SemanticNN: Compressive and Error-Resilient Semantic Offloading for Extremely Weak Devices

Jiaming Huang, Yi Gao*, Fuchang Pan, Renjie Li, and Wei Dong*

Zhejiang Key Lab. of Accessible Perception and Intelligent Systems
College of Computer Science, Zhejiang University, China
{huangjm, gaoyi, panfc, lirj, dongw}@zju.edu.cn

Abstract

With the rapid growth of the Internet of Things (IoT), integrating artificial intelligence (AI) on extremely weak embedded devices has garnered significant attention, enabling improved real-time performance and enhanced data privacy. However, the resource limitations of such devices and unreliable network conditions necessitate error-resilient device-edge collaboration systems. Traditional approaches focus on bit-level transmission correctness, which can be inefficient under dynamic channel conditions. In contrast, we propose SemanticNN, a semantic codec that tolerates bit-level errors in pursuit of semantic-level correctness, enabling compressive and resilient collaborative inference offloading under strict computational and communication constraints. It incorporates a Bit Error Rate (BER)-aware decoder that adapts to dynamic channel conditions and a Soft Quantization (SQ)-based encoder to learn compact representations. Building on this architecture, we introduce Feature-augmentation Learning, a novel training strategy that enhances offloading efficiency. To address encoder-decoder capability mismatches from asymmetric resources, we propose XAI-based Asymmetry Compensation to enhance decoding semantic fidelity. We conduct extensive experiments on STM32 using three models and six datasets across image classification and object detection tasks. Experimental results demonstrate that, under varying transmission error rates, SemanticNN significantly reduces feature transmission volume by 56.82–344.83× while maintaining superior inference accuracy.

Code — https://github.com/zju-emnets/SemanticNN
Extended version — https://arxiv.org/abs/2511.11038

1 Introduction

Internet of Things (IoT) has become increasingly prevalent across a wide range of areas, including manufacturing, agriculture, transportation, and healthcare (Quy et al. 2022; Breda et al. 2023; Ling et al. 2021; Jiang et al. 2021). Deploying intelligent capabilities on weak embedded devices in these settings enhances real-time responsiveness and operational efficiency (Yao et al. 2020; Huang and Gao 2022; Maj et al. 2022; Benazir, Xu, and Lin 2024). For instance, deploying neural networks (NNs) on small drones equipped with IoT cameras enables real-time anomaly detection, which benefits applications such as fire detection, wildlife monitoring, and traffic surveillance (Huang and Gao 2022; Hojjat et al. 2024).

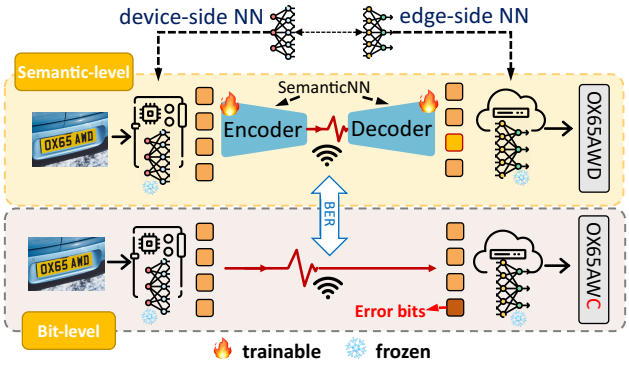


Figure 1: SemanticNN (including encoder and decoder) accepts bit errors in wireless transmissions, pursuing semantic correctness with extremely feature compression in split computing. The original task NN will be frozen, and only the SemanticNN will be optimized.

Nevertheless, executing AI algorithms on weak devices remains challenging due to severe constraints on computation, memory, and power (He et al. 2016; stm 2023). For example, ResNet-50 requires at least 100MB of memory, while a typical embedded device STM32, offers less than 1MB of local memory. To bridge this huge gap, techniques such as TinyML, model compression, and pruning (Kallimani et al. 2024; Zhang, Gu, and Zhang 2021; Lin et al. 2020) have been widely adopted. However, they may lead to non-negligible accuracy loss (Frankle et al. 2020; Huang and Gao 2022) due to aggressive simplification of NN. For example, TinyML achieves over 50% compression via low-precision quantization, but this comes at the cost of reduced model expressiveness and degraded task performance.

