Toward intelligent sensing : intermediate deep feature compression

Chen, Zhuo; Fan, Kui; Wang, Shiqi; Duan, Lingyu; Lin, Weisi; Kot, Alex Chichung

2019

Chen, Z., Fan, K., Wang, S., Duan, L., Lin, W., & Kot, A. C. (2019). Toward intelligent sensing : intermediate deep feature compression. IEEE Transactions on Image Processing, 29, 2230-2243. doi:10.1109/TIP.2019.2941660

https://hdl.handle.net/10356/143979

https://doi.org/10.1109/TIP.2019.2941660

Intermediate Deep Feature Compression: Toward Intelligent Sensing

Zhuo Chen, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, *Fellow, IEEE,* and Alex C. Kot, *Fellow, IEEE*

***Abstract*—The recent advances of hardware technology have made the intelligent analysis equipped at the front-end with deep learning more prevailing and practical. To better enable the intelligent sensing at the front-end, instead of compressing and transmitting visual signals or the ultimately utilized top-layer deep learning features, we propose to compactly represent and convey the intermediate-layer deep learning features with high generalization capability, to facilitate the collaborating approach between front and cloud ends. This strategy enables a good balance among the computational load, transmission load and the generalization ability for cloud servers 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 both lossless and lossy deep feature compression, which provide meaningful investigations and baselines for future research and standardization activities.**

***Index Terms*—Deep learning, intelligent front-end, feature compression.**

I. Introduction

RECENTLY, deep neural networks (DNNs) have demonstrated the incomparable performance in various computer vision tasks, e.g., image classification [1], [2], [3], [4], image object detection [5], [6], visual tracking [7] and visual retrieval [8]. Different from handcrafted features, like Histogram of Oriented Gradient (HOG) [9] and Scale-Invariant Feature Transform (SIFT) [10], deep learning features are directly learned from masses of data. For image classification, which is the fundamental task of computer vision, AlexNet [1] has achieved 9% better classification accuracy than the previous handcrafted methods in the 2012 ImageNet competition [11], 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 [2] and GoogLeNet [12] announced promising performance in the ILSVRC 2014 classification challenge, which demonstrated that deeper and wider 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 [3].

With the advances of network infrastructure, cloud-based applications are springing up in recent years. In particular, the 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 front-end 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 Fig. 1. Many computer vision models powered by deep learning can be applied in such cloud-based paradigm, such as pedestrian detection [13], person [14] and vehicle re-identification [15] in surveillance systems; autopilot [16] and license plate recognition [17] with on-board devices; face recognition [18], [19], landmark retrieval [20] and object detection [5], [6] in portable device (e.g., mobile, smart glasses) applications.

For data communication between front-end and cloud sever, video compression and transmission serve as the foundation infrastructure in the traditional "compress-then-analyse" paradigm. In other words, the front-end 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 front-end 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" [21], 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 [22] and

(Corresponding authors: Ling-Yu Duan and Weisi Lin)

Z. Chen is with the Interdisciplinary Graduate School, Nanyang Technological University, Singapore;

K. Fan is with the School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, Shenzhen, China;

S. Wang is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong;

L.-Y. Duan is with the National Engineering Laboratory of Video Technology, the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China, and also with the Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: lingyu@pku.edu.cn);

W. Lin is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: WSLin@ntu.edu.sg);

A. C. Kot is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes a PDF file. Contact lingyu@pku.edu.cn and WSLin@ntu.edu.sg for further questions about this work.

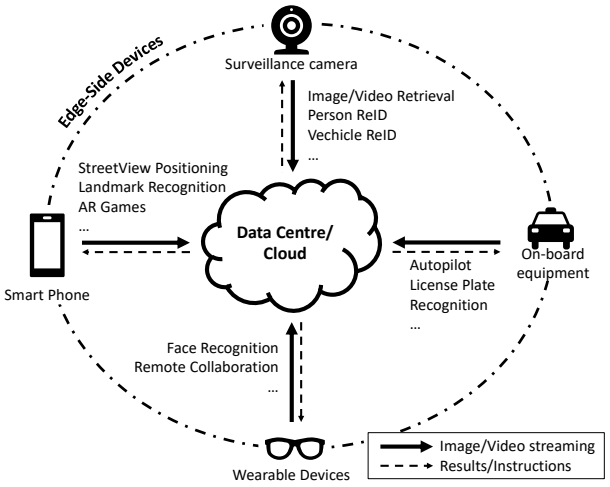


Fig. 1. Diagram of cloud-based visual analysis applications. Images and videos are acquired at the front end and the analysis is performed at the cloud end. The two sides collaborate together through data transmission.

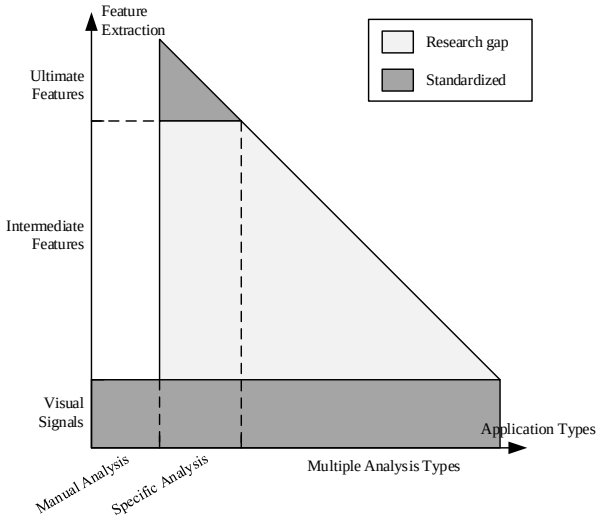


Fig. 2. The compression and transmission of visual signals and ultimate features has been widely investigated and standardized, but the study on transmitting intermediate features is limited which needs further exploration.

MPEG CDVA [23] 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 [19], 128 in Google FaceNet [24], and 300 in SenseTime DeepID3 [25]. 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. 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 this, the intermediate layer feature compression, which shifts the computational load while maintaining the availability of various visual analysis tasks is presented and analyzed in this paper. The presented approach 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.

