. ResNets have three commonly used variants with 50, 101, 152 layers respectively. Benefited from the astonishing performance (top-5 error rate of 5.25%, 4.60%, 4.49%), ResNets are increasingly adopted by various tasks.

We extract the deep features of the aforementioned deep learning models on a subset of the validation set of the ImageNet 2012 dataset [12]. To economize the subsequent compression time while maintaining the variety of test image categories, we randomly choose one image from each of the 1,000 classes. Overall, we evaluate the compression performance on each feature type with 1,000 feature entities.

3.3.1.2 Compression Methods

Analogous to data compression, deep feature compression aims to encode deep learning features with fewer bits than the original, which can be either lossless or lossy. The lossless compression ensures that the decoded feature is identical with the one before encoding. As such, analysis of performance degradation will be avoided. Lossy feature compression reduces the data size by optimizing the information loss and bitrate, which may end in performance loss of corresponding deep learning models. In this section, we evaluate the performance with four conventional lossless data compression methods, and lossy methods will be explored in the next section. The adopted compression methods are described as follows.

GZIP: GZIP [86] was developed in the early 1990's as a replacement for patent-encumbered algorithms such as LZW [87]. The DEFLATE algorithm [88] is the core of GZIP, which employs LZ77 [89] followed by Huffman coding [90]. The GZIP algorithm enjoys very fast compression speed and small memory footprint.

ZLIB: ZLIB [91] was adapted from the GZIP in mid-1990's. It abstracts the DEFLATE algorithm to achieve higher compression ratio and faster speed. ZLIB is now widely used for data transmission and storage.

BZIP2: BZIP2 [92] compresses the initial data with Run-length encoding (RLE) and applies the Burrows-Wheeler transform to rearrange character strings into runs of similar characters. It then uses move-to-front (MTF) transform and a combination of RLE and Huffman coding to efficiently represent the data stream. BZIP2 is generally considered with higher compression ratio than the LZW and Deflate algorithms with relatively slower speed.

LZMA: The Lempel–Ziv–Markov chain algorithm (LZMA) [93] uses a dictionary compression scheme, similar to LZ77 [89], followed by a range encoder. Comparing to LZ77, the dictionary compressor is with huge dictionary sizes (up to 4 GB). The

TABLE 3.6: Lossless feature compression results of VGGNet.

Feat. Type	Feat. Shape	Data Volume	Non-zero	GZIP		ZLIB		BZIP2		LZMA	
				Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost
conv1	$224 \times 224 \times 64$	$12.25M$	0.685 ± 0.021	0.639 ± 0.044	2.016 ± 0.345	0.641 ± 0.044	0.521 ± 0.040	0.648 ± 0.044	3.170 ± 0.572	0.609 ± 0.045	5.146 ± 0.585
pool1	$112 \times 112 \times 64$	$3.0625M$	0.815 ± 0.023	0.752 ± 0.055	0.345 ± 0.050	0.753 ± 0.055	0.147 ± 0.013	0.766 ± 0.053	0.688 ± 0.227	0.719 ± 0.058	0.815 ± 0.056
conv2	$112 \times 112 \times 128$	$6.125M$	0.480 ± 0.012	0.482 ± 0.028	1.369 ± 0.199	0.486 ± 0.028	0.216 ± 0.015	0.475 ± 0.026	1.556 ± 0.489	0.460 ± 0.028	2.451 ± 0.330
pool2	$56 \times 56 \times 128$	$1568K$	0.694 ± 0.033	0.680 ± 0.046	0.378 ± 0.039	0.683 ± 0.046	0.075 ± 0.006	0.672 ± 0.043	0.239 ± 0.076	0.655 ± 0.046	0.493 ± 0.021
conv3	$56 \times 56 \times 256$	$3.0625M$	0.302 ± 0.026	0.319 ± 0.030	0.541 ± 0.098	0.322 ± 0.030	0.077 ± 0.007	0.308 ± 0.028	1.188 ± 0.095	0.301 ± 0.028	1.130 ± 0.120
pool3	$28 \times 28 \times 256$	$784K$	0.484 ± 0.049	0.502 ± 0.050	0.241 ± 0.032	0.506 ± 0.050	0.031 ± 0.003	0.484 ± 0.047	0.139 ± 0.055	0.478 ± 0.049	0.259 ± 0.021
conv4	$28 \times 28 \times 512$	$1568K$	0.127 ± 0.014	0.146 ± 0.016	0.124 ± 0.016	0.148 ± 0.016	0.023 ± 0.002	0.138 ± 0.015	0.511 ± 0.022	0.137 ± 0.014	0.348 ± 0.054
pool4	$14 \times 14 \times 512$	$392K$	0.243 ± 0.030	0.274 ± 0.033	0.071 ± 0.011	0.278 ± 0.033	$9.969m \pm 1.114m$	0.259 ± 0.030	0.131 ± 0.019	0.255 ± 0.030	0.122 ± 0.012
conv5	$14 \times 14 \times 512$	$392K$	0.068 ± 0.017	0.079 ± 0.018	0.024 ± 0.004	0.081 ± 0.019	$4.045m \pm 0.411m$	0.075 ± 0.018	0.108 ± 0.002	0.074 ± 0.017	0.048 ± 0.010
pool5	$7 \times 7 \times 512$	$98K$	0.124 ± 0.028	0.147 ± 0.031	0.013 ± 0.002	0.150 ± 0.032	$1.498m \pm 0.213m$	0.143 ± 0.030	0.029 ± 0.001	0.139 ± 0.029	0.018 ± 0.003
fc1	4096×1	$16K$	0.248 ± 0.046	0.304 ± 0.047	$5.204m \pm 0.662m$	0.307 ± 0.047	$0.537m \pm 0.062m$	0.305 ± 0.046	$2.918m \pm 0.191m$	0.288 ± 0.045	$6.800m \pm 0.849m$
fc2	4096×1	$16K$	0.259 ± 0.061	0.315 ± 0.062	$4.964m \pm 0.485m$	0.318 ± 0.062	$0.540m \pm 0.070m$	0.317 ± 0.061	$2.933m \pm 0.191m$	0.300 ± 0.060	$6.257m \pm 0.249m$
fc3	1000×1	4000	1.000 ± 0.000	0.940 ± 0.002	$0.156m \pm 0.016m$	0.937 ± 0.002	$0.115m \pm 0.008m$	1.086 ± 0.004	$1.310m \pm 0.009m$	0.964 ± 0.006	$2.007m \pm 0.090m$

range encoder employs a complex mechanism to make probability predictions of each bit. LZMA features a generally high compression ratio with a comparable speed [94].

3.3.2 Results

To evaluate the deep feature compression performance, deep features are firstly extracted from different layers of deep models. Subsequently, four classic lossless compression algorithms with default configurations are applied on the extracted features. The feature extractions are performed by Caffe and Tensorflow on a NVIDIA GeForce 1080 GPU. The compression processes are conducted on Intel Xeon CPU E5-2650 v2 @ 2.60GHz with only one thread.

We mainly consider two criteria to evaluate the compression performance: compression rate and computational time cost. In particular, the compression rate is defined as

$$Compression\ rate = \frac{data\ volume\ after\ compression}{data\ volume\ before\ compression}. \tag{3.1}$$

In this thesis, we report the mean compression rate and computational time over 1,000 samples of each type of the deep learning features for the four lossless compression methods. The statistics of each type of features, including the shape, volume and non-zero rate, are also provided. In particular, based on the observation that ReLU function in deep learning models can result in a plenty of identical values (i.e. zeros), which may directly affect the compression rate, we list the mean non-zero rates of each type of feature for compression rates comparison. The results are listed in Tables 3.6 to 3.9 for VGGNet-16, ResNet-50, ResNet-101, ResNet-152.

TABLE 3.7: Lossless feature compression results of ResNet-50.

Feat. Type	Feat. Shape	Data Volume	Non-zero	GZIP		ZLIB		BZIP2		LZMA	
				Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost
*conv*1	$112 \times 112 \times 64$	$3.0625M$	0.686 ± 0.026	0.619 ± 0.019	0.254 ± 0.091	0.619 ± 0.019	0.117 ± 0.006	0.639 ± 0.021	1.067 ± 0.047	0.578 ± 0.023	0.765 ± 0.083
*pool*1	$56 \times 56 \times 64$	$784K$	0.765 ± 0.031	0.529 ± 0.025	0.050 ± 0.009	0.529 ± 0.025	0.024 ± 0.001	0.638 ± 0.029	0.261 ± 0.017	0.459 ± 0.024	0.162 ± 0.014
*conv*2	$56 \times 56 \times 256$	$3.0625M$	0.707 ± 0.028	0.681 ± 0.018	0.595 ± 0.089	0.683 ± 0.018	0.138 ± 0.003	0.681 ± 0.022	0.442 ± 0.069	0.652 ± 0.016	1.067 ± 0.091
*conv*3	$28 \times 28 \times 512$	$1568K$	0.686 ± 0.014	0.672 ± 0.011	0.327 ± 0.037	0.674 ± 0.011	0.068 ± 0.002	0.666 ± 0.012	0.204 ± 0.021	0.649 ± 0.011	0.488 ± 0.049
*conv*4	$14 \times 14 \times 1024$	$784K$	0.541 ± 0.024	0.548 ± 0.021	0.172 ± 0.017	0.550 ± 0.021	0.030 ± 0.001	0.535 ± 0.022	0.126 ± 0.031	0.526 ± 0.021	0.238 ± 0.013
*conv*5	$7 \times 7 \times 2048$	$392K$	0.176 ± 0.032	0.200 ± 0.033	0.065 ± 0.011	0.203 ± 0.034	$7.311m \pm 0.943m$	0.190 ± 0.032	0.115 ± 0.006	0.188 ± 0.032	0.086 ± 0.012
*pool*5	$1 \times 1 \times 2048$	$8K$	0.887 ± 0.063	0.858 ± 0.041	$0.405m \pm 0.176m$	0.857 ± 0.041	$0.265m \pm 0.035m$	0.935 ± 0.053	$1.852m \pm 0.048m$	0.861 ± 0.044	$2.476m \pm 0.081m$
$fc1$	1000×1	4000	1.000 ± 0.000	0.941 ± 0.002	$0.155m \pm 0.015m$	0.938 ± 0.002	$0.115m \pm 0.012m$	1.086 ± 0.004	$1.246m \pm 0.021m$	0.965 ± 0.006	$1.699m \pm 0.038m$

TABLE 3.8: Lossless feature compression results of ResNet-101.

Feat. Type	Feat. Shape	Data Volume	Non-zero	GZIP		ZLIB		BZIP2		LZMA	
				Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost
*conv*1	$112 \times 112 \times 64$	$3.0625M$	0.660 ± 0.023	0.588 ± 0.017	0.216 ± 0.070	0.588 ± 0.018	0.110 ± 0.005	0.610 ± 0.020	1.042 ± 0.031	0.543 ± 0.021	0.640 ± 0.047
*pool*1	$56 \times 56 \times 64$	$784K$	0.713 ± 0.026	0.489 ± 0.023	0.041 ± 0.007	0.490 ± 0.023	0.022 ± 0.001	0.590 ± 0.026	0.261 ± 0.008	0.422 ± 0.022	0.143 ± 0.015
*conv*2	$56 \times 56 \times 256$	$3.0625M$	0.687 ± 0.037	0.663 ± 0.026	0.601 ± 0.091	0.665 ± 0.026	0.131 ± 0.004	0.662 ± 0.030	0.441 ± 0.059	0.632 ± 0.024	0.980 ± 0.067
*conv*3	$28 \times 28 \times 512$	$1568K$	0.724 ± 0.021	0.703 ± 0.015	0.325 ± 0.042	0.705 ± 0.014	0.070 ± 0.002	0.699 ± 0.017	0.201 ± 0.011	0.676 ± 0.013	0.427 ± 0.017
*conv*4	$14 \times 14 \times 1024$	$784K$	0.730 ± 0.032	0.711 ± 0.025	0.180 ± 0.037	0.714 ± 0.024	0.036 ± 0.002	0.704 ± 0.028	0.100 ± 0.005	0.691 ± 0.025	0.202 ± 0.009
*conv*5	$7 \times 7 \times 2048$	$392K$	0.164 ± 0.035	0.187 ± 0.037	0.060 ± 0.011	0.190 ± 0.037	$6.516m \pm 0.957m$	0.178 ± 0.035	0.116 ± 0.006	0.176 ± 0.035	0.079 ± 0.013
*pool*5	$1 \times 1 \times 2048$	$8K$	0.860 ± 0.075	0.841 ± 0.051	$0.455m \pm 0.214m$	0.840 ± 0.051	$0.276m \pm 0.035m$	0.914 ± 0.064	$1.848m \pm 0.062m$	0.843 ± 0.054	$2.544m \pm 0.104m$
$fc1$	1000×1	4000	1.000 ± 0.000	0.941 ± 0.002	$0.146m \pm 0.005m$	0.938 ± 0.002	$0.115m \pm 0.008m$	1.086 ± 0.004	$1.233m \pm 0.030m$	0.965 ± 0.006	$1.746m \pm 0.039m$

TABLE 3.9: Lossless feature compression results of ResNet-152.

Feat. Type	Feat. Shape	Data Volume	Non-zero	GZIP		ZLIB		BZIP2		LZMA	
				Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost	Comp. Rate	Time Cost
*conv*1	$112 \times 112 \times 64$	$3.0625M$	0.595 ± 0.026	0.527 ± 0.022	0.177 ± 0.046	0.527 ± 0.022	0.102 ± 0.005	0.548 ± 0.025	0.981 ± 0.049	0.485 ± 0.024	0.679 ± 0.057
*pool*1	$56 \times 56 \times 64$	$784K$	0.640 ± 0.029	0.441 ± 0.025	0.040 ± 0.008	0.442 ± 0.025	0.020 ± 0.001	0.532 ± 0.029	0.248 ± 0.010	0.379 ± 0.023	0.144 ± 0.012
*conv*2	$56 \times 56 \times 256$	$3.0625M$	0.715 ± 0.030	0.681 ± 0.021	0.538 ± 0.076	0.682 ± 0.021	0.138 ± 0.006	0.685 ± 0.024	0.683 ± 0.186	0.644 ± 0.017	1.121 ± 0.096
*conv*3	$28 \times 28 \times 512$	$1568K$	0.757 ± 0.023	0.729 ± 0.017	0.322 ± 0.048	0.731 ± 0.017	0.072 ± 0.003	0.727 ± 0.020	0.201 ± 0.011	0.704 ± 0.016	0.485 ± 0.032
*conv*4	$14 \times 14 \times 1024$	$784K$	0.765 ± 0.031	0.740 ± 0.023	0.165 ± 0.038	0.742 ± 0.022	0.037 ± 0.002	0.735 ± 0.026	0.098 ± 0.004	0.719 ± 0.023	0.231 ± 0.016
*conv*5	$7 \times 7 \times 2048$	$392K$	0.165 ± 0.034	0.188 ± 0.036	0.062 ± 0.011	0.191 ± 0.036	$6.835m \pm 1.000m$	0.178 ± 0.034	0.121 ± 0.006	0.177 ± 0.034	0.085 ± 0.013
*pool*5	$1 \times 1 \times 2048$	$8K$	0.860 ± 0.074	0.841 ± 0.050	$0.469m \pm 0.218m$	0.840 ± 0.050	$0.276m \pm 0.035m$	0.914 ± 0.063	$1.980m \pm 0.054m$	0.843 ± 0.054	$2.841m \pm 0.096$
$fc1$	1000×1	4000	1.000 ± 0.000	0.941 ± 0.002	$0.155m \pm 0.003m$	0.938 ± 0.002	$0.115m \pm 0.007m$	1.086 ± 0.004	$1.286m \pm 0.023m$	0.965 ± 0.006	$1.986m \pm 0.037m$

From Tables 3.6, 3.7, 3.8, 3.9 we can see that, in terms of compression time cost, the ZLIB method is with the standout compression speed. It takes less time than the other three methods conspicuously on each type of feature. On the contrary, the speed performance of GZIP, BZIP2 and LZMA varies on different feature types. For instance, when compressing the large-volume features (e.g., *conv*1 of VGGNet with 12.25MByte) and small-volume features (e.g., *pool*5 of ResNets and fc features which are under 16KByte), LZMA is the slowest among the four methods. When dealing with features of the volume between 98KByte and 3.0625MByte, BZIP2 takes longer time than LZMA in some cases (e.g., *conv*4-*pool*5 of VGGNet and *conv*1, *pool*1, *conv*4, *conv*5 of ResNet). GZIP takes less time than LZMA and BZIP2 in most cases. However, regarding *pool*2, *pool*3 of VGGNet and *conv*2, *conv*3 of ResNet, which are with the volume between 784KByte and 3.0625MByte, BZIP2 is faster. Overall, the compression speed of ZLIB is orders of magnitude faster than the other three. GZIP, BZIP2 and LZMA are with comparable time cost, while GZIP is generally faster and LZMA is relatively slow among the three. The speed of BZIP2 is not stable and highly depends on feature types.

Regarding the compression performance, we can see that the performance of LZMA is superior to the other three methods on most of feature types except for $fc3$ of VGGNet and *pool*5/$fc1$ of ResNets, while ZLIB performs better on $fc3$ of VGGNet and *pool*5/$fc1$ of ResNets. GZIP has similar performance compared with

ZLIB, though it wins ZLIB a little bit on compressing features except for $fc3$ of VGGNet and $pool5/fc1$ of ResNets. BZIP2 provides comparable compression rates on features except for $fc3$ of VGGNet and $pool5/fc1$ of ResNets, whereas its performance on $fc3$ of VGGNet and $pool5/fc1$ of ResNets is obviously worse than the other three methods. In particular, the compression rates of BZIP2 on the final layer features of all the four tested networks are higher than 1.0, which implies that the compressed data volume is even larger than the uncompressed one. The above observations show that the performance of the four methods on $fc3$ of VGGNet and $pool5/fc1$ of ResNets is largely different from the other feature types. This may be because that the distributions of $fc3$ of VGGNet and $pool5$, $fc1$ of ResNets are different from the others. These three types of features are from the top layers of the corresponding neural networks. Unlike the low level layer features which are stacked 2-D maps with remaining spatial correlations among the elements, the top layer features are in the form of 1-D vector which is lack of correlations between its elements, as mentioned in Section 3.2.1. Furthermore, these three types of features are of higher non-zero rates than the other feature types, which may also affects the performance of the compression methods. From Tables 3.6, 3.7, 3.8, 3.9, we can also find that the compression rates of the four compression methods on one feature type are highly related with the non-zero rates of this feature type. It may be because that ReLU functions in neural networks provides a number of zero values in a feature sample, which produce statistical redundancy in the feature. The non-zero elements in the feature usually distribute in a broad numerical range, making the possibility very low to have several elements with the same value. As such, it is difficult for the compression methods to exploit the statistical redundancy in non-zero elements. Therefore, the performance of the lossless compression methods are largely affected by the non-zero rate of a feature sample.

In summary, regarding lossless compression of the deep learning features, the compression rates of the four benchmark data compression methods are around the non-zero rate of the deep feature. LZMA achieves the best compression rates on most of the feature types with the highest computational complexity. ZLIB performs comparably in term of compression rate with much shorter time. GZIP performs well but not the best in terms of both compression rate and computational cost. The performance of BZIP2 is not stable in terms of both compression rate and time cost, and highly depends on the feature type.

3.3.3 Discussions

By evaluating the four benchmark lossless data compression methods on deep learning features, we observe that the compression rate of a lossless compression method is largely limited by the non-zero rate of the feature to be compressed. The statistical redundancy of the deep learning feature mainly depends on the elements with zero value. It is difficult to identify and eliminate statistical redundancy from the non-zero elements of the deep learning features. As such, compressing the deep features in a lossless manner does not guarantee much room to improve. From the evaluation results, compression ratios of the lossless manner are around 1.5x˜3x, which may not be desirable for real applications. Accordingly, lossy compression on deep features is worth for further investigation. Moreover, it has been shown that the final output result of a neural network is not sensitive to slight changes of the activations in intermediate layers [95], which provides tolerability of the information loss for the lossy compression. In addition, the dynamic range of a deep learning feature is generally much smaller than the value range of the corresponding numeric data type, which provides much room for techniques such as quantization and sampling to compress the deep learning features. It is valuable to conduct further researches on compressing the deep learning features in a lossy way while maintaining the analysis performance, which will be discussed in Chapter 4.

3.4 Conclusion

In this chapter, a new paradigm of intermediate deep learning feature transmission was proposed for visual analysis in edge-cloud collaboration. Such approach helps reducing the computing load at the cloud end while maintaining the availability of various visual analysis applications, such that better trade-off can be achieved in terms of the computational load, communicational cost and generalization capability. To better enable the new paradigm, data coding for intermediate deep learning features is necessary. However, lossless compression is not practical for the intermediate deep learning feature coding, which was demonstrated by extensive evaluations. Lossy compression techniques should be further investigated, which will be discussed in the next chapter.

Chapter 4

Intermediate Deep Learning Feature Coding

As demostrated in Section 3.3, lossless compression methods can hardly provide high compression ratio, which is not practical for intermediate deep learning feature coding. In this chapter, I investigate in lossy compression methods. Section 4.1 proposes a video-codec-based coding framework and corresponding coding tools. Section 4.2 proposes evaluation metrics to reflect the information loss for lossy intermediate deep learning feature compression. Section 4.3 presents evaluation results of the video-codec-based coding framework on three essential computer vision tasks, which prove the generalizing capability of the coding framework on different neural network architectures and tasks. Section 4.4 conducts comprehensive comparative tests on various coding tools in the video-codec-based coding framework and present state-of-the-art performance of intermediate deep learning feature coding.

4.1 Video codec based Coding Framework

In CNNs, the intermediate deep learning features are mainly in the form of feature maps which are the combinations of stacked 2-D arrays with spatial correlations among the elements, as shown in Figure 4.1. Intuitively, a single channel 2-D feature map can be consider as a frame, while an intermediate deep feature can be considered as one video sequence. As such, existing video codecs can be applied

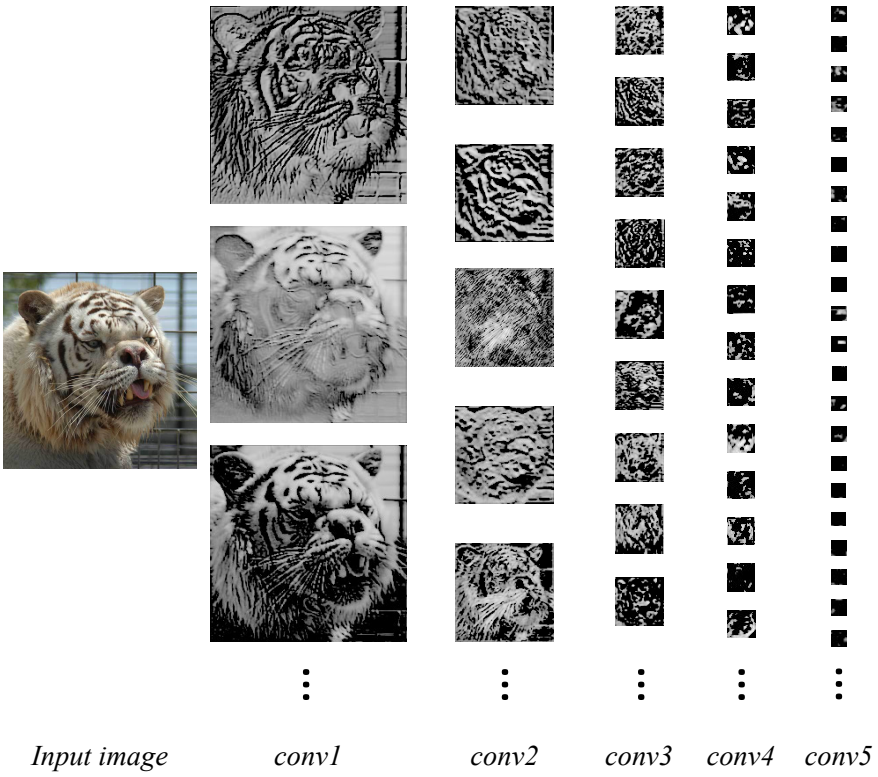


FIGURE 4.1: Visualized feature maps of VGGNet.

to compress deep features in a lossy manner. In this chapter, I propose a video-codec-based coding framework which integrates the traidtional video codec for intermediate deep feature compression.

By integrating video codecs into the proposed compression framework, matured video coding techniques can be borrowed to intermediate feature coding seamlessly. Furthermore, as video encoding/decoding modules (chips, IP cores, etc.) have already been widely deployed in many cloud based systems, it is economically and technically friendly to upgrade the visual devices and systems to support intermediate deep feature conveyance and analysis with our proposed framework.

The video-codec-based coding framework normally contains three modules for encoding, as shown in Figure 4.2. With an input intermediate deep feature, PreQuantization and Repack modules produce video-sequence-like data for the VideoEncoder module to compress. As the video codec is specified in advance, the coding performance will largely depend on how the feature data representation can fit the video codec. In view of this, PreQuantization and Repack modules should be carefully designed.