Recent advances in split computing (Kang et al. 2017; Laskaridis et al. 2020; Huang et al. 2021; Benazir, Xu, and Lin 2024; Bin et al. 2024) significantly mitigate accuracy degradation by offloading parts of NN computations from weak devices to edge servers. Concretely, the original NN is split into the device-side NN and the edge-side NN, with

*Corresponding authors.

the intermediate features extracted on the device offloaded to the edge for further inference. Existing works primarily focus on optimizing model partitioning strategies and compressing device-side outputs to reduce both computational load and communication overhead.

However, these offloaded features are typically compressed before transmission to reduce bandwidth usage, making them highly vulnerable to bit errors introduced by wireless communication (Nogales et al. 2018; Becke et al. 2012). Even a small number of flipped bits can lead to significant semantic drift in downstream predictions. For instance, causing a speech-to-text system to misinterpret "temperature" as "temperate". This sensitivity highlights the importance of transmission robustness in AI systems.

Traditional bit-level error correction techniques, such as network coding(Wicker and Bhargava 1999; MacKay 2005) and retransmission protocols, are designed to ensure bit-level reliability for upper-layer applications. However, they could be impractical for resource-constrained devices, particularly under extremely dynamic or unreliable wireless links (Yun, Kim, and Tan 2010). Orthogonal to bit-level approaches (Bragilevsky and Bajić 2020; Bajić 2021; Dhondea, Cohen, and Bajić 2021; Itahara et al. 2022; Wang and Zhang 2022), we propose a novel paradigm based on semantic-level correctness.

In this paper, we introduce *SemanticNN*, a semantic codec comprising an encoder and decoder, designed for error resilient *Semantic N*eural *N*etwork offloading. Unlike traditional bit-level approaches, SemanticNN treats bit errors as acceptable as long as the essential semantic content is preserved. As illustrated in Figure 1, our approach extends Shannon's second level of communication (Shannon 1948) by defining semantic correctness in terms of end-to-end inference accuracy. The encoder compresses the output of the device-side NN, while the decoder reconstructs the received features in the presence of transmission errors. Thus, the primary objective of SemanticNN is to design an encoder-decoder framework capable of tolerating transmission errors. Two key challenges arise in designing SemanticNN.

The first stems from the combination of extreme feature compression and dynamic transmission errors. Specifically, extreme feature compression increases sensitivity to transmission errors. Moreover, this sensitivity is further amplified by the varying severity of transmission errors (e.g., multi-path interference in Wi-Fi), which hinder the accurate recovery of semantic information at the edge (Appendix A.1). To tackle this, we design a dedicated semantic codec consisting of a Soft Quantization (SQ)-based encoder and a BER-aware decoder. The encoder introduces a learnable, non-uniform quantization mechanism that allows end-to-end optimization of both the quantization and the NN parameters. This ensures that even under severe compression, the encoder retains the most semantically meaningful representations. The decoder integrates an adaptive attention module that dynamically responds to varying error rates, enabling robust reconstruction of corrupted features. Furthermore, we propose Feature-Augmentation Learning, a two-stage training framework. The first stage involves pre-training the codec using an autoencoder objective to denoise compressed features, serving as a universal initialization applicable to any downstream task at a given model split point. In the second stage, task-specific model fine-tuning refines the ability to capture high-level semantic features tailored to individual applications.

The second lies in the architectural asymmetry between the encoder and decoder. Due to resource constraints on weak devices, the encoder deployed at the edge is significantly simpler in structure than the decoder on the server. Our experiments in Appendix A.2 reveal that the decoder often has over ten times the capacity of the encoder. While deeper decoding models generally offer better decoding performance, the limited expressiveness of the lightweight encoder results in suboptimal feature extraction. This architectural asymmetry significantly degrades the task performance. To address this, we propose asymmetry compensation based on eXplainable AI (XAI) techniques for the encoder in SemanticNN. By leveraging explainability maps generated through XAI techniques, the encoder identifies and prioritizes the most informative regions of the feature maps. This lightweight strategy enables the encoder to enhance its semantic feature extraction capability without introducing additional computational or memory overhead.