The rest of the paper is organized as follows. Section II provides a brief review on the compact visual information representation, including video compression and feature compression. Section III describes our proposed collaborating approach for cloud-based visual analysis applications. In Section IV, we discuss and envision the future standardization of deep feature coding. Section V presents the evaluation results of lossless deep feature compression, and Section VI shows the results of lossy compression. Finally, Section VII concludes this paper.

II. Related works

In cloud-based visual analysis tasks, the bitstream transmitted between the edge side and cloud side can be either visual signals or features. As shown in Fig. 2, visual signal can be utilized by all analysis applications, including manual monitoring based on human viewing, as the visual signal is the origin of feature extraction. On the contrary, the ultimately utilized features (it will be denoted as ***"ultimate feature"*** in the following for convenience) can serve specific applications well but lose the generalization capability to deal with other analysis tasks. The transmission and compression for both visual signals and handcrafted ultimate features have been well explored and standardized.

Video Coding Standard: High Efficiency Video Coding (HEVC) [26] is the state-of-the-art video coding standard, which achieves 50% bit-rate reductions for equal perceptual visual quality comparing to H.264/MPEG 4 Advanced Video Coding (AVC) [27]. As a joint video project of ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), the standardization of HEVC was finalized in Jan. 2013. As a video coding standard, HEVC only specifies the decoder. In other words, the decoder conforming to the standard can correctly reconstruct the video based on the bitstream, and the encoder can be feasibly optimized according to the application scenarios and requirements. HEVC can be applied to both image and video in lossy and lossless ways. Recently, in Apr. 2018, the standardization for new generation video coding, Versatile Video Coding (VVC) [28], was launched. It is expected to be completed before 2020, with much superior coding performance compared to HEVC.

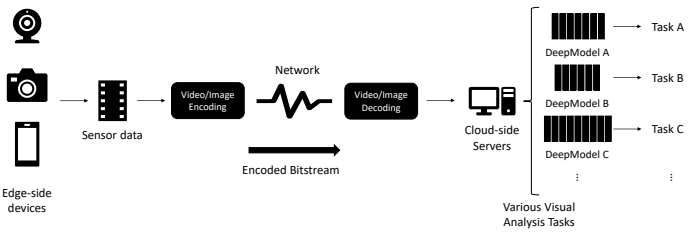
Standardization of Ultimate Features: To provide a standardized bitstream syntax to enable interoperability in the context of image retrieval applications, MPEG published Compact Descriptors for Visual Search (CDVS) [22] in Sep. 2015. CDVS leverages handcrafted local (i.e., SIFT descriptors) and global (i.e., Scalable Compressed Fisher Vector) features to represent the visual characteristics of images. To achieve compact image representation while maintaining the discrimination capability, a series of compression techniques were developed. In particular, with the process of local feature selection, descriptor compression, location compression and descriptor aggregation, CDVS enables interoperability among six different feature sizes from 512B to 16KB.

Based on CDVS, MPEG has moved forward to the standardization of Compact Descriptors for Video Analysis (CDVA) [23] since Feb. 2015. Considering the fact that extracting features frame-by-frame will result in extremely high computational costs and redundancy in the video representations, multi-keyframe based retrieval strategy was adopted by the ongoing CDVA standard. More specifically, to generate compact video descriptors, both local and global descriptors of sampled keyframes are firstly extracted by standardized CDVS. Then, the CDVA descriptors are constituted by compressing and packing these frame-level features. Furthermore, deep learning features were also adopted into the working draft of CDVA to further boost the retrieval performance [8].

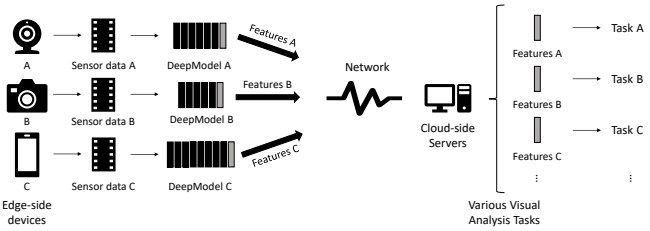
In [29], [30], the joint texture and feature compression strategies were studied, such that the compressed texture and feature can be transmitted simultaneously, and the interactions between them have also been considered. Moreover, the compact representation of deep learning ultimate features (i.e. features from the last layer of neural networks) has also been widely investigated in the literature.

Compact Deep Representations: In computer vision, visual embeddings from deep neural networks have been widely used. To achieve compact and discriminative representations, existing methods can be classified into two categories. The first one aims to design the deep models with small-size embedding layers before training, and the other targets to add a series of dimension-reduction/binarization layers (e.g., hashing and PCA) on top of the trained deep learning models. For the first category, the work in [24] explored the effect of changing embedding layer size in deep face recognition models, and better image retrieval performance can be achieved with smaller embedding size by tailoring the CNN architecture [31]. For the second category, the authors in [32] applied PCA compression on the top layer representations of a pre-trained CNN to achieve state-of-the-art accuracy on a number of image retrieval datasets. Moreover, hashing also plays an important role in deep embedding compression, and different hashing methods on the top of deep neural networks have been investigated [33], [34], [35], [36].

In contrast to the visual signal and the handcrafted feature compression, there are much fewer works studying the transmission and compression of deep learning intermediate features. The comparisons of intermediate features, ultimate features and visual signals are shown in Fig. 2. The recent works [37], [38] conducted deep feature compression on



(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.

Fig. 3. Two commonly used strategies for cloud-based visual analysis.

two specific types of intermediate features in the context of collaborative intelligence and image object detection. In particular, the work in [37] employed HEVC Range extension (REext) to compress deep features extracted by two specific layers (i.e. Max11 and Max17) of the YOLO9000 network. Subsequently, the authors in [38] proposed a near-lossless deep feature compressor and evaluated the performance on four deep networks. However, general deep feature compression should cover different types of deep features from off-the-shelf deep neural networks. There are several differences between this work and [37], [38]. Firstly, we propose to balance the computational load and feature usability in a more generalized way for intermediate features instead of focusing on a specific task. Secondly, both lossless and lossy compression on deep features are studied, while in [37], [38] only the lossy compression results are reported. Thirdly, for the lossy compression methods, feature maps are combined into image by tiling and quilting before applying off-the-shelf image/video encoder in [37], [38], and we compose the feature maps into video sequences which can be more flexibly compatible with varied feature channels.

