This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Feature Coding in the Era of Large Models: Dataset, Test Conditions, and Benchmark

Changsheng Gao[1], Yifan Ma[2], Qiaoxi Chen[2], Yenan Xu[2], Dong Liu[2], Weisi Lin[1†]
[1]Media Interactive Computing Lab, Nanyang Technological University, Singapore 639798
[2]MOE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition,
University of Science and Technology of China, Hefei 230093, China
{changsheng.gao, wslin}@ntu.edu.sg; {mayf,xxii,yenanxu}@mail.ustc.edu.cn; dongeliu@ustc.edu.cn

Abstract

Large models have achieved remarkable performance across various tasks, yet they incur significant computational costs and privacy concerns during both training and inference. Distributed deployment has emerged as a potential solution, but it necessitates the exchange of intermediate information between model segments, with feature representations serving as crucial information carriers. To optimize information exchange, feature coding is required to reduce transmission and storage overhead. Despite its importance, feature coding for large models remains an under-explored area. In this paper, we draw attention to large model feature coding and make three fundamental contributions. First, we introduce a comprehensive dataset encompassing diverse features generated by three representative types of large models. Second, we establish unified test conditions, enabling standardized evaluation pipelines and fair comparisons across future feature coding studies. Third, we introduce two baseline methods derived from widely used image coding techniques and benchmark their performance on the proposed dataset. These contributions aim to provide a foundation for future research and inspire broader engagement in this field. To support a long-term study, all source code and the dataset are made available at https://github.com/chansongoal/LaMoFC.

1. Introduction

Large models have revolutionized artificial intelligence (AI) with their impressive performance. However, they require substantial training data and extensive computational resources, posing significant challenges for practical deployments [19]. To address these challenges, distributed deployment has been proposed as a promising solution [14, 23, 38, 43, 49, 52]. By distributing the computational workload across multiple platforms, this approach reduces the resource strain on individual systems and enables the integration of private client data into training while maintaining data security [4, 49]. As model scale continues to grow, distributed deployment is expected to become a mainstream strategy for large model deployments.

In distributed deployments, effective information exchange between model segments is essential. During forward propagation, features generated by early segments are passed to later segments to complete the inference. Given the vast amount of data in large model applications, it is necessary to encode features to a smaller size to reduce storage and transmission burden. Analogous to image and video coding, feature coding has been proposed to address this problem [6, 16, 18, 44].

Existing feature coding research faces three key limitations: narrow model types, limited task diversity, and constrained data modalities. Most existing feature coding efforts focus on small-scale convolution-based networks, such as ResNet, ignoring large models. Furthermore, existing research primarily targets visual features and neglects the investigation of textual features from large language models (LLMs). This gap indicates a need for broader exploration across both models and data modalities to advance feature coding.

In this work, we shift the feature coding research from small-scale to large-scale models. To address the existing limitations, we construct a comprehensive test dataset with features from five tasks across three large models, covering both visual and textual data for increased modality diversity. Furthermore, we establish unified test conditions to enable fair comparisons, including formulated bitrate computation and task-specific evaluation pipelines.

To lay the groundwork for large model feature coding,

This work was supported by the Ministry of Education of Singapore under Grant T2EP20123-0006. We acknowledge the support of GPU cluster built by MCC Lab of Information Science and Technology Institution, USTC.

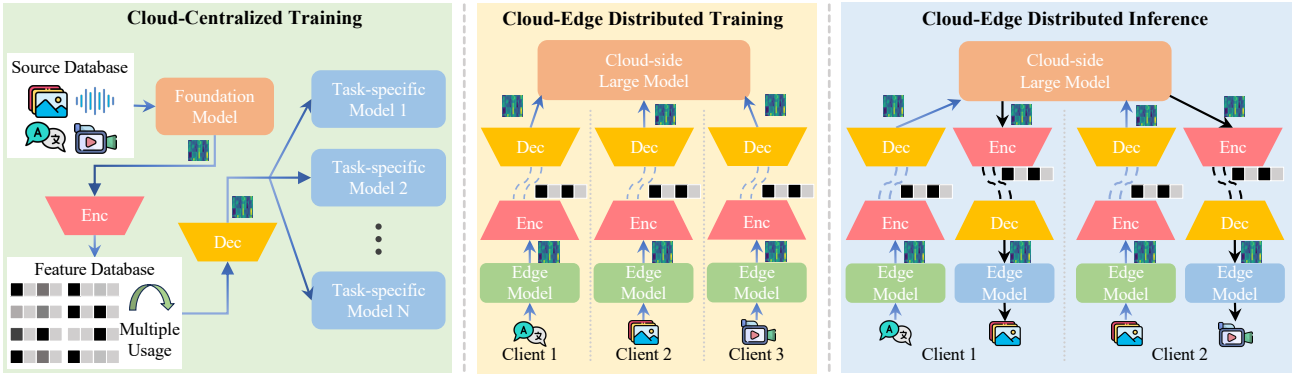

Figure 1. Three application scenarios for **feature coding** in large model deployments. **Cloud-centralized training:** Features generated by foundation models are encoded and stored in a feature database. This database supports various downstream tasks, reducing computational demands by avoiding repeated inference of large foundation models. **Cloud-edge distributed training:** To protect user data privacy, raw data is first processed by an edge model to generate features, which are then encoded and transmitted to the cloud. The cloud decodes these features to complete the remaining forward propagation. **Cloud-edge distributed inference:** A large model is partitioned across multiple parts, with portions offloaded to edge devices to distribute the computational load. The transmitted features ensure privacy protection and enable customized downstream tasks. Please refer to Sec. 3 for detailed illustrations.

we propose two baseline methods and evaluate their applicability to the dataset. The baselines leverage two image coding methods: the handcrafted VVC Intra coding [2] and the learning-based Hyperprior method [1]. We introduce pre-processing and post-processing modules to adapt the baselines to feature characteristics. The resulting benchmark and analyses provide valuable insights into large model feature coding and identify promising directions for further exploration. In summary, our main contributions are as follows:

- We highlight the importance of feature coding in large model deployments, identify relevant application scenarios, and introduce a new research area: large model feature coding.
- We construct a comprehensive feature dataset encompassing three model types, five tasks, and two kinds of source data modalities, along with unified test conditions to support long-term large model feature coding research.
- We propose two baselines and evaluate their usability to the large model feature coding. We build a benchmark and suggest promising directions for further research.

