Towards Reconfigurable Visual Feature Compression

Jiahang Zhang[1], Wenhan Yang[2], Minghao Liu[1], and Jiaying Liu*[1]

[1] Wangxuan Institute of Computer Technology, Peking University, Beijing, China
[2] Pengcheng Laboratory, Shenzhen, China
{zjh2020,liujiaying}@pku.edu.cn, yangwh@pcl.ac.cn, lmhlmh@stu.pku.edu.cn

Abstract. Recently, vision foundation models (VFMs) have demonstrated strong power for versatile downstream task analysis. This paper starts from the problem of how to efficiently deploy VFMs at the cloud *to support various user requests with arbitrary multi-task combinations that end in any scalable fashion*. To achieve feature data transmission between frontend-cloud, traditional feature coding employs the direct compression-reconstruction paradigm. However, this can result in potential redundancy: (1) task-irrelevant information, and (2) repeated coding of cross-task shared knowledge, leading to an inflexible and redundant scheme. To this end, we propose a novel feature coding paradigm, termed ***reconfigurable multi-task feature compression***, which aims to efficiently and adaptively compress the intermediate features to support the requested targeted tasks. Correspondingly, we propose a unified *R*econfigurable *F*eature *C*ompression framework, *RFC*, by feature factorization and recomposition. Specifically, the original feature is first factorized into multiple task-specific features with light-weight adapters. Then to efficiently compress these separate features, we consolidate them in a task-conditional auto-regressive manner, leveraging previously encoded task features as hyperprior conditions to reduce the redundancy of shared information in the current task feature. At the cloud side, a task-attentive recomposition module is further developed to fulfill the multi-task inference power within a single forward pass. Finally, we construct a comprehensive benchmark of reconfigurable feature compression to verify the effectiveness of RFC. Our project page can be found at `https://jhang2020.github.io/Projects/RFC/RFC.html`.

Keywords: Multi-task feature compression · Feature factorization

1 Introduction

Vision foundation models (VFMs) have achieved great advancements recently [19, 28, 29], demonstrating remarkable and versatile generalization ability with large-scale pretrained ViT-based backbones [10]. Through efficient fine-tuning, they

* Corresponding Author.

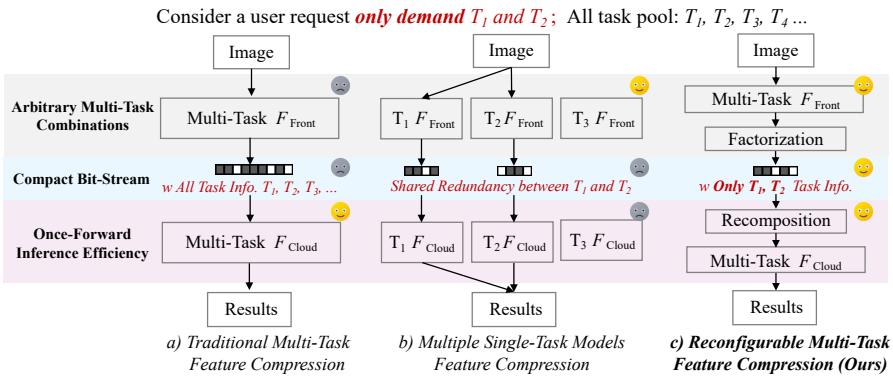

Fig. 1: Comparison of different multi-task feature compression paradigms in frontend-cloud collaboration. Our RFC can efficiently and flexibly support various user requests of arbitrary task subsets with novel factorization and recomposition, enabling *demand-aware transmission* with once-forward inference.

hold great promise for a wide range of intelligent applications, *e.g.*, traffic analysis [16] and smart home [22]. This highlights their potential as general-purpose perception engines in real-world applications.

Despite their strong capabilities, VFMs typically incur substantial computing overhead during inference, posing challenges for real-world deployment. To address this, a growing number of studies have recently explored a model-distributed manner [14, 33, 42] as a promising solution for VFMs deployment. These works mostly follow the end-cloud collaboration paradigm, where the whole model can be split into distributed parts, for the front end and cloud side, respectively. This inevitably yields massive data transmission cost from various users at the front end. In this context, feature compression aims to reduce the bitstream of the pre-extracted features from the front end, and develops techniques to achieve efficient data exchange, *e.g.*, transmission/storage. This approach exploits the computing resources on the front end and alleviates the computing burden on the cloud. Moreover, compared with the image transmission, the bit-rate of features is generally smaller with better privacy preserving, attracting a surge of interest from researchers.

Existing feature compression methods [11,33–35,42] primarily adopt a direct *intermediate feature reconstruction* manner, *i.e.*, decoding the transmitted bitstream back into intermediate features extracted from pre-trained deep models, without sufficient consideration of more practical and versatile feature compression. Specifically, 1) **At model level**, almost all previous methods focus on CNN-based small models, *e.g.*, ResNet-50 [20]. This ignores the strong power of current VFMs, especially the transformer-based features. For transformers, the features to be compressed are from more shallow layers due to the limited resources of front end, yielding new challenges for training and compression. 2) **At task level**, most works only study a single task for one model, *e.g.*, classification

or detection, leaving the versatile multi-task capacity of VFMs underexplored. 3) **At feature level**, current works directly perform simple bit-rate constrained reconstruction, without further analysis, *e.g.*, factorization and recomposition. This can lead to the problem of redundant coding caused by task-irrelevant information and the shared redundant information between multiple features.

To this end, we propose a new feature coding paradigm within the end-cloud collaboration category, ***reconfigurable multi-task feature compression***, designed to generate effective and compact feature bit-streams for arbitrary multi-task combinations, allowing more flexible coding targeted at versatile machine analysis. It has three advantages: *multi-task flexibility, smaller bit-rate*, and *inference efficiency* as shown in Figure 1. Specifically, in the context of multi-task end-cloud service, traditional multi-task feature compression directly reconstructs and then sends the intermediate feature to the cloud for analysis. However, it assumes that the users need to solve all tasks, which is relatively rare in real applications. This leads to the redundant bit-stream caused by the *task-irrelevant information*. Another solution is to deploy multiple single-task models to send on demand. However, this approach requires inefficient multi-model deployment and multiple forward inferences. Meanwhile, it also suffers from coding redundancy due to the shared information among different task features. In contrast, our reconfigurable feature compression aims to factorize the universal multi-task features into sub-features targeted at the demanded task group, achieving both flexibility and efficiency. In particular, for large-scale foundation model inference or service scenarios, our method enables demand-aware transmission adaptation and naturally supports asynchronous inference requests, thereby improving responsiveness and resource efficiency.