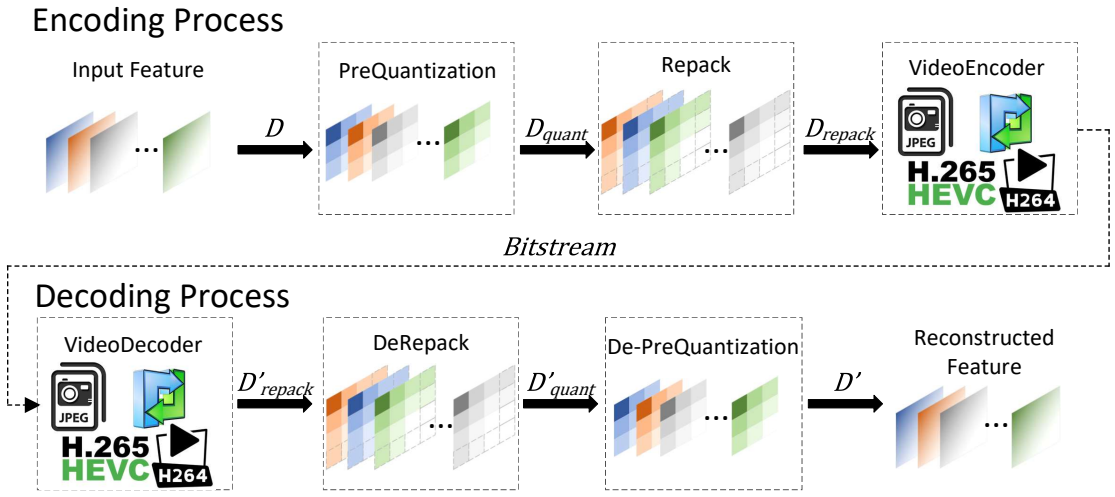


FIGURE 4.2: Flowchart of intermediate deep feature compression with the video-codec-based coding framework.

4.1.1 PreQuantization and De-PreQuantization

Deep learning models are usually trained with high bit-depth floating point numbers ensuring high precision in back propagation process. Accordingly, intermediate deep learning features are also commonly in high bit-depth floating point format. However, in the neural network inference, small errors of intermediate feature elements usually do not affect the final output [95], which makes it possible to reduce the feature volume by applying quantization methods. Besides, the supported bit depths of video codecs are usually lower than the features' bit depths. For instance, HEVC range extensions [96] are designed to support bit depths of up to 16 bits per sample, while the deep learning features are commonly of 32 bits or even 64 bits. A quantization process before the video codec can help match the numeric type of intermediate feature with the input requirement of video codec.

Let an input intermediate deep learning feature $D \in \mathbb{R}^{W \times H \times C}$ comprise a plurality of 2-D arrays $\in \mathbb{R}^{W \times H}$ (a.k.a., feature maps). We say this intermediate deep learning feature D is of C channels. $\mathbb{R}$ denotes the set of real numbers, $W \times H \times C$ is the shape of the intermediate deep learning feature. In PreQuantization module, the input feature D will be converted to integer format with lower (or equal) bit depth, while the shape of the feature remains the same. PreQuantization module outputs the quantized feature $D_{quant} \in {\mathbb{N}_0}^{W \times H \times C}$, where $\mathbb{N}_0$ denotes the set of nonnegative integers. Inversely, in De-PreQuantization module, the output of

De-Repack $D'_{quant} \in {\mathbb{N}_0}^{W \times H \times C}$ will be dequantized to the reconstructed feature $D' \in \mathbb{R}^{W \times H \times C}$.

Theoretically, any scalar quantizer can be applied to PreQuantization and De-PreQuantization modules. In the literature, uniform quantizer and logarithmic quantizer have been utilized for intermediate deep feature compression. In this chapter, I introduce a learning based adaptive quantizer as another option.

4.1.1.1 Uniform quantizer

As a straightforward approach, uniform quantizer evenly samples elements of intermediate deep features. [61, 62] employed such quantizer to transfer 32-bit floating point feature values to 8-bit integer. The quantization and dequantization processes are formulated as

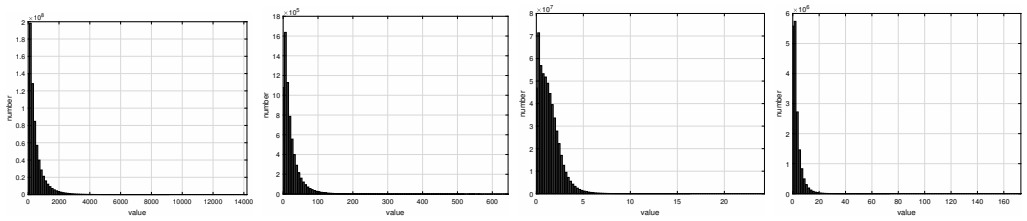
$$D_{quant} = rint(\frac{D - min(D)}{max(D) - min(D)} \cdot (2^n - 1)) \tag{4.1}$$

$$D' = \frac{max(D) - min(D)}{2^n - 1} \cdot D'_{quant} + min(D) \tag{4.2}$$

truncate where $rint(\cdot)$ rounds the floating point input to the nearest integer and n sets the bit depth of the quantized integers. From 4.2, we can know that, to perform the dequantization, the maximum value $max(D)$ and the minimum value $min(D)$ of the input feature and the bit depth value n should be available in the decoding process. As the ReLU function [97] is widely used as activation function in neural networks, which chops off negatives from features making $min(D)$ always be 0. Only $max(D)$ and n should be included in the bitstream in this case.

4.1.1.2 Logarithmic quantizer

In deep learning training, weight decay (a.k.a., L2 regularization) [98] is widely employed to prevent the weights from growing too large. It encourages the feature values towards zero along with the weight values. Rectified by the activation function, distribution of the features are usually of a right-skewed exponential behavior, shown as Figure 4.3. In view of this, as the inverse to exponentiation, my



(a) VGGNet *conv*2 (b) VGGNet *conv*5 (c) ResNet *conv*2 (d) ResNet *conv*5

FIGURE 4.3: Examples of feature distributions of VGGNet-16 and ResNet-50.

works [1, 59] applied logarithmic quantizer to intermediate deep feature compression. The quantization and dequantization processes are formulated as

$$D_{quant} = rint(\frac{log(D - min(D) + 1)}{log(max(D) - min(D) + 1)} \cdot (2^n - 1)) \tag{4.3}$$

$$D' = 2^{\frac{log(max(D)-min(D)+1)}{2^n-1} \cdot D'_{quant}} + min(D) - 1 \tag{4.4}$$

where $log(\cdot)$ is the logarithm function. From 4.4, to perform the dequantization, $max(D)$, $min(D)$ and n values should be available in the decoding process. When the feature is generated with ReLU function, $min(D)$ is not necessary to be included in the bitstream.

4.1.1.3 Learning based adaptive quantizer

Although statistics of the features is taken into consideration, logarithmic quantizer is able to perfectly quantize the features only when the sensitivity to quantization errors is linearly correlated with the density of the feature distribution. However, such assumption cannot always be true. To better minimize the information loss, we propose a learning based method to optimize the partition for intermediate deep feature quantization.

In scale quantization, a n-bit quantizer divides the entire data range into 2^n intervals. If each interval contributes equally to the feature information loss, the overall information loss due to quantization error will be minimized. To this end, we can iteratively perform bipartitioning as described in Algorithm 1. The algorithm penalizes (by bipartitioning) the interval where the feature is most sensitive in each

iteration. With a large enough number of iterations, the information loss caused at each interval will be evenly distributed.

To determine the information loss caused at a certain interval, we can quantize the training feature data T only in this interval. Concretely, saying it is the j-th interval, any element of training features T in the j-th interval is mapped to the middle value of the interval, while the other elements keep the same. The obtained partially quantized training feature data is denoted as T_j^{quant}. After that, T_j^{quant} is fed to the corresponding neural network to perform inference. The inference results of T_j^{quant} can be further compared with the inference results of pristine features T to evaluate the information loss. Here, we do not straightforwardly compare T_j^{quant} and T with signal-level metrics, such as SNR and PSNR, since deep learning features are with high level semantic information. The comparison of inference results can better reflect the information loss rather than the signal-level comparison of features. Furthermore, in practice, we found that the comparison of inference results can not be in terms of task performance (e.g., classification accuracy, mean average precision). Elaborated later in Section 4.4.2.1, task performance with T_j^{quant} and T may become identical when the iteration number goes large while the number of training samples is limited. It may hinder the algorithm from finding correct interval to perform bipartitioning. Alternatively, inspired by knowledge distillation [99], we calculate the L2 distance between the logits inferred by T_j^{quant} and T, to measure the information loss.

Having the partition determined by Algorithm 1 with a training feature set, the quantization for a new input feature can be performed by mapping the feature element values in different intervals to the indexes of corresponding intervals. The indexes of intervals $\in \{0, 1, 2, \cdots, 2^n - 1\}$ is considered as the codebook for the quantizer. The dequantization process can be performed by mapping the inputs of different index values to the middle values of corresponding intervals. To this end, the partition should be available in both encoding and decoding side. Since the feature extraction model and the feature type to be transmitted will not be frequently changed in real applications, the partition can be trained offline and stored in both encoding and decoding devices. As such, the partition is not necessary to be included in the bitstream.

Algorithm 1 Determining the partition for intermediate feature quantization

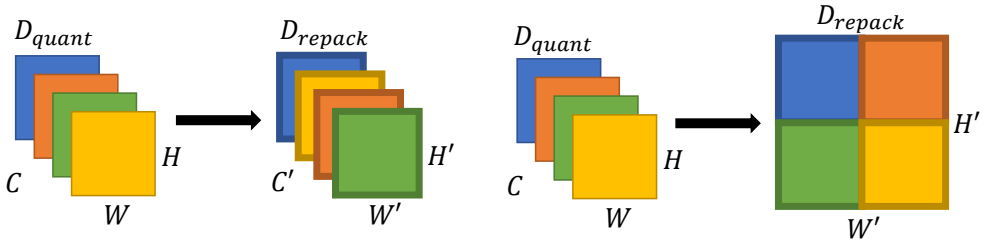
Require: The training feature data T; Target number of partition intervals, $N_{interval}$;
Ensure: The partition
1: Determine the entire data range $[min(T), max(T)]$;
2: The i-th partition interval is to be bipartitioned in next; Set $i = 1$, as there is only one interval now;
3: **repeat**
4: Bipartition the i-th interval; The number of intervals I increases by 1;
5: **for** Each interval j in the entire I intervals **do**
6: Quantize the training feature data T only in this interval, obtaining T_j^{quant};
7: Send T_j^{quant} back to the deep neural network to evaluate information loss L_j;
8: Find the maximum information loss from $L_{1,2,\cdots,I}$, assign the index to i;
9: **until** I equals to $N_{interval}$

4.1.2 Repack and DeRepack

After PreQuantization, the Repack module will further reorganize intermediate deep features to video sequence format as the inputs for the subsequent VideoEncoder module. The way of feature repacking has great influence on the efficiency of traditional video coding tools for feature compression.

Repack module reorganizes the quantized feature data $D_{quant} \in {\mathbb{N}_0}^{W \times H \times C}$ to $D_{repack} \in {\mathbb{N}_0}^{W' \times H' \times C'}$. In this process, the values and numerical type of elements of the feature data D_{quant} are changed, while the shape of feature data and the indices of the elements are altered. Particularly, the operations for repacking process may include mapping the elements of D_{quant} to D_{repack}, and inserting new elements to the repacked feature D_{repack}. So, the element number of D_{quant} (i.e., $W \times H \times C$) is not necessarily equal to the element number of D_{repack} (i.e., $W' \times H' \times C'$). Inversely, in De-Repack module, the output of Video Decoder $D'_{repack} \in {\mathbb{N}_0}^{W' \times H' \times C'}$ will be reconstructed to $D'_{quant} \in {\mathbb{N}_0}^{W \times H \times C}$.

As the intermediate deep learning feature $D \in \mathbb{R}^{W \times H \times C}$ comprises C 2-D arrays $in\mathbb{R}^{W \times H}$ (a.k.a., feature maps) where typical spatial correlations are remained, concatenation and tiling of the feature maps (channels) are two ways to repack the feature, as shown in Figure 4.4.



(a) Channel concatenation (b) Channel tiling

FIGURE 4.4: Two ways of repacking for intermediate deep features.

4.1.2.1 Naive channel concatenation

Naive channel concatenation is one straightforward way to repack the intermediate deep features, which was employed in my works [1, 59]. In this way, channels of a feature considered as gray-scale frames are simply concatenated together to form a video-sequence-like structure. As such, spatial correlations within a channel can be explored by intra coding tools of the subsequent video codecs, while the redundancies among channels can be eliminated by inter coding tools. To meet the requirements of input frame size of video encoders, the feature may need to be padded. For example of HEVC whose input frame size should be the integral multiple of 8, the concatenated feature of shape (W, H, C) should be padded to of (W', H', C), where $W' = \lceil W/8 \rceil \times 8$ and $H' = \lceil H/8 \rceil \times 8$. To introduce less high frequency patterns maximizing the compression efficiency, a replicate padding method, which adds elements of padding to the end of specified dimensions with the values of last array elements on the corresponding dimensions, can be applied.

On the contrary, in De-Repack module, the padded elements in $D'_{repack} \in {\mathbb{N}_0}^{W' \times H' \times C'}$ will be removed to produce $D'_{quant} \in {\mathbb{N}_0}^{W \times H \times C}$ as the input for De-PreQuantization module. Since the naive channel concatenation do not change the channel orders of features, only the numbers of elements padded in W and H dimensions should be available in the decoding side as the supplemental data.

4.1.2.2 Channel concatenation by distance

On top of naive channel concatenation, channels of intermediate deep feature can be sorted to maximize the similarity between adjacent channels. With channel concatenation methods, inter-channel redundancy is largely identified by inter coding

tools of subsequent video encoder. By maximizing the similarity between adjacent channels, it is expected to minimize the energy of residual frames for video encoder, thereby increasing the compression efficiency. In this thesis, I propose to sort the feature channels by L2 distance between channels as described in Algorithm 1. Concretely, we set the first channel of the feature as an anchor. Then, we iteratively find a channel from the rests with the smallest distance with the anchor channel, and set this newly found channel as the new anchor. After sorting the channels by distance, the padding for W and H dimension will be performed as described before. As such, in De-Repack process, the two numbers for padding and a list of indices that sort D_{quant} along the C axis are required to invert D'_{repack} to D'_{quant}.

Algorithm 2 Sort feature channels by distance

Require: D_{quant}: Quantized feature maps with shape of (W, H, C)
Ensure: D_{sorted}: Repacked feature maps

procedure SORT(D_{quant})
    $D_{sorted} = zeros([W, H, C])$
    $D_{sorted}[:, :, 0] = D_{quant}[:, :, 0]$
    $channel\_list = list(range(1, C))$
    for i in $range(C - 1)$ **do**
        $anchor\_map = D_{sorted}[:, :, i]$
        $rest\_map = D_{quant}[:, :, channel_list]$
        $optimal\_index = argmin(distance(anchor\_map, rest\_map))$
        $D_{sorted}[:, :, i + 1] = D_{quant}[:, :, channel\_list.pop(optimal\_index)]$

4.1.2.3 Channel tiling

Other than the concatenation methods, channel tiling can also help video codecs to explore the inter-channel redundancy. Shown as Figure 4.4(b), in channel tiling, channels of an intermediate deep feature are considered as patches of a large frame rather than single frames. In this way, inter-channel redundancy will be largely eliminated by intra coding tools of the video encoder, but not by inter coding tools like the channel concatenation methods. As it can embed the entire intermediate deep feature into a single 2-D array, [61, 62] employed such approach before compressing the feature with traditional image codecs (e.g., JPEG, JPEG2000). For decoding, the numbers of channels contained in height and width axes of the large 2D array and the numbers for padding should be made available to invert D'_{repack} to D'_{quant}.

4.1.3 VideoEncoder and VideoDecoder

The repacked feature data $D_{repack} \in \mathbb{N}_0^{W' \times H' \times C'}$ are then converted to corresponding format as the input for video/image codecs. For example of HEVC, we can convert the 3D array D_{repack} to YUV400 format with the height of H', the width of W', and the frame number of C'. Technically, any existing video/image codec can be employed in these two modules. In this chapter, I use the widely-used video codec HEVC [44] to conduct the following experiments.

4.2 Evaluation metrics

Similar to video coding, the evaluation of intermediate deep feature coding should take both compression performance and information loss into consideration. In the experiments, compression rate, defined in Equation 3.1, is employed to evaluate the compression performance. To evaluate information loss, the comparison of output results of the tasks performed after the feature transmission should be considered. It is because signal-level comparison (e.g., SNR, PSNR) for features is bootless, as deep features are with high level semantic information. It is also not proper to utilize task performance metrics (e.g., accuracy for image classification task, mAP for image retrieval task) to evaluate the performance of feature codecs. The reason is threefold. Firstly, the variation of a task performance metric may not reflect the fidelity level of the features before/after compression. Concretely, in terms of the direction of change, information loss of the features before/after compression can result in either positive or negative change to a task performance metric (e.g., classification accuracy varies from 0.80 to 0.75 or 0.85); in terms of the amount of change, same change amount of a task performance metric may refer to different information loss levels. The task performance metric may not be linearly proportional to inforation loss. Secondly, it is not well normalized to use task performance metrics to evaluate information loss. On the one hand, task performance metrics are with different value ranges (e.g., image classification accuracy is with the range of 0 to 1, while image captioning CIDEr [100] can reach more than 1); on the other hand, the task performance value on pristine features (i.e., the reference value) may vary depending on the test dataset, which makes it hard to compare information loss with task performance metrics. Thirdly, using the

task performance metric to evaluate the information loss, paired values (before/after compression) should be involved which is not elegant.

Therefore, we propose new metrics to evaluate information loss of features on different tasks. In this chapter, I choose three popular computer vision tasks in surveillance applications: image classification, image retrieval and image object detection, respectively. For image classification, we calculate the fidelity by comparing the pristine classification DNN outputs (i.e., the onehot classification results) with the outputs inferred from the reconstructed intermediate deep features, as below

$$F_{classification} = 1 - \frac{1}{2N} \sum_{i}^{N} \frac{Hamming(Y_p^i, Y_r^i)}{length(Y_p^i)} \tag{4.5}$$

where Y_p^i is the pristine onehot output of the tested neural network inferred from i-th test image sample, Y_r^i is the onehot output inferred from the corresponding reconstructed intermediate feature, $length(\cdot)$ returns the dimension of input, N denotes the total number of tested samples.

For retrieval task, given a query, a ranked sequence of documents will be returned by the system. In task performance metrics like mean average precision (mAP), the order of the ranked sequence is taken into consideration to calculate the average precision (AP). Here, we propose to calculate the fidelity by comparing the pristine output document sequence with the one inferred from the reconstructed intermediate deep features:

$$F_{retrieval} = \frac{1}{N} \sum_{i}^{N} bubble\_index(S_p^i, S_r^i) \tag{4.6}$$

where S_p^i and S_r^i are the ranked sequence of documents returned by the retrieval system with pristine features and reconstructed respectively for the i-th query, N denotes the total number of tested queries, $bubble\_index(\cdot, \cdot)$ is proposed to measure the similarity between two ranked sequences by counting the number of swap operations during sorting the reconstructed sequence into the pristine with bubble sort method. We name the similarity measurement after "bubble sort" method as "bubble index". The workflow of bubble index is described in Algorithm 3. It is worth noting that a naive implementation of bubble index is computational heavy ($\mathcal{O}(n^2)$) especially when the length of input sequence is large. The computing complexity can be significantly reduced (less than $\mathcal{O}(n \log(n))$) by

applying dichotomy in the for-loop. The code implementation can be found at [1].

Algorithm 3 Calculate similarity between two ranked sequences of documents

Require: S_p: ranked sequence of documents returned by the retrieval system with pristine features; S_r: the sequence inferred from the reconstructed intermediate deep features.

Ensure: Similarity score

```
procedure BUBBLE_INDEX(S_p,S_r)
    for element in S_r do                          ▷ generate list to be sort
        ordered_list.append(S_p.index(element))
    cnt_swap, cnt_total = 0
    for i in range(len(ordered_list)-1) do                          ▷ Bubble sort
        for j in range(len(ordered_list)-i-1) do
            cnt_total += 1
            if ordered_list[j] > ordered_list[j + 1] then
                cnt_swap += 1
                swap ordered_list[j] and ordered_list[j + 1]
    Similarity score = cnt_swap/cnt_total
```

As to object detection task, the detection model predicts both location and category of detected objects. We use Intersection over Union (IoU) to measure the fidelity on the location of the predictions, and a relative change rate to monitor the predicted classification confidences. Moreover, considering that predictions with different confidence level contribute differently to the task performance, we weighted each prediction with the confidence inferred by the pristine feature. Overall, the fidelity in object detection task is calculated as below:

$$F_{detection} = \frac{1}{N}\sum_{i}^{N}\frac{\sum_{j}^{M} UoI(B_p^{ij}, B_r^{ij}) \cdot max(0, 1 - \frac{C_p^{ij} - C_r^{ij}}{C_p^{ij}}) \cdot C_p^{ij}}{\sum_{j}^{M} C_p^{ij}} \tag{4.7}$$

where B is the predicted bounding box and C is the confidence value of the predicted category, N is the number of tested images and M is the number of predicted objects of i-th image. The implement code can be found at [1].

[1] http://chenzhuo.info/repos/acmmm19.html

4.3 Evaluations on a broad range of computer vision tasks

To provide evidences for the feasibility of the strategy of transmitting intermediate deep feature and the effectiveness of the proposed lossy compression framework, we present comprehensive experiments of intermediate deep feature compression on three widely-used visual surveillance tasks with two commonly-used backbone neural networks.

4.3.1 Experiment Setup

4.3.1.1 Evaluation tasks and datasets

As discussed in Section 3.1, one advantage of our proposed data transmission strategy relies on that intermediate deep features are with good generic ability which can be applied to a broad range of tasks. As such, we compress the intermediate features from unified backbone networks, then evaluate the information loss on three notable tasks in visual surveillance, which are image classification, image retrieval and image object detection, respectively.

Image classification: As a fundamental task in computer vision, image classification has been widely adopted in training and evaluating deep learning architectures. Many generic networks trained on image classification (e.g., VGGNet, ResNet) are employed as feature extractors or backbone networks in other computer vision tasks. Following [58], in this chapter, we evaluate information loss in feature compression on image classification task with a subset of the validation set of the ImageNet 2012 dataset [12]. To economize the compression time while maintaining the variety of test image categories, we randomly choose one image from each of the 1,000 classes.

Image retrieval: Content-based image retrieval is another key problem in computer vision. Among image retrieval problems, vehicle retrieval, as an unique application, has been drawing more and more attention due to the explosively growing requirement on surveillance security field. In this chapter, we adopt the "Small" test split of PKU VehicleID dataset [16] to perform feature compression evaluation on image retrieval task, which contains 800 query images and 5,693

reference images. In the experiments, only the features extracted from query images are to be compressed. Features extracted from reference images serve as reference during fidelity evaluations.

Image object detection: Image object detection task predicts both object location and category in the mean time, which contains both regression and classification. It is a fundamental task for surveillance analyses. We test our compression algorithm on image object detection with the test set of Pascal Visual Object Classes (VOC) 2007 dataset [101], which contains 4,952 images and 12,032 objects.

4.3.1.2 Deep learning architectures and features

In the experiments, we extract intermediate deep features with VGGNets and ResNets, which are the common choices for image feature extraction in many computer vision applications as their features can be regarded as generic.

VGGNet: Simonyan and Zisserman developped VGGNet [3] at the ILSVRC 2014. VGGNet-16 outstands from the six variants of VGGNet for its good balance among performance and computing complexity. VGG-16 is very appealing thanks its neat architecture consisting of 16 convolutional layers which only performs 3×3 convolution and 2×2 pooling all the way through. Currently it is the most preferred choice to extract features from images in computer vision community. In this chapter, we extract $conv1$ to $pool5$ features from VGGNet-16 architecture to compress and evaluate in image classification; $pool3$ and $pool4$ are not included in image retrieval task due to [102] perform feature downsampling in $conv1$ and $conv2$ by setting convolution stride instead of $pool3$ and $pool4$; $pool5$ is not included in detection task due to the region proposal network (RPN) of faster RCNN is built on top of $conv5$ feature of VGGNet. The implementation of image classification follows [103], image retrieval follows [102] and image object detection follows [104].

ResNet: At the ILSVRC 2015, He *et al.* introduced Residual Neural Network (ResNet) [4] which contains a novel technique called "skip connections". Thanks to this new structure, the network architectures are able to go into very deep with lower complexity than VGGNet. ResNets have three commonly used variants with 50, 101, 152 layers respectively. In this chapter, the $conv1$ to $conv5$ and $pool1$ features (ResNets do not have pooling layers for the last four blocks) are investigated in image classification and retrieval tasks, the $conv1$ to $conv4$ and $pool1$ (RPN of

faster RCNN is built on top of *conv*4 feature of ResNets, so *conv*5 is not include here) features are involved in image object detection task. To broadly investigate the features of three variants of ResNets while economizing the implementation difficulty, we apply ResNet-152 for image classification following [105], ResNet-50 for image retrieval following [102], and ResNet-101 for image object detection following [104].

4.3.1.3 Configurations for compression

The proposed video-codec-based coding framework in Section 4.1 is applied in the experiments. Specifically, for the PreQuantization and De-PreQuantization modules, the intermediate deep features are quantized/dequantized with the 8-bit logarithmic quantizer. For Repack module, we use naive channel concatenation method here. As to the VideoEncoder and VideoDecoder modules, the reference software (HM16.12) of HEVC Range extension (REext) is employed in the experiments. The compression is performed with four quantization parameter (QP) values, i.e., $[12, 22, 32, 42]$.