2. Background and motivation

2.1. Feature coding

Originally introduced within the Coding for Machines framework [10, 48], feature coding is one of two primary branches alongside visual coding. Unlike visual coding [15, 25, 26, 28, 31, 35, 39, 40, 50], which encodes and reconstructs the original visual data, feature coding involves encoding features extracted from source data into bitstreams. While visual coding has already been explored in the context of large models [20, 41], feature coding remains unexplored in this domain.

Despite recent progress, feature coding research still faces three key limitations. First, there is a **restricted variety of model types** in use. The existing feature coding focuses on small-scale CNN features such as [3, 17, 21, 24, 27, 30, 37]. However, as Transformer architectures now dominate large model research [11, 12, 33, 42], it is crucial to investigate feature coding for Transformer-based models. Second, the field is constrained by the **limited scope of task types**. Existing studies primarily address discriminative tasks such as image classification and segmentation [5, 7, 13, 32, 47, 51], while generative tasks remain largely unexplored. Given the significance of generative models, extending feature coding to include these tasks, such as image synthesis, is essential. The third limitation is the **constrained range of source modality**. Current feature coding research predominantly focuses on features extracted from visual data, overlooking the increasingly important features generated from textual data, particularly in light of advancements in LLMs. Expanding to multiple modalities is crucial as the diversity of AI applications grows.

Furthermore, two challenges hinder progress in large model feature coding: **the lack of a public test dataset and unified test conditions**. Without a standardized dataset and evaluation conditions, fair comparisons across methods can not be conducted. A public dataset ensures consistent input data across studies, while unified test conditions establish a standard pipeline to evaluate performance.

These limitations motivate us to construct a diverse test dataset and formulate unified test conditions to support large model feature coding research.

Model	Parameters	Task	Split Point	Feature Shape	Source Data
DINOv2	1.1B	Image Classification (Cls)	40^{th} ViT Block	257×1536	ImageNet, 500 samples
		Semantic Segmentation (Seg)	40^{th} ViT Block	$2 \times 1370 \times 1536$	VOC2012, 100 samples
		Depth Estimation (Dpt)	10^{th}, 20^{th}, 30^{th}, 40^{th} ViT Blocks	$2 \times 4 \times 1611 \times 1536$	NYU-Depth-v2, 80 samples
Llama3	8B	Common Sense Reasoning (CSR)	32^{th} Decoder Layer	$N \times 4096$	Arc-Challenge, 500 samples
SD3	8B	Text-to-Image Synthesis (TTI)	Input Layer of VAE Decoder	$16 \times 128 \times 128$	COCO2017, 500 samples

Table 1. Summary of the model and task selection, split point decision, and source data collection.

2.2. Large model deployment

The impressive performance of large models is driven by scaling laws [34, 36], which emphasize the importance of vast model sizes and extensive training data. For example, GPT-3 was built with 175 billion parameters and trained on 499 trillion tokens, both contributing to substantial computational demands. To alleviate these resource requirements, distributed training and inference methods have been proposed in [14, 49, 52].

User privacy protection presents another challenge in client services [4, 29, 46]. Split learning has been proposed as an effective solution to protect data privacy by partitioning models between clients and the cloud [49].

In distributed deployments, efficient information exchange between model segments is crucial. However, academic research has largely overlooked the storage and transmission costs associated with feature exchange. To address this gap, we propose to encode features into compact bitstreams to improve the efficiency of large model deployments in distributed environments.

3. Feature coding in large models

In Fig. 1, we illustrate three application scenarios of feature coding in large model deployments, highlighting their roles in optimizing the deployment efficiency of large models.

3.1. Large model training

Large models achieve their high performance through extensive training on large-scale datasets. Cloud-centralized training is commonly employed to leverage the vast computational resources and data in the cloud. However, in sensitive domains like healthcare, high-quality public data may be scarce, necessitating access to private data through cloud-edge collaborative training.

3.1.1. Cloud-centralized training

In cloud-centralized training, source databases are maintained in the cloud, where intensive computational demands become the main challenge. Complex tasks are increasingly addressed through the collaborative use of multiple models. For instance, in vision-language tasks, visual data is first processed by a vision model, and its features are then passed to an LLM for advanced reasoning. Similarly, foundation models often serve as core backbones for various tasks. For example, features extracted from DINOv2 are repurposed for downstream tasks like segmentation [9]. In such collaborative workflows, features generated by one model become inputs for others. However, during training, each data sample may undergo repeated inference, resulting in substantial and unnecessary computational costs.

To address this, an effective solution is to generate features once, store them, and reuse them across tasks. The workflow is outlined on the left of Fig. 1. First, a foundation model extracts features from the source database, which are then encoded and stored in a feature database. For subsequent task-specific training, only the precomputed feature database is needed, enabling efficient access to features without redundant feature extraction. This approach significantly reduces computational load and accelerates downstream task training. However, raw feature data consumes considerable storage space, particularly with large datasets. Therefore, encoding raw features into compact bitstreams becomes essential to reduce storage demands and support efficient large model training.

3.1.2. Cloud-edge distributed training

In addition to computational resources, large model training requires access to large-scale datasets, which are often only available to a few major high-tech companies. Furthermore, sensitive data, such as medical records and financial information, cannot be freely shared due to privacy concerns and regulatory constraints. To leverage distributed, privately owned data, split learning has emerged as a viable solution [43]. Split learning divides the training process into two phases, with the bulk of the computational load handled by the cloud, while edge devices, which are often resource-constrained, handle a smaller portion of the task. This approach minimizes the computational demand on the edge while enabling collaboration with powerful cloud resources.