Technically, to deal with this challenging task, we present a unified framework in factorization-consolidation-recomposition manner. Specifically, we design the adapter-based feature factorization at the front end to obtain the task-specific feature components. Then, to efficiently compress these factorized features, we consolidate this separate information with a cross-task hyperprior conditioning mechanism. It performs information reuse in a task auto-regressive manner, achieving compact feature coding for all kinds of user requests. Finally, a task-attentive fusion module is developed at the cloud side to recompose multiple features into a single feature, enabling single-forward inference advantages of VFMs. A comprehensive large-scale benchmark for reconfigurable multi-task feature coding is established, demonstrating the significant advantages of our method. Our contributions can be summarized as follows:

- To the best of our knowledge, **we are the first to propose and formulate the problem of reconfigurable multi-task feature compression**, targeting compact and adaptive bitstreams for arbitrary multi-task combinations to enable flexible VFM deployment in end-cloud collaboration. A novel factorization-consolidation-recomposition paradigm is proposed to achieve both task-specific compactness and combination-agnostic adaptability within a unified framework, offering an efficient solution for serving various requests.

– At the front end, a parameter-isolated **adapter-based information bottleneck** is designed for feature factorization, to obtain atomic task-specific knowledge with lower bit-rate by reducing task-irrelevant information. At the cloud side, we propose a light-weight **task-attentive recomposition module** to achieve the fusion of multiple factorized features into a single feature with task affinity guidance. The entire pipeline supports the inference of arbitrary multi-task combinations via a single forward pass, and is readily compatible with existing compression standards and VFM inference frameworks.
– To further reduce the redundancy caused by the shared knowledge among different tasks, we present a novel **task auto-regressive consolidation coding mechanism**, which leverages previously encoded task features as hyperprior conditions to improve entropy encoding of current task features. In this way, our framework effectively reduces cross-task shared redundancy to achieve efficient feature coding of arbitrary targeted task combinations.

2 Related Work

2.1 Feature Compression / Coding

Different from visual signal coding [2, 6, 24], *e.g.*, image compression, which encodes and reconstructs the original visual data, feature coding [11, 23, 42] focuses on the extracted features for compression. The decoder-side decompresses the features for machine analysis or human vision, which is crucial in collaborative intelligence, *e.g.*, end-cloud collaboration paradigm [32].

Some works have explored intermediate feature compression using standard image/video codec [5, 30], *e.g.*, High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC). Generally, it cannot achieve desirable performance, due to the image/video-targeted codecs instead of features. Therefore, researchers have explored learned feature compression methods. Hyperprior models [3], widely used in image compression, are also introduced for feature compression. Omini-ICM [11] proposed to split the self-supervised pre-trained model at mid-layer to compress more general features. Finder [12] proposed a high-resolution convolution operation based on Haar-wavelet transform, achieving efficient feature compression. However, as discussed in Sec. 1, these works ignore the further feature factorization, leading to the redundant bit-stream of task-irrelevant information. Moreover, prior scalable/collaborative coding works [1, 7, 8, 17, 18] in both task flexibility and coding mechanism. Rather than relying on static dual-task architectures or predefined task dependencies (*e.g.*, machine vision $\subset$ human vision) that additively stack enhancement layers atop a fixed base, RFC subtractively factorizes semantically rich VFM features into compact, on-demand representations for arbitrary task subsets. Correspondingly, instead of exploiting cross-layer redundancy only along fixed base-to-enhancement or residual paths, RFC introduces a dynamic cross-task consolidation mechanism that flexibly reduces shared semantic redundancy across arbitrary task combinations.

2.2 Knowledge Decomposition

To effectively and efficiently leverage VFMs, various works have studied how to decompose large models and extract intrinsic knowledge. Some works focus on disentangled representation learning [4, 13]. The goal is typically to obtain the disentangled variables through adversarial learning or a variational auto-encoder. Besides, model editing methods [9, 43] change the knowledge to edit the model behavior. For example, Meng *et al*. [25] regarded the feed-forward-networks as a key-value knowledge memory, tracing the effect of specific neurons. Another popular approach is knowledge distillation [21], which aims to transfer the specific knowledge from large models to smaller student networks. Recently, knowledge factorization [36] has been proposed to learn multiple factor networks that are modularized and interpretable for further effective knowledge assembling. However, these methods are either constrained by complex learning strategies and module designs, or cannot be efficiently transferred to multi-task scenarios. In this paper, we jointly consider the feature factorization of VFMs in the end-cloud collaboration paradigm, as well as the subsequent compression and recomposition, providing a more comprehensive and practical framework.

3 Proposed Method: A Unified Framework

3.1 Preliminaries

Problem Formulation. Consider a multi-task pretrained vision foundation model $F(\cdot)$ (mainly focusing on Transformer-based models), targeted at N tasks, *i.e.*, $T = \{T_1, ..., T_N\}$. Following the canonical end-cloud collaboration feature coding paradigm, $F(\cdot)$ is split into a light-weight sub-network $F_{front}(\cdot)$ deployed at the front-end and the remaining tail network with N task heads $F_{cloud}(\cdot)$ at cloud side. We aim to develop an multi-task reconfigurable feature coding framework by addressing factorization, compression, and recomposition problems.