Our main contributions can be summarized as follows:

- We propose SemanticNN, a semantic codec for error resilient compressive device-edge collaboration, comprising SQ-based encoder and BER-aware decoder, trained through Feature-augmentation Learning.
- We introduce XAI-based Asymmetry Compensation that enables the weak encoder in SemanticNN to refine feature map based on explainability analysis for extracting more valuable information without extra overhead.
- We conduct extensive experiments and the results show that SemanticNN drastically reduces the offloaded features by 56.82-344.83$\times$ while outperforming all baselines. Real-world case study further demonstrates its robustness to dynamic BERs.

The rest is organized as follows. Section 2 reviews related work. Section 3 shows SemanticNN architecture and training process. Section 4 details the encoder compensation mechanism. Section 5 presents experimental results and real-world cases. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Efficient On-device Inference

Model Compression. To align the computational requirements of tasks with the weak devices, researchers have pursued model compression techniques (Lin et al. 2020; Hinton et al. 2015; Sau and Balasubramanian 2016), which reduce model size and computational overhead while maintaining accuracy. Alternatively, many lightweight architectures (Sandler et al. 2018) are designed directly to achieve efficiency without compression.

However, when deploying models on weak devices, existing approaches often oversimplify the architecture, leading to significant accuracy degradation (Huang and Gao 2022; Zhuo et al. 2022).

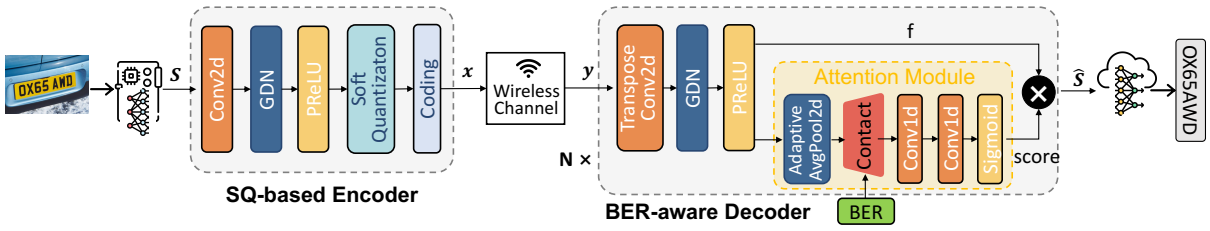


Figure 2: The detailed model structures of the SQ-based encoder and BER-aware decoder in SemanticNN.

Device-edge collaboration. Research on device-edge collaboration has explored optimization of computation allocation and communication overhead between the device and edge. For communication efficiency, techniques like semantic caching (Xu et al. 2018), input filtering, and semantic similarity indexing (Yuan et al. 2022; Kang et al. 2022) have been proposed. For computation, early approaches (Kang et al. 2017; Li et al. 2018) introduced partial NN offloading to enable collaborative inference, paving the way for split learning. Subsequent work (Laskaridis et al. 2020; Huang et al. 2021), building upon the early-exit mechanism (Teerapittayanon, McDanel, and Kung 2016), adopted progressive inference to generate intermediate outputs. Further optimizations (Yao et al. 2020; Huang and Gao 2022) compressed intermediate data to enhance real-time performance.

However, none of the aforementioned works take into consideration the fact that the potential occurrence of transmission errors can substantially affect task performance.

2.2 Error-Resilient Communication

Shannon's theory of communication defines three hierarchical levels (Shannon 1948). The first focuses on the *physical* transmission of bits. The second emphasizes the correct conveyance of *semantic* meaning. This paper leverages semantic-level communication for robust split AI inference.

Bit-Level Reliable Communication. A wide range of techniques have been developed to improve the reliability and efficiency of bit-level transmission, including sophisticated network coding and modulation schemes (Svensson 2007; Sundararajan, Shah, and Médard 2008). With the rise of AI, recent efforts have extended these communication principles for AI tasks. For example, tensor completion methods (Bajić 2021; Dhondea, Cohen, and Bajić 2021) aim to recover lost packets during transmission, while NeuroMessenger (Wang and Zhang 2022) improves resilience under low SNR conditions through adaptive retransmission mechanisms. BottleNet++ (Shao and Zhang 2020) further integrates Joint Source Channel Coding (JSCC) into original AI models, enabling end-to-end optimization for error-resilient bit transmission.