The cloud-edge distributed training pipeline is depicted in the middle of Fig. 1. The whole pipeline is divided into two parts: edge models and cloud models. During forward propagation, the edge model processes private data to generate features, which are then encoded into bitstreams and transmitted to the cloud. Upon receiving the bitstreams, the cloud decodes them back into features and feeds them into

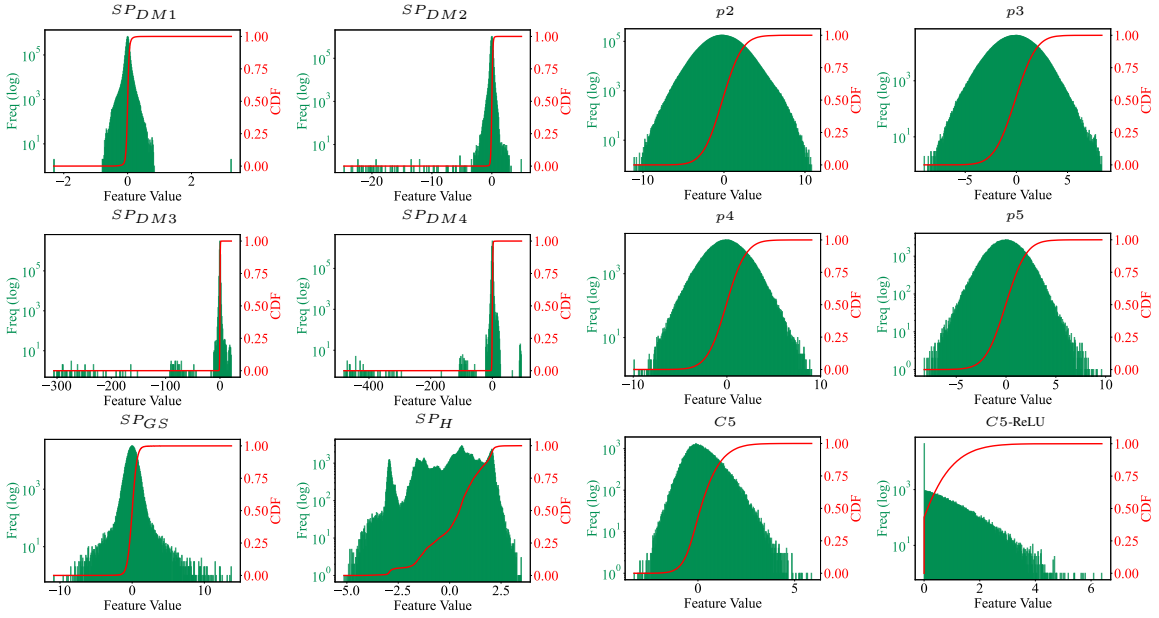

Figure 2. Frequency and CDF comparisons between features in the proposed dataset and commonly used existing features. The proposed features exhibit distinct distributions compared to existing features, highlighting the necessity and value of the dataset. In addition, the proposed features demonstrate greater diversity, enhancing their representativeness and suitability for long-term use. Frequencies are scaled logarithmically for better visualization.

the cloud model. In the backward pass, gradients are computed and backpropagated along the reverse path. To reduce bandwidth cost and minimize latency, it is critical to apply feature coding before transmission.

3.2. Cloud-edge distributed inference

Once large models are trained, they are deployed in products to provide client services. For a product or service to succeed, two primary concerns must be addressed: supporting a high volume of simultaneous cloud server access and protecting user privacy. For the first concern, managing computational load is essential, particularly in generative applications that involve repetitive diffusion processes. A practical solution is to offload part of this computational demand to edge devices. For the second concern, the same strategy in the cloud-edge distributed training, as discussed in Sec. 3.1.2, can be applied.

The cloud-edge distributed inference pipeline is illustrated on the right of Fig. 1. In the upload phase, source data is converted into features by an edge model on the client side. These features are then encoded and transmitted to the cloud, where they are further processed by large models. In the download phase, the processed features are sent back to the client to complete a specific task. Depending on the task head used, various functionalities can be performed with the returned features. For example, Stable Diffusion 3 (SD3) features derived from an image can be used to synthesize video or generate segmentation masks. This approach not only reduces the computational load on the cloud server but also enhances data privacy by keeping raw user data on the edge side.

4. Dataset construction and analysis

4.1. Dataset construction

To guarantee the dataset's representativeness and support the long-term study of feature coding research, we curate the test dataset with consideration of three key aspects. Table 1 outlines our selected models and tasks, split points, and source data.

4.1.1. Model and task selection

Given the vast number of large models available, it is impractical to include all of them in a single research paper. Therefore, we focus on selecting representative models. To ensure the dataset's representativeness and long-term relevance, we choose one widely recognized model for each type of large model. Specifically, we select DINOv2 [33], Llama3 [11], and SD3 [12] as representatives of discriminative models, generative models, and hybrid models, respectively. The trend of adopting Transformer architectures in large models is expected to persist in the coming years. As a result, the selected models are anticipated to retain their research value and practical utility in the near future. DINOv2 and Llama3 handle visual and textual inputs, respectively, while SD3 processes textual inputs to generate visual out-