Specifically, for any front-end request on targeted tasks $T_Q \subseteq T$, the intermediate feature f_i (i denotes the i_{th} sample) output by $F_{front}(\cdot)$ is factorized to reduce irrelevant information, and then sent to the cloud side. In other words, **the sent feature should contain and only contain T_Q-relevant information to reduce the bit-rate as a compact and effective bit-stream**. This can be formulated as

$$\underset{F,G,C,D}{argmin} \sum_{\mathbf{T_Q} \subseteq \mathbf{T}} \sum_{\mathbf{t_k} \in \mathbf{T_Q}} \lambda \mathcal{L}_{t_k}(\widetilde{f}_i^{T_Q}, Y_i^{t_k}) + \mathcal{R}(f_i^{T_Q}) + \lambda_c \mathcal{C}, \tag{1}$$

where the first term is the loss objective of task t_k with ground-truth label $Y_i^{t_k}$ and the decompressed feature $\widetilde{f}_i^{T_Q} = D \cdot C(f_i^{T_Q})$; $f_i^{T_Q} = G(f_i)$ is the factorized feature map for T_Q tasks by a factorization module $G(\cdot)$. The second term $R(\cdot)$ denotes the bit-rate optimization function; the complexity term $\mathcal{C}$ is introduced to encourage inference efficiency, *i.e.*, changing as few parameters of

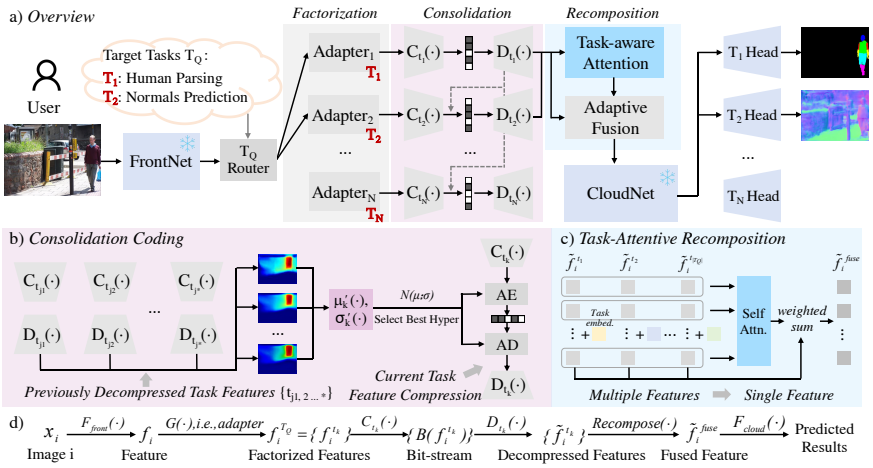

Fig. 2: a) is the overview of our proposed RFC. Specifically, given the target tasks T_Q, we factorize the original feature into task-specific features by T_Q-corresponding adapters *to reduce T_Q-irrelevant information*. Then, as shown in b), we consolidate these separate features with hyperprior-based cross-task conditioning, *to effectively compress them by reducing the coding redundancy of shared information*. Finally, a recomposition module detailed in c) is employed at the cloud to recompose the multiple features into a single feature *to enable once-forward inference efficiency*. d) is the main data flow with symbols.

$F_{front/cloud}(\cdot)$ as possible to retain the single-forward inference power and maintain generalization capacity of $F(\cdot)$. λ and λ_c are coefficients. A clearer data flow with symbols can be found in Fig. 2 (d).

Method Overview. A unified framework is proposed to tackle this challenging task, as shown in Fig. 2. It works in three steps: 1) *Factorization*: obtain the task-specific knowledge as composable feature components; 2) *Consolidation*: reduce the coding redundancy in compression of the factorized features; 3) *Recomposition*: recompose the multiple factorized features to derive the predictions with a single model forward. Details of each design are unfolded below.

3.2 Factorize: Adapter Information Bottleneck

In factorization, the goal is to derive compact features targeted at T_Q. However, it is an extremely challenging task because of the huge number of possible task combinations as T_Q, *i.e.*, $2^N - 1$. Directly learning a unified representation for all combinations is computationally intractable. Therefore, we adopt a **divide-and-conquer** strategy: atomic *factorization* for flexibility, followed by *consolidation* and *recomposition* for bitstream-level and inference-level efficiency (later Sec.).

For the factorization of task t_k, our optimization goal is

$$\mathcal{L}_{factor} = H(f_i|Y_i^{t_k}) - \beta I(f_i; Y_i^{t_k}). \tag{2}$$

The conditional entropy term $H(\cdot)$ reduces t_k-irrelevant information, while maximizing the second mutual information term $I(\cdot)$ is for maintaining the t_k-relevant information. β is a weight coefficient. Given f_i is the deterministic function of input image x_i, Eq. 2 can be reformulated as

$$\mathcal{L}_{factor} = I(x_i; f_i) - (\beta + 1)I(f_i; Y_i^{t_k}), \tag{3}$$

which aligns the formulation of Information Bottleneck theory [31] to derive the effective and compact representations. Therefore, we can alternatively optimize the following objective corresponding to Eq. 3 (full derivation in Supp. Sec. 3)

$$\mathcal{L}_{factor} = \mathcal{R}(f_i^{t_k}) + \lambda \mathcal{L}_{t_k}(\widetilde{f}_i^{t_k}, Y_i^{t_k}) + \lambda' ||f_i^{t_k} - \widetilde{f}_i^{t_k}||_2, \tag{4}$$

where $f_i^{t_k}$ is the t_k task feature factorized from f_i, output by a factorization module, and $\widetilde{f}_i^{t_k} = D_{t_k} \cdot C_{t_k}(f_i^{t_k})$ (we use task-specific compression networks for better compression). The last l_2 term serves as a compression regularization for better convergence of training. In this way, we can obtain N task-specific factorized features $\{f_i^{t_k}\}_{t_k}$.

For the instantiation of the factorization module $G(\cdot)$, the simplest way is to copy N front-end networks, *i.e.*, $\{F_{front}^{t_k}(\cdot)\}$ to derive the corresponding task-specific knowledge. However, this raises another concern about the computational cost of multiple front-end network copies, especially considering the expensive attention calculation in VFMs. Therefore, we propose to share all the attention mechanisms, and only finetune the newly introduced task-specific adapters at the end of $F_{front}(\cdot)$ as shown in Fig. 2 (a). We find this simple strategy both efficient and effective. Here, we do not focus on the specific adapter architecture design and directly adopt the recent work [40].