4.3.2 Results

The intermediate deep features are firstly extracted by neural networks, then passed to the feature encoder to generate compact bitstreams. The compression rate is subsequently calculated with the volume of original intermediate deep features and the corresponding bitstreams by Equation 3.1. As to the fidelity evaluation, the reconstructed features are passed to their birth-layer of the corresponding neural network to infer the network outputs, which will be compared with pristine outputs to evaluate the information loss of the lossy compression methods by the proposed metrics described in Section 4.2. The exhaustive results are listed in Table 4.1.

Comparing with lossless compression results reported in [58], we can see that lossy deep feature compression methods have more potential to compress the feature data into smaller volume than the lossless methods. In the extreme case, i.e. ResNet *conv*4 feature on retrieval dataset can reach over 500x compression ratio at $QP44$, while the lossless methods can only reach 2-5x. However, greater compression ratio will result in greater information loss. For each feature type, the fidelity value decreases as the QP value rises. Looking into the table, it can be also observed

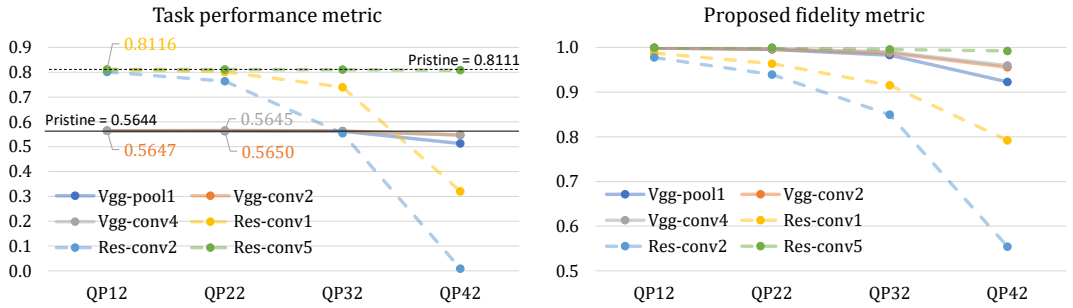
TABLE 4.1: Lossy feature compression results (Comp.Rate—Fidelity).

Feat.	Classification				Retrieval				Detection			
	QP12	QP22	QP32	QP42	QP12	QP22	QP32	QP42	QP12	QP22	QP32	QP42
VGGNet												
*conv*1	0.116 — 0.996	0.080 — 0.985	0.048 — 0.955	0.020 — 0.839	0.070 — 0.999	0.041 — 0.997	0.019 — 0.989	0.006 — 0.955	0.096 — 0.982	0.065 — 0.954	0.037 — 0.913	0.013 — 0.850
*pool*1	0.145 — 0.994	0.099 — 0.984	0.057 — 0.914	0.023 — 0.693	0.071 — 0.999	0.039 — 0.996	0.017 — 0.983	0.005 — 0.923	0.127 — 0.977	0.085 — 0.942	0.047 — 0.893	0.018 — 0.820
*conv*2	0.130 — 0.992	0.098 — 0.972	0.066 — 0.952	0.035 — 0.790	0.089 — 0.999	0.069 — 0.996	0.050 — 0.987	0.027 — 0.955	0.121 — 0.980	0.090 — 0.950	0.059 — 0.907	0.030 — 0.858
*pool*2	0.185 — 0.995	0.138 — 0.982	0.090 — 0.947	0.047 — 0.745	0.157 — 0.999	0.119 — 0.997	0.079 — 0.990	0.037 — 0.945	0.172 — 0.981	0.128 — 0.953	0.082 — 0.900	0.040 — 0.815
*conv*3	0.102 — 0.995	0.080 — 0.986	0.057 — 0.960	0.034 — 0.840	0.112 — 0.999	0.089 — 0.998	0.070 — 0.993	0.048 — 0.976	0.040 — 0.981	0.033 — 0.954	0.025 — 0.912	0.015 — 0.845
*pool*3	0.179 — 0.989	0.140 — 0.981	0.102 — 0.955	0.063 — 0.819	–	–	–	–	0.080 — 0.982	0.066 — 0.955	0.050 — 0.908	0.032 — 0.826
*conv*4	0.065 — 0.992	0.053 — 0.984	0.041 — 0.967	0.028 — 0.865	0.088 — 0.999	0.070 — 0.997	0.057 — 0.990	0.043 — 0.959	0.022 — 0.983	0.019 — 0.960	0.014 — 0.920	0.008 — 0.877
*pool*4	0.160 — 0.992	0.127 — 0.974	0.097 — 0.969	0.065 — 0.864	–	–	–	–	0.049 — 0.984	0.041 — 0.960	0.031 — 0.914	0.019 — 0.847
*conv*5	0.059 — 0.997	0.046 — 0.989	0.037 — 0.969	0.023 — 0.920	0.049 — 0.998	0.041 — 0.995	0.036 — 0.984	0.030 — 0.952	0.031 — 0.983	0.023 — 0.956	0.014 — 0.902	0.005 — 0.741
*pool*5	0.162 — 0.995	0.129 — 0.986	0.106 — 0.967	0.075 — 0.908	0.243 — 0.998	0.200 — 0.996	0.173 — 0.986	0.146 — 0.960	–	–	–	–
ResNet												
*conv*1	0.070 — 0.982	0.041 — 0.935	0.019 — 0.818	0.005 — 0.356	0.044 — 0.987	0.018 — 0.964	0.006 — 0.915	0.001 — 0.792	0.056 — 0.948	0.029 — 0.915	0.011 — 0.872	0.002 — 0.713
*pool*1	0.074 — 0.979	0.043 — 0.937	0.017 — 0.732	0.004 — 0.087	0.055 — 0.989	0.025 — 0.963	0.009 — 0.900	0.002 — 0.720	0.061 — 0.939	0.030 — 0.889	0.010 — 0.779	0.002 — 0.470
*conv*2	0.139 — 0.995	0.095 — 0.986	0.052 — 0.957	0.014 — 0.765	0.061 — 0.978	0.027 — 0.939	0.005 — 0.849	0.001 — 0.554	0.154 — 0.981	0.107 — 0.949	0.061 — 0.883	0.016 — 0.660
*conv*3	0.184 — 0.996	0.134 — 0.989	0.083 — 0.973	0.028 — 0.854	0.071 — 0.996	0.041 — 0.986	0.012 — 0.930	0.003 — 0.551	0.162 — 0.989	0.118 — 0.971	0.071 — 0.922	0.021 — 0.684
*conv*4	0.231 — 0.997	0.170 — 0.992	0.101 — 0.977	0.034 — 0.932	0.050 — 0.998	0.035 — 0.997	0.022 — 0.978	0.012 — 0.799	0.082 — 0.984	0.056 — 0.964	0.029 — 0.926	0.008 — 0.833
*conv*5	0.168 — 1.000	0.131 — 0.998	0.100 — 0.988	0.063 — 0.961	0.057 — 0.999	0.040 — 0.999	0.023 — 0.996	0.014 — 0.992	–	–	–	–

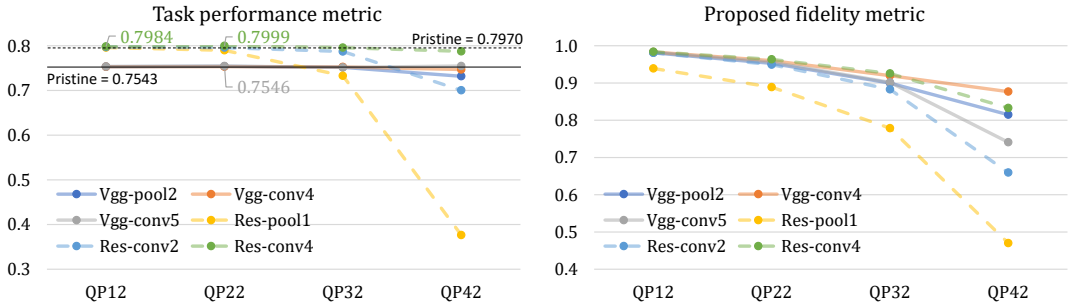
that $QP22$ can generally provide high fidelity and fair compression ratio in the meantime. Moreover, upper layer features, like *conv*4 to *pool*5, are generally more robust to heavy compression. It is a great character for practical implementation of intermediate feature transmission, since the high layer features can largely save the computing load while provide great usability at the cloud end, as mentioned in Table 3.2.

4.3.2.1 Metric comparison

To check the effectiveness of our proposed fidelity metrics, we compare them with the task performance metrics. As shown in Figure 4.5, such two types of metrics are compared on three selected feature types for each network architecture on retrieval and detection tasks. For the task performance tasks, to evaluate the information loss, the performance values on reconstructed features should be compared with the pristine performance values. As the performance values may vary subject to difference models, datasets and tasks, it is not a straightforward way to measure the information loss. Moreover, although task performance metrics and our proposed fidelity metrics generally share similar trend as QP changes, some task performance values are counterintuitive. As marked in Figure 4.5, reconstructed ResNet *conv*1 at $QP12$ and VGGNet *conv*2 at $QP12\&22$ on retrieval task, ResNet *conv*4 at $QP22\&32$ and VGGNet *conv*5 at $QP32$ can even produce higher performance values than the pristine input. It is contrary to the fact that lossy compression will introduce information loss and higher QP will make the information loss greater. So we believe that our proposed metrics can better evaluate the performance of coding methods which should mainly consider how the reconstructed feature is fidelity to the original feature signal.



(a) Metric comparison on Retrieval



(b) Metric comparison on Detection

FIGURE 4.5: Comparisons between task performance metrics and the proposed fidelity metrics.

4.4 Comparisons of coding tools

In this section, we present comprehensive intermediate deep learning feature compression results to compare the performance of different coding tools for the modules of the video-codec-based coding framework.

4.4.1 Experiment Setup

Evaluation of lossy intermediate deep feature compression on different computer vision tasks shows strong positive correlation of fidelity maintenance performance on different computer vision tasks, as demonstrated in Section 4.3. So, in this section, to economize the compression time, we apply feature extraction and fidelity evaluation only on image classification task. Concretely, we extract intermediate deep features from convolutional layers and pooling layers of VGGNet and ResNet, following the settings of Section 4.3. Officially released models of VGGNets [3] and ResNets [4] trained on ImageNet 2012 dataset [12] are employed. One thousand

feature samples extracted from a subset of ILSVRC 2012 validation dataset will be compressed for each type of features of the aforementioned two neural networks. In the following experiments, feature extraction and neural network inference were performed with Caffe on a NVIDIA GeForce 1080 GPU. The compression was performed with Intel Xeon CPU E5-2650 v2 @ 2.60GHz single thread.

4.4.2 Comparison on PreQuantization methods

As introduced in Section 4.1.1, there are three quantization methods to be compared in PreQuantization and De-PreQuantization module. They are uniform quantizer, logarithmic quantizer and learning-based adaptive quantizer. In particular, uniform quantizer was applied to compress intermediate deep feature in [61, 62]; logarithmic quantizer was employed in my works [1, 59]; learning-based adaptive quantizer is newly proposed in this chapter.

To evaluate the task performance drop due to quantization error, we first apply quantization and de-quantization to the intermediate deep features, then perform neural network inference with the reconstructed features to calculate fidelity with Equation. 4.5. Table 4.2 lists the fidelity results of the three quantizers on features of VGGNet. From the table, we can easily find that performance of the learning-based adaptive quantizer is superior to the other two quantizers. At almost each bit depth point and on each feature type, the learning-based adaptive quantizer introduces less information loss than uniform quantizer and logarithmic quantizer. The superiority becomes even greater at lower bit depth. Logarithmic quantizer can achieve comparable fidelity with the learning-based adaptive quantizer in some cases, such as *conv*3 - *pool*4 at 8-bit, *conv*1 - *pool*1 and *conv*5 - *pool*5 at 6-bit, and *conv*5 - *pool*5 at 4-bit. As the partition and codebook do not depend on any training, logarithmic quantizer is a good choice when training samples or the cloud-side neural network is not available. Uniform quantizer generally presents inferior performance in this experiment. Table 4.2 shows the fidelity comparison results on features of VGGNet. The adaptive quantizer and logarithmic quantizer can also achieve good performance on features of ResNet. Experimental results on ResNet are presented in Appendix A as Table A.1.

More precisely, we look into the information loss at different numbers of quantization levels. Figure 4.6 shows the comparison of the three quantizers. In the figures, the

number of quantization levels varies from 3 to 256 covering 2-bit (4 levels) to 8-bit (256 levels). Figure 4.6(a) compares the quantizers with fidelity metric (i.e., task performance). We can see that performance of learning-based adaptive quantizer is notably lower than the other two at a small number of quantization levels (i.e., 3-5 levels). However, the learning-based adaptive quantizer improves rapidly with the increasing of quantization levels. It surpasses the other two quantizers from 3-bit (8 levels) onward. After around 50 quantization levels, the three curves get converged. The difference among their fidelity performance is less than 1%. Changes of information loss are hard to be identified in this stage. It is one reason why we do not use the task performance metrics to determine the information loss guiding the training of the learning-based adaptive quantizer. In Figure 4.6(b), we use the sum of square error (SSE) between the pristine logits and the logits inferred with reconstructed features to measure the information loss. It is much easier to tell the differences among the three quantizers. From this figure, we can observe that, at small numbers of quantization levels, SSE of the learning-based adaptive quantizer is greater than the other two. In this case, such quantizer introduces more information loss to the features, which is in line with the observation in Figure 4.6(a) that the fidelity of adaptive quantizer is lower than the uniform and logarithmic quantizers. It is because the training process of the adaptive quantizer is based on iteratively bipartitioning the input range. At small numbers of quantization levels, the training loop can only perform few times, thereby leading the poor performance. However, with the increasing number of quantization levels, adaptive quantizer surpasses the other two quantizers notably.

Previous experiments shown in Table 4.2 and Figure 4.6 compare the quantizers when they work solely on the features. Next, we investigate the performance of the three quantizers in the video-codec-based coding framework. In this experiment, the quantizers work as the coding tools of PreQuantization and De-PreQuantization modules. To control variates, the method for Repack and De-Repack modules will be fixed to naive channel concatenation, configurations of HEVC Range extension (REext) HM-16.12 for VideoEncoder and VideoDecoder module follow the common test condition [106]. The compression is performed at five quantization parameter (QP) values, i.e., $[0, 12, 22, 32, 42]$. Figure 4.7 provides two examples of visualized results on *conv*2 and *conv*5 features of VGGNet respectively. More visualized comparison results can be referred in Figure A.1 in Appendix A. The numerical results of the visualized figures can also be found in Tables A.2, A.3, A.4 in

Appendix A. From the figures, we can see that the performance of logarithmic quantizer is notably inferior to the ones of the other two quantizers. Uniform quantizer and adaptive quantizer generally present comparable performance. On high-layer features (e.g., *conv*5 features as shown in Figure 4.7(b)) and at high QPs, the coding results with uniform quantizer even have slightly better performance than the ones with adaptive quantizer. Such observation is quite different from what we learned in the previous two experiments that, when only applying quantization to compress the features, the adaptive quantizer is superior to the logarithmic quantizer while the logarithmic quantizer is superior to the uniform quantizer. We believe it may be because of the uneven partitioning of logarithmic and adaptive quantizer. Concretely, in the video-codec-based coding framework, information loss is largely introduced by PreQuantization and VideoEncoder module, where VideoEncoder is performed after PreQuantization. As such, new distortion will be introduced to the feature elements in the encoding process. Since the video codecs are designed for visual signals, the distortion level is evenly distributed in the range of pixels (quantized feature elements). However, the partitioning of logarithmic and adaptive quantizer is not even. A large value of the quantized data corresponds to a big range of quantization interval. Therefore, a small value change introduced by the video encoder may result in a great error in the reconstructed feature. On the contrary, uniform quantizer partitions the input range evenly, which make the feature quantized by it suffers less from the subsequent video encoder. As a result, uniform quantization can fit better with the video codecs than the logarithmic quantization, and be comparable to adaptive quantizer, even it is inferior to the others when applying solely to the features. However, looking into the numerical results in Tables A.2, A.3, A.4 in Appendix A, we can also find that the upper limits of fidelity (obtained when QP is 0) with uniform quantizer is generally lower than the ones with logarithmic or adaptive quantizer. For example, the video-codec-based coding framework can achieves up to 0.999 fidelity on *conv*1 features of VGGNet with adaptive quantizer, and can achieve up to 0.997 fidelity with logarithmic method, but can only achieve up to 0.995 fidelity with uniform quantization. It is because the upper limits of fidelity is largely hinged on the quantization error occurred by the three different quantizers.

Moreover, we also evaluated the coding performance of the video-codec-based coding framework with different bit depth settings for PreQuantization module. Figure 4.8 visualized the comparison results with 6-bit and 8-bit quantizers on

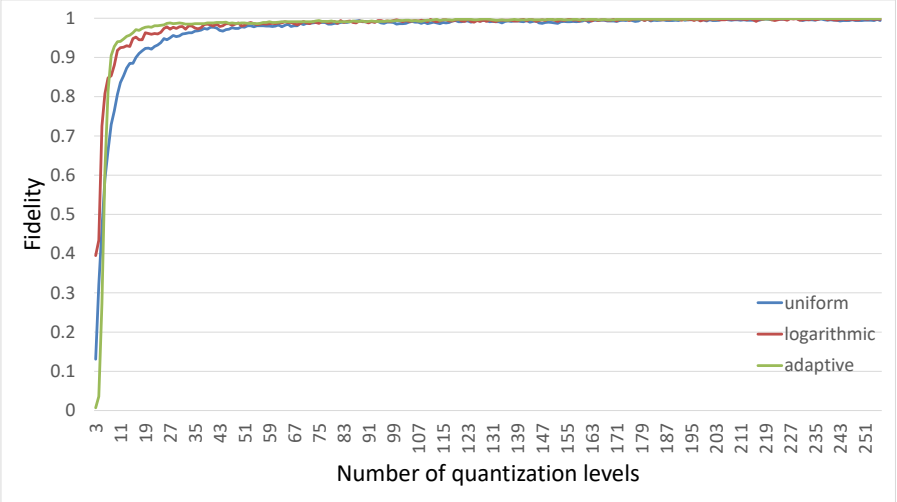
*conv*2 and *conv*5 features of VGGNet. (The corresponding numerical results can be referred in Table A.3, A.4, A.5, A.6, and more visualized figures can be found in Figure A.2 in Appendix A.) From the figures, we can see that different bit depth settings for the quantizer do not have much impact on the overall performance of the video-codec-based coding framework, but the type of quantization method does. The bit saving coming from the quantization bit depth reduction can also be achieved by setting larger QP factor in the video codec without affecting the fidelity performance. However, quantizer of higher bit depth can provide higher upper limits of fidelity for the entire coding framework. Furthermore, quantizer of high bit depth performs better in the high QP condition (i.e., QP 32 and 42) as shown in the figures. We feel it is better to use high bit depth settings for the PreQuantization module.

TABLE 4.2: Fidelity comparison of three quantizers on different feature types of VGGNet.

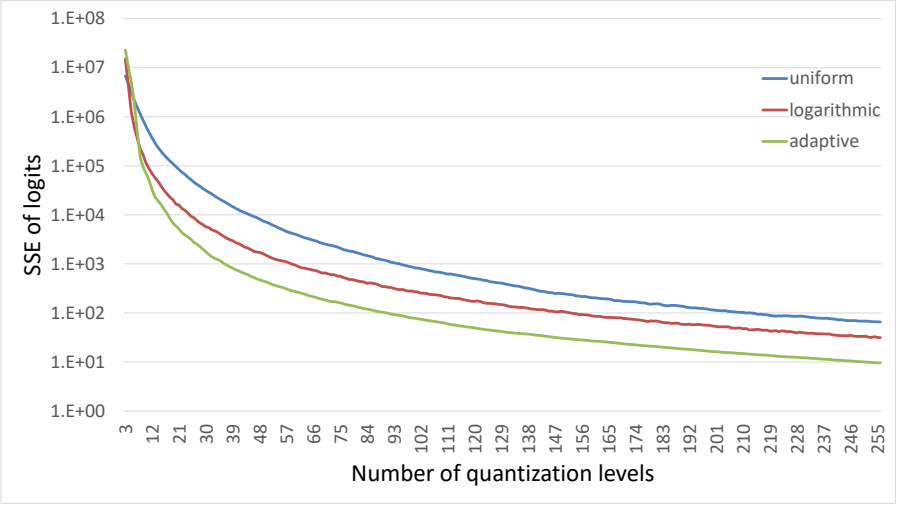
	8-bit			6-bit			4-bit		
	uniform	log	adaptive	uniform	log	adaptive	uniform	log	adaptive
conv1	0.994	0.997	**0.999**	0.982	**0.991**	**0.991**	0.900	0.952	**0.971**
pool1	0.994	0.997	**0.999**	0.982	**0.991**	**0.991**	0.900	0.952	**0.971**
conv2	0.994	0.997	**0.999**	0.988	0.989	**0.994**	0.921	0.967	**0.975**
pool2	0.994	0.997	**0.999**	0.988	0.989	**0.994**	0.921	0.967	**0.975**
conv3	0.996	**0.997**	**0.997**	0.986	0.986	**0.994**	0.970	0.968	**0.983**
pool3	0.996	**0.997**	**0.997**	0.986	0.986	**0.994**	0.970	0.968	**0.983**
conv4	0.997	**0.998**	**0.998**	0.992	0.991	**0.994**	0.975	0.972	**0.984**
pool4	0.997	**0.998**	**0.998**	0.992	0.991	**0.994**	0.975	0.972	**0.984**
conv5	0.998	0.998	**1.000**	0.994	**0.997**	**0.997**	**0.980**	**0.980**	**0.980**
pool5	0.998	0.998	**1.000**	0.994	**0.997**	**0.997**	**0.980**	**0.980**	**0.980**

4.4.2.1 Discussions on the learning-based adaptive quantizer

In the training of the learning-based adaptive quantizer, we use L2 distance between logits inferred by quantized feature T_j^{quant} and pristine feature T to determine the information loss, rather than using the fidelity or task performance metric, as described in Section 4.1.1.3. The reason is that, with the increase of training iterations (i.e., increasing of quantization levels), the fidelity will become saturated where the changes of information loss will be hard to be identified. It may lead poor performance of the adaptive quantizer. To prove this, we conduct experiment to compare the quantizers trained with fidelity and distance between logits respectively. Figure 4.9 shows the comparison results on *conv*1 features of VGGNet. It can be easily found that there are several steps in the curve of the quantizer trained with



(a) Fidelity changes with the number of quantization levels.

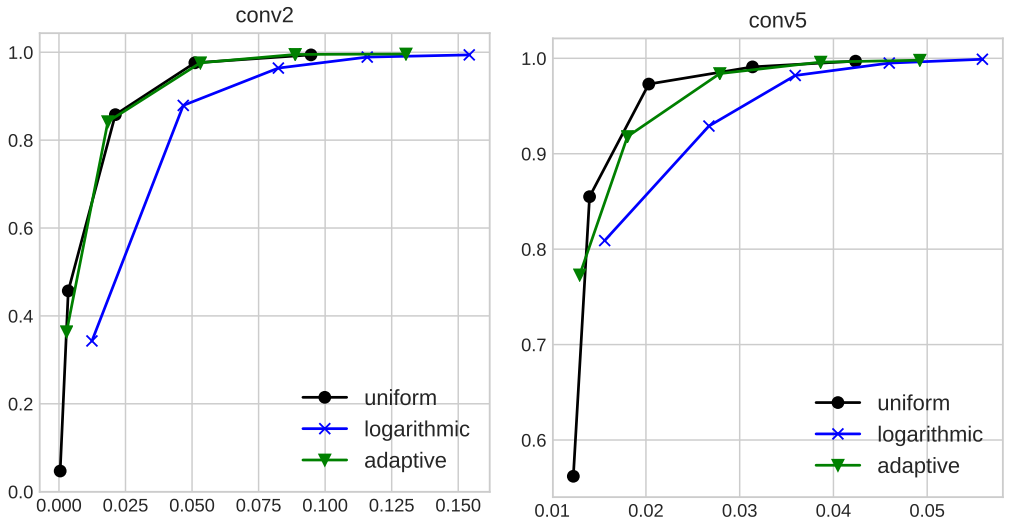


(b) SSE of logists changes with the number of quantization levels.

FIGURE 4.6: Comparison of the quantizers on *conv*1 fetaures of VGGNet. Information loss changes with the number of quantization levels.

fidelity where the information loss (i.e., SSE of logits) descends slowly. At around 110 quantization levels, the information loss stops decreasing. On the contrary, performance of the quantizer trained with distance of logits keeps improving along with the training continuing. As a result, distance between logits is more suitable for the training of the learning-based adaptive quantizer.