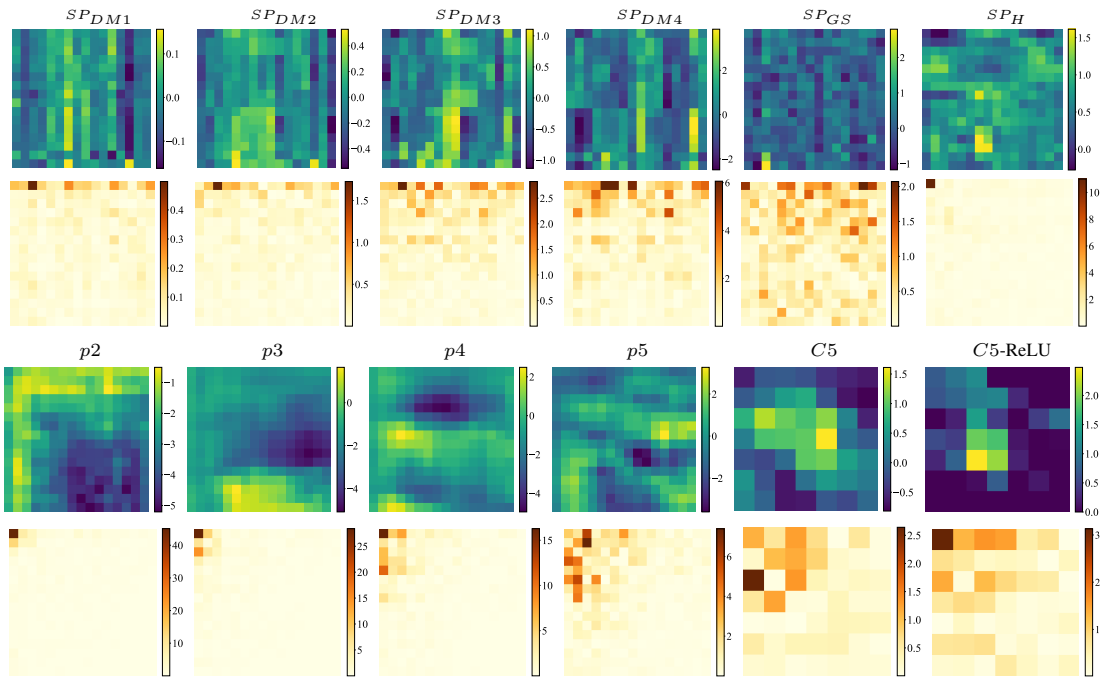

Figure 3. Visualization of the original feature blocks and their corresponding DCT blocks (absolute values). In the feature domain, the proposed features exhibit higher vertical redundancy, whereas the existing features vary smoothly in both directions. In the DCT domain, the proposed features display more dispersed energy distributions. These distinct redundancy characteristics indicate different coding properties. We use 7×7 blocks for $C5$ and $C5$-ReLU and 16×16 blocks for other features.

puts. Together, these models provide comprehensive feature representations across visual, textual, and cross-modal applications.

For DINOv2, we include image classification (Cls), semantic segmentation (Seg), and depth estimation (Dpt) tasks. Llama3 is applied to the common sense reasoning task (CSR), and SD3 is used for text-to-image synthesis (TTI). The selection covers three visual tasks, one textual task, and one text-to-visual task, forming a comprehensive foundation for analyzing feature coding across diverse tasks and model types.

4.1.2. Split point decision

The split points are decided to support various downstream tasks. For DINOv2, we define two types of split points: 1) SP_{DS}, the output of the 40^{th} ViT layer, and 2) SP_{DM}, which aggregates outputs from the 10^{th}, 20^{th}, 30^{th}, and 40^{th} ViT layers. SP_{DS} is used for tasks utilizing single-layer features, while SP_{DM} supports tasks that require multi-layer features. In this study, SP_{DS} is applied to Cls and Seg, and SP_{DM} to Dpt. The corresponding features are designated as F_{Cls}, F_{Seg}, and F_{Dpt}, respectively.

For Llama3, the split point SP_G is set at the output of the 32^{nd} decoder layer. Features extracted at SP_G can support both task-specific heads and integration with other large models [45]. In CSR, SP_G produces $N \times 4096$ features, where N is the number of tokens.

For SD3, we take the input layer of the VAE decoder as the split point SP_H. This split point generates features of size $16 \times 128 \times 128$, denoted as F_{TTI}, which are then passed to the VAE decoder for image synthesis.

4.1.3. Source data collection

We organize our feature test dataset into five classes, each supported by a curated subset of publicly available datasets used as source data. For Cls, we select 500 images from ImageNet [8], each representing a unique class accurately classified by DINOv2. For Seg, we collect 100 images from VOC2012, ensuring coverage of all 20 object classes. For Dpt, we source 80 images from NYU-Depth-v2, adhering to the train-test split proposed in [22] and selecting 3-8 images for each scene category. For CSR, we utilize 500 samples from the Arc-Challenge dataset, chosen for their longest input prompts and correct prediction by Llama3. For TTI, we first select 500 images from COCO2017 and then collect their longest captions.

We noticed that learning-based methods require a dedicated training set. Please follow the source code to generate training data and find our training data in the supplementary.

4.2. Feature analysis

In this section, we compare the proposed features with those commonly used in existing research to demonstrate the necessity and importance of introducing a new dataset.

Task	Head	Metric
Cls	Norm + Linear1	Accuracy
Seg	Norm + Linear1	mIoU
Dpt	Linear4	RMSE
CSR	Norm + Linear + Softmax	Accuracy
TTI	VAE Decoder	FID

Table 2. Summary of task head and accuracy metric settings. Find definitions of Heads in the original papers [11, 12, 33].

4.2.1. Distribution analysis

We first compare frequency distributions and cumulative distribution functions (CDFs) of the proposed and commonly used existing features in Fig. 2. $C5$ and $C5$-ReLU are derived from the official ResNet50 pre-trained on ImageNet, while P-layer features are generated by the ResNeXT101-based MaskRCNN pre-trained on COCO2017.

For multi-layer split points, we observe four key distinctions between SP_{DM} and P-layer features: (1) SP_{DM} distribution range expands progressively with deeper layers; (2) SP_{DM} distribution is more asymmetric; (3) SP_{DM} feature values are more concentrated, as evidenced by CDFs; and (4) SP_{DM} features contain more regions with sparse feature points. For single-layer split points, SP_{GS} has a more concentrated distribution, while SP_H shows more peaks. The proposed dataset omits $C5$-ReLU-like features, as ReLU is rarely used in large models.