III. 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 Fig. 3.

As the most conventional paradigm, the visual signal compression and transmission methods have been well developed

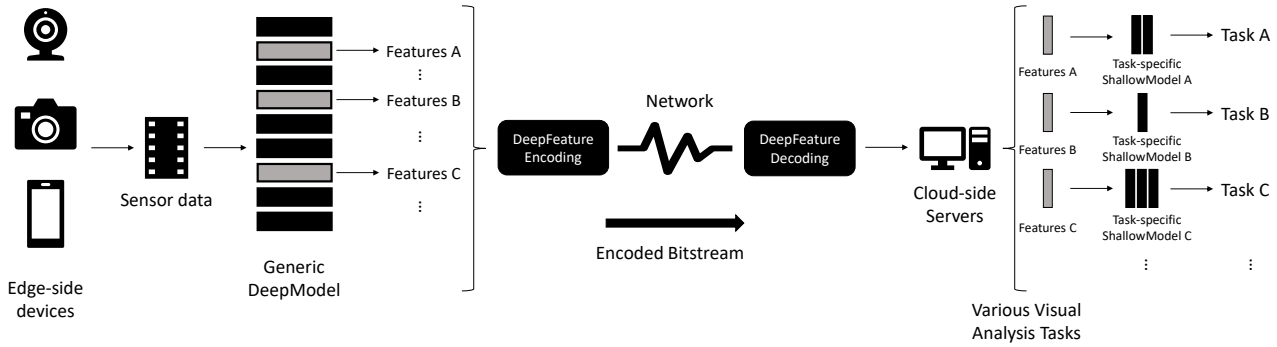


Fig. 4. Diagram of the proposed approach. The intermediate-layer features of a generic deep model can be applied to a broad range of tasks. The features of specific layers will be transmitted based on the analysis requirements on the cloud side. On top of these transmitted features, shallow task-specific models will be applied for visual analysis.

and standardized. As shown in Fig. 3(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 [6], which is the best performance ever reported to our 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 [39], [40], [41], 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 Fig. 3(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 [22], [23]. 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 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.

Therefore, the 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 Fig. 4, 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 Fig. 5. For instances, in image captioning tasks, the work [42] leverages the $conv5$ features (the output feature maps of the fifth convolutional block, we use the shorthands for convenience in the rest of this paper, more details can be found in Table II) of VGGNet to represent given images. The authors in [43] encoded the full image with the ResNet to extract both spatial and semantic information from its $conv4$ layer. In visual tracking tasks, $pool4$ and $pool5$ features of VGGNet were employed in [7]. In image object detection tasks, the work in [5] used the $fc2$ features and $pool5$ feature of VGGNet was employed in [44], [45]. In visual retrieval, the $pool5$ features of VGGNet were modified to introduce translation, scale and rotation invariances for image retrieval [8]. Handcrafted features and $fc1$ features of VGGNet were combined to achieve better retrieval performance [46]. In image QA tasks, the $conv4$ feature of ResNet was leveraged as the visual representation [47], and $pool5$ features of VGGNet and $conv5$ features of ResNet were used in [48]. 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 ($conv4$ or higher) as their input. As the computing load are mainly laid on low layers in neural networks (as shown in Table I), it can help saving great computing cost for the server-end with our proposed strategy. Thus, the deployment of our 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, our proposed approach will have more advantages over the former ones.

IV. 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 II.

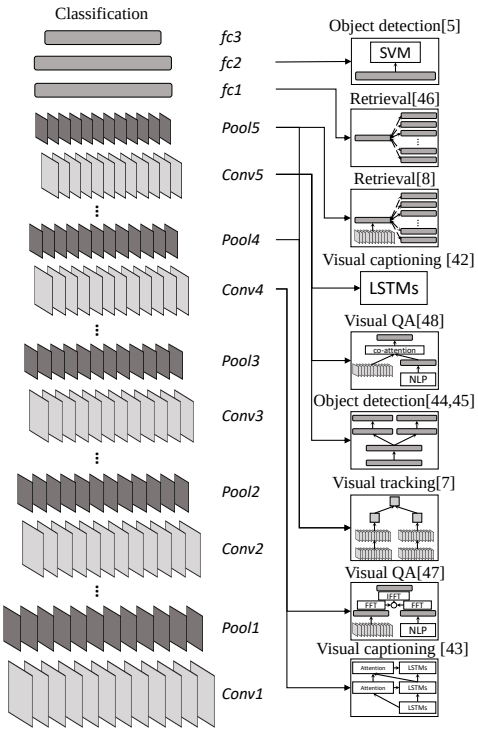


Fig. 5. Generic deep models (e.g. VGGNet and ResNet trained on ImageNet classification task) are widely employed as backbone networks in many computer vision tasks. Task-specific networks are designed on top of the intermediate layers of generic models.

TABLE I
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	$15.47G$	$3.86G$

As such, to optimize the bandwidth, compression for deep features is necessary.

A. 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 paper. 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 the outputs of neuron clusters at previous convolutional layer into a single neuron. The output of last pooling layer in $\mathbf{i}-$th block is denoted as $pool\mathbf{i}$ 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 neurons in 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 $\mathbf{i}-$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 paper.

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 II lists four milestone CNN architectures in image classification tasks, including AlexNet [1], VGGNet [2], ResNet [3] and DenseNet [4]. 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 to the task-related semantic space, which will largely erase 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 of these benchmark CNNs use ReLU as the activation function, which constrains the numerical distribution of deep features in a similar range. This property is useful for the deep feature compression.

B. 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, as mentioned in Section II. It is envisioned that our 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 [22], [23]. 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. [49]. 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 IV-A. 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

TABLE II
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

further investigated during the standardization explorations.

V. 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.

A. 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.

1) Deep Learning Models and Datasets: In this paper, 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 III, for many computer vision applications, the task-specific models are designed on top of the features of VGGNets and ResNets, such as image captioning [42], [43], visual tracking [7], image object detection [5], [44], [45], [6], visual retrieval [8], [46], image QA [47], [48].

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 [11]. 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.

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 [50] was developed in the early 1990's as a replacement for patent-encumbered algorithms such as LZW [51]. The DEFLATE algorithm [52] is the core of GZIP, which employs LZ77 [53] followed by Huffman coding [54]. The GZIP algorithm enjoys very fast compression speed and small memory footprint.