3.3 Consolidate: Cross-task Hyperprior Condition

After factorization, we have already obtained the task-specific factorized features $\{f_i^{t_k}\}, t_k \in T_Q$, reducing the task-irrelevant information. However, it is still redundant to directly compress them one by one. Concretely, the first factorization stage introduces redundancy caused by the shared knowledge of different task features, which is especially serious in the features at shallow layers of the encoder, *e.g.*, the output of $F_{front}(\cdot)$. One might consider using existing feature decoupling methods to reduce this shared redundancy. However, there is a dilemma between using a deeper network for better decoupled features and the limited computing resources at the front end.

To address this problem, we propose to perform the redundancy removal at the *coding level* instead of the feature level. Specifically, we design a cross-task hyperprior conditioning scheme with the philosophy of information reuse. We begin with the basic hyperprior compression model [3], whose data flow can be simplified as:

$$\begin{aligned} f_i^{t_k} \longrightarrow hyper\ z_i^{t_k} \longrightarrow \mu(z_i^{t_k}), \sigma(z_i^{t_k}) \xrightarrow{f_i^{t_k}} B(f_i^{t_k}), \\ \mu = 0, \sigma = 1 \xrightarrow{z_i^{t_k}} B(z_i^{t_k}), \end{aligned}$$

where the feature $f_i^{t_k}$ is first encoded into downsampled hyper $z_i^{t_k}$ to roughly estimate the mean and variance of $f_i^{t_k}$ for better entropy coding into bit-stream $B(f_i^{t_k})$, while the hyper $z_i^{t_k}$ directly uses $\mathcal{N}(0;1)$ for entropy encoding (quantization is omitted here). The trade-off among them is that a larger hyper $z_i^{t_k}$ can lead to more accurate distribution estimation of $f_i^{t_k}$, *i.e.*, larger $B(z_i^{t_k})$ leads to smaller $B(f_i^{t_k})$. In our context, there are multiple factorized features to send to the cloud, and the task features that have been sent can be used as hypers for the unsent task features to fully exploit their redundancy. Specifically, our method can be represented as

$$f_i^{t_j} \longrightarrow hyper\ z_i^{t_j \to k} \longrightarrow \mu(z_i^{t_j \to k}), \sigma(z_i^{t_j \to k}) \xrightarrow{f_i^{t_k}} B(f_i^{t_k}),$$

$f_i^{t_j}$ is the feature of the t_j task that is already sent (it can also be obtained by simulating the decoding process at the encoding end). Intuitively, we reuse the information of other task features for better entropy coding of the current task, which effectively reduces the overall bit-rate in two aspects:

• Since $f_i^{t_j}$ has been decompressed at the cloud, we can derive a higher-quality hyper $z_i^{t_j \to k}$ than $z_i^{t_k}$ without downsampling information loss, leading to a better estimation of μ and σ. Actually, we find $z_i^{t_j \to k}$ is almost always significantly better than its own hyper $z_i^{t_k}$ for entropy encoding. This indicates that in the shallow layer, there is still much correlation redundancy between different task features after factorization.

• No need to send the original hyper $z_i^{t_k}$ if using another task feature as hypers, directly leading to $\sim 30\%$ bit-stream reduction.

Overall, our compression works in a task auto-regressive manner. The first task feature is compressed using its own-derived hyper $z_i^{t_k}$, and the later task can select the best hyper from its own $z_i^{t_k}$ and other hypers $\{z_i^{t_j \to k}\}$ according to the bit-rate objective, where $\{t_j\}$ are all the tasks that have been sent. The only cost is to add an identifier field to indicate which task feature is used as hyper, which is negligible. *Note different hypers do not affect the compression distortion and task performance*, and only affect the bit-rate.

In implementation, for t_k task compression network, we employ $\mu_{t_k}(\cdot), \sigma_{t_k}(\cdot)$ modules for prediction of its own hyper $z_i^{t_k}$ and add $\mu'_{t_k}(\cdot), \sigma'_{t_k}(\cdot)$ for prediction of other task hypers $z_i^{t_j \to k}$. These modules are simply instantiated as 3-layer residual MLPs. Furthermore, to determine the task transmission sequence including the order and the used hyperprior, we globally maintain an optimal task sequence table indexed by each task combination in training. This will be used in the inference process as a hard rule, *i.e.*, directly obtaining the task transmission sequence by looking up the table based on the requested task combination, to avoid the additional cost of sample-wise searching. We found that the bit-rate obtained by this efficient heuristic strategy is very close to that of optimal sample-wise searching.

3.4 Recomposite: Task-attentive Feature Fusion

We have already achieved an effective multi-task reconfigurable coding framework by factorization and consolidation. However, the inference efficiency of the cloud is ignored. Concretely, due to the factorized multiple features, a single request from the front end needs $|T_Q|$ times forward of $F_{cloud}(\cdot)$, which reduces the throughput of the cloud and loses the advantages of multi-task VFMs, *i.e.*, single forward for multiple tasks. Therefore, we propose an efficient task-attentive recomposition module for composing multiple features into a single feature before being fed into $F_{cloud}(\cdot)$.

Specifically, we first use task-specific MLPs to project all $|T_Q|$ task features into a fusion latent space, and concatenate-reshape them as $\widetilde{f}_i^{all} \in \mathbb{R}^{BN_p \times |T_Q| \times C}$, where B, N_p, C are batch size, patch token number, and dimension. Then, a self-attention layer is applied with learnable task-specific embeddings to capture the task relations:

$$w = sigmoid\left(FFN\left(MHSA(\widetilde{f}_i^{all} + TE[T_Q])\right)\right), \tag{5}$$

where FFN and $MHSA$ are the feed-forward network and multi-head self-attention in the transformer. TE are N learnable parameters in $\mathbb{R}^D$ as task embeddings, which work similarly to the position embedding. Note that this attention operation is light-weight because attention is calculated along the $|T_Q|$ dimension, which is very small. After *sigmoid*, we obtain the fusion weights $w \in \mathbb{R}^{BN \times |T_Q| \times C}$. Then, we utilize w to perform weighted sum of $\widetilde{f}_i^{all}$ along $|T_Q|$ dimension to obtain the final fused features $\widetilde{f}_i^{fuse} \in \mathbb{R}^{B \times N \times C}$, which are fed into the $F_{cloud}(\cdot)$ to obtain the corresponding predictions of targeted tasks T_Q with only single forward.