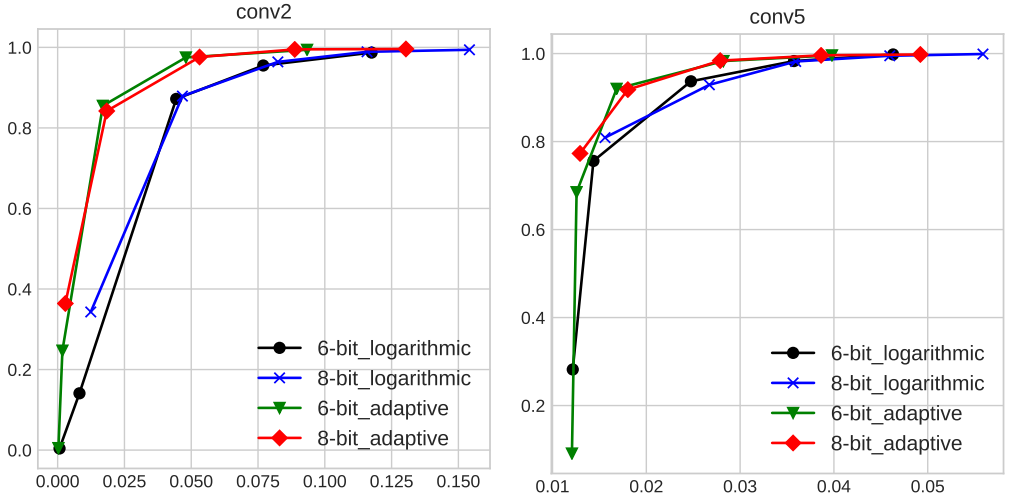
As introduced in Section 4.1.1.3, training process of the adaptive quantizer aims to optimize the partition which assigns smaller quantization interval (a.k.a., quantization step) to where is more sensitive to the quantization error, and assigns bigger quantization interval to where is less sensitive. As such, length of a quantization



(a) Performance comparison on *conv*2 features of VGGNet.

(b) Performance comparison on *conv*5 features of VGGNet.

FIGURE 4.7: Coding performance comparison of the video-codec-based coding framework with three different PreQuantization tools. The horizontal axis represents compression rate, while the vertical axis represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.



(a) Performance comparison on *conv*2 features of VGGNet.

(b) Performance comparison on *conv*5 features of VGGNet.

FIGURE 4.8: Coding performance comparison of the video-codec-based coding framework with different bit depth settings for PreQuantization module. The horizontal axis represents compression rate, while the vertical axis represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

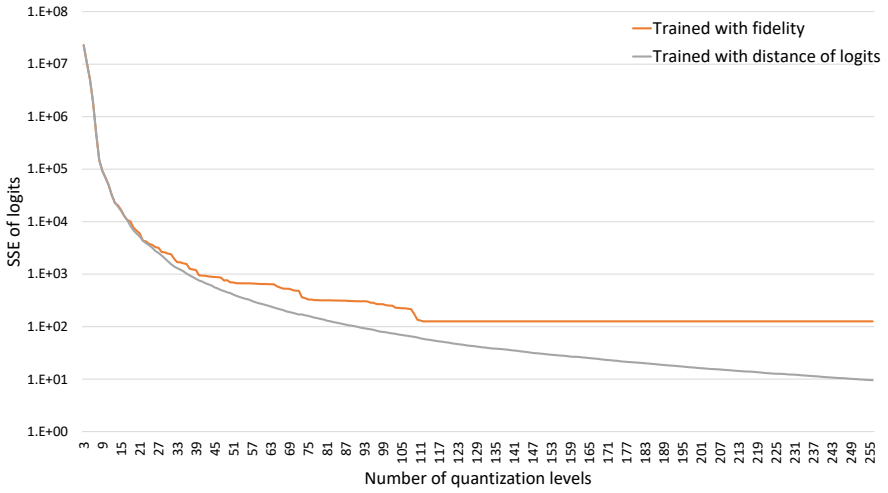
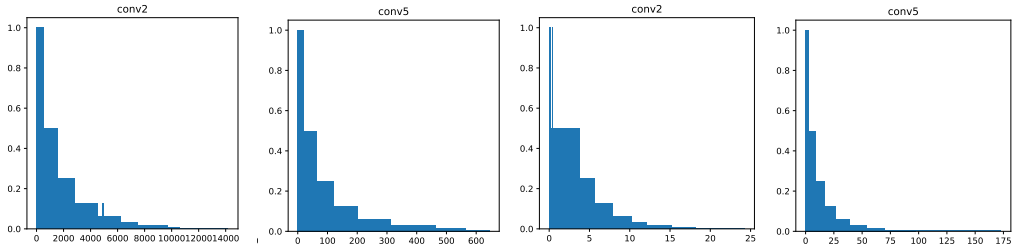


FIGURE 4.9: Comparison of two learning-based adaptive quantizers trained with fidelity and distance of logits respectively.

interval is the reciprocal of the sensitivity in that range. Figure 4.10 depicts the sensitivity distribution of the intermediate deep features learned by the adaptive quantizer at 8-bit. (More results can be found in Figure A.3 in Appendix A.) In the figures, we use the reciprocal of quantization interval length to present the sensitivity. The greater value means the feature is more sensitive to the quantization error at the corresponding interval. From the figures, we can observe that the distribution of sensitivity coarsely shows a exponential behavior. It may be a reason why the logarithmic quantizer has a good performance on the intermediate deep features. However, it can be also found that such exponential behavior is far from strict. For instance, there are several outliers (pulses) in the figure on *conv*2 of VGGNet and *conv*2 of ResNet. Comparing with the data distributions of the corresponding features in Figure 4.3, the sensitivity distributions are less zero-concentrated. It is in line with our assumption made in Section 4.1.1.3 that the sensitivity is not linearly correlated with the density of the feature distribution, which makes the performance of logarithmic quantization inferior to the adaptive quantization.

4.4.3 Comparison on Repack methods

Introduced in Section 4.1.2, there are methods to be compared in Repack and DeRepack module. They are naive channel concatenation, channel concatenation by distance and channel tiling respectively. In particular, naive channel concatenation

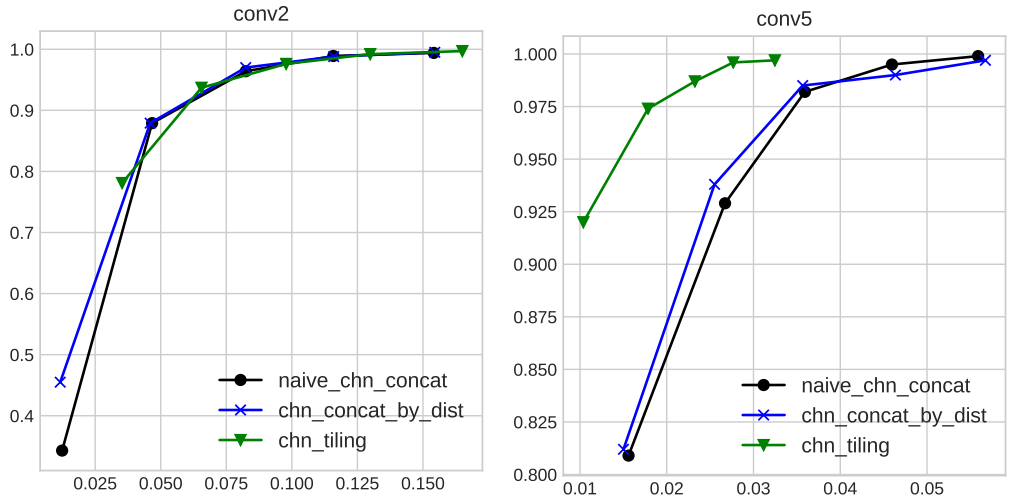


(a) VGGNet *conv*2 (b) VGGNet *conv*5 (c) ResNet *conv*2 (d) ResNet *conv*5

FIGURE 4.10: Examples of sensitivity distributions derived by the learning-based adaptive quantizer on features of VGGNet and ResNet. The horizontal axis represents the data range of the features, while the vertical axises is the reciprocal of quantization step size reflecting the sensitivity.

method was employed in my works [1, 59], while [1, 59] only explored the intra-channel redundancy. Channel concatenation by distance method is improved from the naive channel concatenation method in this chapter. Channel tiling method was applied in [61, 62], whereas [61, 62] only reported results on three features of backbone networks on image detection task. To investigate the performance of the three Repack methods in the video-codec-based coding framework, the method for PreQuantization and De-PreQuantization modules is fixed as 8-bit logarithmic quantization; configurations of HEVC Range extension (REext) HM-16.12 for VideoEncoder and VideoDecoder module follow the common test condition [106]; and the compression is performed at five QP values, i.e., $\{0, 12, 22, 32, 42\}$.

Coding performance comparison of the video-codec-based coding framework with three different Repack methods is visualized in Figure 4.11. (More visualized figures on other feature types can be found in Appendix A as Figure A.4, and the corresponding numerical results can be referred in Tables A.3, A.7, A.8.) From the results, it can be observed that the curves of the three Repack methods approximately coincide with each other on the low layer features from *conv*1 to *conv*3. However, when QP is of small values (e.g., QP0 and QP12), channel tiling method will result in higher compression rate than the other two with the same fidelity level. At large QP values (e.g., QP42), channel concatenation by distance method can achieve higher fidelity than the naive channel concatenation method with the same compression rate level. On the contrary, on the high layer features from *pool*3 to *pool*5, channel tiling method shows the superior performance to the concatenation methods. For instance, on the *conv*5 features, compression rate with channel tiling method can reach 0.032 while the concatenation methods can only achieve 0.056



(a) Performance comparison on *conv*2 features of VGGNet.

(b) Performance comparison on *conv*5 features of VGGNet.

FIGURE 4.11: Coding performance comparison of the video-codec-based coding framework with three different Repack tools. The horizontal axis represents compression rate, while the vertical axis represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

and 0.057 when the task performance drop are less than 0.3% (i.e., in case of QP0), where the channel tiling method helps save more than 42% data volume comparing with the other two. It may be because there are rich similar patterns in spatial domain among the channels in the high layer feature. Such patterns share similar spatial structures with various brightness and contrast. As such, intra coding tools in the video codec is more efficient to eliminate redundancies among these patterns, while inter coding tools can not work well as there is no explicit motion among channels. In view of this, channel tiling method enabling inter-channel redundancy elimination via intra prediction can result in better performance on high layer features.

4.4.4 Comparison to the state-of-the-art

Evaluated in the previous experiments, we know that channel tiling method can generally provide good performance for Repack and De-Repack module, especially on high layer features. Uniform quantization and learning-based adaptive quantization can work well in PreQuantization and De-PreQuantization module, while the adaptive quantization can reach higher upper limits of fidelity. So we believe

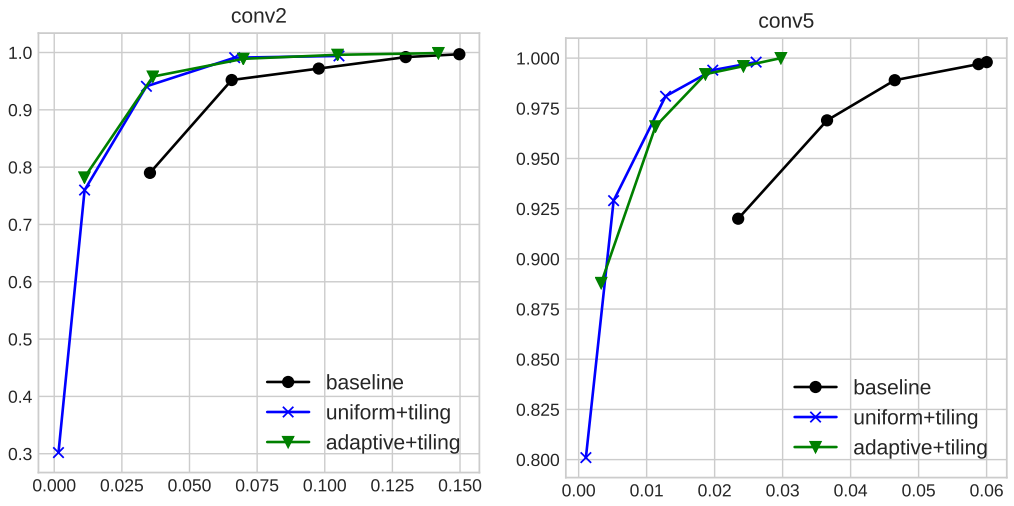
the combinations of "adaptive quantization and channel tiling" and "uniform quantization and channel tiling" may be the best settings for the video-codec-based coding framework. In this experiment, the quantizers are set to be 8-bit, and the configurations for VideoEncoder and VideoDecoder modules keep the same as the previous. We take the results reported in my previous work [1] as the baseline for comparison, which is also a benchmark in AVS [67] and MEPG VCM [67] standard. The baseline results are based on logarithmic quantization and channel concatenation methods. Figure 4.12 shows the visualized comparison results on *conv*2 and *conv*5 features of VGGNet. More visualized results on other feature types can be found in Appendix A as Figure A.5, and the corresponding numerical results can be referred in Tables A.9, A.10, A.11.

From the comparison results, we can easily find that the two combinations with the new coding tools are significantly better then the baseline, where the superiority becomes even greater on the high layer features. For examples, on *conv*2 features, the baseline can achieve 0.130 compression rate when the task performance (i.e., fidelity) drop is less than 1%, while the "uniform + tiling" setting can reach 0.067 which saves 48.5% more data volume than the baseline. Moreover, on *conv*5 features, the baseline can compress the features to 0.046 times of original volume when maintains the fidelity at 0.989, while the "adaptive + tiling" setting can reach 0.019 compression rate with higher fidelity which saves 58.7% more volume than the baseline. In average over the ten types of VGGNet features, "adaptive + tiling" and "uniform + tiling" settings can save over 40% data volume than the baseline when task performance drop is less than 1% (i.e., fidelity value is greater than 0.99).

As to the "adaptive + tiling" and "uniform + tiling" settings, we can see that the two curves are coincided with each other, which demonstrates the comparable performance. However, "adaptive + tiling" setting always has higher upper limits of fidelity. So, we feel "adaptive + tiling" setting is more suitable when task performance of the visual analytic system is crucial.

4.4.5 Comparison to visual transmission strategy

To further compare the proposed intermediate deep feature coding scheme with the conventional visual signal coding scheme, we conduct experiment to evaluate how the task performance change with the lossy compressed visual signal at different



(a) Performance comparison on *conv*2 features of VGGNet.

(b) Performance comparison on *conv*5 features of VGGNet.

FIGURE 4.12: Coding performance comparison to the state-of-the-art. The baseline results are reported in my previous work [1]. The horizontal axis represents compression rate, while the vertical axis represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

quality factors. With the ImageNet dataset on image classification task, we compress both full images and cropped images with JPEG codec at QF (quality factor) from 5 to 100 with a stride of 5. The full images are the ones of original resolution provided by ImageNet dataset, whose average resolution is 482/*times*418 pixels. The cropped images are of the input size of VGGNet, which is 224/*times*224 pixels. After compression, the restored images are fed into VGGNet to evaluate the fidelity. Evaluation results are as shown in Figure 4.13, noting that the features are compressed with the ransmission strategy. Images are compressed with JPEC codec, while features are compressed with "adaptive + tiling" setting mentioned in Section 4.4.4.

From the figure, we can see that even with high quality settings, images, especially full resolution images, can not maintain the fidelity well. It may be because image codec is designed for human vision but not optimized for machine analysis. Human may not notice small JPEG artifact and the noise introduced by image resizing, but deep learning model is sensitive to them. Low level features, such as *pool*3 and *conv*4 features, may not be efficient for compression and transmission comparing with images, especially in high QP cases (i.e., QP42). However, high level feature transmission (i.e., *pool*4, *conv*5, *pool*5) can fully surpasses image transmission in

terms of encoded volume and fidelity. It further proves the efficiency and feasibility of our proposed paradigm of intermediate deep feature compression.

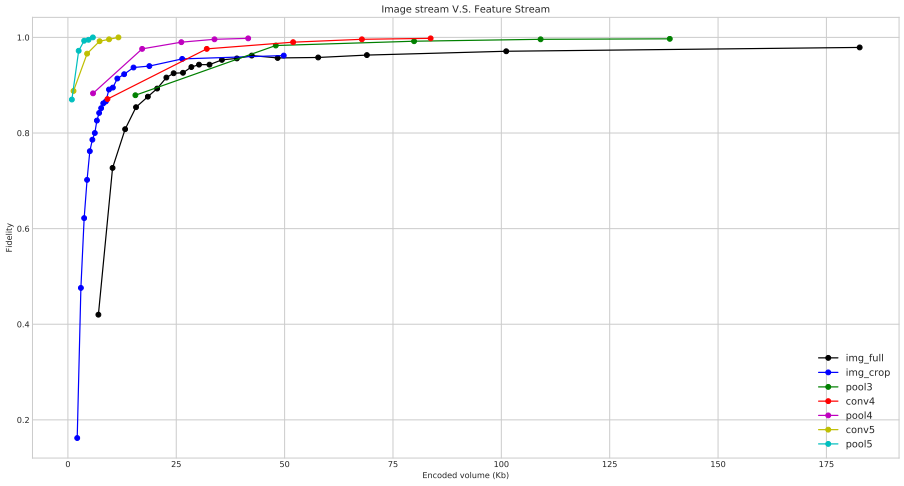


FIGURE 4.13: Coding performance comparison to the conventional visual transmission strategy. Images are compressed with JPEC codec, while features are compressed with "adaptive + tiling" setting.

4.4.6 Discussions

With the new coding tools and optimized settings, the video-codec-based coding framework can save over 40% more data volume than the previous works. It makes the intermediate deep feature compression more practical. Based on the evaluation of the "adaptive + tiling" setting, we calculate the compressed feature data volume in average at QP22 where the fidelity is around 0.99. The results are summarized in Table 4.3. We can find that the compressed data volume of *conv*5 and *pool*5 features are less than 10 KByte which is usually much smaller than high-quality images. Also knowing that *conv*5 feature extraction may take 99.2% computing load out of the whole analysis task [59], transmission of such compressed feature can largely economize the visual analysis system in edge-cloud collaboration by reducing the computing load for cloud and saving the bandwidth usage.

The video-codec-based coding framework is proposed to expediently borrow the matured video coding techniques and ease the software and hardware development for intermediate deep feature compression in the early research stage. However, as demonstrated in Section 4.4.2, the video codec can not perfectly match the

feature data distribution since it is designed and optimized for natural images and videos. Moreover, some advanced video coding tools are redundant to be existed in the coding framework to compress the features, which do not contribute to the final results but taking the computing resources. In view of this, more coding tools designed for intermediate deep features are anticipated. As such, new coding frameworks for intermediate deep feature compression can be proposed without integrating the video codecs. More efficient data transmission for visual analytics in edge-cloud collaboration can be expected at that stage.

TABLE 4.3: Feature volume berfore and after compression. Task performance drop is around 1% due to the lossy compression.

	Before Compression (Kb)	After compression (Kb)
conv1	12544	672.0
pool1	3136	253.2
conv2	6272	438.3
pool2	1568	173.5
conv3	3136	182.7
pool3	784	79.9
conv4	1568	52.0
pool4	392	26.2
conv5	392	7.3
pool5	98	3.8

4.5 Conclusion

This chapter explored lossy compression of intermediate deep learning feature. A video-codec-based coding framework and evaluation metrics were proposed. Various coding tools for the proposed coding framework were also developed. In particular, uniform quantizer and learning-based adaptive quantizer have comparable good performance in PreQuantization module, while the learning-based adaptive quantizer can help the entire coding framework achieve higher upper limits of fidelity. Channel tiling method presents good performance, especially on high layer features, in the Repack module. The optimal combination of "adaptive quantization working together with channel tiling" can achieve over 50x compression ratio with less than 1% task performance drop, where the volume of the compressed intermediate deep learning features can be much more less than the volume of input image bit stream.

Chapter 5

Image Quality Assessment Based Label Smoothing in Neural Network Training

There are three major processes in the proposed intermediate deep feature transmission paradigm: feature extraction, feature coding, and feature reuse . In feature extraction, to generate high-quality features for machine vision tasks, we present a label smoothing method to train more robust and generic backbone networks for feature extraction at edge in this chapter. The image quality assessment (IQA) based label smoothing method is evaluated in image classification task for neural network training. The proposed technique leads us to the robust neural network generalized to images with a broad range of quality levels. As such, more robust and generic backbone neural networks can be implemented for intermediate deep learning feature extraction in edge-side devices for visual analysis applications in edge-cloud collaboration. Section 5.1 analyses the existing data augmentation with distorted images for deep learning. Section 5.2 introduces the proposed scheme with IQA-based label smoothing. Section 5.3 provides experimental results and analyses on the proposed method.

5.1 Data Augmentation with Distorted Images for Deep Learning

It is generally acknowledged that the capability of the deep model largely depends on the training data. As such, to deal with the low quality test images, the straightforward solution is augmenting the training data with low quality versions and mixing them together with the original high quality data. In view of this, in this work we involve the pristine as well as the distorted images as the input data to learn the CNN model. The distorted versions are generated by injecting different levels of distortion into the pristine training images manually. As one of the first attempts using distorted images as the training data, here we are particularly interested in three types of commonly encountered distortions: blur, noise and JPEG compression.

Blurring artifacts are introduced due to the limitations of acquisition process such as camera motion or out-of-focus of the optical system. In addition, necessary image processing operations may also create blurring artifacts, such as denoising and compression. Therefore, investigating the blurring artifacts enables us to simulate the real-world application scenarios in recognizing and understanding the blurry images. Blurring artifacts will dramatically reduce the sharpness of images and distort the structural information of the texture content, which directly influence the interpretation of the images.

Noise may result from acquiring images with low quality camera sensors or in poor illumination conditions. Generally speaking, Gaussian noise can be used to model this scenario by adding the disturbance to each pixel. The introduced noise will make the image more disorderly with higher uncertainties for human perception from the perspective of the free energy theory [107].

Lossy compression techniques are also commonly used in computer vision processing, which introduce distortion in visual signals while reducing storage or memory footprint requirement simultaneously. Among the digital image coding standards, JPEG compression is the most wildly-used one, which can bring strong blocking boundaries and ringing artifacts in low bit coding scenarios.

All these types of distortions may introduce disturbance in representing the semantic information of images by damaging the image content. According to the study

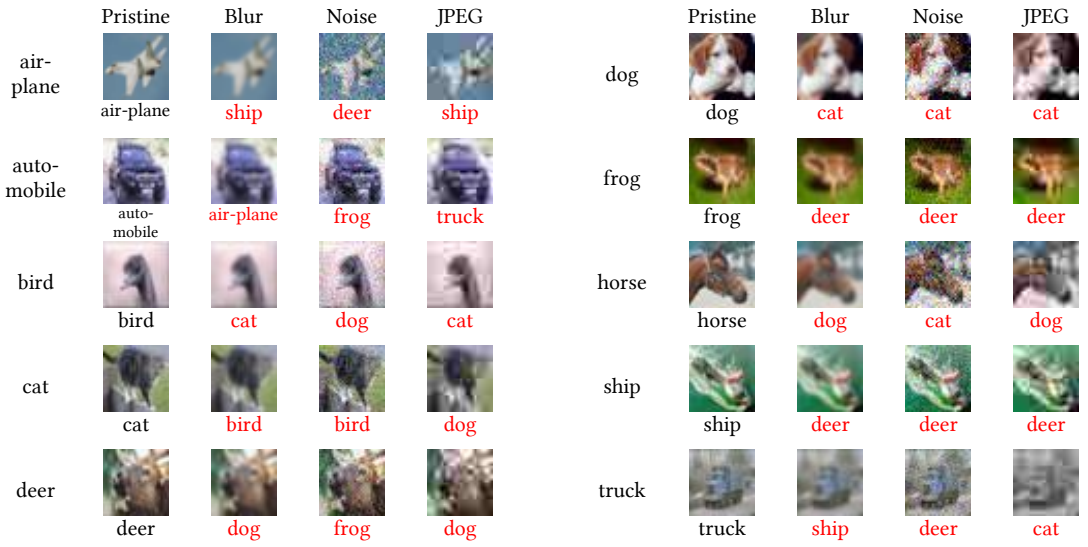


FIGURE 5.1: Illustration of the classification results of the 10 classes in CIFAR-10, where each class set including one pristine sample image and its corresponding distorted images. The first column of a class set is the pristine sample image of CIFAR-10. The second, third and fourth columns are the images corrupted by blur, noise and JPEG compression respectively. Given the pristine images, a DNN model can correctly identify them. However, when distortions are injected, the DNN model may misclassify the inputs. The labels below thumbnails are the prediction results by the DNN model.

in [108], not only the human visual system, but also computer vision algorithms cannot efficiently recognize and understand the visual information with these three kinds of distortions. Therefore, it is meaningful to start from these artifacts, which pose a unique set of challenges to computer vision tasks. However, there are also side effects when both pristine and distorted images are used as the training data. In particular, although the model can provide promising prediction performance on the distorted images in principle, it may also significantly degrade the testing performance when we feed the high quality images as the input. Thus, in this work we further seek a good balance between the high quality and low quality training images with IQA to learn a more robust model.

5.2 IQA-based Label Smoothing

In typical image classification tasks, a softmax layer is usually laid on the top of a neural network to predict the probabilities of an input image belonging to each

given class. Alternatively, the probability can also be treated as the perceptive confidence of the model for the input image belonging to a specific class. For each input image x, the softmax layer computes the probability $p(k|x)$ for each given label $k \in \{1 \cdots K\}$. The ground-truth distribution is denoted by $q(k|x)$. For brevity, we omit the dependence of p and q on the input x. During the training process, the loss can be calculated with p and q as follows,

$$l = -\sum_{k=1}^{K}(p(k) - q(k))^2. \tag{5.1}$$

As such, minimizing the loss is equivalent to maximizing the expected Euclidean-likelihood of a label, where the label is selected according to its ground-truth distribution $q(k)$. In most situations, given a single ground-truth label y, $q(y) = 1$ for $k = y$ while $q(k) = 0$ when $k \neq y$. This encourages the deep model to be robust and confident in the classification tasks of pristine images.