ZLIB: ZLIB [55] 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 [56] 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) [57] uses a dictionary compression scheme, similar to LZ77 [53], followed by a range encoder. Comparing to LZ77, the dictionary compressor is with huge dictionary sizes (up to 4 GB). The range encoder employs a complex mechanism to make probability predictions of each bit. LZMA features a generally high compression ratio with a comparable speed [58].

B. 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

$$Compression\ rate = \frac{data\ volume\ after\ compression}{data\ volume\ before\ compression}. \quad (1)$$

In this paper, 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 III to VI for VGGNet-16, ResNet-50, ResNet-101, ResNet-152. Visualized results are also presented in supplemental material as Fig. 8.

From Tables III to VI 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., $conv1$ of VGGNet with 12.25MByte) and small-volume features (e.g., $pool5$ 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., $conv4$-$pool5$ of VGGNet and $conv1$, $pool1$, $conv4$, $conv5$ of ResNet). GZIP takes less time than LZMA and BZIP2 in most cases. However, regarding $pool2$, $pool3$ of VGGNet and $conv2$, $conv3$ 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 $pool5/fc1$ of ResNets, while ZLIB performs better on $fc3$ of VGGNet and $pool5/fc1$ 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 IV-A. 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 III to VI, 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.

TABLE III
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	$2.007m \pm 0.090m$

TABLE IV
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
$conv1$	$112 \times 112 \times 64$	$3.0625M$	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
$pool1$	$56 \times 56 \times 64$	$784K$	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
$conv2$	$56 \times 56 \times 256$	$3.0625M$	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
$conv3$	$28 \times 28 \times 512$	$1568K$	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
$conv4$	$14 \times 14 \times 1024$	$784K$	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
$conv5$	$7 \times 7 \times 2048$	$392K$	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
$pool5$	$1 \times 1 \times 2048$	$8K$	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 V
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
$conv1$	$112 \times 112 \times 64$	$3.0625M$	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
$pool1$	$56 \times 56 \times 64$	$784K$	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
$conv2$	$56 \times 56 \times 256$	$3.0625M$	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
$conv3$	$28 \times 28 \times 512$	$1568K$	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
$conv4$	$14 \times 14 \times 1024$	$784K$	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
$conv5$	$7 \times 7 \times 2048$	$392K$	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
$pool5$	$1 \times 1 \times 2048$	$8K$	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 VI
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
$conv1$	$112 \times 112 \times 64$	$3.0625M$	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
$pool1$	$56 \times 56 \times 64$	$784K$	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
$conv2$	$56 \times 56 \times 256$	$3.0625M$	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
$conv3$	$28 \times 28 \times 512$	$1568K$	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
$conv4$	$14 \times 14 \times 1024$	$784K$	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
$conv5$	$7 \times 7 \times 2048$	$392K$	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
$pool5$	$1 \times 1 \times 2048$	$8K$	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	$1.986m \pm 0.037m$

C. 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 [59], 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.

VI. EVALUATIONS ON LOSSY COMPRESSION OF INTERMEDIATE DEEP LEARNING FEATURES

As discussed in the previous section, the lossless compression methods can hardly provide high compression ratio, which is not desirable for practical applications. In this section, we present lossy compression results of intermediate deep learning features to show the potential.

A. Compression Methods

In CNNs, the feature of a convolutional layer is in the form of feature maps which is a combination of stacked 2-D arrays with spatial correlations among the elements. Intuitively, one 2-D feature map can be consider as a frame, while the feature of the convolutional layer can be consider as the video

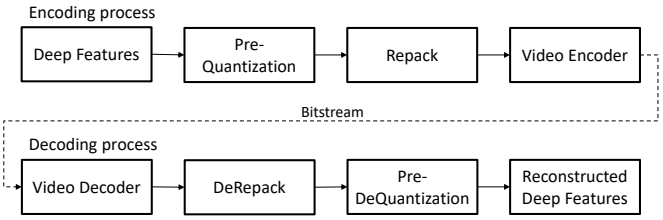


Fig. 6. Flow chart of lossy compression for intermediate deep learning feature maps.

sequence. As such, existing video codecs can be applied to compress deep features in the lossy manner [37], [38]. In this paper, we conduct the lossy compression experiments with the video codec based compression method. Fig. 6 describes the flow of the compression method.

In the encoding phase, pre-quantization module is first applied to convert the floating point deep learning feature data to integers. It is necessary since that deep learning features, like the vanilla VGGNets and ResNets features, are in $float32$ format, which are not compatible with the desired input format of most video codecs. For instance, HEVC and AVC require 8-bit (or higher) integers as the input. In view of that the intermediate deep feature generally has a right-skewed exponential distribution with a wide data span (histograms of intermediate deep features are presented in the supplemental material as Fig. 7), we quantize the features to 8-bit precision with logarithmic sampling in this paper. The quantization and corresponding dequantization (i.e. inverse quantization) are performed as

$$X_{quant} = round(\frac{log_B(X - min(X) + 1)}{max(log_B(X - min(X) + 1))} \cdot (2^{bitdepth} - 1)) \quad (2)$$

$$X_{dequant} = 2^{\frac{X_{quant} \cdot max(log_B(X - min(X) + 1))}{2^{bitdepth} - 1}} + min(X) - 1 \quad (3)$$

where X can be the feature tensor of a certain input from any specific layer of the neural network; X_{quant} and $X_{dequant}$ are the corresonding quantized and dequantized feature tensor; $round(\cdot)$ rounds the input float value to the nearest integer; B is the base of logarithm which can be any real number; and $bitdepth$ denotes the bit depth of the quantized integers, we set it as 8 in this paper.

After quantization, integer feature maps $\mathbb{N}_0^{H \times W \times C}$ will then be repacked to YUV 4:0:0 format $\mathbb{N}_0^{H' \times W' \times C}$ to feed the video encoder. Pratically, the height H and width W of feature maps will be extended to H' and W', by padding after the last array element along each dimension with repeating border elements, to fullfill the frame size requirement that the input height and width should be integral multiple of 8, where $H' = \lceil H/8 \rceil \times 8$ and $W' = \lceil W/8 \rceil \times 8$. Each feature map will be then considered as a frame in the yuv format data. It is worth noting that the order of the frames can be reorganized during the repack phase, which may affect the compression performance if inter-frame correlations are considered. In this paper, we only apply intra coding while the frame order reorganization is not investigated.