The proposed features exhibit different distributions compared to the existing features, highlighting the necessity of introducing the new dataset. Additionally, the high diversity of the proposed features enhances the dataset's representativeness and suitability for long-term research.

4.2.2. Redundancy analysis

Since coding aims to remove redundancy in inputs, we conduct a redundancy comparison on feature blocks. We visualize the original feature blocks and their DCT blocks (absolute values) in Fig. 3. In the feature domain, clear spatial redundancy is observed in both SP_{DM} and SP_H features. SP_H exhibits redundancy in both horizontal and vertical directions, while SP_{DM} shows stronger vertical redundancy. In contrast, SP_{GS} features vary rapidly in both directions, resulting in weak spatial redundancy. This is because Llama3 only processes textual data, which lacks inherent spatial correlation. Existing features, on the other hand, exhibit smoother spatial variation and higher spatial redundancy. We attribute this to the translation invariance of convolutional layers, which preserve the relative positions of pixels in the feature domain, resulting in consistent redundancy patterns. In contrast, transformer-based vision models partition images into patches before transforming them, shifting spatial redundancy primarily to the vertical direction, as seen in SP_{DM}.

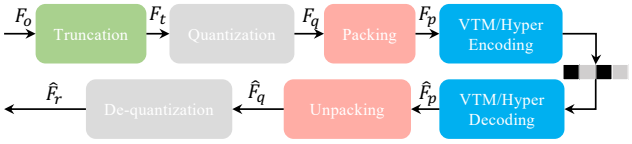

Figure 4. The pipeline of the proposed baseline methods.

In the DCT domain, spatial redundancy is reflected in the energy distribution. The proposed features display more dispersed energy, whereas existing features concentrate energy in the top-left DC component. SP_H has the highest energy concentration as it closely resembles images.

These distinct redundancy characteristics further demonstrate the necessity of the proposed dataset and suggest that future studies should explore feature coding methods tailored to these unique properties.

5. Unified test conditions

Even with a test dataset, fair comparisons are still challenging without identical test conditions. To address this, we establish unified test conditions here.

5.1. Bitrate computation

We introduce a new bitrate measurement, Bits Per Feature Point (BPFP), replacing the commonly used Bits Per Pixel (BPP). BPFP is calculated by dividing the total coding bits by the number of feature points. Our rationale for shifting from BPP to BPFP is twofold. First, BPP is unsuitable for features extracted from non-visual data where pixels do not exist. Second, we believe bitrate should reflect the actual encoded data (the feature itself) rather than the source data. Using BPP to calculate bitrate on source data introduces ambiguity and limits fair comparisons between coding methods. For instance, applying the same BPP to different features extracted from the same image can be misleading. In addition, BPP may incentivize downscaling source images to reduce feature size, yielding lower bitrates that do not represent true coding efficiency. In contrast, BPFP measures bitrate on the feature itself, enabling fair comparisons across coding methods as long as the same feature is used.

5.2. Task accuracy evaluation

The decoded features undergo task heads to perform accuracy evaluation. To isolate the impact of task heads, we choose simple task heads and fix their parameters during evaluation. The task heads and accuracy metrics are presented in Table 2.

6. Baselines and benchmark

As a starting point, we introduce two baselines and establish a benchmark. We aim to examine their applicability to

Task	Image Classification			Semantic Segmentation			Depth Estimation			Common Sense Reasoning			Text-to-Image Synthesis		
Metric	**BPFP**	**Accuracy**	**MSE**	**BPFP**	**mIoU**	**MSE**	**BPFP**	**RMSE**	**MSE**	**BPFP**	**Accuracy**	**MSE**	**BPFP**	**FID**	**MSE**
Original	32	100	0	32	83.39	0	32	0.3695	0	32	100	0	32	0	0
VTM Baseline	1.94	99.80	2.9499	1.76	80.42	1.8668	2.21	0.5021	0.6108	2.69	99.80	0.0109	1.41	6.14	0.0066
	1.03	97.40	3.1693	0.88	79.31	2.0836	1.27	0.6511	0.6826	1.83	100	0.0260	0.74	19.94	0.0184
	0.23	74.60	3.6993	0.23	73.47	2.5440	0.33	0.9530	0.9225	0.89	98.80	0.0810	0.29	58.42	0.0421
	0.04	25.80	4.0651	0.04	56.53	2.9526	0.03	1.4850	1.0631	0.16	82.20	0.1868	0.11	109.23	0.0715
	0.01	7.20	4.4400	0.01	37.42	3.2762	0.003	2.1174	1.1072	0.04	23.80	0.2455	0.05	171.63	0.1072
Hyperprior Baseline	2.01	93.20	3.4501	1.71	77.96	2.2010	1.51	0.4256	0.6600	6.34	91.40	0.0776	1.44	25.25	0.0138
	1.13	88.80	3.6966	1.30	77.27	2.2795	1.01	0.4965	0.6844	3.60	87.80	0.0808	0.66	52.12	0.0300
	0.91	83.60	3.8327	0.54	74.82	2.5845	0.43	0.6699	0.7837	1.68	82.40	0.1622	0.28	95.03	0.0541
	0.37	29.00	4.2909	0.12	62.58	3.0151	0.08	1.0351	1.0844	1.50	57.80	0.1355	0.15	124.84	0.0734
	0.23	14.60	4.7814	0.03	37.11	3.5449	0.01	2.7302	1.1866	1.35	34.80	0.1624	0.08	170.01	0.1006

Table 3. Rate-accuracy (R-A) performance evaluations on the two baseline methods (MSE is calculated between F_o and $\hat{F}_r$).