Note that since we only introduce small task-specific parameters by adapters with most parameters shared by different tasks, their feature distributions are still similar, which makes this simple additive fusion strategy feasible. In our experiments, it is observed that this fusion can slightly affect the task performance compared with no fusion, due to the potential conflicts or collaboration of different features. Overall, this fusion design promotes the cloud throughput and further allows advanced fusion policies, *e.g.*, which tasks should be fused together and how many groups should be fused into, considering the computing burden and task affinity, which is an interesting topic as future work.

3.5 Training and Inference

Integrating all the above designs, we effectively reduce both the task-irrelevant information (*factorization*) and the shared information coding redundancy (*consolidation*) with well inference efficiency (*recomposition*), to achieve a compact and flexible reconfigurable coding paradigm. Next, we describe the training and inference pipeline.

Training. Given a pre-trained multi-task model partitioned into front and cloud sub-networks, *we only add/update the parameters of adapters, compression models, and a recomposition module for training.* In the following, the modules not

mentioned remain fixed. 1) We first copy and finetune N front end adapters and compression models for each task described in Sec. 3.2. 2) Then, we only train $\mu'_{t_k}(\cdot)$ and $\sigma'_{t_k}(\cdot)$ (task-specific modules of t_k) in each task compression model using only bit-rate optimization objective $R(\cdot)$, as cross-task hyper learning. 3) Finally, we train the recomposition module using only task loss $\sum_{t_k \in T_Q} \mathcal{L}_{t_k}(\widetilde{f}_i^{t_k}, Y_i^{t_k})$ by enumerating all possible T_Q combinations (the recomposition module is only 0.4% size of the backbone encoder and hence training converges very fast). All detailed formulations of the training objectives can be found in Supp.

Inference. With the targeted tasks T_Q and input image data, the $F_{front}(\cdot)$ with corresponding task-specific adapters first outputs $|T_Q|$ factorized features. Then these features are compressed along with a bit-stream field for T_Q indication and a bit-stream field to show which task hyper is used. The cloud side decompresses, fuses them[3], and perform model forward to obtain final predictions of T_Q to return to users.

4 Experiment

4.1 Datasets, Tasks, and Metrics

We present a new benchmark on multi-task reconfigurable feature coding, covering regression-based, low-level, global, and dense semantic understanding tasks. Specifically, we utilize two commonly used multi-task learning datasets, PASCAL-Context [26] and NYUD v2 [27]. PASCAL-Context includes high-quality annotations of 5 tasks, *i.e.*, *semantic segmentation, human parsing, saliency detection, surface normals*, and *edge boundary detection*, with 4,998 training images and 5,105 test images. NYUD v2 provides *semantic segmentation, monocular depth estimation, surface normals*, and *edge boundary detection* task annotations with 795 training images and 654 test images. Besides, we also collect the scene labels of each image to involve an additional *scene classification* task.

For the downstream task evaluation metrics, we follow previous multi-task learning works [38, 39]. Specifically, 1) Segmentation and Human parsing: mean intersection-over-union (mIoU); 2) Depth: root mean square error (RMSE); 3) Saliency: maximum F-measure (maxF); 4) Normals: mean error (mErr); 5) Edge: optimal-dataset-scale F-measure (odsF); 6) Top-1 accuracy is used for scene classification task. Meanwhile, bits-per-pixel (bpp) is adopted for bit-rate evaluation.

4.2 Implementation Details

We adopt multi-task supervised pretrained models on the above datasets with ViT backbones. Specifically, we split the ViT at the 3rd layer as front-end model partition point for both ViT-Base (12 layers) and ViT-Large (24 layers). The feature channel to be compressed is 768 for ViT-Base and 1024 for ViT-Large. For training, Adam optimizer with a learning rate as 2e-5 is adopted. The batch

[3] Recomposition is optional and compatible with more advanced fusion policies. We use simple $|T_Q|$-to-1 fusion strategy by default.

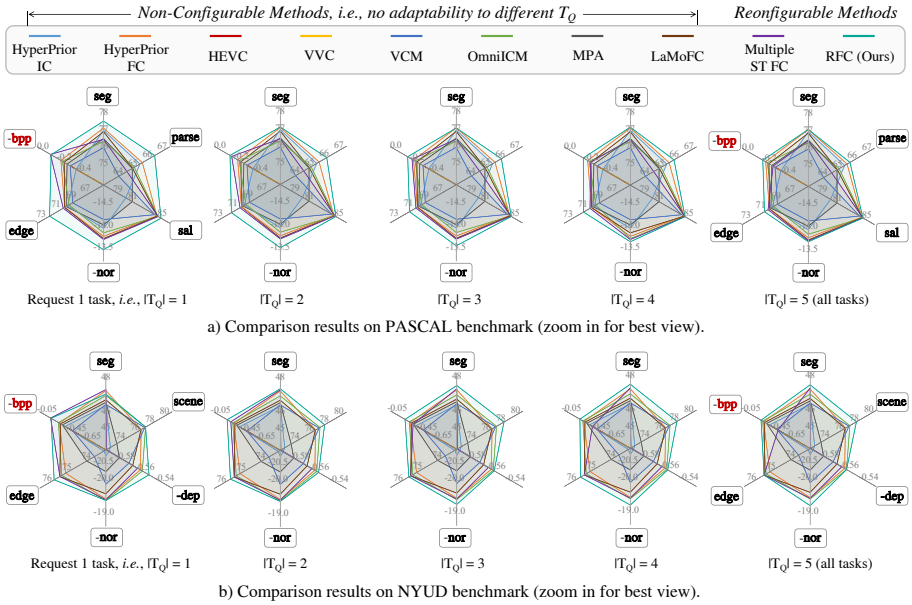

Fig. 3: Overall comparison on PASCAL (a) and NYUD (b). We comprehensively display the task performance with corresponding metrics and bpp. Note we take the opposite numbers for normals predictions, depth estimation, and bpp to ensure that **the larger metric is better** (denoted as "**-metric**"). The average performance is reported. For example, when task number is 2, task t_k score is averaged on all possible $T_Q, |T_Q| = 2$ *and* $t_k \in T_Q$ and bpp is averaged on all possible $T_Q, |T_Q| = 2$. Here, for the convenience of comparison, we adopt the form of radar charts. The complete numerical results can be found in Supp.

size is 3. λ is 1.0 and λ' is 5.0. The input images are cropped as 512×512 for PASCAL-Context and 448×576 for NYUD v2, following previous works. The models are trained 60k, 20k, and 20k iterations for factorization, consolidation, and recomposition stages, respectively. Details can be found in Supp.