As mentioned early, although the corrupted images are involved in the training process to deal with the scenario of low quality images as the input, such method may not permanently solve the problem, and there are several challenging issues:

- Since both high and low quality images are used as the training data, the learned model is lack of the generalization capability and exhibit strong bias to the low quality images. As such, though we can improve the accuracy of low quality images, the prediction performance of the high quality images will be degraded.

- The human visual perception has been largely ignored in the learning process. It is generally hypothesized that the human visual system evolves through learning from the natural images that possess certain statistical properties. As such, low quality images which belong to unnatural images should play a less importance role compared with pristine images since low-quality images are more difficult to understand. As such, a reasonable way to manipulate this is to make the expected probabilities corresponding to the ground-truth labels of the low-quality images lower. However, the commonly used ground-truth distribution does not follow this trend.

To avoid these drawbacks and incorporate the brain-like perception in the deep learning framework, an IQA-based label smoothing method (IQA-LS) is proposed.

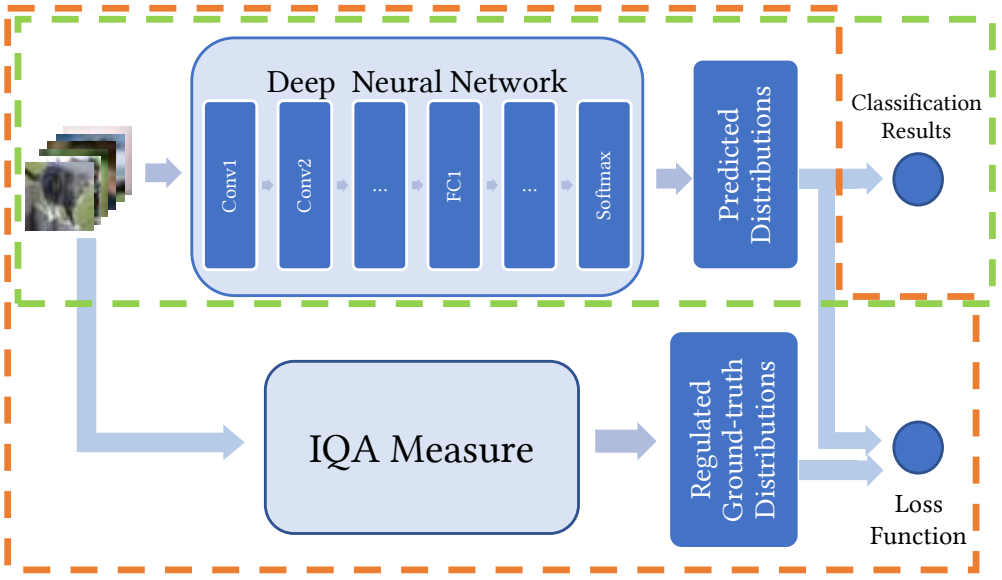


FIGURE 5.2: The deep learning framework with IQA-based label smoothing. The modules in the orange dashed box represent the training process while the modules in the green dashed box stand for the inference process.

In particular, given the label k and a single input x with ground-truth label y, the label distribution $q(k|x)$ is reformulated as follows

$$q'(k|x) = \begin{cases} T(s(x)) & k = y \\ (1 - T(s(x)))/(K-1) & k \neq y \end{cases} \tag{5.2}$$

where $s(\cdot)$ denotes the score of IQA measure and $T(\cdot)$ transforms the IQA score to the range of $(0, 1]$. This implies that the distribution of the label k is obtained based on the IQA score of the input image x when $k = y$, while the uniform distribution is employed for the rest labels. Therefore, the confidence value is directly determined by the image quality, and better quality implies higher confidence in the network learning. This is in line with the human perception when understanding the image content, as low quality images are usually perceived with higher uncertainties from the perspective of free-energy theory [107].

In this work, we adopt the SSIM [60] as the IQA measure due to its good trade-off between the accuracy and computational complexity. In particular, it is computed by comparing the original and distorted images based on the degradation of the structural information. IQA measure is only employed in the training procedure where we can get access to both the distorted and its corresponding pristine images, as shown in Figure 5.2. It is also worth mentioning that other IQA algorithms

including reduced-reference (RR) and no-reference (NR) methods are also compatible with my proposed IQA-based label smoothing framework.

5.3 Experimental Results

To evaluate the reliability of DNN models learned by the proposed IQA-based label smoothing method, experiments are conducted on the CIFAR-10 database. First, we briefly introduce the database and the training data enhancement strategy. Subsequently, the parameter settings in the experiments are detailed. Finally, we illustrate and analyse the classification results with different training strategies.

5.3.1 Dataset description

In this chapter, the proposed scheme is evaluated on CIFAR-10 dataset which is a labelled subset of 80 million tiny images [109]. As shown in Figure 5.1, CIFAR-10 is composed of colour images with size 32×32, and in total there are 10 different classes for performing the classification tasks. Moreover, the dataset contains 50,000 training samples and 10,000 testing samples. In order to compare the models trained with different strategies, except for specific image distortion, data enhancement strategies (e.g. image contrast, brightness and saturation adjustment) are prohibited in the training process. Here, as discussed in Section 5.1, we only apply Gaussian blur, Gaussian noise and JPEG compression.

5.3.2 Benchmark learning architecture

The learning architecture is designed following the model described in the Tensorflow [110]. In particular, it is a Tensorflow based duplication of Alex Krizhevsky's work [2] with a few modifications. Specific descriptions regarding the proposed learning architecture is illustrated in Table 5.1.

This model follows the common multi-layer convolutional neural network (CNN) architecture which consists of alternating convolutions and nonlinearities. This architecture takes the full-size image as the input. The padded input is filtered with 64 kernels of size $5 \times 5 \times 3$ with a stride of 1 pixel to produce 64 feature maps

TABLE 5.1: Descriptions of the learning architecture. The input and output sizes are specified as $rows \times cols \times channels$, and the kernel is characterized in terms of $rows \times cols, stride$.

layer	size-in	size-out	kernel
$conv1$	$32 \times 32 \times 3$	$32 \times 32 \times 64$	$5 \times 5, 1$
$pool1$	$32 \times 32 \times 64$	$16 \times 16 \times 64$	$3 \times 3, 2$
$lrn1$	$16 \times 16 \times 64$	$16 \times 16 \times 64$	/
$conv2$	$16 \times 16 \times 64$	$16 \times 16 \times 64$	$5 \times 5, 1$
$lrn2$	$16 \times 16 \times 64$	$16 \times 16 \times 64$	/
$pool2$	$16 \times 16 \times 64$	$8 \times 8 \times 64$	$3 \times 3, 2$
$fc1$	4096	384	/
$fc1$	384	192	/
$softmax$	192	10	/

TABLE 5.2: Performance comparisons of the models with different training strategies.

Strategy	Regularization Method	Training Set	Pristine	Blur			Noise			JPGE		
				Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
1	Original	*Pristine*	0.794	0.676	0.524	0.436	0.677	0.566	0.392	0.600	0.529	0.391
2	Original	MIX_{blur}	0.781	0.777	0.766	**0.751**	0.735	0.681	0.585	0.669	0.621	0.469
3	IQA-LS	MIX_{blur}	**0.798**	**0.782**	**0.766**	0.749	0.749	0.698	0.607	0.681	0.624	0.465
4	Original	MIX_{noise}	0.794	0.716	0.606	0.526	0.779	0.773	**0.749**	0.693	0.640	0.496
5	IQA-LS	MIX_{noise}	**0.807**	0.736	0.612	0.525	**0.792**	**0.776**	0.743	0.699	0.642	0.503
6	Original	MIX_{JPEG}	0.753	0.728	0.711	0.656	0.720	0.662	0.616	0.710	0.687	**0.607**
7	IQA-LS	MIX_{JPEG}	**0.791**	0.751	0.704	0.622	0.742	0.667	0.607	**0.722**	**0.693**	0.590
8	Original	MIX_{all3}	0.767	0.753	0.737	0.721	0.761	0.744	0.725	0.721	0.686	**0.599**
9	IQA-LS	MIX_{all3}	**0.790**	**0.773**	**0.753**	**0.738**	**0.771**	**0.757**	**0.731**	**0.726**	**0.692**	0.585

with identical size as the input. After the nonlinear transformation by Rectified Linear Units (ReLU), the feature maps are response-normalized and down-sampled by Local Response Normalization (LRN) and Overlapping max-Pooling respectively [2] to generate the input of second convolutional layer. Subsequently, the similar process is applied again to produce the input of the following fully connected layers whose activation function is also ReLU. Finally, the softmax layer takes the representation learned by the network to create the final classification results, which characterize the probabilities of the input belonging to each given class.

5.3.3 Parameter setting

The proposed deep architectures are trained with stochastic gradient descent method on a NVIDIA GeForce 980Ti GPU with batch size 100 for 2,000 epochs. All our experiments use the initial learning rate of 0.1 which decays for every 350 epochs

with an exponential rate of 0.1. In addition, L2Loss weight decay multiplied by 0.004 is added to the two full-connected layers.

Regarding the training data, pristine images and the mixture data with both pristine and distorted images are used for validations. The pristine data are totally from the CIFAR-10 training dataset, and the mixture data are the combination of pristine and distorted images with different distortion levels in a fixed ratio. In each training epoch, 60% of the pristine training samples are maintained, 15% of them are with *level* 1 distortion, another 15% of them are with *level* 2 distortion, and the rest 10% samples are distorted in *level* 3.

Specifically, for the blur distortion, we use the Gaussian kernels with $\sigma = 0.7, 1.0, 1.2$ for *levels* from 1 to 3 respectively. With respect to the noise artifacts, three levels of white Gaussian noise with variance values $v = 0.005, 0.01, 0.02$ are employed. Regarding to JPEG compression, we compress the images with the JPEG quality factors of $12, 8, 4$ corresponding to the distortion *levels* from 1 to 3. In particular, the images generated by the selected parameters can severely influence the inference performance of deep networks, which has also been pointed out in [108].

In summary, five different types of training set are utilized in the experiments: one pristine training set and four sets of mixture data with both pristine and distorted samples. For convenience, the four sets of mixture data are denoted as MIX_{blur}, MIX_{noise}, MIX_{JPEG} and MIX_{all3}, where MIX_{blur}, MIX_{noise} and MIX_{JPEG} represent the training sets of mixture data with pristine and distorted images degraded by one certain type of distortion (blur, noise and JPEG respectively), while MIX_{all3} is the combination of pristine and all the three types of distorted samples.

With respect to the testing data, a pristine set and nine distorted sets (with three different types and each one has three distortion levels) are generated for evaluating each learned model. Example images with different distorted levels on the three types of artifact are also illustrated in Figure 5.1.

5.3.4 Performance comparisons

Nine different training approaches are implemented to evaluate the performance of the proposed scheme. All these nine approaches share the same architecture,

as introduced in the sub-section 5.3.2, but different training strategies. These training strategies are with different combinations of data augmentation strategies and label distributions. Here, we will detail these training strategies and analyse their performance in terms of the classification accuracy, as illustrated in Table 5.2.

5.3.4.1 Training on pristine dataset

Strategy 1 in Table 5.2 aims to train deep models with the pristine CIFAR-10 training samples without the augmented data. Moreover, the label distribution of each training image is the classical 0-1 distribution. Such training strategy follows the widely adopted benchmark models such as AlexNet and VGG. Therefore, we consider this approach as the baseline.

From Table 5.2, we can see that the model trained with this strategy is sensitive to all the three involved distortions as the classification performance decreases dramatically when the distortion level increases. For instance, even with the blur *level* 1 where the σ value of Gaussian kernel is moderate, the accuracy drops more than 10%. Similar trends can be observed for noise and JPEG compression artifacts. This is in accordance with the evaluation results in [108]. This phenomenon can be explained by the fact that the distortions can heavily remove the texture and edge information in an image, which is important to the DNN models learned with pristine images since such DNN models may always attempt to look for specific textures and edges for the classification task.

5.3.4.2 Training on mixture dataset

Strategies 2,4,6,8 in Table 5.2 train DNN models with the mixture data of pristine and distorted images while the label distribution maintains the typical 0-1 distribution. Such kind of training approach is a straightforward solution to make the network better adapt to the distorted images.

As shown in Table 5.2, the results exhibit that training strategy with low quality samples improves the performance on the corresponding distorted images. For instance, the classification accuracy of Strategy 2 only decays about 1% when the distortion level rises from pristine to level 1. Such decreasing speed is an order of magnitude slower than that of the baseline strategy. However, it is noticed that the

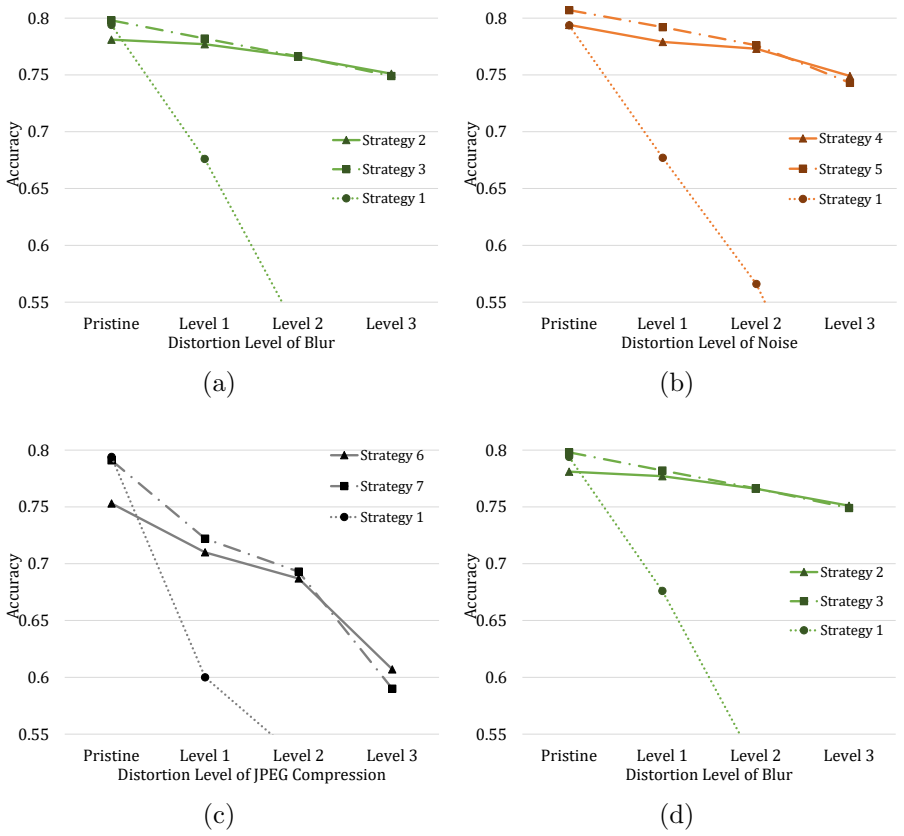


FIGURE 5.3: Performance comparisions. (a) Performance comparisons of the models trained and tested with the distortion type of blur, which correspond to Strategies 2 and 3; (b) Performance comparisons of the models trained and tested on the distortion type of noise, which correspond to Strategy 4 and 5; (c) Performance comparisons of the models trained and tested on the distortion type of JPEG, which correspond to Strategy 6 and 7; and subfigure (d) Performance comparisons of the models trained on all three types of distorted images, and tested for each distortion type individually (e.g., *S. 8 - blur* means strategy 8 and the blurred images are used for testing).

performance of these strategies on high-quality pristine images cannot approach as high as the baseline method, which can be intuitively observed in Figure 5.3 (where the dot lines denotes the baseline model while the solid lines denote the models naively trained on mixture data). As discussed in Section 5.2, this is due to the fact that 0-1 label distribution teaches the model to be equally confident about the classification results of both high and low quality images.

5.3.4.3 Training with IQA-based label smoothing

Strategies 3,5,7,9 in Table 5.2 target at training the model based on the mixture data as well. Moreover, in contrast to the previous strategies, the label distribution is regularized by the proposed IQA-based label smoothing method.

As shown in Figure 5.3, the performance of models trained with IQA-LS is credibly better than the original ones on relative high quality images (e.g.,pristine, distortion level 1 and 2). Regarding to the performance on strongly distorted images (e.g., distortion level 3), although the models trained with IQA-LS are slightly weaker than those without IQA-LS, the performance drop is marginal and acceptable. Therefore, it is concluded that, comparing to the straightforward way that trains the deep models on mixture data, our proposed IQA-LS technique is not only effective in maintaining the high classification performance for distorted samples, but also promising in improving the accuracy on high quality test data. Moreover, from Figure 5.3(d), it is observed that when training on mixture of samples with multiple types of artifacts rather than a certain type the superiority of IQA-LS is more apparent. This can be explained by the reason that the regularized label distribution penalises the false inference based on the quality levels, which provides the DNN models with stronger generalizing ability.

5.3.4.4 Generalizing ability of IQA-LS on unknown artifacts

Table 5.2 also presents results of models trained on certain distortion type (denoted as known artifact) but tested on other distortion types (denoted as unknown artifacts). In particular, models trained with MIX_{blur} data (strategies 2 and 3) are tested on images corrupted by Gaussian noise and JPEG artifact; models trained with MIX_{noise} data (strategies 4 and 5) are tested on images corrupted by Gaussian blur and JPEG artifact; models trained with MIX_{JPEG} data (strategies 6 and 7) are tested on images corrupted by Gaussian blur and Gaussian noise. From the visualized results as shown in Figure 5.5, we can easily observe that models trained with IQA-LS (denoted as dash dot lines) can generally achieve better performance than models trained without IQA-LS (denoted as solid lines) on the same test data. It demonstrates that our proposed IQA-LS method can provide the deep learning models with higher generalizing capability. Namely, deep learning models trained with IQA-LS is generally more robust against both known and unknown artifacts.

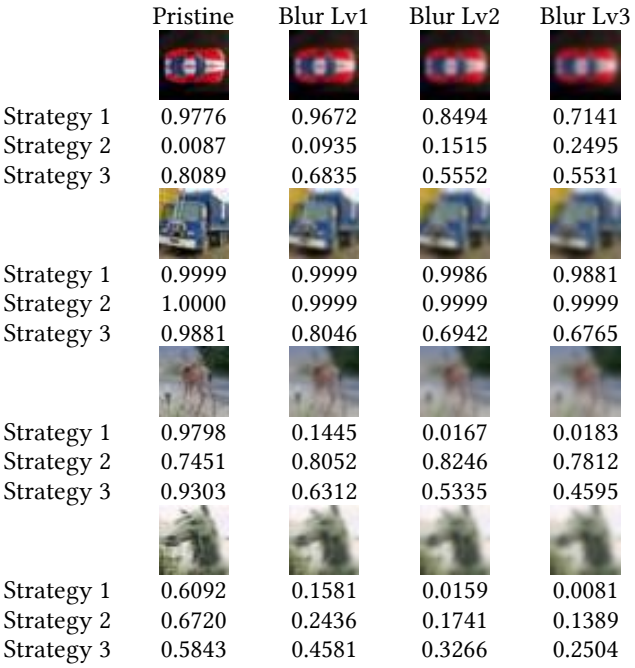
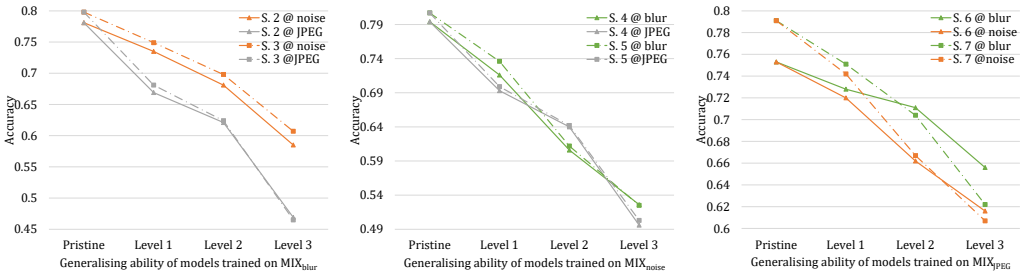


FIGURE 5.4: Examples of the test images and the corresponding confidence values predicted by models trained with Strategy 1 (baseline), Strategy 2 (trained on mixture data straightforwardly) and Strategy 3 (trained on mixture data with proposed IQA based label smoothing technique).

However, it may not be always true when dealing with JPEG artifact at a strong distortion level. As shown in Figure 5.5(a), when testing on level-3 JPEG distorted data, model without IQA-LS (strategy 2) surpasses the model with IQA-LS (strategy 3), although difference of accuracy between the two ($0.469 - 0.465 = 0.004$) is negligible. Moreover, as shown in Figure 5.5(c), when training with JPEG distorted data (MIX_{JPEG}), models with IQA-LS do not perform better at strong distorted images (level-3 Gaussian noise and level-2,3 Gaussian blur). It may be because the pattern of JPEG artifact is far different from patterns of Gaussian blur and Gaussian noise, as JPEG artifact is patch-wise while Gaussian blur and Gaussian noise is more of pixel-wise.

5.3.4.5 Discussions

Here, we perform an in-depth analysis on the proposed scheme to gain a better understanding on the IQA based label-smoothing technique. Let us focus on a certain type of distortion, i.e. blur. Figure 5.4 provides some example images along with the confidence values that indicate the prediction results for the correct



(a) (b) (c)

FIGURE 5.5: Generalizing ability comparisons. (a) Comparison of models trained on MIX_{blur} but tested on distortion types of noise and JPEG; (b) Comparison of models trained on MIX_{noise} but tested on distortion types of blur and JPEG; (c) Comparison of models trained on MIX_{JPEG} but tested on distortion types of blur and noise.

class with three strategies, which are the baseline and the two trained on MIX_{blur}. In particular, the confidence values are generated by the corresponding node of the softmax layer. Specifically, the value close to 0 implies that the model is unconfident about the prediction, while the value close to 1 corresponds to a high confidence prediction. It can be observed that Strategy 3, which utilizes IQA-based label smoothing technique, can always provide a smooth and moderate variation trend on the confidence values with the increase of the blur level. This is in line with the human perception, as blurry images may usually lead to uncertainty in image understanding, thus creating lower confidence values. By contrast, the confidence values of Strategies 1 and 2 are more consistent when the distortion level changes. For some cases, the prediction cannot be robust when the distortion level is extremely high (the images in the third and fourth rows of Figure 5.4). Moreover, Strategy 2 may suffer from the over-fitting problem to the high-level distorted images (images in the first row of Figure 5.4).

Therefore, the proposed IQA-based label smoothing technique leads to the model that not only provides fairly good prediction performance for both high and low quality images, but also simulates human-like perception from the perspective of uncertainty. Moreover, the proposed method may also help to reduce the over-fitting problem when severely distorted images are used for training. In the future, more distortion types, IQA methods and deep learning models will be investigated in this framework.

5.4 Conclusion

In this chapter, an image quality assessment (IQA) based label smoothing approach was proposed for deep neural network training. The novelty of the proposed approach lies in that the distorted images are included in the training process in learning the reliable neutral network model, and IQA is adopted in regularizing the label distribution of training samples to obtain a more robust representation. The performance of the proposed scheme was evaluated based on image classification. Such training method achieved high prediction accuracy across different distortion types and levels, which can help providing more robust and generic backbone network for visual analysis.

Chapter 6

Deep Holographic Networks for Similarity Metrics Learning

There are three major processes in the proposed intermediate deep feature transmission paradigm: feature extraction, feature coding, and feature use. In feature use , we present a deep holographic network for more efficient data retrieval on top of the intermediate deep features in this chapter. The deep holographic network with a holographic composition operator is designed to learn similarity metrics of deep learning features with multiple labels. The proposed method improves the performance for retrieval tasks while reduces network parameters and memory costs for cloud implementation in edge-cloud collaboration. In experiments, the proposed deep holographic network takes intermediate deep learning feature extracted by the backbone network (i.e., GoogLeNet) as input to measure similarity scores of paired videos for sementic video retrieval task. Section 6.1 introduces the existing methods for metric learning in retrieval tasks. Section 6.2 proposes the deep holographic network. Section 6.2.5 discuss pros and cons of the proposed method comparing with existing deep learning approaches. Section 6.3 presents the evaluation results.