However, in long-distance or interference-prone scenarios, achieving perfect bit-level fidelity often requires highly conservative coding strategies. These methods, while effective in minimizing transmission errors, can incur significant overhead due to the dynamic nature of wireless channels.

Semantic-Level Reliable Communication. In contrast to prior approaches prioritizing bit-level accuracy, semantic-level communication tolerates minor bit errors as long as the essential semantic meaning is preserved. Bit-level and semantic-level optimizations are orthogonal, i.e., combining them can yield complementary benefits (Shao, Mao, and Zhang 2021; Lee et al. 2019). Recent advances leverage deep learning (DL) to extract semantic information directly from data, forming the basis of semantic communication (Bourtsoulatze, Kurka, and Gündüz 2019; Xie et al. 2021). Most DL-based frameworks adopt the JSCC paradigm, which jointly optimizes source encoding, channel characteristics, and task-specific decoding. This design enables efficient transmission of multimodal data under poor channel conditions (Xie et al. 2021; Xu et al. 2021; Weng et al. 2023; Han et al. 2022; Dai et al. 2022).

SemanticNN aligns with this trend, but introduces that instead of transmitting raw data, semantic communication can be applied to intermediate feature maps in split computing. This approach enhances error resilience and supports extreme compression for deploying AI on weak devices.

3 Building the Model

In this paper, we focus on resource-constrained devices and fix the split after the first convolutional layer. Given the large number of possible split positions and the fact that earlier splits cause greater performance degradation (Appendix A.2), this configuration represents the most challenging scenario for compressive offloading under transmission errors.

3.1 Model Structure

The model architecture of SemanticNN forms the foundation for achieving efficient and error-resilient semantic offloading. As illustrated in Figure 2, it consists of an SQ-based encoder and a BER-aware decoder.

SQ-based Encoder The *SQ-based encoder* is responsible for compressing feature representations under extreme resource constraints. It consists of one convolutional layer, followed by Generalized Divisive Normalization (GDN), Parametric Rectified Linear Unit (PReLU), and a learnable soft quantization layer coupled with fixed-length bit coding.

NN Operations. GDN is a normalization technique, and it has been widely adopted in perceptual compression tasks due to its effectiveness in capturing statistical dependencies among neural activations. PReLU further improves the expressiveness of the encoder by mitigating the *dying ReLU* problem, allowing a small gradient for negative inputs to maintain neuron activity during training.

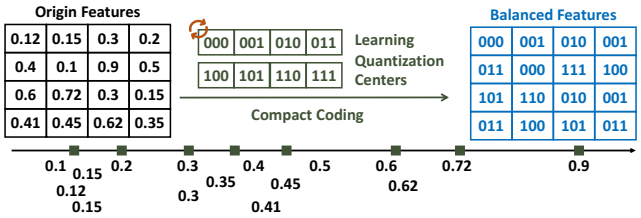


Figure 3: Balanced quantization with eight learning quantization centers and 3-bit compact coding.

Feature Compression Mechanism. The soft quantization and bit coding perform further compression on extracted features to reduce transmission overhead. Motivated by findings in Appendix A.1, we adopt fixed-length coding to enhance stability and avoid error propagation caused by variable-length codes like Huffman coding.

As illustrated in Figure 3 (left), traditional hard (non-uniform) quantization is employed to partition the feature space into intervals, each represented by a centroid. However, it introduces two major limitations: (1) suboptimal centroid placement when dealing with skewed or outlier-prone distributions, and (2) it is non-differentiable, which blocks gradient propagation during training. To address these issues, we introduce a learnable soft quantization mechanism that enables end-to-end optimization of both the quantization parameters and the overall network. As shown in Figure 3, the encoder learns eight quantization centers, each representable using three bits, through backpropagation. This quantization process is treated as a differentiable neural layer, allowing joint optimization of model parameters and data distribution statistics.