The repacked data are then compressed with the video encoder. We adopt the reference software (HM16.12) of HEVC Range extension (RExt) to conduct the experiment. To evaluate on a broad range of bitrate condition, we test the compression performance and accuracy loss with five quantization parameter (QP) values, i.e., $[0, 12, 22, 32, 42]$.

The decoding phase is an inverse of the encoding flow, the bitstream is decoded by video decoder, DeRepack module and DeQuantization module in sequence. The reconstructed deep feature maps will be further passed to their birth-layer in 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.

Different from the feature maps, feature vectors from the fc layers do not have the 2-D map structure. Therefore, video codec based lossy compression methods cannot be applied to the feature vectors. Instead of repacking and passing the feature to the video codec, we apply lossless compression on the quantized feature vectors. As LZMA performs well in Section V, we adopt LZMA to further eliminate the redundancy in the quantized feature data. It is worth noting that the upper and lower bound values of each feature sample (i.e. $max(log_2(X - min(X) + 1))$ and $min(X)$ in Eq. 2 and Eq. 3) are also included in the bitstream for further dequantization.

B. Results

The experimental settings of lossy compression are basically identical with the lossless compression experiments. Deep features are extracted from different layers of four deep models on a subset of ImageNet dataset which is mentioned in Section V-A1. The extracted deep features are then compressed with lossy compression methods described in Section VI-A.

In contrast with lossless compression which introduces no information loss to reconstructed features, the lossy compression reduces the data volume by eliminating less important information which may result in performance loss of corresponding deep learning models. As such, in this section, we evaluate the lossy compression performance in both terms of compression rate and information loss. For the compression rate, we follow Eq. 1 to obtain the compression ratio. As to the information loss, we calculate the fidelity by comparing the pristine DNN outputs with the outputs inferred from the reconstructed intermediate deep features, as below

$$Fidelity = 1 - \frac{1}{2N} \sum_{i}^{N} \frac{Hamming(Y_i, Y_i')}{length(Y_i)} \quad (4)$$

where Y_i is the onehot vector (output result of a deep learning model) inferred with i-th test image sample, Y_i' is the onehot vector inferred with the corresponding reconstructed deep feature, $length(\cdot)$ returns the dimension of input, N denotes the total number of tested samples. There is a negative relationship between fidelity and information loss. Namely, higher fidelity values tend to be associated with lower information loss, and vise verse.

Tables VII to X list lossy compression results on VGGNet-16, ResNet-50, ResNet-101, ResNet-152 respectively. Visualized results are also presented in supplemental material as Fig. 9. From the results, we can see that compression rates of lossy methods get dramatically improved comparing with lossless methods. The mean compression rates of lossy methods over tested feature vectors and feature maps on QP12 are around 0.205 and 0.140 respectively, while the lossless methods can only provide around 0.777 and 0.469 in the best cases (i.e. with LZMA). Beyond that, for lossy compression on feature maps, with the QP value increasing, the bitstream can be even more compact. For instance, the compression rate for $conv1$ of VGGNet-16 is 0.116 on QP12, while it gets more than 5x smaller (i.e. 0.020) on QP42. However, increasing QP values will also result in loss of information. For example, for VGGNet $conv1$ features, the fidelity of the reconstructed features on QP12 is 0.996 which denotes that the discriminative capability of the features is not significantly affected during the lossy compression process, while the fidelity on QP42 is only 0.839 which means almost 20% of the reconstructed features are mismatched with the pristine ones.

Since feature vectors are compressed without video codecs, the compression performance on feature vectors will not be affected by the quantization parameter (QP) as shown in the tables. Only quantized by the straightforward 8-bit quantization, the lossy compression for feature vectors are with very high fidelity while the data volume is nearly 4x smaller than the result of lossless methods. In particular, the fidelity on final layer features (i.e. $fc3$ of VggNet and $fc1$ of ResNets) are slightly lower than which on others. It may be because the elements of last layer features are directly associated with the final classification results, which suppresses the tolerability for the information loss.

Regarding feature maps, we can see that applying quantization (i.e., the cases other than QP0) inside video codec can generally result in greater compression ratio. However, for higher layer features, such as $conv5$, $pool5$ of VGGNet and $conv5$ of ResNets, the difference of compression ratio between with (i.e., QP12) and without (i.e., QP0) quantization is not significant comparing with the lower layer features. It may be due to that high layer features are usually abstract and dense, and in this case, feature maps are relatively smooth and highly spatially correlated. The existing intra prediction scheme can remove the spatial redundancy efficiently, and only little energy remains in the residual signal accordingly. In this context, quantization with small QP can hardly have effect on the residuals. As a result, the compression ratio will not change too much. When enlarging QP values, compression ratio increases along with the degradation of the fidelity. From the experimental results, it can be observed that QP22 generally provides a good trade-off between the compression rate and fidelity. Comparing to QP12, the compressed bitstream at QP22 is around 1.5x smaller while the information loss does not significantly increase. The deep features compressed at QP32 enjoy 2x smaller volume than QP22, but the quality of the feature is out of control, especially on low layer features like $conv1$ and $pool1$ of ResNet. It can also be observed that, the fidelity of some features does not change much with the QP values, like $conv5$ of ResNets. They can be compressed to extremely low volume with high QP values, while maintaining most of useful information. As such, we can expect that lossy compression methods with adaptive parameter selection will be more effective to achieve a better trade-off between the compression rate and information loss.

In summary, lossy deep feature compression methods are more promising to compress the feature data into smaller volume than the lossless methods. However, lossy methods will also introduce information loss where the lossless methods will not. The compression parameters, like the QP value in this paper, can be adjusted to control the trade-offs between the compression rate and the fidelity of the compressed deep feature.

VII. Conclusions

We have investigated a new strategy that exploits the redundancy of intermediate deep learning features instead of visual signal or ultimately utilized features. The advantage of this strategy 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. We further conducted comprehensive lossless and lossy compression evaluations on deep features of four widely used neural networks. As the first attempt to the problem, the proposed strategy and the evaluation results in this paper provide a good reference for further studies and investigations along this vein.