6.1 Preliminaries

Content-based video search is to retrieve videos in a database that are the most similar to a query video. Feature representations and similarity metrics are the key components to a video search system. Previous work [111] mainly concentrated

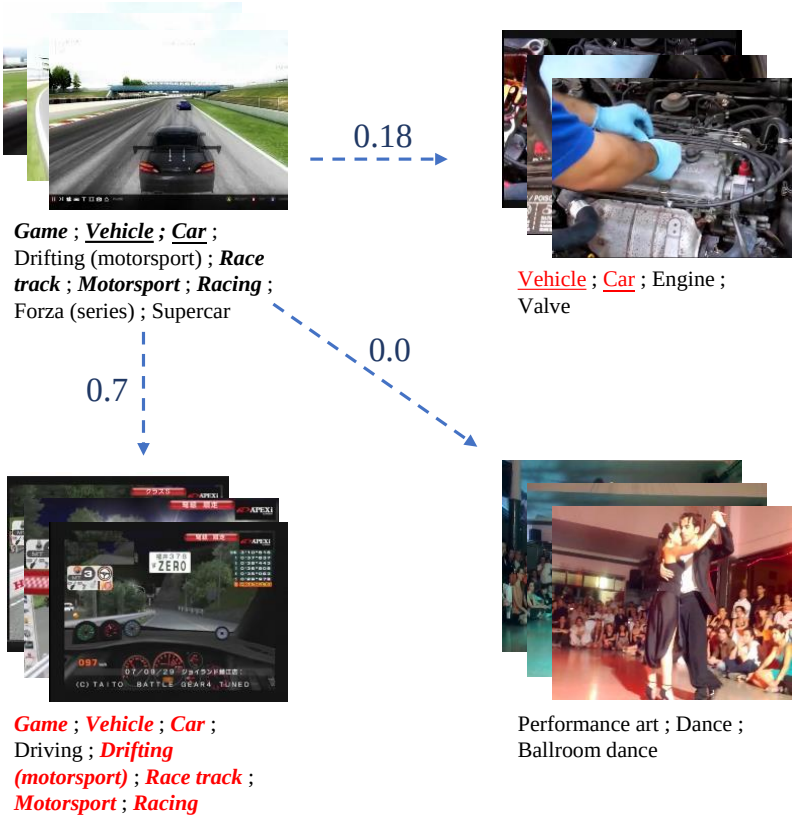


FIGURE 6.1: Examples of YouTube videos with coarse to fine-grained labels. Intuitively, the similarity metric between a video pair shall depend on the overlap ratio of their labels. The higher the overlap ratio, the more similar they are.

on instance retrieval applications such as video copy detection and near-duplicate search, in which low-level features that can distinguish textured differences between fine-grained objects/scenes are developed, followed by standard distance computation between these features. These approaches did not consider high-level concepts depicted in videos, which are essential for semantic video search (e.g. event retrieval [112, 113]) to close the so called semantic gap. This problem becomes more challenging when videos are associated with multiple labels, e.g. the number of truth labels per video varies from 1 to 31 for the recently introduced YouTube-8M dataset [114] (Figure 6.1). In this work, we focus on the joint learning of feature representations and similarity metrics, tailored for multi-label video search.

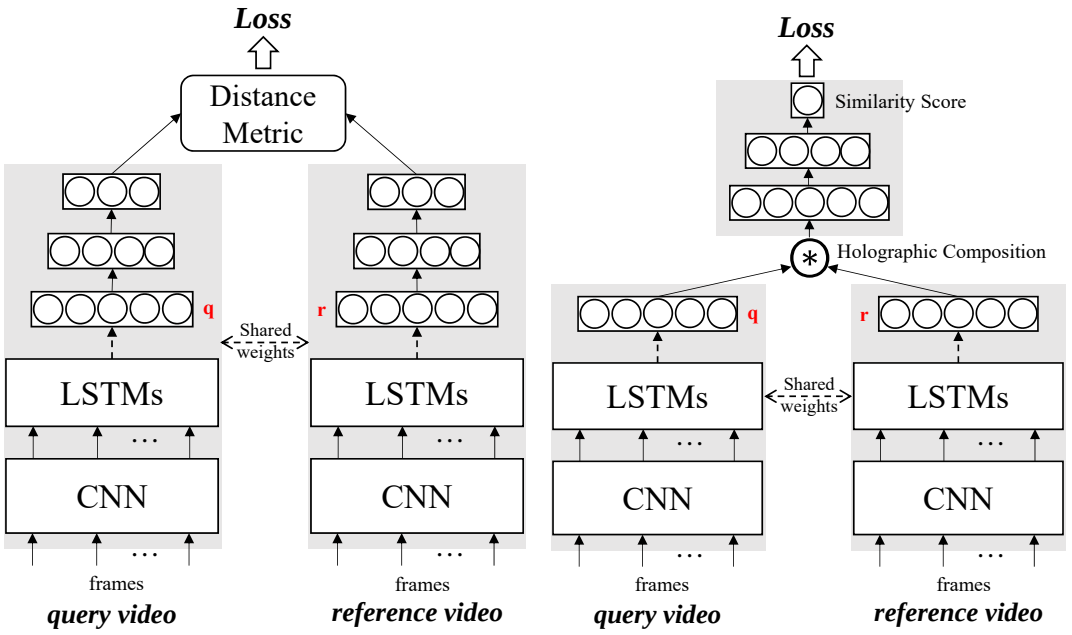
Traditional video instance search systems [9, 111] are mainly built upon handcrafted features including local (e.g. SIFT [11]) and global descriptors (VLAD [115] and FV [116]), followed by standard distance computations. Recently, MPEG started

the standardization of Compact Descriptors for Visual Analysis (CDVA) [9], with the aim to come up with a normative bitstream of standardized features for mobile visual search and augmented reality applications.

In recent years, deep learning has also been developed for video applications [9, 117]. [117] proposed FV and VLAD aggregation techniques over dense local features of CNN activation maps from video frames for event retrieval. [9] generated video representations by performing pooling over CNN activation maps extracted from video keyframes, which are subsequently used for video instance retrieval of objects, scenes and landmarks. LSTM has also been applied over frame-level CNN activations [118] for modeling spatial-temporal clues [119]. Recent work on large scale video classification challenges such as ActivityNet [120] and YouTube8M [114] shown that the combination of CNN and LSTM obtains the best accuracy than simple aggregation techniques over CNN activations. In this work, we use the combined architecture of CNN and LSTM as the backbone network for learning video representations.

A number of recent works demonstrated that feature representation together with similarity metric can be learned in a unified deep learning framework [121–123]. [124, 125] proposed siamese network to learn similarity metrics with contrastive loss for face verification and image instance retrieval, respectively. There are also triplet-based metric learning methods [25, 126, 127] that extends siamese network with triplet input {query, positive, negative}. [128] proposed to learn patch-level image descriptors by integrating a feature network and a metric network together.

Finally, our approach draws inspirations from recent work on holographic composition for link prediction on knowledge graph [129]. Holographic composition has also demonstrated its effectiveness for question answering in natural language processing [130, 131]. However, to our knowledge, this work is the first to exploit holographic composition for deep metric learning in computer vision.



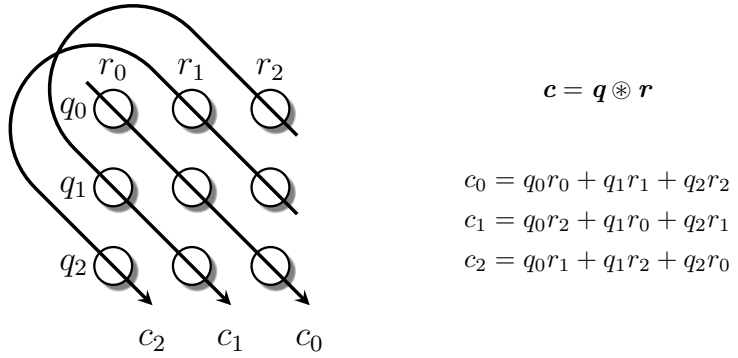
(a) Non-compositional method (b) Compositional method

FIGURE 6.2: (a) Siamese Network (Non-compositional method) and (b) compositional network architectures for learning similarity metrics.

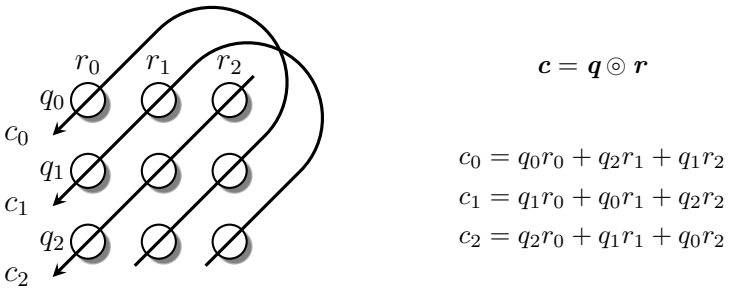
6.2 Deep Holographic Networks

6.2.1 Overview

As shown in Figure 6.2(b), the proposed deep holographic networks (DHN) can be decomposed into three key components: First, twin feature networks with shared architecture and weights for video-level feature extraction of a video pair independently. Following [114], we use Inception network to extract CNN features for frames sampled from a video. Subsequently, the frame-level features are input to a LSTM network, and aggregated into a video-level feature representation with temporal feature encoded. In addition, multiple LSTMs can be stacked together to explore longer term dependencies of video frames. Second, the pair of video-level features are transformed to a new compositional feature representation by holographic composition layer, which we introduced in the following section. Finally, the compositional feature vector is fed to a stack of fully connected layers, and end with a sigmoid layer which directly predicts a similarity score of the video pair.



(a) Circular correlation



(b) Circular convolution

FIGURE 6.3: Holographic composition with either (a) circular correlation or (b) circular convolution. Circular arrow denotes summation operation.

6.2.2 Holographic Composition

Instead of learning similarity metrics with ranking loss computed by distances of video pairs with representations output by the top layer, an alternative way for modeling similarity metrics is to explicitly interact activations of the pair at the intermediate holographic composition layer. Let $\boldsymbol{q}$ and $\boldsymbol{r}$ denote video-level feature vectors respectively extracted from query and reference videos using the twin feature networks. the ultimate goal is to predict the similarity score $s(\boldsymbol{q}, \boldsymbol{r})$ of the pair with

$$s(\boldsymbol{q}, \boldsymbol{r}) = \sigma(W^T(\boldsymbol{q} \bigcirc \boldsymbol{r}) + b), \tag{6.1}$$

where $\bigcirc$ denotes a compositional function operated on the pair $\{\boldsymbol{q}, \boldsymbol{r}\}$, resulting in a compositional feature vector as input to the subsequent fully connected layers. $\boldsymbol{W}$ and b denote learnable weights and bias of fully connected layers (for simplicity, Equation 6.1 contains only one fc layer), σ is the sigmoid function. Particularly, holographic composition can be implemented with either circular correlation or circular convolution [132].

Circular correlation. Let $\circledast : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ denotes the compositional operator of circular correlation, which is computed as

$$[\boldsymbol{q} \circledast \boldsymbol{r}]_k = \sum_{i=0}^{d-1} \boldsymbol{q}_i \boldsymbol{r}_{(k+i) \bmod d}, \quad k \in [0, d-1] \tag{6.2}$$

Figure 6.3(a) shows an example of circular correlation. Basically, circular correlation performs pairwise multiplications followed by summation with certain patterns. In addition, the computation of Equation 6.2 can be significantly accelerated with fast Fourier transform (FFT) and inverse FFT, resulting in computational complexity of $\mathcal{O}(d \log d)$.

Circular convolution. As shown in Figure 6.3(b), circular convolution is closely related to circular correlation,

$$[\boldsymbol{q} \circledcirc \boldsymbol{r}]_k = \sum_{i=0}^{d-1} \boldsymbol{q}_i \boldsymbol{r}_{(k-i) \bmod d}, \quad k \in [0, d-1] \tag{6.3}$$

where $\circledcirc$ denotes the compositional operator of circular convolution.

The main differences between circular correlation and circular convolution are two-fold. First, circular correlation is non-commutative, i.e., $\boldsymbol{q} \circledast \boldsymbol{r} \neq \boldsymbol{r} \circledast \boldsymbol{q}$, while circular convolution is commutative. Second, as shown in Figure 6.3(a), the first component of the compositional representation from circular correlation, $[\boldsymbol{q} \circledast \boldsymbol{r}]_0$ (i.e. c_0), represents the dot product of $\boldsymbol{q}$ and $\boldsymbol{r}$, which closely relates to the cosine similarity of the pair, which is preferred for video search.

6.2.3 Loss function

Intuitively, the similarity of multi-label videos shall depend on the overlap ratio of their labels. In view of this, we compute the ground-truth semantic similarity between query and reference videos via Jaccard index,

$$\hat{s} = \jmath(q, p) = \frac{|q \cap p|}{|q \cup p|} \tag{6.4}$$

where q and p denotes the set of labels from query and reference videos. One may note that the holographic composition layer is parameter-free. Thus, we jointly

train the twin feature networks and fully connected layers end to end by Stochastic Gradient Descent, with the objective to minimize either the simple mean squared error loss or the cross-entropy loss $-(\hat{s} * \log(s(\boldsymbol{q}, \boldsymbol{r})) + (1 - \hat{s}) \log(1 - s(\boldsymbol{q}, \boldsymbol{r})))$.

6.2.4 Inference

We apply the DHN for the task of multi-label video search. Given a query video, the goal is to rank the reference videos by their similarity scores to the query, which are directly output by the DHN for each video pair. With M queries and N reference videos, the number of video pairs $\{\boldsymbol{q}, \boldsymbol{r}\}$ is $M \times N$. Performing a forward pass through the DHN for each pair is ineffective as there are duplicated computations for feature networks. Instead, one can significantly accelerate the inference speed over multiple queries with a two-stage retrieval pipeline. First, video-level feature vectors are extracted with a single forward pass through the feature network for all the $M + N$ videos. Second, the holographic composition layer and fully connected layers are used to generate similarity score for each of the $M \times N$ pairs.

6.2.5 Discussions

6.2.5.1 DHN vs. Siamese network

Siamese network can be modeled by embedding a pair of samples into a low-dimensional Euclidean space independently, followed by standard distance comparison,

$$s(\boldsymbol{q}, \boldsymbol{r}) = cos(f(\boldsymbol{q}), f(\boldsymbol{r})) \tag{6.5}$$

where $f(\boldsymbol{x}) = \sigma(W^T \boldsymbol{x} + b)$, σ is the sigmoid function, $cos(\,\cdot\,,\,\cdot\,)$ denotes cosine similarity. The parameters are optimized by minimizing the contrastive loss $-(y * s(\boldsymbol{q}, \boldsymbol{r}) + (1 - y) max(0, m - s(\boldsymbol{q}, \boldsymbol{r})))$, where $y = 1$ when $\{\boldsymbol{q}, \boldsymbol{r}\}$ is match, otherwise, $y = 0$. m is a constant margin.

First, the main difference of our DHN and siamese network is that the former explicitly interacts features of a pair at intermediate layer of the network architecture, while the latter does not. We argue that encoding metric at intermediate layer could potentially enable the network to learn richer pairwise relationships. Second, as

shown in Table 6.1, the memory complexity of DHN is much smaller than siamese network, while with very small computational overhead $\mathcal{O}(d \log d)$ introduced by circular correlation. Finally, DHN is storage free as it directly predicts similarity score, but siamese network needs to store feature representations output by the top layer. Due to limited space, we exclude triplet network from the discussion as its inference complexities are the same with siamese network.

6.2.5.2 DHN vs. Other compositional networks

Deep Concatenation Network (DCN). A straightforward approach for feature composition is to directly concatenate features from a pair of samples [128]. Let $\oplus : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^{2d}$ denotes concatenation operator, one may note that concatenation operator doubles the dimensionality the compositional vector, i.e. from d from $2d$. Thus, it increases the memory and computational cost of the fully connected layer by a factor of 2.

Neural Tensor Network (NTN). Concatenation does not interact features of a pair, another compositional operator termed tensor product is proposed [130], which is in accordance with the outer product. An exhaustive pairwise multiplications between $\boldsymbol{q}$ and $\boldsymbol{r}$ is performed,

$$[\boldsymbol{q} \otimes \boldsymbol{r}]_{ij} = \boldsymbol{q}_i \boldsymbol{r}_j, \quad i, j \in [0, d-1] \tag{6.6}$$

where $\otimes$ denotes the tensor product operator. In this sense, tensor product allows to capture all pairwise interactions, with the cost of dramatically increased dimensionality of the compositional vector (from d to d^2). This significantly increases the memory and computational cost of the subsequent fully connected layers. Also, It is worth noting that vanilla NTN usually combines concatenation and tensor product together. Thus, the resulted compositional vector is with dimension of $(d^2 + 2d)$.

DHN differs from DCN in that holographic composition directly interacts features, instead of simple concatenation of them. On the other hand, holographic composition generates features with half the size of concatenation, resulting in less memory and computational complexity by a factor of 2 (see Table 6.1). A major advantage of DHN over NTN is that DHN enables modeling interactions of features, without significantly increasing the number of parameters of the subsequent layers (see Table 6.1).

TABLE 6.1: Theoretical memory complexity of different methods. We only count the number of parameters for fully connected layers after the twin feature networks, by assuming there is only one hidden fc layer with h neurons and bias is not counted (see Figure 6.2). For Siamese/Triplet Network, t represents the number of neurons at the top layer ($t >> 1$).

Network	# Parameters
Siamese / Triplet Network	$(d+t)h$
Deep Concatenation Network	$(2d+1)h$
Neural Tensor Network	$(d^2+2d+1)h$
DHN (Ours)	$(d+1)h$

6.3 Experiments

6.3.1 Dataset

The **YouTube-8M** video dataset [114] contains around 8 million multi-label videos categorized into 4,716 classes. The number of ground truth labels per video varies from 1 to 31, with an average of 3.4 per video. This dataset presents two major challenges: diversity and class imbalance. The videos cover many different possible topics (music, politics, etc.), styles (CGI, documentary, etc.), and formats (conversation, action, etc.). Furthermore, there is significant class imbalance, with only 10 labels accounting for over half of all label appearances in the dataset, while a large portion of classes has a very small number of videos.

Class imbalance in queries would cause the evaluation accuracy driven by majority classes. To encourage a fair comparison, we build a new dataset termed **YouTube-MLR** designed for the multi-label video search task, based on the YouTube-8M set. We rank the classes by the number of videos in descending order, choose the top 1,000 classes, the rest classes are removed. As YouTube-8M ground-truth are only available for training and validation sets, we create the YouTube-MLR training and test set from the original YouTube-8M training and validation sets, respectively. In particular, the YouTube-MLR test set is constructed by sampling videos from the trimmed YouTube-8M validation set, randomly and evenly for each of the 1,000 classes selected. There is no duplicate video in YouTube-MLR test set.

To evaluate retrieval accuracy, the YouTube-MLR test set is further split into query and reference subsets. Furthermore, in order to evaluate performance trend

TABLE 6.2: Training and test dataset statistics.

	# Classes	Training	Validation	
YouTube-8M	4,716	4.906M	1,401,828	
			Test	
	Classes	Training	# Query	# Refs
Ours (small)	1,000	4.776M	4,569	20,133
Ours (large)			9,526	187,442

at different scales, we generate a small-scale and a large-scale test set termed **YouTube-MLR-S** and **YouTube-MLR-L**, respectively. Table 6.2 summaries the statistics of YouTube-8M and the new YouTube-MLR-S/YouTube-MLR-L data sets.

6.3.2 Evaluate Metric

Normalized Discounted Cumulative Gain (NDCG) is a standard evaluation metric of ranking quality in information retrieval community, which takes the similarity level into consideration. NDCG is calculated as

$$NDCG@p = \frac{DCG@p}{IDCG@p} \tag{6.7}$$

where $DCG@p = \sum_{i=1}^{p} \frac{2^{\hat{s}_i}-1}{log(i+1)}$ and $IDCG@p = \sum_{i=1}^{p} \frac{2^{s_i}-1}{log(i+1)}$. p is the truncated rank position; s_i and $\hat{s}_i$ are the ground-truth similarity score and the predicted similarity score, respectively for the i-th position in a ranking list. For all experiments, we set $p = \{1, 10, 100, 1000\}$ and compute mean NDCG (mNDCG) for all queries.

6.3.3 Model Training

It's impractical to process the hundreds of Terabytes YouTube-8M videos [114]. Therefore, the organizers released pre-extracted features for video frames (1 frame per second, at most 300 frames per video), by extracting 2048-d feature vectors from the Inception network pre-trained on ImageNet, followed by PCA whitening to reduce feature dimension to 1024. In this work, we regard the 1024-d features as the CNN feature representations. The CNN features serve as the input to a two-layer LSTMs with 1024 hidden nodes. The output of LSTMs, i.e. concatenation

of hidden state and cell state of each LSTM, is a 4096-d video-level feature vector as the input to the holographic composition layer.

To learn parameters of the layers after Inception network, we sample matching and non-matching video pairs from the YouTube-MLR training dataset. For each epoch, we randomly sample 10,000 from the training data as queries and 400 video pairs for each query, with the criterions that (1) the sampled queries are distributed evenly across classes; (2) query and positive videos are matching pair only if they have at least 1 label overlapped, otherwise, they are non-matching pair; (3) for each query, 60% of the 400 video pairs are matching pairs, the rest are non-matching pairs. Batch size is 100, 60% of them are matching pairs. We use learning rate 0.001, which is divided by 5 for every 10 epochs. To accelerate the convergency, we initialized the parameters of LSTMs with the model pre-trained on YouTube-8M [133]. We train the network for 100 epochs with the Adam optimizer. Training takes around 1 week on a single NVIDIA Tesla K40m, while inference speed for a video pair is around 55 ms.

In the following section, we refer DHN with circular correlation and circular convolution to **DHN-COR** and **DHN-CON**, respectively. We compare DHN to other compositional and non-compositional approaches, including Deep Concatenation Network (**DCN**) [128], Neural Tensor Network (**NTN**) [130], **Siamese Net** [125] and **Triplet Net** [127]. We tune hyper-parameters (e.g. architectures for the fully connected layers, margin for Siamese/Triplet Net, etc) to explore the performance trade-offs for each network architecture. At last, we also include a **Baseline** method by applying sum pooling over the 1024-d CNN features to form a new 1024-d video-level features. The similarity between a video pair is computed as the cosine similarity of their 1024-d video features.

6.3.4 Results

DHN vs. Other compositional networks. Table 6.3 shows comprehensive comparisons of DHN over other compositional variants, in terms of mNDCG on the YouTube-MLR-S test set. All compositional networks are trained with mean square error loss. We only count projection matrix of the FC layers, bias is ignored as it is insignificant. First, DHN obtains the best mNDCG@N scores across different rank positions (N=1,10,100,1000), with significantly lower memory complexity over the

TABLE 6.3: Comparisons of the proposed DHN with other compositional methods on the YouTube-MLR-S test set.

Method	FC Layers	# Params	mNDCG@1	mNDCG@10	mNDCG@100	mNDCG@1000
Baseline	–	–	0.463	0.454	0.462	0.448
DCN	8K - 1	8.19K	0.059	0.050	0.078	0.133
	8K - 1K - 1	8.39M	0.341	0.387	0.496	0.616
	8K - 4K - 1K - 1	37.75M	0.255	0.298	0.404	0.589
NTN	16,785,408 - 2 - 1	33.57M	0.288	0.352	0.463	0.551
	16,785,408 - 4 - 1	67.14M	0.318	0.390	0.509	0.605
	16,785,408 - 6 - 1	100.71M	0.374	0.430	0.525	0.620
	16,785,408 - 8 - 1	134.28M	0.409	0.456	0.521	0.593
DHN-CON	4K - 1K - 1	4.20M	0.459	0.506	0.604	0.683
DHN-COR	4K - 1	4.10K	0.452	0.486	0.562	0.645
	4K - 1K - 1	4.20M	**0.527**	**0.563**	**0.628**	**0.691**

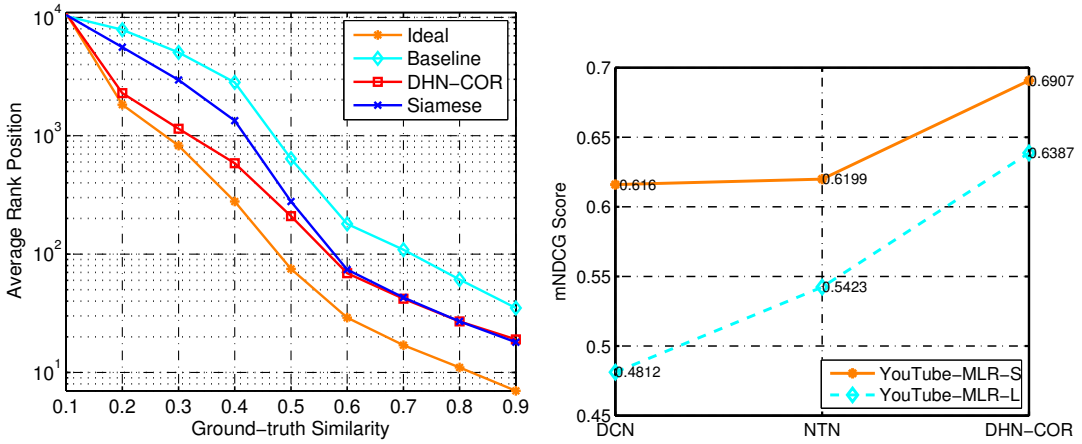
TABLE 6.4: Effect of loss functions, i.e. mean square error (MSE) and cross entropy loss, on compositional methods in terms of mNDCG@1000 with the YouTube-MLR-S test set.

Method	# Params	MSE	Cross Entropy
DCN	8.39M	0.616	0.670
NTN	100.71M	0.620	0.591
DHN-CON	4.20M	0.683	0.684
DHN-COR	4.20M	0.691	0.694

TABLE 6.5: Comparisons of the proposed DHN with non-compositional methods, on the YouTube-MLR-S test set.

Method	FC Layers	Margin	mNDCG@1000
Siamese	4K - 1K - 512	0.3	0.520
		0.5	0.535
		0.7	0.562
Triplet	4K - 1K - 512	0.3	0.585
		0.5	0.638
		0.7	0.596
DHN-CON	4K - 1K - 1	–	0.683
DHN-COR	4K - 1K - 1	–	0.691

rest. Compositional networks are superior to the baseline as N increases. Second, as expected, DHN-COR outperforms DHN-CON, suggesting that the circular correlation is more favored than circular convolution for the matching problem. Third, we study the effect of FC layer structures for compositional operators. There are consistent performance improvements if marginally increasing the number of FC parameters, the retrieval accuracy of DCN and NTN tends to drop if FC layers are too large, which is probably due to overfitting. Finally, mNDCG scores for the



(a) (b)

FIGURE 6.4: (a) Statistic of rank positions of reference videos as a function of their ground-truth similarity with queries, for the Baseline, Siamese Network and DHN-COR respectively. The higher the overlap ratio of labels between query and reference, the higher the ground-truth similarity. (b) Comparisons of retrieval performance trends between DHN and other compositional networks, as test dataset scales up.

baseline are stable as N changes. However, in most cases, mNDCG scores increases with N for all compositional networks.