Task	Split Point	Org. Region	Trun. R. (VTM)	Trun. R. (Hyperprior)
Cls	SP_{DS}	[-542.30, 94.14]	[-20, 20]	[-5, 5]
Seg	SP_{DS}	[-506.97, 105.95]	[-20, 20]	[-5, 5]
Dpt	SP_{DM1}	[-2.38, 3.27]	[-1, 1]	[-1, 1]
	SP_{DM2}	[-26.44, 5.03]	[-2, 2]	[-2, 2]
	SP_{DM3}	[-323.30, 25.05]	[-10, 10]	[-10, 10]
	SP_{DM4}	[-504.43, 100.27]	[-20, 20]	[-10, 10]
CSR	SP_G	[-71.50, 47.75]	[-5, 5]	[-5, 5]
TTI	SP_H	[-5.79, 4.46]	[-5.79, 4.46]	[-5.79, 4.46]

Table 4. Summary of the original and truncation region settings.

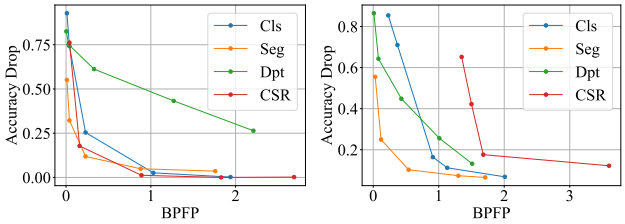

Figure 5. Rate-accuracy-drop (R-AD) comparisons among different tasks. Left: VTM baseline. Right: Hyperprior baseline.

large model feature coding and provide insights for interested researchers.

6.1. The baseline methods

The pipeline for the proposed baselines, illustrated in Fig. 4, includes pre-processing, core codec, and post-processing modules. First, the original feature F_o is truncated to a smaller range and then quantized into integers. Next, the quantized F_q is packed into a 2D feature F_p. The codec takes pre-processed F_p as the input and outputs decoded feature $\hat{F}_p$. Two core codecs are used: a handcrafted codec VTM and a learning-based codec Hyperprior. In the post-processing module, $\hat{F}_p$ is unpacked to $\hat{F}_q$ and then converted back into a floating-point feature $\hat{F}_r$.

6.1.1. Pre-processing and post-processing

The truncation operation is proposed to remove feature points that deviate significantly from the central region. The specific truncated regions for each task are listed in Table 4. Different truncations are used for the two baselines for their distinct coding strategies. The quantization operation uniformly quantizes the truncated features to 10-bit integers. Given the input requirement of VTM, we pack the features into a 2D YUV-400 format. The same packing applies in the Hyperprior baseline. F_{Seg}, F_{Dpt}, and F_{TTI} are packed into the shapes of 2740×1536, 3222×6144, and 512×512, respectively. Please refer to supplementary materials for the packing details. The post-processing performs unpacking and de-quantization, which are inverse processes of packing and quantization.

6.1.2. Encoding and decoding

VTM baseline: We employ VTM-23.3 as the codec. The Intra coding is applied using the main configuration file *encoder_intra_vtm.cfg*. The *InputChromaFormat* is set to *400*, and *ConformanceWindowMode* is enabled. The *InternalBitDepth*, *InputBitDepth*, and *OutputBitDepth* are all set to *10*. All other configurations are set to default. In our experiments, five QPs $\{22, 27, 32, 37, 42\}$ are used.

Hyperprior baseline: We modify the channel of the input and output of the original Hyperprior as 1 to accommodate 2D features. The loss function is defined as:

$$L = BPFP + \lambda \times ||(F_o - \hat{F}_r)||^2 \quad (1)$$

where λ is a scaling factor used to adjust the bitrate. Please find more details in the supplementary.

6.2. Rate-accuracy (R-A) analysis

The R-A evaluation results are presented in Table 3. Both baselines can achieve significant bitrate savings, while the VTM baseline outperforms the Hyperprior baseline. To further assess the impact of bitrate on accuracy, we introduce a new metric, accuracy drop (AD), defined as the percentage decrease in accuracy. For Dpt, the reciprocal of RMSE is employed. The R-AD curves, shown in Fig. 5, reveal

Codec	Cls	Seg	Dpt	CSR	TTI
VTM Baseline	0.9413	0.8955	0.8549	0.7490	0.9983
Hyperprior Baseline	0.9296	0.9280	0.7437	0.4713	0.9996

Table 5. Distortion-accuracy (D-A) analysis on the two baselines.

that most tasks exhibit an inflection point where accuracy starts to decrease significantly. However, for Dpt, this point is observed only in the Hyperprior baseline. In the Hyperprior baseline, inflection points are more distracted across tasks, which may caused by the fact that different models are trained for different tasks. TTI is excluded as images generated from the original features have an FID of 0.

6.3. Distortion-accuracy (D-A) analysis

We evaluate the linear correlation between feature MSE (D) and task accuracy (A) and present coefficients of determination in Table 5. Among the baselines, only TTI exhibits high linear correlations. This is reasonable since TTI involves generating images, and the baselines employ image codecs. In contrast, CSR fails to achieve a high linear correlation, as its features are derived from textual data. Overall, the VTM baseline outperforms the Hyperprior baseline, demonstrating its superior adaptability to large model features. Our analysis reveals that feature MSE struggles to characterize semantic distortion, particularly for textual data. This finding suggests that feature MSE is not an effective metric for measuring semantic distortion.

6.4. Complexity analysis

We run the VTM baseline and Hyperprior baseline on an Intel® Xeon® E5-2690 v4 CPU and a single NVIDIA GeForce RTX 4090 GPU, respectively. The VTM baseline exhibits encoding times ranging from a few seconds to several hours, while its decoding time remains within a few seconds. In contrast, the Hyperprior baseline exhibits comparable encoding and decoding times, most of which are on the order of milliseconds. Detailed results are provided in the supplementary. We note that runtime depends heavily on both hardware capabilities and implementation details. A well-optimized codec and high-performance hardware can significantly accelerate feature coding.