4.3 Evaluation Protocol

We aim to present a comprehensive evaluation of bit-rate and task performance for different requests, with the following two evaluation protocols designed.

• **Overall**: As a comprehensive and regular evaluation protocol, we present detailed results of different requested task numbers, *i.e.*, 1 to N. The average task and bit-rate performance on all possible T_Q combinations are reported.

• **Simulated**: As a more practical and realistic protocol, we simulate a request stream containing different requested numbers and evaluate the average task and bit-rate performance. Specifically, we recommend two stream settings: 1) Few tasks dominated requests (*F-request*) with 40%, 30%, 15%, 10%, and 5%

Table 1: Comparison on PASCAL-Context benchmark under *simulated evaluation protocol*. $|T_Q|$ is the number of requested tasks.

Method	*F-request Flow*						*M-request Flow*						Forward
	Seg.↑	Parse↑	Sal.↑	Nor.↓	Edge↑	bpp↓	Seg.↑	Parse↑	Sal.↑	Nor.↓	Edge↑	bpp↓	Times
Uncompressed	78.95	66.83	84.64	13.79	72.4	-	78.95	66.83	84.64	13.79	72.4	-	1
Hyperprior IC	77.29	65.53	80.11	14.02	70.0	0.3150	77.29	65.53	80.11	14.02	70.0	0.3150	1
Hyperprior FC	77.43	65.70	84.43	13.84	70.4	0.1554	77.43	65.70	84.43	13.84	70.4	0.1554	1
HEVC	17.12	19.16	56.20	22.15	24.1	0.2648	17.12	19.16	56.20	22.15	24.1	0.2648	1
VVC	45.98	37.38	71.67	19.18	44.3	0.2907	45.98	37.38	71.67	19.18	44.3	0.2907	1
VCM	76.25	64.06	84.14	14.22	69.3	0.2531	76.25	64.06	84.14	14.22	69.3	0.2531	1
OmniICM	76.63	65.14	84.58	13.92	69.8	0.1713	76.63	65.14	84.58	13.92	69.8	0.1713	1
MPA	76.80	65.03	81.02	14.08	70.2	0.2197	76.80	65.03	81.02	14.08	70.2	0.2197	1
LaMoFC	77.21	65.02	84.50	13.79	70.4	0.1894	77.21	65.02	84.50	13.79	70.4	0.1894	1
Multiple ST FC	76.75	64.63	84.59	13.77	70.6	0.0976	76.75	64.63	84.59	13.77	70.6	0.1557	$\|T_Q\|$
RFC (Ours)	**77.61**	**66.31**	**84.81**	**13.59**	**72.0**	**0.0725**	**77.47**	**66.11**	**84.67**	**13.66**	**71.5**	**0.0977**	1

Table 2: Ablation study on different components of RFC. $|T_Q|$ denotes the number of requested tasks by users.

Method	$\|T_Q\|$	Segmentation	Parsing	Saliency	Normals	Edge	Bit-Rate	Forward
		mIoU ↑	mIoU ↑	maxF ↑	mErr ↓	odsF ↑	bpp ↓	Times
Only Factorize - Full $F_{front}(\cdot)$	5	76.98	65.79	84.49	13.63	71.9	0.2700	5
Only Factorize - Adapters		77.85	66.56	85.00	13.54	72.5	0.2675	5
Factorize+Fuse		77.43	65.90	84.60	13.75	71.1	0.2675	1
Factorize+Consolidate+Fuse		77.43	65.90	84.60	13.75	71.1	0.1253	1

frequency for 1, 2, 3, 4, and 5 target tasks; 2) Multiple tasks dominated requests (*M-request*) use 5%, 20%, 30%, 25% and 20% frequency for 1-5 tasks.

4.4 Comparison Results

We compare our method with the recent methods on feature compression and image compression for machine. Most compared methods are non-configurable methods, *i.e.*, cannot adapt to different T_Q, including 1) image/video codec standard HEVC [30] and VVC [5]; 2) Hyperprior [3] IC and FC: image / feature compression using classic hyperprior model; 3) VCM [35]: feature compression using a structural codebook; 3) OmniICM [11]: feature compression for machine with task-specific cloud tail networks; 4) MPA [41]: image compression method with multiple task-specific decoding paths for better performance; 5) LaMoFC [15]: feature compression method with a hyperprior model applied to features after a channel-flatten transformation. Note that previous work pays little attention to the reconfigurable methods. We present 6) Multiple ST FC: feature compression separately for multiple single-task models, as a naive reconfigurable implementation for comparison. The results are shown in Figure 3.

1) Compared to traditional non-configurable methods, **RFC can organize the bit-stream on-demand**, demonstrating remarkable bit-stream reduction with strong task performance when requested task number is small. As shown in the charts with $|T_Q| < 4$, our bpp is much smaller than non-configurable methods. This is the most remarkable advantage of our method, *i.e.*, flexibility and adap-

Table 3: Ablation study on different split points under simulated evaluation protocol of F-request Flow. Our baseline is Hyper FC.