Acknowledgment

This research is supported by the NTU-PKU Joint Research Institute, a collaboration between the Nanyang Technological University (NTU), Singapore, and Peking University (PKU), China, which is sponsored by a donation from the Ng Teng Fong Charitable Foundation. The research work was done at the Rapid-Rich Object Search (ROSE) Lab at NTU. This work is also supported in part by Singapore Ministry of Education Tier-2 Fund MOE2016-T2-2-057(S), in part by the National Natural Science Foundation of China under Grant 61661146005 and Grant U1611461, in part by Hong Kong RGC Early Career Scheme 9048122 (CityU 21211018), in part by City University of Hong Kong under Grant 7200539/CS, and in part by the National Research Foundation, Prime Minister's Office, Singapore, through the NRF-NSFC Grant, under Grant NRF2016NRF-NSFC001-098.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

TABLE VII
LOSSY FEATURE COMPRESSION RESULTS OF VGGNET-16.

Feat. Maps	QP0		QP12		QP22		QP32		QP42	
	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity
$conv1$	0.148 ± 0.022	0.997	0.116 ± 0.022	0.996	0.080 ± 0.018	0.985	0.048 ± 0.013	0.955	0.020 ± 0.007	0.839
$pool1$	0.180 ± 0.025	0.997	0.145 ± 0.024	0.994	0.099 ± 0.019	0.984	0.057 ± 0.013	0.914	0.023 ± 0.006	0.693
$conv2$	0.150 ± 0.015	0.997	0.130 ± 0.016	0.992	0.098 ± 0.013	0.972	0.066 ± 0.010	0.952	0.035 ± 0.007	0.790
$pool2$	0.214 ± 0.022	0.997	0.185 ± 0.020	0.995	0.138 ± 0.017	0.982	0.090 ± 0.012	0.947	0.047 ± 0.007	0.745
$conv3$	0.114 ± 0.013	0.997	0.102 ± 0.013	0.995	0.080 ± 0.011	0.986	0.057 ± 0.008	0.960	0.034 ± 0.005	0.840
$pool3$	0.196 ± 0.021	0.997	0.179 ± 0.020	0.989	0.140 ± 0.016	0.981	0.102 ± 0.012	0.955	0.063 ± 0.007	0.819
$conv4$	0.070 ± 0.008	0.998	0.065 ± 0.007	0.992	0.053 ± 0.006	0.984	0.041 ± 0.005	0.967	0.028 ± 0.004	0.865
$pool4$	0.164 ± 0.018	0.998	0.160 ± 0.017	0.992	0.127 ± 0.014	0.974	0.097 ± 0.011	0.969	0.065 ± 0.008	0.864
$conv5$	0.060 ± 0.009	0.998	0.059 ± 0.009	0.997	0.046 ± 0.007	0.989	0.037 ± 0.005	0.969	0.023 ± 0.002	0.920
$pool5$	0.162 ± 0.018	0.998	0.162 ± 0.017	0.995	0.129 ± 0.013	0.986	0.106 ± 0.009	0.967	0.075 ± 0.005	0.908

Feat. Vectors	Comp. Rate	Fidelity
$fc1$	0.112 ± 0.016	1.000
$fc2$	0.116 ± 0.021	1.000
$fc3$	0.245 ± 0.005	0.970

TABLE VIII
LOSSY FEATURE COMPRESSION RESULTS OF RESNET-50.

Feat. Maps	QP0		QP12		QP22		QP32		QP42	
	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity
$conv1$	0.122 ± 0.021	0.996	0.091 ± 0.021	0.983	0.054 ± 0.017	0.945	0.026 ± 0.010	0.841	0.007 ± 0.003	0.441
$pool1$	0.126 ± 0.020	0.996	0.097 ± 0.020	0.980	0.058 ± 0.016	0.923	0.024 ± 0.008	0.744	0.006 ± 0.002	0.122
$conv2$	0.179 ± 0.009	0.999	0.148 ± 0.011	0.997	0.103 ± 0.010	0.991	0.059 ± 0.008	0.952	0.017 ± 0.003	0.812
$conv3$	0.198 ± 0.008	1.000	0.168 ± 0.009	0.997	0.121 ± 0.008	0.993	0.073 ± 0.008	0.967	0.023 ± 0.003	0.884
$conv4$	0.210 ± 0.006	1.000	0.186 ± 0.006	0.998	0.136 ± 0.006	0.991	0.080 ± 0.005	0.965	0.028 ± 0.002	0.883
$conv5$	0.180 ± 0.016	1.000	0.175 ± 0.015	0.999	0.136 ± 0.011	0.999	0.104 ± 0.008	0.987	0.063 ± 0.002	0.962

Feat. Vectors	Comp. Rate	Fidelity
$pool5$	0.216 ± 0.014	0.998
$fc1$	0.243 ± 0.005	0.979

TABLE IX
LOSSY FEATURE COMPRESSION RESULTS OF RESNET-101.

Feat. Maps	QP0		QP12		QP22		QP32		QP42	
	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity
$conv1$	0.111 ± 0.019	0.995	0.081 ± 0.019	0.981	0.048 ± 0.015	0.949	0.022 ± 0.009	0.817	0.006 ± 0.003	0.400
$pool1$	0.112 ± 0.017	0.995	0.085 ± 0.018	0.985	0.049 ± 0.013	0.923	0.020 ± 0.006	0.718	0.005 ± 0.001	0.081
$conv2$	0.174 ± 0.008	0.997	0.143 ± 0.010	0.994	0.100 ± 0.009	0.985	0.056 ± 0.008	0.948	0.016 ± 0.003	0.783
$conv3$	0.205 ± 0.007	0.999	0.174 ± 0.007	0.997	0.127 ± 0.007	0.996	0.077 ± 0.007	0.968	0.025 ± 0.004	0.844
$conv4$	0.260 ± 0.007	0.999	0.224 ± 0.007	0.998	0.164 ± 0.007	0.994	0.097 ± 0.007	0.984	0.032 ± 0.002	0.904
$conv5$	0.173 ± 0.018	1.000	0.168 ± 0.016	0.997	0.130 ± 0.012	0.992	0.100 ± 0.008	0.989	0.063 ± 0.002	0.953

Feat. Vectors	Comp. Rate	Fidelity
$pool5$	0.211 ± 0.016	0.999
$fc1$	0.243 ± 0.005	0.982

TABLE X
LOSSY FEATURE COMPRESSION RESULTS OF RESNET-152.