6.5. Generalizability analysis

We assess the generalizability of trained models by compressing features extracted from one task using models trained on a different task and evaluating their R-A performance. We select three tasks and present the results in Table 6. Overall, for a given task (along column), three kinds of models behave differently in both bitrate and accuracy. For example, models trained on CSR struggle to achieve low bitrates when applied to Seg and TTI, while models trained on Seg fail to achieve low FID in TTI. More analyses can be found in the supplementary. In practical applications, an effective compression model should be capable of handling diverse inputs. Therefore, a compression method with high generalizability is highly desirable.

Task	Seg		CSR		TTI	
Models	**BPFP**	**mIoU**	**BPFP**	**Acc.**	**BPFP**	**FID**
Trained on Seg	1.71	77.96	2.08	83.80	0.97	119.83
	1.30	77.27	1.37	66.60	0.62	208.50
	0.54	74.82	0.78	0.20	0.31	333.75
	0.12	62.58	0.25	0.00	0.12	363.32
	0.03	37.11	0.09	0.00	0.05	354.22
Trained on TTI	2.58	78.30	2.28	97.60	1.44	25.25
	1.96	76.34	1.46	77.80	0.66	52.12
	1.12	73.37	0.65	32.00	0.28	95.03
	0.53	56.90	0.30	0.00	0.15	124.84
	0.26	42.94	0.11	0.00	0.08	170.01
Trained on CSR	5.29	78.20	6.34	91.40	4.77	58.19
	3.27	77.94	3.60	87.80	2.59	73.19
	1.80	76.77	1.68	82.40	1.02	196.56
	1.73	75.82	1.50	57.80	1.02	191.69
	1.54	75.80	1.35	34.80	0.94	193.08

Table 6. Generalizability evaluation on Seg, CSR, and TTI tasks.

6.6. Discussion

Large model applications require a balance of high accuracy, low complexity, and strong generalizability. However, both baselines struggle to maintain high accuracy at low bitrates, underscoring the need for more efficient feature coding strategies tailored to the unique characteristics of large model features. In addition, semantic distortion measurement remains a critical challenge in feature coding. Advancing semantic distortion metrics could further optimize codec performance and enhance overall coding efficiency.

The two baseline approaches present distinct advantages and drawbacks. The handcrafted approach aligns seamlessly with existing codecs but suffers from excessive runtime, limiting its real-time applicability. In contrast, the learning-based approach benefits from efficient GPU acceleration but lacks generalizability, as separate models must be trained for different features. Therefore, developing lightweight and generic codecs remains a promising research direction for feature coding.

7. Conclusion

In this paper, we highlight the importance of feature coding in large model deployments and introduce a new research area: large model feature coding. As the first step, we provide a test dataset, unified test conditions, baseline methods, and a benchmark. We examine the applicability of the two baselines and identify promising directions for future research. Our aim is to encourage collaboration between image/video coding and large model communities to drive the development of feature coding.

References

[1] Johannes Ballé, David C. Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *ArXiv*, abs/1802.01436, 2018. 2

[2] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021. 2

[3] Yangang Cai, Peiyin Xing, and Xuesong Gao. High efficient 3D convolution feature compression. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022. 2

[4] Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. When federated learning meets privacy-preserving computation. *ACM Computing Surveys*, 56(12): 1–36, 2024. 1, 3

[5] Qiaoxi Chen, Changsheng Gao, and Dong Liu. End-to-end learned scalable multilayer feature compression for machine vision tasks. In *ICIP*, pages 1781–1787, 2024. 2

[6] Zhuo Chen, Kui Fan, Shiqi Wang, Lingyu Duan, Weisi Lin, and Alex Chichung Kot. Toward intelligent sensing: Intermediate deep feature compression. *IEEE Transactions on Image Processing*, 29:2230–2243, 2020. 1

[7] Hyomin Choi and Ivan V. Bajić. Latent-space scalability for multi-task collaborative intelligence. In *ICIP*, pages 3562–3566, 2021. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: a large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[9] Ronan Docherty, Antonis Vamvakeros, and Samuel J Cooper. Upsampling DINOv2 features for unsupervised vision tasks and weakly supervised materials segmentation. *arXiv preprint arXiv:2410.19836*, 2024. 3

[10] Lingyu Duan, Jiaying Liu, Wenhan Yang, Tiejun Huang, and Wen Gao. Video coding for machines: A paradigm of collaborative compression and intelligent analytics. *IEEE Transactions on Image Processing*, 29:8680–8695, 2020. 2

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 2, 4, 6

[12] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 2, 4, 6

[13] Ruoyu Feng, Xin Jin, Zongyu Guo, Runsen Feng, Yixin Gao, Tianyu He, Zhizheng Zhang, Simeng Sun, and Zhibo Chen. Image coding for machines with omnipotent feature learning. In *ECCV*, pages 510–528. Springer, 2022. 2

[14] Othmane Friha, Mohamed Amine Ferrag, Burak Kantarci, Burak Cakmak, Arda Ozgun, and Nassira Ghoualmi-Zine. LLM-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*, 5:5799–5856, 2024. 1, 3

[15] Changsheng Gao, Dong Liu, Li Li, and Feng Wu. Towards task-generic image compression: A study of semantics-oriented metrics. *IEEE Transactions on Multimedia*, 25:721–735, 2023. 2

[16] Changsheng Gao, Yiheng Jiang, Li Li, Dong Liu, and Feng Wu. DMOFC: discrimination metric-optimized feature compression. In *PCS*, pages 1–5, 2024. 1

[17] Changsheng Gao, Zhuoyuan Li, Li Li, Dong Liu, and Feng Wu. Rethinking the joint optimization in video coding for machines: A case study. In *DCC*, pages 556–556, 2024. 2

[18] Changsheng Gao, Yiheng Jiang, Siqi Wu, Yifan Ma, Li Li, and Dong Liu. IMOFC: identity-level metric optimized feature compression for identification tasks. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(2):1855–1869, 2025. 1