Split-Point	Method	Seg.	Parse	Sal.	Nor.	Edge	BPP
2nd Layer	Baseline	76.84	65.09	84.54	13.94	70.6	0.1566
	RFC	**77.28**	**65.92**	**84.83**	**13.68**	**71.6**	**0.0838**
6th Layer	Baseline	75.85	65.40	84.83	13.60	70.8	0.1841
	RFC	**76.26**	**66.00**	**85.15**	**13.31**	**72.0**	**0.0771**

tation capacity to different requests, which is achieved by the factorization to reduce the T_Q-irrelevant information.
2) In addition to the strong and flexible adaptation capacity, **RFC can still perform well when requesting all tasks**. See the charts with $|T_Q| = 5$. Most traditional methods target at the all-task-requested scenario with reconstructed compression manner. However, in this case, the compared reconfigurable methods can seriously suffer from the coding redundancy caused by the shared knowledge between different factorized task features, leading to poor bpp performance. Thanks to the proposed consolidation coding mechanism, our methods can effectively reduce this redundancy, leading to a compact bitstream. Moreover, we find RFC even outperforms the traditional methods, which we attribute to the benefits of explicit feature factorization and task-specific compression learning.
3) Compared with another naive reconfigurable method (Multiple ST FC), **RFC retains the native efficiency of VFMs, *i.e.*, single forward for inference of multiple tasks**. Multiple ST FC lose the inference efficiency. Specifically, Multiple ST FC employs different models for each task, leading to N model forwards for N-task inference. In contrast, we design efficient front adapters and a cloud recomposition strategy to retain the single forward inference efficiency.
4) For more practical simulation protocol, RFC demonstrates significant advantages over other methods. As shown in Table 1, we report the average performance of various requests of different task numbers to cover all possible requests. Our method can achieve the best performance with significant improvement, verifying our motivation and effectiveness for *reconfigurable feature compression*.

4.5 Ablation Study

Effect of Main Designs in Table 2: **(1)** First, we develop the factorization design to endow model with basic adaptation capacity to different T_Q. We compare the directly fine-tuning full $F_{front}(\cdot)$ with our adapter-based strategy. We can see that employing adapters performs better by alleviating the over-fitting problem. Meanwhile, it keeps the parameters of $F_{front}(\cdot)$ unchanged to avoid multiple model forward of $F_{front}(\cdot)$. **(2)** The recomposition design achieves inference efficiency, *i.e.*, single forward for multi-task inference. It is also observed that it can slightly affect the task performance. We have provided more detailed

Table 4: Overall comparison on PASCAL-Context benchmark with **ViT-Large backbone**. Task number denotes the number of requested tasks from the front end, and the average performance is reported.

Method	Task Number	Segmentation mIoU ↑	Parsing mIoU ↑	Saliency maxF ↑	Normals mErr ↓	Edge odsF ↑	Bit-Rate (bpp) ↓	Forward Times
Non-Configurable Methods								
Uncompressed	1-5	81.03	69.83	84.44	13.80	72.8	-	1
Hyperprior IC		80.24	68.69	84.35	13.86	71.0	0.3059	1
Hyperprior FC		80.17	68.71	84.40	13.84	71.5	0.2566	1
Reconfigurable Methods								
RFC (Ours)	1	80.15	68.96	84.42	13.70	72.7	0.0720	1
	2	80.12	69.06	84.46	13.73	72.8	0.0933	
	3	80.31	69.11	84.45	13.76	72.6	0.1115	
	4	80.44	69.13	84.46	13.80	72.5	0.1284	
	5	80.47	69.14	84.46	13.82	72.3	0.1451	

quantitative results of different task combinations in Supp. to depict the impact of recomposition in each case. This is an interesting topic that how to design a more advanced task grouping strategy for future work. **(3)** Finally, our consolidation coding strategy effectively reduces the redundancy caused by the shared knowledge between different task features, leading to a significant bit-stream reduction, especially when $|T_Q|$ is large.

Effect of Different Split Points in Table 3: We compare different split points with baseline, *i.e.*, hyperprior feature compression method. We can see that our method can achieve a consistent performance improvement with different split layers, verifying the generalization ability of our method. There is a less important technical detail, *i.e.*, when split at $6th$ layer, the feature layers aggregated by the cloud decoder is changed to $\{6, 9, 12\}$ instead of the default $\{3, 6, 9, 12\}$ [37], which can affect the whole model performance to some extent.

Larger Backbones in Table 4: We present the results using ViT-Large (24 layers in total) as the backbone on the PASCAL-Context benchmark. The same model partition point is selected, *i.e.*, the 3rd layer of the whole model. Compared with the 3rd layer in ViT-Base (12 layers in total) for partition, the partition point of the deeper ViT-Large is relatively shallower with less decoupled features of different tasks and a higher bit-rate (The extreme case is Layer 0, where the image involves complete information for analysis, with the poorest decoupling and the highest bit rate). As we can see, our method can still achieve desirable adaptation capacity and flexibility. This indicates that our method has good compatibility with different model backbones and different partition relative depth.

4.6 Practical Analysis of RFC

Beyond its algorithmic novelty and effectiveness, RFC provides several practical advantages for real-world deployment, offering more flexibility in actual use:
1) Adaptive factorization granularity. While per-task decomposition maximizes representational efficiency, we can also group highly correlated or co-requested tasks, *e.g.*, $\{(t_1, t_2), (t_3), (t_4, t_5)\}$, into unified units based on application needs, reducing training and deployment complexity for more tasks. **2) On-demand front-end execution.** Only parameters for the requested tasks need to be loaded at runtime. Since most user requests involve just a few tasks, the front-end computation remains efficient and scales naturally with demand. **3) Beyond faster transmission.** The smaller bitstream also saves storage space, simplifies data handling, and improves privacy by reducing raw data exposure.

5 Conclusion and Future Work

We study a new problem termed reconfigurable multi-task feature compression, designed to efficiently tackle various requests from the front end with VFMs in an end-cloud collaboration paradigm. Remarkably, we propose a unified framework, RFC, which involves novel designs to tackle feature *factorization* as task-specific knowledge, joint *consolidation* and compression of the factorized features, and multiple feature *recomposition* for inference efficiency. Finally, we construct a comprehensive benchmark to verify the effectiveness and remarkable adaptation capacity of our method.
Future Work. The proposed RFC demonstrates promising results with desirable adaptation capacity to different T_Q. For future work, it can extends the model to support high-fidelity image reconstruction for human vision, while this paper currently only prioritizes high-level semantic feature compression for machine analysis. Besides, as observed in the Supp., slight performance fluctuations are found under different task combinations due to complex task interactions. Exploring advanced task assignment and routing strategies remains a meaningful direction for real-world applications.

Acknowledgements

This work was supported in part by the Beijing Major Science and Technology Project under Contract No.Z251100008425023 and in part by the Interdisciplinary Frontier Research Project of PCL (PCL2025QYB013).