Effect of Loss functions. Table 6.4 studies the effect of loss functions (mean square error and cross entropy loss) for compositional networks, in terms of mNDCG@1000 on the YouTube-MLR-S test set. For simplicity, the FC layer structure for DCN, NTN, DHN-CON and DHN-COR follows the best performing model in Table 6.3, respectively. We observe that DCN with mean square error performs much worse than DCN with cross entropy, while NTN prefers mean square error. DHN-CON and DHN-COR are robust to both loss functions.

DHN vs. Siamese/Triplet network. Table 6.5 compares DHN to non-compositional methods including siamese network and triplet network on the YouTube-MLR-S test set. One can see that the DHN variants significantly outperform siamese and triplet networks, while with smaller memory complexity of FC layers. It's worth noting that the best performing triplet network is much better than the best performing siamese network (i.e. 0.638 vs. 0.562)

How does DHN help? One hypothesis is that holographic composition shall be helpful to boost the rank of {query, reference} pairs with higher overlap ratio of labels. To verify this, we compute the average rank position of reference videos

as a function of their ground-truth similarity with queries (Equation 6.4), given the predict ranking list of the Baseline, Siamese Network and DHN-COR on the YouTube-MLR-S test set, respectively. With a collection of query-reference video pairs with ground-truth similarity ranging from 0.1 to 0.9 with step size 0.1, average rank position is computed as the median of the positions where reference videos ranked in the predict list. As shown in Figure 6.4(a), we observe that DHN-COR is able to bridge the gap between the Baseline/Siamese Network and "Ideal", especially for lower ground-truth similarities. This means that DHN-COR is capable of ranking relevant reference videos with low overlap ratio of labels at top positions.

Accuracy vs. Scale. Finally, we study the performance trends of compositional networks as test data scales up. Figure 6.4(b) compares the performance loss when data scale increases from YouTube-MLR-S to YouTube-MLR-L, for DCN, NTN and DHN-COR respectively. As one can see, the relative performance loss of DHN-COR is the lowest (-7.54%, vs. NTN -12.53% and DCN -21.89%), implying that holographic composition is more robust to scale change than tensor product and concatenation operators.

6.4 Conclusion

In this chapter, I proposed a deep holographic network to learn similarity metrics of videos with multiple labels. The holographic composition with circular correlation was introduced to explicitly model the interaction of a video pair with intermediate features of the network architecture. It provides an alternative solution for deep metric learning, instead of the widely used ranking loss driven approaches built on ultimate features. In addition, the holographic operator is parameter-free and enables less memory footprint than state-of-the-art. Promising results demonstrated the effectiveness of the proposed method.

Chapter 7

Summary and Future Works

7.1 Summary

In this thesis, I proposed a new data transmission paradigm for visual analysis in edge-cloud collaboration. To enable this new paradigm, compression techniques for intermediate deep learning features have been developed. Furthermore, an advanced training method with IQA label smoothing technique was proposed to provide more robust and generic backbone neural networks for feature extraction at edge side. Besides, a deep learning architecture with holographic composition was investigated to provide high-performance task-specific models on top of the intermediate deep features for the cloud side.

The proposed paradigm transmits intermediate deep learning features instead of visual signal or ultimately utilized features. The advantage of this approach lies in that the generalization ability is greatly enhanced to achieve multiple analyses tasks performed simultaneously at the cloud side, such that better trade-off can be achieved in terms of the computational load, communicational cost and generalization capability. I further conducted comprehensive lossless compression evaluations on features of four widely used neural networks, finding that the lossless approach is not practical for intermediate deep learning feature coding.

I further explored intermediate deep learning feature coding in a lossy manner. I developed a video-codec-based coding framework and evaluation metrics for the lossy compression. Various coding tools have been introduced and evaluated.

Comprehensive comparative experiments demonstrated the pros and cons of each coding tools. In particular, uniform quantizer and learning-based adaptive quantizer have comparable good performance in PreQuantization module, while the learning-based adaptive quantizer can help the entire coding framework to achieve higher upper limits of fidelity. Channel tiling method presents good performance, especially on high layer features, in the Repack module. The optimal combination of "adaptive quantization working together with channel tiling" can achieve over 50x compression ratio with less than 1% task performance drop, where the volume of the compressed intermediate deep learning features can be far less than the volume of input image bit stream. The drawbacks of utilizing video codecs to compress intermediate deep feature have also been analysed. New coding framework without integrating video codecs is anticipated in the future study.

The novelty of the proposed quality assessment based label smoothing approach for deep neural network learning lies in that the distorted images are taken into account in the training process for learning the reliable neutral network model. Besides, IQA is adopted in regularizing the label distribution of training samples to obtain a more robust representation. The performance of the proposed scheme is evaluated based on image classification which shows that the proposed scheme achieves high prediction accuracy across different distortion types and levels, making the neural network model more robust and generic.

I also presented a deep holographic network to learn similarity metrics of videos with multiple labels on top of the intermediate deep learning features. I introduced holographic composition with circular correlation to explicitly model the interaction of a video pair with intermediate representations of the network architecture. This provides an alternative solution for deep metric learning, instead of the widely used ranking loss driven approaches built on the top layer (ultimate features). In addition, the holographic operator is parameter-free and enables less memory footprint than state-of-the-art.

7.2 Future Works

The newly proposed paradigm may not only change the way of data transmission and coding, but also change the way of deep learning model implementation.

Regarding to the intermediate deep learning feature coding, current coding framework introduced in this thesis is a video-codec-based coding framework, which should integrate the traditional video codec. However, the video codec can not perfectly match the feature data distribution since it is designed and optimized for natural images and videos. Moreover, some advanced video coding tools are redundant in the coding framework to compress the features, which do not contribute to the final results but taking the computing resources. In view of this, more coding tools designed for intermediate deep features are anticipated. As such, new coding frameworks for intermediate deep feature compression can be proposed without integrating the video codecs. More efficient data transmission for visual analytics in edge-cloud collaboration can be expected at that stage. In addition, feature selection can also be further investigated to choose which feature in a neural network should be transmitted to maximize the generality for tasks while minimize the computing and communication costs.

Traditional coding methods aim for the best video/image under certain bit-rate constraint for human consumption. However, the new paradigm provides new requirements to capture the video and extract the useful information for machine vision to perform description and compression of features for machine analysis, and also there are requirements for combined machine and human vision to perform combined human/machine-oriented video representation/compression. In view of this, joint optimization of video coding and feature coding can be investigated in the future. In particular, features are good at maintaining high-level semantic information with a low bit rate, while visual signals are suitable to maintain structural information. It can be explored that how to use both feature bitstream and video bitstream to reconstruct high quality videos for human vision, or to achieve better task performance for machine vision.

As to the deep learning models, in the proposed paradigm, one neural network may be split into two parts for implementation in both edge-side devices and cloud-side servers. As the backbone network deployed in the edge-side devices can directly access to the raw visual signals, instead of compressed RGB or YUV data. As the image signal processing (ISP) will introduce noises and distortions to the visual signal, neural networks with raw image signal input may obtain better task performance. It could be an interesting topic for future research. Besides, as the feature coding is performed in a lossy compression way, coding-aware backbone

network and task-specific model deign can be another meaningful research direction. In particular, coding-aware backbone networks at the edge side should be able to generate features which are robust against the lossy compression. The coding-aware task-specific model at the cloud side should maintain high task performance with corrupted features. New neural network architectures and training methods for such models are anticipated in the future.

To better enable the proposed paradigm in the future, many interdisciplinary researches are worth to be investigated. For instances, this thesis only studied the source coding for intermediate deep feature transmission, but channel coding was not explored. It is worth to investigate whether existing communication protocol can well handle the data transmission error for intermediate deep feature. Channel coding techniques optimized for deep feature transmission can also be explored. Furthermore, network system can be studied to reduce the latency in feature communication for delay-sensitive applications such as autonomous driving. The proposed paradigm may also work well with federated learning, as sensor data is not necessary to be transmitted to the cloud servers in this paradigm, but intermediate feature containing rich information for machine learning is available. The proposed paradigm is intuitively of high data privacy and security level for federated learning. Encryption method for intermediate deep learning feature is also worth further investigating.

Appendix A

Supplemental data of Chapter 4

TABLE A.1: Fidelity comparison of three quantizers on different feature types of ResNet.

	8-bit			6-bit			4-bit		
	uniform	log	adaptive	uniform	log	adaptive	uniform	log	adaptive
conv1	0.989	**0.996**	0.994	0.960	0.971	**0.974**	0.791	**0.900**	0.898
pool1	0.989	**0.996**	0.994	0.960	0.971	**0.974**	0.791	**0.900**	0.898
conv2	0.996	0.999	**1.000**	0.990	**0.992**	0.990	0.945	0.967	**0.974**
conv3	0.998	**1.000**	0.999	0.988	**0.999**	0.996	0.961	**0.984**	0.982
conv4	0.997	**1.000**	**1.000**	0.992	0.995	**0.997**	0.948	0.983	**0.984**
conv5	**1.000**	**1.000**	**1.000**	0.995	0.998	**0.999**	0.988	0.991	**0.992**

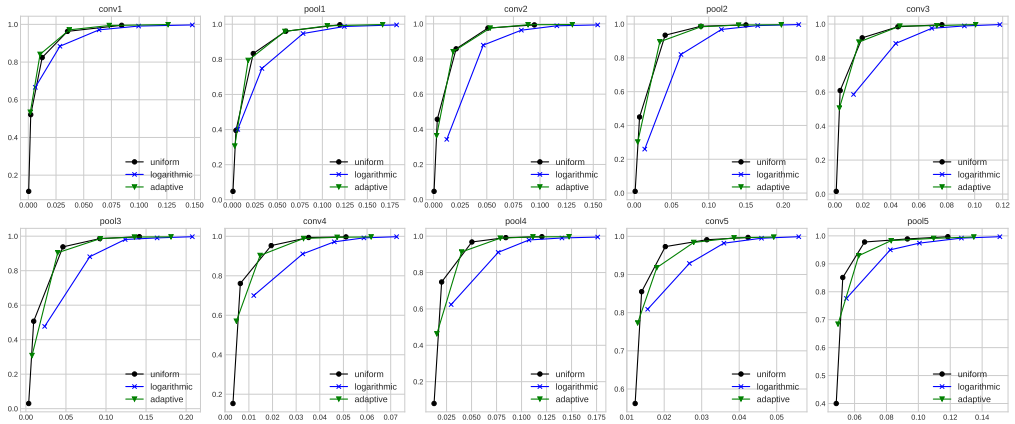


FIGURE A.1: [Extension of Figure 4.7] [Detailed numerical data can be found in Tables A.2, A.3, A.4]Coding performance comparison of the hybrid coding framework with three different PreQuantization tools. The horizontal axises represents compression rate, while the vertical axises represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

TABLE A.2: [Numerical results for Figures 4.7 and A.1] Compression results of 8-bit uniform quantizer working together with naive channel concatenation on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.084	0.995	0.035	0.963	0.013	0.824	2.20E-03	0.521	3.46E-04	0.114
pool1	0.119	0.997	0.059	0.960	0.023	0.833	4.04E-03	0.395	7.22E-04	0.048
conv2	0.095	0.994	0.051	0.976	0.021	0.858	3.50E-03	0.457	4.60E-04	0.047
pool2	0.150	0.995	0.089	0.987	0.042	0.934	7.15E-03	0.450	1.23E-03	0.010
conv3	0.077	0.996	0.045	0.984	0.020	0.919	3.79E-03	0.609	9.93E-04	0.016
pool3	0.142	0.997	0.092	0.986	0.046	0.938	0.010	0.507	3.49E-03	0.030
conv4	0.051	0.997	0.035	0.994	0.019	0.953	6.38E-03	0.761	3.26E-03	0.153
pool4	0.120	0.997	0.084	0.992	0.050	0.968	0.020	0.748	0.012	0.080
conv5	0.042	0.997	0.031	0.991	0.020	0.973	0.014	0.855	0.012	0.562
pool5	0.118	0.997	0.093	0.989	0.066	0.978	0.053	0.851	0.049	0.400

TABLE A.3: [Numerical results for Figures 4.7, 4.8, 4.11 and A.1, A.2, A.4] Compression results of 8-bit logarithmic quantizer working together with naive channel concatenation on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.148	0.997	0.100	0.991	0.064	0.971	0.029	0.884	6.30E-03	0.666
pool1	0.182	0.996	0.124	0.987	0.079	0.947	0.033	0.748	6.05E-03	0.401
conv2	0.154	0.994	0.116	0.989	0.082	0.964	0.047	0.879	0.012	0.343
pool2	0.221	0.997	0.166	0.991	0.117	0.968	0.063	0.820	0.014	0.260
conv3	0.118	0.997	0.093	0.989	0.069	0.975	0.044	0.886	0.013	0.586
pool3	0.208	0.997	0.164	0.990	0.124	0.981	0.080	0.881	0.023	0.477
conv4	0.072	0.998	0.059	0.992	0.046	0.972	0.033	0.911	0.012	0.700
pool4	0.176	0.995	0.140	0.990	0.107	0.979	0.077	0.911	0.030	0.624
conv5	0.056	0.999	0.046	0.995	0.036	0.982	0.027	0.929	0.016	0.809
pool5	0.151	0.997	0.127	0.992	0.101	0.974	0.082	0.950	0.055	0.776

TABLE A.4: [Numerical results for Figures 4.7, 4.8 and A.1, A.2] Compression results of 8-bit adaptive quantizer working together with naive channel concatenation on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.126	0.999	0.073	0.994	0.037	0.971	0.011	0.841	1.84E-03	0.534
pool1	0.167	0.998	0.106	0.991	0.058	0.959	0.018	0.794	2.72E-03	0.306
conv2	0.130	0.996	0.089	0.995	0.053	0.976	0.018	0.842	2.87E-03	0.364
pool2	0.197	0.996	0.140	0.994	0.090	0.984	0.034	0.895	4.69E-03	0.302
conv3	0.101	0.997	0.073	0.991	0.047	0.988	0.017	0.894	3.28E-03	0.506
pool3	0.181	0.996	0.136	0.995	0.093	0.986	0.040	0.904	7.64E-03	0.308
conv4	0.062	0.997	0.047	0.995	0.033	0.989	0.015	0.903	4.71E-03	0.570
pool4	0.147	0.997	0.111	0.996	0.079	0.989	0.040	0.913	0.015	0.463
conv5	0.049	0.998	0.039	0.996	0.028	0.984	0.018	0.918	0.013	0.773
pool5	0.135	0.996	0.110	0.991	0.083	0.983	0.063	0.929	0.050	0.684

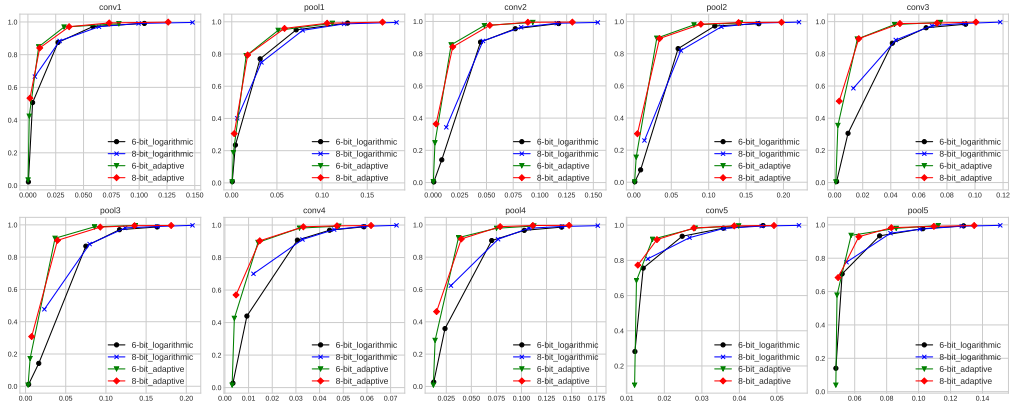


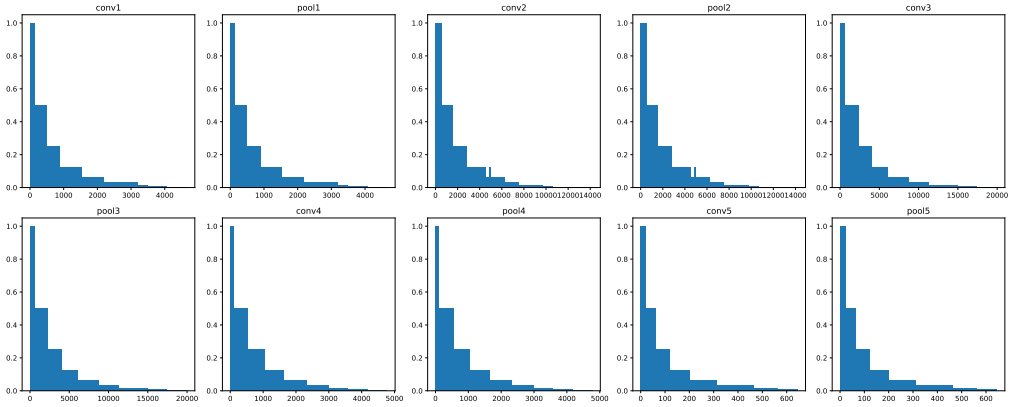
FIGURE A.2: [Extension of Figure 4.8] [Detailed numerical data can be found in Tables A.3, A.4, A.5, A.6] Coding performance comparison of the hybrid coding framework with different bit depth settings for PreQuantization module. The horizontal axises represents compression rate, while the vertical axises represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

TABLE A.5: [Numerical results for Figures 4.8 and A.2] Compression results of 6-bit logarithmic quantizer working together with naive channel concatenation on VGGNet.

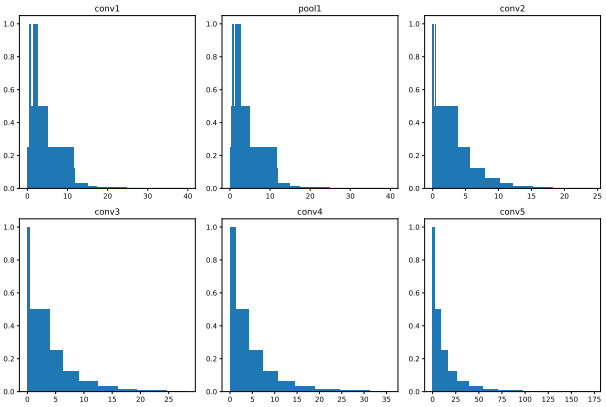
	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.105	0.991	0.059	0.972	0.027	0.875	4.21E-03	0.507	4.44E-04	0.022
pool1	0.128	0.992	0.071	0.950	0.031	0.770	4.05E-03	0.235	5.64E-04	0.008
conv2	0.118	0.987	0.077	0.955	0.044	0.872	8.17E-03	0.141	7.05E-04	0.004
pool2	0.167	0.987	0.108	0.973	0.059	0.832	8.98E-03	0.077	1.22E-03	0.003
conv3	0.093	0.984	0.065	0.962	0.041	0.866	0.010	0.306	1.39E-03	0.006
pool3	0.164	0.987	0.117	0.970	0.075	0.868	0.016	0.142	3.85E-03	0.012
conv4	0.059	0.989	0.044	0.968	0.031	0.907	9.29E-03	0.440	3.38E-03	0.027
pool4	0.140	0.986	0.103	0.966	0.070	0.903	0.024	0.358	0.012	0.027
conv5	0.046	0.998	0.036	0.983	0.025	0.937	0.014	0.756	0.012	0.282
pool5	0.128	0.994	0.103	0.977	0.076	0.934	0.052	0.706	0.048	0.140

TABLE A.6: [Numerical results for Figures 4.8 and A.2] Compression results of 6-bit adaptive quantizer working together with naive channel concatenation on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.082	0.989	0.032	0.969	0.010	0.848	1.22E-03	0.424	1.80E-04	0.034
pool1	0.111	0.991	0.051	0.947	0.016	0.789	1.85E-03	0.188	3.97E-04	0.006
conv2	0.093	0.994	0.048	0.975	0.017	0.855	1.78E-03	0.247	2.88E-04	0.005
pool2	0.143	0.992	0.080	0.980	0.031	0.897	3.05E-03	0.156	9.27E-04	0.002
conv3	0.075	0.995	0.043	0.982	0.016	0.892	2.32E-03	0.355	8.52E-04	0.004
pool3	0.137	0.996	0.086	0.988	0.037	0.917	5.70E-03	0.172	3.20E-03	0.007
conv4	0.048	0.993	0.032	0.981	0.014	0.893	3.99E-03	0.426	3.05E-03	0.015
pool4	0.112	0.993	0.075	0.979	0.037	0.921	0.014	0.285	0.012	0.008
conv5	0.040	0.996	0.028	0.983	0.017	0.920	0.013	0.685	0.012	0.091
pool5	0.112	0.996	0.086	0.979	0.058	0.937	0.049	0.579	0.048	0.039



(a) VGGNet



(b) ResNet

FIGURE A.3: [Extension of Figure 4.10] Examples of sensitivity distributions derived by the learning-based adaptive quantizer on features of VGGNet and ResNet. The horizontal axises represents the data range of the features, while the vertical axises is the reciprocal of quantization step size reflecting the sensitivity.

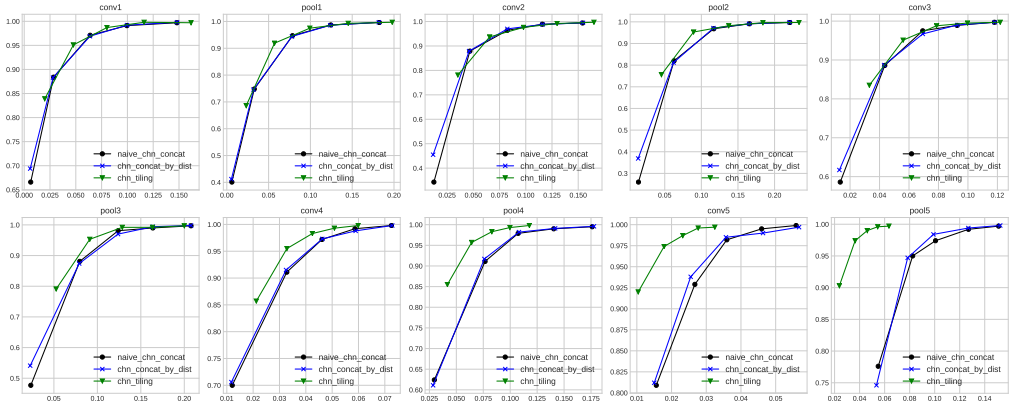


FIGURE A.4: [Extension of Figure 4.11] [Detailed numerical data can be found in Tables A.3, A.7, A.8] Coding performance comparison of the hybrid coding framework with three different Repack tools. The horizontal axises represents compression rate, while the vertical axises represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

TABLE A.7: [Numerical results for Figures 4.11 and A.4] Compression results of 8-bit logarithmic quantizer working together with channel concatenation by distance on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.148	0.998	0.100	0.992	0.064	0.969	0.028	0.883	5.81E-03	0.694
pool1	0.182	0.997	0.124	0.986	0.078	0.944	0.032	0.746	5.49E-03	0.412
conv2	0.155	0.995	0.116	0.988	0.082	0.970	0.046	0.879	0.012	0.455
pool2	0.220	0.998	0.165	0.991	0.116	0.971	0.062	0.812	0.014	0.369
conv3	0.118	0.997	0.093	0.990	0.069	0.967	0.043	0.887	0.013	0.617
pool3	0.208	0.998	0.164	0.994	0.124	0.970	0.079	0.874	0.023	0.541
conv4	0.073	0.998	0.059	0.988	0.046	0.973	0.032	0.915	0.012	0.706
pool4	0.178	0.996	0.142	0.991	0.108	0.982	0.076	0.917	0.028	0.611
conv5	0.057	0.997	0.046	0.990	0.036	0.985	0.026	0.938	0.015	0.812
pool5	0.152	0.998	0.127	0.994	0.099	0.984	0.078	0.947	0.053	0.746

TABLE A.8: [Numerical results for Figures 4.11 and A.4] Compression results of 8-bit logarithmic quantizer working together with channel tiling on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.162	0.997	0.116	0.998	0.080	0.987	0.048	0.951	0.020	0.839
pool1	0.198	0.997	0.145	0.993	0.099	0.975	0.057	0.919	0.023	0.686
conv2	0.165	0.997	0.130	0.992	0.098	0.976	0.066	0.937	0.035	0.781
pool2	0.233	0.997	0.184	0.996	0.137	0.982	0.089	0.953	0.045	0.756
conv3	0.122	0.997	0.100	0.995	0.079	0.988	0.056	0.951	0.033	0.835
pool3	0.200	0.997	0.163	0.992	0.129	0.992	0.091	0.953	0.053	0.791
conv4	0.060	0.998	0.051	0.993	0.042	0.983	0.033	0.955	0.021	0.857
pool4	0.118	0.998	0.100	0.993	0.083	0.983	0.064	0.957	0.042	0.855
conv5	0.032	0.997	0.028	0.996	0.023	0.987	0.018	0.974	0.010	0.920
pool5	0.063	0.997	0.054	0.996	0.046	0.990	0.036	0.974	0.024	0.903

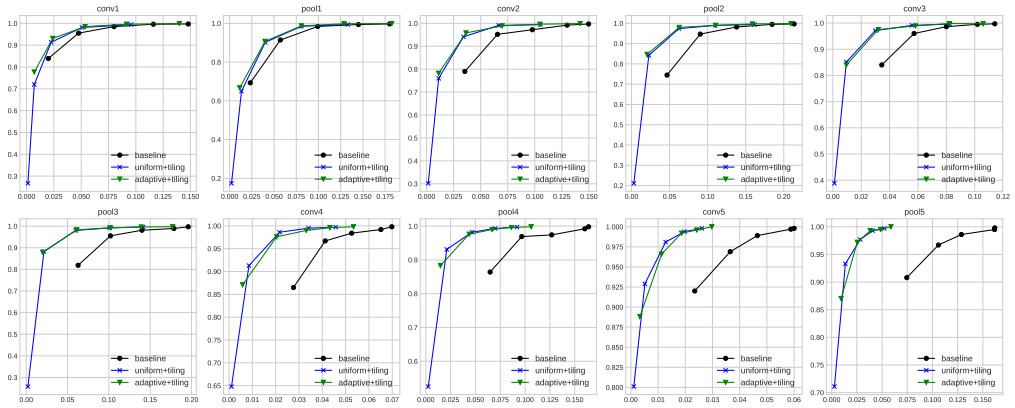


FIGURE A.5: [Extension of Figure 4.12] [Detailed numerical data can be found in Tables A.9, A.10, A.11] Coding performance comparison to the state-of-the-art. The horizontal axises represents compression rate, while the vertical axises represents fidelity. The closer a curve is to the upper left corner, the more efficient is the corresponding method.