[19] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023. 1

[20] Chia-Hao Kao, Cheng Chien, Yu-Jen Tseng, Yi-Hsin Chen, Alessandro Gnutti, Shao-Yuan Lo, Wen-Hsiao Peng, and Riccardo Leonardi. Bridging compressed image latents and multimodal large language models. *arXiv preprint arXiv:2407.19651*, 2024. 2

[21] Yeongwoong Kim, Hyewon Jeong, Janghyun Yu, Younhee Kim, Jooyoung Lee, Se Yoon Jeong, and Hui Yong Kim. End-to-end learnable multi-scale feature compression for VCM. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2023. 2

[22] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 5

[23] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. 1

[24] Shibao Li, Chenxu Ma, Yunwu Zhang, Longfei Li, Chengzhi Wang, Xuerong Cui, and Jianhang Liu. Attention-based variable-size feature compression module for edge inference. *The Journal of Supercomputing*, 2023. 2

[25] Zhuoyuan Li, Junqi Liao, Chuanbo Tang, Haotian Zhang, Yuqi Li, Yifan Bian, Xihua Sheng, Xinmin Feng, Yao Li, Changsheng Gao, et al. Ustc-td: A test dataset and benchmark for image and video coding in 2020s. *arXiv preprint arXiv:2409.08481*, 2024. 2

[26] Zhuoyuan Li, Zikun Yuan, Li Li, Dong Liu, Xiaohu Tang, and Feng Wu. Object segmentation-assisted inter prediction for versatile video coding. *IEEE Transactions on Broadcasting*, 2024. 2

[27] Tie Liu, Mai Xu, Shengxi Li, Chaoran Chen, Li Yang, and Zhuoyi Lv. Learnt mutual feature compression for machine vision. In *ICASSP*, pages 1–5, 2023. 2

[28] Guo Lu, Xingtong Ge, Tianxiong Zhong, Qiang Hu, and Jing Geng. Preprocessing enhanced image compression for machine vision. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2024. 2

[29] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 35 (7):8726–8746, 2024. 3

[30] Yifan Ma, Changsheng Gao, Qiaoxi Chen, Li Li, Dong Liu, and Xiaoyan Sun. Feature compression with 3d sparse convolution. In *VCIP*, pages 1–5, 2024. 2

[31] Rui Mao, Xinmin Feng, Changsheng Gao, Li Li, Dong Liu, and Xiaoyan Sun. Perceptual image compression with conditional diffusion transformers. In *VCIP*, pages 1–5, 2024. 2

[32] Kiran Misra, Tianying Ji, Andrew Segall, and Frank Bossen. Video feature compression for machine tasks. In *ICME*, pages 1–6, 2022. 2

[33] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2, 4, 6

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 3

[35] Xihua Sheng, Li Li, Dong Liu, and Houqiang Li. Vnvc: A versatile neural video coding framework for efficient human-machine vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):4579–4596, 2024. 2

[36] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-CLIP: improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*, 2023. 3

[37] Satoshi Suzuki, Shoichiro Takeda, Motohiro Takagi, Ryuichi Tanida, Hideaki Kimata, and Hayaru Shouno. Deep feature compression using spatio-temporal arrangement toward collaborative intelligent world. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(6):3934–3946, 2022. 2

[38] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. FedBERT: when federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology*, 13(4):1–26, 2022. 1

[39] Yuan Tian, Guo Lu, Guangtao Zhai, and Zhiyong Gao. Non-semantics suppressed mask learning for unsupervised video semantic compression. In *ICCV*, pages 13564–13576, 2023. 2

[40] Yuan Tian, Guo Lu, and Guangtao Zhai. SMC++: masked learning of unsupervised video semantic compression. *arXiv preprint arXiv:2406.04765*, 2024. 2

[41] Yuan Tian, Guo Lu, and Guangtao Zhai. Free-VSC: free semantics from visual foundation models for unsupervised video semantic compression. In *ECCV*, pages 163–183, 2025. 2

[42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2

[43] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018. 1, 3

[44] Shurun Wang, Shiqi Wang, Wenhan Yang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. Towards analysis-friendly face representation with scalable feature and texture compression. *IEEE Transactions on Multimedia*, 24:3169–3181, 2022. 1

[45] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-any multimodal LLM. In *ICML*, pages 53366–53397, 2024. 5

[46] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. On protecting the data privacy of large language models (LLMs): A survey. *arXiv preprint arXiv:2403.05156*, 2024. 3

[47] Ning Yan, Changsheng Gao, Dong Liu, Houqiang Li, Li Li, and Feng Wu. SSSIC: Semantics-to-signal scalable image coding with learned structural representations. *IEEE Transactions on Image Processing*, 30:8939–8954, 2021. 2

[48] Wenhan Yang, Haofeng Huang, Yueyu Hu, Ling-Yu Duan, and Jiaying Liu. Video coding for machines: Compact visual representation compression for intelligent collaborative analytics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2024. 2

[49] Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. OpenFedLLM: Training large language models on decentralized private data via federated learning. In *ACM SIGKDD*, pages 6137–6147, 2024. 1, 3

[50] Xu Zhang, Peiyao Guo, Ming Lu, and Zhan Ma. All-in-one image coding for joint human-machine vision with multi-path aggregation. *arXiv preprint arXiv:2409.19660*, 2024. 2

[51] Zhicong Zhang, Mengyang Wang, Mengyao Ma, Jiahui Li, and Xiaopeng Fan. MSFC: Deep feature compression in multi-task network. In *ICME*, pages 1–6, 2021. 2

[52] Jiaying Zheng, Hainan Zhang, Lingxiang Wang, Wangjie Qiu, Hongwei Zheng, and Zhiming Zheng. Safely learning with private data: A federated learning framework for large language model. *arXiv preprint arXiv:2406.14898*, 2024. 1, 3