References

1. de Andrade, A., Harell, A., Foroutan, Y., Bajić, I.V.: Conditional and residual methods in scalable coding for humans and machines. In: ICMEW (2023)
2. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. arXiv preprint arXiv:1611.01704 (2016)

3. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: ICLR (2018)
4. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE TPAMI **35**(8), 1798–1828 (2013)
5. Bross, B., Wang, Y.K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.: Overview of the versatile video coding (vvc) standard and its applications. IEEE TCSVT **31**(10), 3736–3764 (2021)
6. Cheng, Z., Sun, H., Takeuchi, M., Katto, J.: Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: IEEE/CVF CVPR. pp. 7939–7948 (2020)
7. Choi, H., Bajić, I.V.: Latent-space scalability for multi-task collaborative intelligence. In: ICIP. pp. 3562–3566 (2021)
8. Choi, H., Bajić, I.V.: Scalable image coding for humans and machines. IEEE TIP **31**, 2739–2754 (2022)
9. De Cao, N., Aziz, W., Titov, I.: Editing factual knowledge in language models. arXiv preprint arXiv:2104.08164 (2021)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
11. Feng, R., Jin, X., Guo, Z., Feng, R., Gao, Y., He, T., Zhang, Z., Sun, S., Chen, Z.: Image coding for machines with omnipotent feature learning. In: ECCV. pp. 510–528 (2022)
12. Finder, S.E., Zohav, Y., Ashkenazi, M., Treister, E.: Wavelet feature maps compression for image-to-image CNNs. NeurIPS pp. 20592–20606 (2022)
13. Fu, T., Deng, R., Gao, Y., Zhang, F.: Representing scenes as compositional generative neural feature fields based on giraffe for 3d reconstruction of classroom scenes. In: IIHMSP. pp. 227–237 (2022)
14. Gao, C., Ma, Y., Chen, Q., Xu, Y., Liu, D., Lin, W.: Feature coding in the era of large models: Dataset, test conditions, and benchmark. In: ICCV (2025)
15. Gao, C., Ma, Y., Chen, Q., Xu, Y., Liu, D., Lin, W.: Feature coding in the era of large models: Dataset, test conditions, and benchmark. In: ICCV. pp. 1068–1077 (2025)
16. Gu, B., Zhan, J., Gong, S., Liu, W., Su, Z., Guizani, M.: A spatial-temporal transformer network for city-level cellular traffic analysis and prediction. IEEE TWC **22**(12), 9412–9423 (2023)
17. Guo, S., Chen, Z., Zhao, Y., Zhang, N., Li, X., Duan, L.: Toward scalable image feature compression: A content-adaptive and diffusion-based approach. In: ACM MM. pp. 1431–1442 (2023)
18. Harell, A., Foroutan, Y., Bajić, I.V.: VVC+M: Plug and play scalable image coding for humans and machines. In: ICMEW (2023)
19. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. IEEE/CVF CVPR (2022)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE/CVF CVPR. pp. 770–778 (2016)
21. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
22. Huang, X., Zhang, S.: Human activity recognition based on transformer in smart home. In: CACML. p. 520–525 (2023)
23. Huang, Y., Wu, Z., Wang, L., Tan, T.: Feature coding in image classification: A comprehensive study. IEEE TPAMI **36**(3), 493–506 (2013)

24. Kuang, H., Ma, Y., Yang, W., Guo, Z., Liu, J.: Consistency guided diffusion model with neural syntax for perceptual image compression. In: ACM MM. pp. 1622–1631 (2024)
25. Meng, K., Bau, D., Andonian, A., Belinkov, Y.: Locating and editing factual associations in gpt. NeurIPS **35**, 17359–17372 (2022)
26. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: IEEE/CVF CVPR (2014)
27. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
28. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
29. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML. pp. 8748–8763 (2021)
30. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE TCSVT **22**(12), 1649–1668 (2012)
31. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. arXiv preprint arXiv/0004057 (2000)
32. Wang, Y., Yang, C., Lan, S., Zhu, L., Zhang, Y.: End-edge-cloud collaborative computing for deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials **26**(4), 2647–2683 (2024)
33. Yamazaki, M., Kora, Y., Nakao, T., Lei, X., Yokoo, K.: Deep feature compression using rate-distortion optimization guided autoencoder. In: ICIP. pp. 1216–1220 (2022)
34. Yang, M., Yang, F., Murn, L., Blanch, M.G., Sock, J., Wan, S., Yang, F., Herranz, L.: Task-switchable pre-processor for image compression for multiple machine vision tasks. IEEE TCSVT **34**(7), 6416–6429 (2024)
35. Yang, W., Huang, H., Hu, Y., Duan, L.Y., Liu, J.: Video coding for machines: Compact visual representation compression for intelligent collaborative analytics. IEEE TPAMI **46**(7), 5174–5191 (2024)
36. Yang, X., Ye, J., Wang, X.: Factorizing knowledge in neural networks. In: ECCV. pp. 73–91 (2022)
37. Yang, Y., Jiang, P.T., Hou, Q., Zhang, H., Chen, J., Li, B.: Multi-task dense prediction via mixture of low-rank experts. In: IEEE/CVF CVPR (2024)
38. Ye, H., Xu, D.: Inverted pyramid multi-task transformer for dense scene understanding. In: ECCV (2022)
39. Ye, H., Xu, D.: Taskprompter: Spatial-channel multi-task prompting for dense scene understanding. In: ICLR (2023)
40. Yin, D., Hu, L., Li, B., Zhang, Y., Yang, X.: 5%> 100%: Breaking performance shackles of full fine-tuning on visual recognition tasks. In: IEEE/CVF CVPR. pp. 20071–20081 (2025)
41. Zhang, X., Guo, P., Lu, M., Ma, Z.: All-in-one image coding for joint human-machine vision with multi-path aggregation. NeurIPS pp. 71465–71503 (2024)
42. Zhang, Z., Wang, M., Ma, M., Li, J., Fan, X.: Msfc: Deep feature compression in multi-task network. In: ICME. pp. 1–6 (2021)
43. Zhu, C., Rawat, A.S., Zaheer, M., Bhojanapalli, S., Li, D., Yu, F., Kumar, S.: Modifying memories in transformer models. arXiv preprint arXiv:2012.00363 (2020)