TABLE A.9: [Numerical results for Figures 4.12 and A.5] Compression results of Baseline.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.148	0.997	0.116	0.996	0.080	0.985	0.048	0.955	0.020	0.839
pool1	0.180	0.997	0.145	0.994	0.099	0.984	0.057	0.914	0.023	0.693
conv2	0.150	0.997	0.130	0.992	0.098	0.972	0.066	0.952	0.035	0.790
pool2	0.214	0.997	0.185	0.995	0.138	0.982	0.090	0.947	0.047	0.745
conv3	0.114	0.997	0.102	0.995	0.080	0.986	0.057	0.960	0.034	0.840
pool3	0.196	0.997	0.179	0.989	0.140	0.981	0.102	0.955	0.063	0.819
conv4	0.070	0.998	0.065	0.992	0.053	0.984	0.041	0.967	0.028	0.865
pool4	0.164	0.998	0.160	0.992	0.127	0.974	0.097	0.969	0.065	0.864
conv5	0.060	0.998	0.059	0.997	0.046	0.989	0.037	0.969	0.023	0.920
pool5	0.162	0.998	0.162	0.995	0.129	0.986	0.106	0.967	0.075	0.908

TABLE A.10: [Numerical results for Figures 4.12 and A.5] Compression results of 8-bit uniform quantizer working together with channel tiling on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.096	0.994	0.051	0.981	0.022	0.913	7.09E-03	0.720	1.24E-03	0.267
pool1	0.133	0.994	0.081	0.985	0.040	0.902	0.013	0.650	2.09E-03	0.174
conv2	0.105	0.994	0.067	0.991	0.034	0.941	0.011	0.760	1.57E-03	0.302
pool2	0.162	0.994	0.110	0.990	0.062	0.974	0.023	0.842	2.97E-03	0.211
conv3	0.083	0.996	0.056	0.991	0.030	0.971	0.010	0.851	1.12E-03	0.388
pool3	0.141	0.996	0.101	0.992	0.060	0.980	0.021	0.882	2.01E-03	0.258
conv4	0.046	0.997	0.034	0.995	0.022	0.986	8.55E-03	0.913	1.06E-03	0.648
pool4	0.092	0.997	0.070	0.993	0.047	0.982	0.021	0.931	2.51E-03	0.525
conv5	0.026	0.998	0.020	0.994	0.013	0.981	5.14E-03	0.929	1.06E-03	0.801
pool5	0.052	0.997	0.041	0.993	0.028	0.977	0.014	0.933	2.72E-03	0.711

TABLE A.11: [Numerical results for Figures 4.12 and A.5] Compression results of 8-bit adaptive quantizer working together with channel tiling on VGGNet.

	Qp0		Qp12		Qp22		Qp32		Qp42	
	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity	Comp. Ratio	fidelity
conv1	0.140	0.999	0.092	0.996	0.054	0.985	0.024	0.931	7.04E-03	0.777
pool1	0.183	0.999	0.129	0.998	0.081	0.987	0.040	0.907	0.011	0.667
conv2	0.142	0.999	0.105	0.996	0.070	0.989	0.036	0.958	0.011	0.782
pool2	0.209	0.999	0.159	0.997	0.111	0.990	0.063	0.980	0.020	0.847
conv3	0.106	0.997	0.082	0.998	0.058	0.989	0.032	0.975	0.010	0.840
pool3	0.177	0.997	0.139	0.996	0.102	0.992	0.061	0.983	0.020	0.879
conv4	0.053	0.998	0.043	0.996	0.033	0.990	0.020	0.976	5.82E-03	0.871
pool4	0.106	0.998	0.086	0.996	0.067	0.990	0.044	0.976	0.015	0.883
conv5	0.030	1.000	0.024	0.996	0.019	0.992	0.011	0.966	3.31E-03	0.888
pool5	0.059	1.000	0.049	0.995	0.038	0.993	0.025	0.972	9.35E-03	0.870

List of Author's Awards, Patents, and Publications[1]

Awards

- Won 1st place in track 1 of EmotionNet Challenge 2017.
- Won 3rd place in Ad-hoc Video Search (AVS) task of TRECVID Challenge 2018.
- Gold Level (personal) in 2019 WorldQuant Challenge
- The top-10 team in 2019 WorldQuant Challenge National (Singapore) Final

Patents

- **CHEN Zhuo**, FAN Kui, LIN Weisi, DUAN Lingyu, KOT Chi Chung, "Lossy Compression Framework For Intermediate Deep Feature Compression", PCT/SG2020/050526.

Journal Articles

- **Zhuo Chen**, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Toward intelligent sensing: Intermediate deep feature compression. IEEE Transactions on Image Processing 29 (2019): 2230-2243.

[1]The superscript * indicates joint first authors

- **Zhuo Chen**, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Video Codec Based Framework for Intermediate Deep Feature Compression. IEEE Transactions on Circuits and Systems for Video Technology. (submitted)

- Qiuping Jiang, Feng Shao, Wei Gao, **Zhuo Chen**, Gangyi Jiang, and Yo-Sung Ho. Unified no-reference quality assessment of singly and multiply distorted stereoscopic images. IEEE Transactions on Image Processing 28, no. 4 (2018): 1866-1881.

Conference Proceedings

- **Zhuo Chen**, Weisi Lin, Shiqi Wang, Long Xu, and Leida Li. Image quality assessment based label smoothing in deep neural network learning. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6742-6746. IEEE, 2018.

- **Zhuo Chen***, Jie Lin*, Zhe Wang, Vijay Chandrasekhar, and Weisi Lin. Beyond Ranking Loss: Deep Holographic Networks for Multi-Label Video Search. In 2019 IEEE International Conference on Image Processing (ICIP), pp. 879-883. IEEE, 2019.

- **Zhuo Chen**, Kui Fan, Shiqi Wang, Ling-Yu Duan, Weisi Lin, and Alex Kot. Lossy intermediate deep learning feature compression and evaluation. In Proceedings of the 27th ACM International Conference on Multimedia, pp. 2414-2422. 2019.

- **Zhuo Chen**, Ling-Yu Duan, Shiqi Wang, Weisi Lin, and Alex Kot. Data Representation in Hybrid Coding Framework for Feature Maps Compression. In 2020 IEEE International Conference on Image Processing (ICIP), pp. 3094-3098. IEEE, 2020.

Standard Contributions

- Kui Fan, **Zhuo Chen**, "Framework for Lossy Compression of Deep Feature Maps", Audio Video Coding Standard (AVS) document AI M1013, 65th AVS Meeting, Jun 2018.
- **Zhuo Chen**, Kui Fan, "Test Condition for Lossy Compression of Deep Feature Maps", Audio Video Coding Standard (AVS) document AI M1014, 65th AVS Meeting, Jun 2018.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, Tiejun Huang, "Lossy Compression of Deep Feature Maps with Video Codecs", Audio Video Coding Standard (AVS) document AI M1030, 66th AVS Meeting, Aug 2018.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, Tiejun Huang, "The Framework and Test Condition for Lossy Compression of Deep Feature Maps", Audio Video Coding Standard (AVS) document AI M1061, 67th AVS Meeting, Dec 2018.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Upgrades for Framework and Test Condition for Lossy Compression of Deep Feature Maps", Audio Video Coding Standard (AVS) document AI M1088, 68th AVS Meeting, Mar 2019.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "To Support Integer Input for the Lossy Compression Framework of Deep Feature Maps", Audio Video Coding Standard (AVS) document AI M1089, 68th AVS Meeting, Mar 2019.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Minimum Standardization for Lossy Deep Feature Map Compression", Audio Video Coding Standard (AVS) document AI M1100, 69th AVS Meeting, Jun 2019.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Qiang Fu, Lingyu Duan, Alex C. Kot, "Mode extension for Pre-quantization and Repack modules", Audio Video Coding Standard (AVS) document AI M1122, 70th AVS Meeting, Aug 2019.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Additional evaluations for the video-codec-based coding framework", Audio Video Coding Standard (AVS) document AI M1154, 71st AVS Meeting, Dec 2019.

- **Zhuo Chen**, Wenhan Zhang, Lingyu Duan, "Discuss on hardware implementation for feature maps coding", Audio Video Coding Standard (AVS) document AI M1168, 71st AVS Meeting, Dec 2019.
- **Zhuo Chen**, Ziqian Chen, Ling-Yu Duan, Weisi Lin, Alex Chichung Kot, " CNN Feature Map Coding for Cloud-based Visual Analysis", ISO/IEC JTC1/SC29/WG11 MPEG2020/m52162, MPEG 129, Jan 2020.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Updates on Quantization Techniques for Feature Map Compression", Audio Video Coding Standard (AVS) document AI M1220, 72nd AVS Meeting, Mar 2020.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Amendment of Working Draft for Feature Map Compression", Audio Video Coding Standard (AVS) document AI M1286, 73nd AVS Meeting, Jun 2020.
- **Zhuo Chen**, Kui Fan, Weisi Lin, Lingyu Duan, Alex C. Kot, "Cross Validation for Deep Feature Coding", Audio Video Coding Standard (AVS) document AI M1327, 74nd AVS Meeting, Aug 2020.

Bibliography

[1] Zhuo Chen, Kui Fan, Shiqi Wang, Ling-Yu Duan, Weisi Lin, and Alex Kot. Lossy intermediate deep learning feature compression and evaluation. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2414–2422, 2019. xviii, 12, 35, 38, 48, 55, 57, 58

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 11, 22, 66, 67

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 22, 44, 47

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2, 22, 44, 47

[5] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017. 1, 22

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016. 1, 2, 17, 25

[7] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 2, 15, 19, 25

[8] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015. 1, 17, 25

[9] Jie Lin, Ling-Yu Duan, Shiqi Wang, Yan Bai, Yihang Lou, Vijay Chandrasekhar, Tiejun Huang, Alex Kot, and Wen Gao. Hnip: Compact deep invariant representations for video matching, localization, and retrieval. *IEEE Transactions on Multimedia*, 19(9):1968–1983, 2017. 1, 11, 17, 25, 76, 77

[10] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 1

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1, 76

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 11, 26, 43, 47

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1

[14] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2056–2063, 2013. 2

[15] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 1249–1258. IEEE, 2016. 2

[16] Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016. 2, 43

[17] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2

[18] Rohith Polishetty, Mehdi Roopaei, and Paul Rad. A next-generation secure cloud-based deep learning license plate recognition for smart cities. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pages 286–293. IEEE, 2016. 2

[19] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014. 2

[20] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014. 2, 3

[21] Yang Wang, Xuemin Lin, Lin Wu, and Wenjie Zhang. Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval. *IEEE Transactions on Image Processing*, 26(3):1393–1404, 2017. 2

[22] ALESSANDRO ENRICO CESARE Redondi, Luca Baroffio, Lucio Bianchi, Matteo Cesana, and Marco Tagliasacchi. Compress-then-analyze vs analyze-then-compress: what is best in visual sensor networks? *IEEE Transactions on Mobile Computing*, 15(12):3000–3013, 2016. 3

[23] Ling-Yu Duan, Vijay Chandrasekhar, Jie Chen, Jie Lin, Zhe Wang, Tiejun Huang, Bernd Girod, and Wen Gao. Overview of the mpeg-cdvs standard. *IEEE Transactions on Image Processing*, 25(1):179–194, 2016. 3, 11, 15, 23

[24] Ling-Yu Duan, Vijay Chandrasekhar, Shiqi Wang, Yihang Lou, Jie Lin, Yan Bai, Tiejun Huang, Alex Chichung Kot, and Wen Gao. Compact descriptors for video analysis: the emerging mpeg standard. *arXiv preprint arXiv:1704.08141*, 2017. 3, 11, 15, 23

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 3, 11, 77

[26] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015. 3

[27] WM Goodall. Television by pulse code modulation. *Bell System Technical Journal*, 30(1):33–49, 1951. 9

[28] Ernest R Kretzmer. Statistics of television signals. *The bell system technical journal*, 31(4):751–763, 1952. 9

[29] BM Oliver. Efficient coding. *The Bell System Technical Journal*, 31(4): 724–750, 1952.

[30] CW Harrison. Experiments with linear prediction in television. *Bell System Technical Journal*, 31(4):764–783, 1952. 9

[31] Robert M Fano. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics , 1949. 9

[32] AE Laemmel. Coding processes for band-width reduction in picture transmission. In *PROCEEDINGS OF THE INSTITUTE OF RADIO ENGINEERS*, volume 39, pages 293–293, 1951. 9

[33] HC Andrews and WK Pratt. Fourier transform coding of images. In *Proc. Hawaii Int. Conf. System Sciences*, pages 677–679, 1968. 9

[34] H Enomoto and K Shibata. Features of hadamard transformed television signal. In *Nat. Conf. IECE Jpn*, page 881, 1965. 9

[35] William K Pratt, Julius Kane, and Harry C Andrews. Hadamard transform image coding. *Proceedings of the IEEE*, 57(1):58–68, 1969. 9

[36] William Lea. *Video on demand*. House of Commons Library, 1994. 9

[37] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974. 9

[38] Toshio Koga. Motion compensated interframe coding for video-conferencing. In *Proc. Nat. Telecommun. Conf.*, pages G5–3, 1981. 10

[39] Jaswant R Jain and Anil K Jain. Interframe adaptive data compression techniques for images. Technical report, CALIFORNIA UNIV DAVIS SIGNAL AND IMAGE PROCESSING LAB, 1979. 10

[40] JC Candy, MA Franke, BG Haskell, and FW Mounts. Transmitting television as clusters of frame-to-frame differences. *Bell System Technical Journal*, 50 (6):1889–1917, 1971. 10

[41] CCITT SGXV Working Party XV et al. Video codec for audiovisual services at px64 kbit/s. *Draft Revision of Recommendation H*, 261, 1989. 10

[42] Eric Hamilton. Jpeg file interchange format version 1.02. *C-Cube Microsystems*, 1992. 10

[43] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003. 10

[44] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 10, 40

[45] Jianle Chen, Yan Ye, and Seung Hwan Kim. Algorithm description for versatile video coding and test model 8 (vtm 8). *Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-Q2002-v3*, 2020. 10

[46] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. 11

[47] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009. 11

[48] Antonio Torralba, Rob Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 11

[49] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 11

[50] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.

[51] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006. 11

[52] Vijay Chandrasekhar, Gabriel Takacs, David Chen, Sam S Tsai, Jatinder Singh, and Bernd Girod. Transform coding of image feature descriptors. In *Visual Communications and Image Processing 2009*, volume 7257, page 725710. International Society for Optics and Photonics, 2009. 11

[53] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016. 11

[54] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014. 11

[55] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*, pages 27–35. IEEE, 2015.

[56] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1556–1564. IEEE, 2015.

[57] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. *arXiv preprint arXiv:1504.03410*, 2015. 11

[58] Zhuo Chen, Weisi Lin, Shiqi Wang, Lingyu Duan, and Alex C Kot. Intermediate deep feature compression: the next battlefield of intelligent sensing. *arXiv preprint arXiv:1809.06196*, 2018. 12, 43, 45

[59] Zhuo Chen, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Toward intelligent sensing: Intermediate deep feature compression. *IEEE Transactions on Image Processing*, 29:2230–2243, 2019. 12, 35, 38, 48, 55, 59

[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 12, 65

[61] Hyomin Choi and Ivan V Bajić. Deep feature compression for collaborative object detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3743–3747. IEEE, 2018. 12, 34, 39, 48, 55

[62] Hyomin Choi and Ivan V Bajić. Near-lossless deep feature compression for collaborative intelligence. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2018. 12, 34, 39, 48, 55

[63] Amir Erfan Eshratifar, Amirhossein Esmaili, and Massoud Pedram. Towards collaborative intelligence friendly architectures for deep learning. In *20th International Symposium on Quality Electronic Design (ISQED)*, pages 14–19. IEEE, 2019. 12

[64] Saeed Ranjbar Alvar and Ivan V Bajić. Multi-task learning with compressible features for collaborative intelligence. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1705–1709. IEEE, 2019. 12

[65] Amir Erfan Eshratifar, Amirhossein Esmaili, and Massoud Pedram. Bottlenet: A deep learning architecture for intelligent mobile cloud computing services. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019. 12

[66] Siwei Ma. Investigation report on ai-related coding techniques. *Audio Video Coding Standard (AVS) document AVS N2510*, 2017. 12

[67] Chen Zhuo, Fan Kui, Lin Weisi, Duan Lingyu, Kot Alex, C., and Huang Tiejun. The framework and test condition for lossy compression of deep feature maps. *Audio Video Coding Standard (AVS) document AI M1061*, 2018. 12, 57

[68] Yuan Zhang and Patrick Dong. Video coding for machine: Use cases. *AhG on VCM of ISO/IEC JTC1/SC29/WG11, N18662*, 2019. 12

[69] Ling-Yu Duan, Jiaying Liu, Wenhan Yang, Tiejun Huang, and Wen Gao. Video coding for machines: A paradigm of collaborative compression and intelligent analytics. *arXiv preprint arXiv:2001.03569*, 2020. 12

[70] Zhuo Chen, Ziqian Chen, Lingyu Duan, Weisi Lin, and Alex C Kot. Cnn feature map coding for cloud-based visual analysis. *AhG on VCM of ISO/IEC JTC1/SC29/WG11, m52162*, 2020. 12

[71] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 449–460. IEEE Computer Society, 2012. 15

[72] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang, and Huazhong Yang. A survey of fpga based neural network accelerator. *arXiv preprint arXiv:1712.08934*, 2017.

[73] Mircea Horea Ionica and David Gregg. The movidius myriad architecture's potential for scientific computing. *IEEE Micro*, 35(1):6–14, 2015. 15

[74] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 17, 25

[75] Jiuxiang Gu, Jianfei Cai, Gang Wang, and Tsuhan Chen. Stack-captioning: Coarse-to-fine learning for image captioning. *arXiv preprint arXiv:1709.03376*, 2017. 17, 25

[76] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 17, 25

[77] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 17, 25

[78] Vijay Chandrasekhar, Jie Lin, Olivier Morere, Hanlin Goh, and Antoine Veillard. A practical guide to cnns and fisher vectors for image instance retrieval. *Signal Processing*, 128:426–439, 2016. 17, 25

[79] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016. 17, 25

[80] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016. 17, 25

[81] Wikipedia. Nvidia tesla, 2021. URL `https://en.wikipedia.org/wiki/Nvidia_Tesla`. [Online; accessed 16-Jun-2021]. 19, 21

[82] Wikipedia. Nvidia geforce, 2021. URL `https://en.wikipedia.org/wiki/GeForce`. [Online; accessed 16-Jun-2021]. 19, 21

[83] Wikipedia. Nvidia jetson, 2021. URL `https://en.wikipedia.org/wiki/Nvidia_Jetson`. [Online; accessed 16-Jun-2021]. 20, 21

[84] Gyrfalcon Technology Inc. Lightspeeur 5801s, 2021. URL `https://www.gyrfalcontech.ai/solutions/lightspeeur-5801/`. [Online; accessed 16-Jun-2021]. 20

[85] Lingyu Duan, Yihang Lou, Shiqi Wang, Wen Gao, and Yong Rui. Ai oriented large-scale video management for smart city: Technologies, standards and beyond. *arXiv preprint arXiv:1712.01432*, 2017. 23

[86] Jean-loup Gailly, Mark Adler. Gzip, 2018. URL `https://www.gnu.org/software/gzip/`. [Online; accessed 28-August-2018]. 26

[87] Terry A. Welch. Technique for high-performance data compression. *Computer*, (52), 1984. 26

[88] Peter Deutsch. Deflate compressed data format specification version 1.3. Technical report, 1996. 26

[89] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977. 26

[90] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. 26

[91] Jean-loup Gailly, Mark Adler. Zlib, 2018. URL `https://zlib.net/`. [Online; accessed 28-August-2018]. 26

[92] Julian Seward. Bzip2, 2018. URL `https://en.wikipedia.org/wiki/Bzip2`. [Online; accessed 28-August-2018]. 26

[93] Igor Pavlov. Lempel–ziv–markov chain algorithm, 2018. URL `http://en.wikipedia.org/wiki/LZMA`. [Online; accessed 28-August-2018]. 26

[94] Lasse Collin. A quick benchmark: Gzip vs. *Bzip2 vs. LZMA*, 2005. 27

[95] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 30, 33

[96] David Flynn, Detlev Marpe, Matteo Naccari, Tung Nguyen, Chris Rosewarne, Karl Sharman, Joel Sole, and Jizheng Xu. Overview of the range extensions for the hevc standard: Tools, profiles, and performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):4–19, 2015. 33

[97] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 34

[98] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992. 34

[99] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 36

[100] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015. 40

[101] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 44

[102] Yihang Lou, Yan Bai, Jun Liu, Shiqi Wang, and Ling-Yu Duan. Embedding adversarial learning for vehicle re-identification. *IEEE Transactions on Image Processing*, 2019. 44, 45

[103] K. Simonyan and A. Zisserman. Ilsvrc-2014 model (vgg team) with 16 weight layers. https://gist.github.com/ksimonyan/211839e770f7b538e2d8. 44

[104] Xinlei Chen and Abhinav Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017. 44, 45

[105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. https://github.com/KaimingHe/deep-residual-networks. 45

[106] Frank Bossen. Common hm test conditions and software reference configurations (jctvc-l1100). *Joint Collaborative Team on Video Coding*, 2013. 49, 55

[107] Guangtao Zhai, Xiaolin Wu, Xiaokang Yang, Weisi Lin, and Wenjun Zhang. A psychovisual quality metric in free-energy principle. *IEEE Transactions on Image Processing*, 21(1):41–52, 2012. 62, 65

[108] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*, pages 1–6. IEEE, 2016. 63, 68, 69

[109] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008. 66

[110] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 66

[111] Andre Araujo and Bernd Girod. Large-scale video retrieval using image queries. *IEEE Transactions on CSVT*, 2017. 75, 76

[112] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *ACM international conference on Image and video retrieval*, pages 494–501, 2007. 76

[113] Cees Snoek, Kvd Sande, OD Rooij, Bouke Huurnink, J Uijlings, M van Liempt, M Bugalhoy, I Trancosoy, F Yan, M Tahir, et al. The mediamill trecvid 2009 semantic video search engine. In *TRECVID workshop*, 2009. 76

[114] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 76, 77, 78, 83, 84

[115] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010. 76

[116] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010. 76

[117] Zhongwen Xu, Yi Yang, and Alex G Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, pages 1798–1807, 2015. 77

[118] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014. 77

[119] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015. 77

[120] FC Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015. 77

[121] Zechao Li and Jinhui Tang. Weakly supervised deep metric learning for community-contributed image retrieval. *IEEE Transactions on Multimedia*, 17(11):1989–1999, 2015. 77

[122] Hyun Oh-Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016.

[123] Zechao Li and Jinhui Tang. Weakly supervised deep matrix factorization for social image understanding. *IEEE Transactions on Image Processing*, 26(1): 276–288, 2017. 77

[124] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005. 77

[125] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*, pages 3–20. Springer, 2016. 77, 85

[126] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, pages 1386–1393, 2014. 77

[127] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307, 2016. 77, 85

[128] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, pages 3279–3286, 2015. 77, 82, 85

[129] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961, 2016. 77

[130] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013. 77, 82, 85

[131] Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. Learning to rank question answer pairs with holographic dual lstm architecture. *arXiv preprint arXiv:1707.06372*, 2017. 77

[132] Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641, 1995. 79

[133] Zhe Wang, Kingsley Kuan, Mathieu Ravaut, Gaurav Manek, Sibo Song, and et al. Truly multi-modal youtube-8m video classification with video, audio, and text. *arXiv preprint arXiv:1706.05461*, 2017. 85