Feat. Maps	QP0		QP12		QP22		QP32		QP42	
	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity	Comp. Rate	Fidelity
$conv1$	0.097 ± 0.017	0.989	0.070 ± 0.017	0.982	0.041 ± 0.013	0.935	0.019 ± 0.008	0.818	0.005 ± 0.002	0.356
$pool1$	0.099 ± 0.016	0.989	0.074 ± 0.016	0.979	0.043 ± 0.012	0.937	0.017 ± 0.006	0.732	0.004 ± 0.001	0.087
$conv2$	0.169 ± 0.010	0.997	0.139 ± 0.011	0.995	0.095 ± 0.010	0.986	0.052 ± 0.008	0.957	0.014 ± 0.003	0.765
$conv3$	0.216 ± 0.007	1.000	0.184 ± 0.008	0.996	0.134 ± 0.008	0.989	0.083 ± 0.007	0.973	0.028 ± 0.004	0.854
$conv4$	0.268 ± 0.008	0.999	0.231 ± 0.007	0.997	0.170 ± 0.007	0.992	0.101 ± 0.008	0.977	0.034 ± 0.003	0.932
$conv5$	0.173 ± 0.017	1.000	0.168 ± 0.016	1.000	0.131 ± 0.012	0.998	0.100 ± 0.008	0.988	0.063 ± 0.002	0.961

Feat. Vectors	Comp. Rate	Fidelity
$pool5$	0.212 ± 0.015	1.000
$fc1$	0.243 ± 0.005	0.986

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[6] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[7] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3119–3127.

[8] J. Lin, L.-Y. Duan, S. Wang, Y. Bai, Y. Lou, V. Chandrasekhar, T. Huang, A. Kot, and W. Gao, "Hnip: Compact deep invariant representations for video matching, localization, and retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 1968–1983, 2017.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[13] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2056–2063.

[14] T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 1249–1258.

[15] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2167–2175.

[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[17] R. Polishetty, M. Roopaei, and P. 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*. IEEE, 2016, pp. 286–293.

[18] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[20] Y. Wang, X. Lin, L. Wu, and W. Zhang, "Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1393–1404, 2017.

[21] A. E. C. Redondi, L. Baroffio, L. Bianchi, M. Cesana, and M. Tagliasacchi, "Compress-then-analyze vs analyze-then-compress: what is best in visual sensor networks?" *IEEE Transactions on Mobile Computing*, vol. 15, no. 12, pp. 3000–3013, 2016.

[22] L.-Y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao, "Overview of the mpeg-cdvs standard," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 179–194, 2016.

[23] L.-Y. Duan, V. Chandrasekhar, S. Wang, Y. Lou, J. Lin, Y. Bai, T. Huang, A. C. Kot, and W. Gao, "Compact descriptors for video analysis: the emerging mpeg standard," *arXiv preprint arXiv:1704.08141*, 2017.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[25] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[26] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[27] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[28] B. Bross, J. Chen, and S. Liu, "Working draft 3 of versatile video coding," *Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-L1001-v9*, 2018.

[29] S. Ma, X. Zhang, S. Wang, X. Zhang, C. Jia, and S. Wang, "Joint feature and texture coding: Towards smart video representation via front-end intelligence," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[30] X. Zhang, S. Ma, S. Wang, X. Zhang, H. Sun, and W. Gao, "A joint compression scheme of video feature descriptors and visual content," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 633–647, 2017.

[31] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *European Conference on Computer Vision*. Springer, 2016, pp. 241–257.

[32] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *European conference on computer vision*. Springer, 2014, pp. 584–599.

[33] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning." in *AAAI*, vol. 1, 2014, p. 2.

[34] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*. IEEE, 2015, pp. 27–35.

[35] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 1556–1564.

[36] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," *arXiv preprint arXiv:1504.03410*, 2015.

[37] H. Choi and I. V. Bajic, "Deep feature compression for collaborative object detection," *arXiv preprint arXiv:1802.03931*, 2018.

[38] ——, "Near-lossless deep feature compression for collaborative intelligence," *arXiv preprint arXiv:1804.09963*, 2018.

[39] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2012, pp. 449–460.

[40] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of fpga based neural network accelerator," *arXiv preprint arXiv:1712.08934*, 2017.

[41] M. H. Ionica and D. Gregg, "The movidius myriad architecture's potential for scientific computing," *IEEE Micro*, vol. 35, no. 1, pp. 6–14, 2015.

[42] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[43] J. Gu, J. Cai, G. Wang, and T. Chen, "Stack-captioning: Coarse-to-fine learning for image captioning," *arXiv preprint arXiv:1709.03376*, 2017.

[44] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[45] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[46] V. Chandrasekhar, J. Lin, O. Morere, H. Goh, and A. Veillard, "A practical guide to cnns and fisher vectors for image instance retrieval," *Signal Processing*, vol. 128, pp. 426–439, 2016.

[47] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," *arXiv preprint arXiv:1606.01847*, 2016.

[48] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Advances In Neural Information Processing Systems*, 2016, pp. 289–297.

[49] L. Duan, Y. Lou, S. Wang, W. Gao, and Y. Rui, "Ai oriented large-scale video management for smart city: Technologies, standards and beyond," *arXiv preprint arXiv:1712.01432*, 2017.

[50] Jean-loup Gailly, Mark Adler, "Gzip," 2018, [Online; accessed 28-August-2018]. [Online]. Available: https://www.gnu.org/software/gzip/

[51] T. A. Welch, "Technique for high-performance data compression," *Computer*, no. 52, 1984.

[52] P. Deutsch, "Deflate compressed data format specification version 1.3," Tech. Rep., 1996.

[53] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.
[54] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.
[55] Jean-loup Gailly, Mark Adler, "Zlib," 2018, [Online; accessed 28-August-2018]. [Online]. Available: https://zlib.net/
[56] Julian Seward, "Bzip2," 2018, [Online; accessed 28-August-2018]. [Online]. Available: https://en.wikipedia.org/wiki/Bzip2
[57] Igor Pavlov, "Lempel–ziv–markov chain algorithm," 2018, [Online; accessed 28-August-2018]. [Online]. Available: http://en.wikipedia.org/wiki/LZMA
[58] L. Collin, "A quick benchmark: Gzip vs," *Bzip2 vs. LZMA*, 2005.
[59] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

Zhuo Chen received the B.S. degree from the School of Electronic and Information Engineering, Beijing Jiaotong University. He is currently pursuing the Ph.D. degree with the Rapid-Rich Object Search Lab, Nanyang Technological University, Singapore. Before that he was a Deep Learning engineer with LLVISION Pte. Ltd and a research assistant with National Astronomical Observatories, Chinese Academy of Sciences. His research interests include image processing, visual feature coding and deep learning.

Kui Fan received the B.E. and M.E. degrees from Beijing Jiaotong University (BJTU), China, in 2011 and 2014, respectively. He received the Ph.D. degree from Peking University, China in 2019. He was a visiting Ph.D. intern with Nanyang Technological University, Singapore from 2018 to 2019. His research interests include image and video compression, and multimedia processing. Since 2016, he has been a co-chair of ISO/IEC MPEG Internet Video Coding (IVC) group. He serves as the primary software coordinator for AVS3 standard since 2018. He is currently a senior researcher with Bytedance Inc., San Diego, CA, USA.

Shiqi Wang received the B.S. degree in computer science from the Harbin Institute of Technology in 2008, and the Ph.D. degree in computer application technology from the Peking University, in 2014. From Mar. 2014 to Mar. 2016, he was a Postdoc Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. From Apr. 2016 to Apr. 2017, he was with the Rapid-Rich Object Search Laboratory, Nanyang Technological University, Singapore, as a Research Fellow. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. He has proposed over 40 technical proposals to ISO/MPEG, ITU-T and AVS standards, and authored more than 170 refereed journal/conference papers. He received the Best Paper Award of IEEE Multimedia 2018, the Best Paper Award at the 2017 Pacific-Rim Conference on Multimedia (PCM), the Best Paper Award at the 2019 IEEE International Conference on Multimedia and Expo, and is the coauthor of a paper that received the Best Student Paper Award in IEEE International Conference on Image Processing 2018. His research interests include video compression, image/video quality assessment, and image/video search and analysis.

Lingyu Duan (M'06) is a Full Professor with the National Engineering Laboratory of Video Technology (NELVT), School of Electronics Engineering and Computer Science, Peking University (PKU), China, and has served as the Associate Director of the Rapid-Rich Object Search Laboratory (ROSE), a joint lab between Nanyang Technological University (NTU), Singapore, and Peking University (PKU), China since 2012. He is also with Peng Cheng Laboratory, Shenzhen, China, since 2019. He received the M.Sc. degree in automation from the University of Science and Technology of China, Hefei, China, in 1999, the M.Sc. degree in computer science from the National University of Singapore (NUS), Singapore, in 2002, and the Ph.D. degree in information technology from The University of Newcastle, Callaghan, Australia, in 2008. His research interests include multimedia indexing, search, and retrieval, mobile visual search, visual feature coding, and video analytics, etc. He received the IEEE ICME 2019 Best Paper Award and EURASIP Journal on Image and Video Processing Best Paper Award in 2015, the Ministry of Education Technology Invention Award (First Prize) in 2016, the National Technology Invention Award (Second Prize) in 2017, China Patent Award for Excellence (2017), the National Information Technology Standardization Technical Committee "Standardization Work Outstanding Person" Award in 2015. He was a Co-Editor of MPEG Compact Descriptor for Visual Search (CDVS) Standard (ISO/IEC 15938-13), and is serving as a Co-Chair of MPEG Compact Descriptor for Video Analytics (CDVA). Currently he is an Associate Editor of ACM Transactions on Intelligent Systems and Technology (ACM TIST) and ACM Transactions on Multimedia Computing, Communications, and Applications (ACM TOMM).

Weisi Lin (M'92–SM'98–F'16) received his Ph.D. from King's College, London University, U.K. He is a Professor in the School of Computer Science and Engineering, Nanyang Technological University. His areas of expertise include image processing, perceptual signal modeling, video compression, and multimedia communication, in which he has published 200+ journal papers, 230+ conference papers, filed 7 patents, and authored 2 books. He has been an AE for IEEE Trans. on Image Processing, IEEE Trans. on Circuits and Systems for Video Tech., IEEE Trans. on Multimedia, and IEEE Signal Processing Letters. He has been a Technical Program Chair for IEEE ICME 2013, PCM 2012, QoMEX 2014 and IEEE VCIP 2017. He has been an invited/panelist/keynote/tutorial speaker in 20+ international conferences, as well as a Distinguished Lecturer of IEEE Circuits and Systems Society 2016-2017, and Asia-Pacific Signal and Information Processing Association (APSIPA), 2012-2013. He is a Fellow of IET, and an Honorary Fellow of Singapore Institute of Engineering Technologists.

Alex Chichung Kot (F'06) has been with the Nanyang Technological University, Singapore since 1991. He headed the Division of Information Engineering at the School of Electrical and Electronic Engineering for eleven years and served as Associate Chair/ Research and Vice Dean Research for the School of Electrical and Electronic Engineering and eight years as Associate Dean for College of Engineering. He is currently Professor and Director of Rapid-Rich Object SEarch (ROSE) Lab and NTU-PKU Joint Research Institute. He has published extensively in the areas of signal processing for communication, biometrics, image forensics, information security and computer vision and machine learning.

Dr. Kot served as Associate Editor for more than ten journals, mostly for IEEE transactions. He has served the IEEE SP Society in various capacities such as the General Co-Chair for the 2004 IEEE International Conference on Image Processing and the Vice-President for the IEEE Signal Processing Society. He received the Best Teacher of the Year Award and is a co-author for several Best Paper Awards including ICPR, IEEE WIFS and IWDW. He was elected as the IEEE Distinguished Lecturer for the Signal Processing Society and the Circuits and Systems Society. He is a Fellow of IES, a Fellow of IEEE, and a Fellow of Academy of Engineering, Singapore.