BER-aware Decoder The *decoder* comprises N blocks of similar operations to the encoder supplemented by an attention module each. Corresponding to the encoding operation, we utilize transpose convolution and inverse GDN to reconstruct the intermediate feature maps during the decoding process. The *attention module* in the decoder calculates the attention score to quantify the level of attention paid to intermediate features under the prevailing noise conditions. It accepts input from the intermediate features and the current communication state (i.e., BER), learning the correlation between the transmission error and the reconstructed feature. With this attention module, SemanticNN to dynamically adapt to unpredictable changing communication link environments. Through iterative repetitions of this process, the intended offloaded feature map will be restored at the receiver size. Finally, the more complex trained decoder will be deployed to edge servers as the receiver.

3.2 Feature-augmentation Learning

The designed model needs to be trained to compress offloaded activations efficiently while preserving semantic features crucial for task performance. We propose progressive Feature-augmentation Learning. It first trains the denoising autoencoder to achieve initial noise reduction while reducing the possibility of overfitting for downstream tasks.

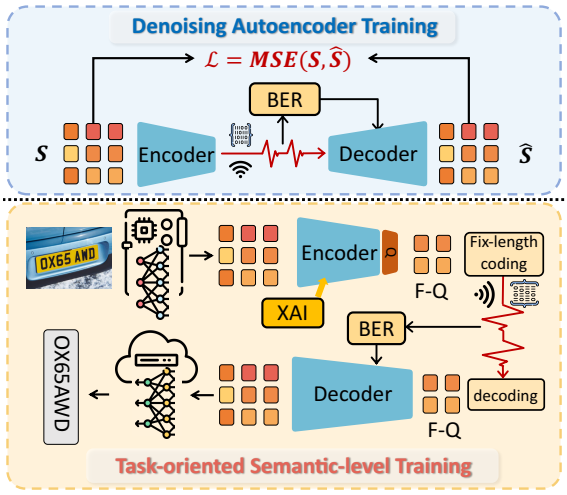


Figure 4: Feature-augmentation Learning first performs Denoising Autoencoder Training to initially fight against transmission errors. Subsequently, Task-Oriented Semantic-level Training is performed to extract and transmit semantic information on specific downstream tasks for better performance.

Subsequently, Task-oriented semantic-level training is performed on the downstream task to extract semantic information while adapting to transmission errors. Figure 4 shows the details of the training process.

Denoising Autoencoder Training This phase aims to align transmitted and received features at the bit level, enhancing resilience to counter interference caused by transmission errors. As illustrated on the top of Figure 4, we introduce Mean Squared Error (MSE) to minimize the difference as shown in Equation 1:

$$\mathcal{L}_{\text{bit}} = \text{MSE}\left(\mathbf{s}, \hat{\mathbf{s}}\right). \tag{1}$$

As training progresses and $\mathcal{L}_{\text{bit}}$ diminishes, the encoder and decoder gradually converge toward a consistent representation of the input features, achieving bit-level alignment. This pre-training stage enables SemanticNN to develop an initial resilience to transmission errors, endowing it with error-mitigation capabilities even on unseen tasks. Furthermore, this process significantly enhances the learning efficiency of downstream task adaptation while reducing the risk of overfitting during semantic feature extraction. Notably, this phase is entirely task-agnostic and depends only on the size of the intermediate feature maps at the offloading point. Therefore, encoders and decoders trained under this phase can be directly reused across different tasks, as long as the feature dimensions match.

Task-Oriented Semantic-level Training At this phase, we draw on the idea of semantic communication to transmit the most task-relevant information with a degree of tolerance for bit errors for better task performance. SemanticNN and the origin NN model are both considered for the Task-oriented Semantic-level Training, which achieves the goals of error-resilient compressed semantic inference. The origin NN model is solely engaged during the training process

to produce intermediate features and does not participate in the subsequent gradient updates. The performance of tasks is safeguarded by aligning the offloaded inference results at the edge with the ground truth. It enables the transmission of the most task-relevant data. To further improve the efficiency of fixed-length coding, we incorporate a distribution regularization term into the loss function. Given the task-dependent nature of feature distributions, we define the semantic-level training objective as:

$$\mathcal{L}_{\text{sem}} = \alpha \cdot \mathcal{L}_{\text{div}}\left(\mathbf{p}, \frac{1}{n}\mathbf{1}\right) + \beta \cdot \mathcal{L}_{\text{cls}}(y, \hat{y}) + \gamma \cdot \mathcal{L}_{\text{XAI}}, \quad (2)$$

where n is the number of quantization centers and $\mathbf{p} \in \mathbb{R}^n$ denotes the normalized frequency distribution of their occurrences. The distribution regularization loss $\mathcal{L}_{\text{div}}$ encourages frequency distributions to approach a uniform distribution. The task-specific loss $\mathcal{L}_{\text{cls}}(y, \hat{y})$, such as cross-entropy (CE) for classification or MSE for regression, measures the discrepancy between the ground truth y and the predicted output $\hat{y}$. The $\mathcal{L}_{\text{XAI}}$ term, introduced in Section 4, promotes semantic interpretability by encouraging sparse and faithful feature attributions. Together, the multi-objective formulation balances model performance, feature utilization, and explainability. Hyperparameters are justified in Appendix F.

Adapting to transmission errors in denoising autoencoder training enhances the efficiency of semantic extraction and transmission in this stage. Bypassing the first stage and directly imposing this constraint on SemanticNN may lose the fight against transmission errors, potentially introducing instability. Appendix D shows the results of ablation studies. To further improve the ability of the asymmetric encoder in SemanticNN, we impose an additional loss constraint $\mathcal{L}_{\text{XAI}}$ in Equation 2, which will be introduced in Section 4.

4 XAI-based Asymmetry Compensation

XAI-based Asymmetry Compensation is primarily employed to optimize the performance of the asymmetric device-side encoder. We propose to enhance the encoder by extracting important semantic information and further compress the transmission data by removing non-important features with the help of external XAI tools, shown at the top of Figure 5.

Channel-level analysis (Huang and Gao 2022) utilizes XAI tools to differentiate between important and unimportant features at the channel level in task offloading scenarios. In contrast to conducting feature map importance analysis at the channel level (the bottom of Figure 5), SemanticNN proposed *pixel-level* feature map importance analysis, motivated by Appendix A.3.

The top of Figure 5 illustrates the detailed utilization of XAI tools in the training of the encoder and the decoder within SemanticNN. In contrast to channel-level analysis, SemanticNN involves pixel-level analysis, aiming to transmit crucial pixels that hold significance across multiple channels during the training process. SemanticNN harnesses XAI in two facets: firstly, to transmit fewer data incorporating a maximal number of important features, and secondly, to optimize the significance of these important features. This optimization of importance is shown as Equation 3, which is

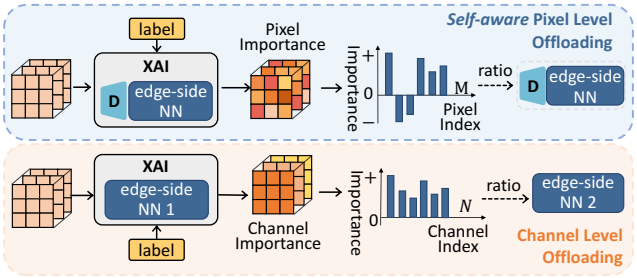


Figure 5: Self-aware pixel-level feature importance analysis using XAI and partial feature offloading.

Algorithm 1: Pixel-Level Feature Slicing

Input: Intermediate feature map $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, spatial ratio $\alpha \in (0, 1]$.
Output: Cropped feature map $\mathbf{F}_\alpha$ of size proportional to α.

$H \leftarrow \text{height}(\mathbf{F}), W \leftarrow \text{width}(\mathbf{F})$
$S_h, S_w \leftarrow \lfloor\sqrt{\alpha}, H\rfloor, \lfloor\sqrt{\alpha}, W\rfloor$
$p_h, p_w \leftarrow \left\lfloor\frac{H-S_h}{2}\right\rfloor, \left\lfloor\frac{W-S_w}{2}\right\rfloor$
$\mathbf{F}_\alpha \leftarrow \mathbf{F}[:, p_h : p_h + S_h, p_w : p_w + S_w]$
return $\mathbf{F}_\alpha$

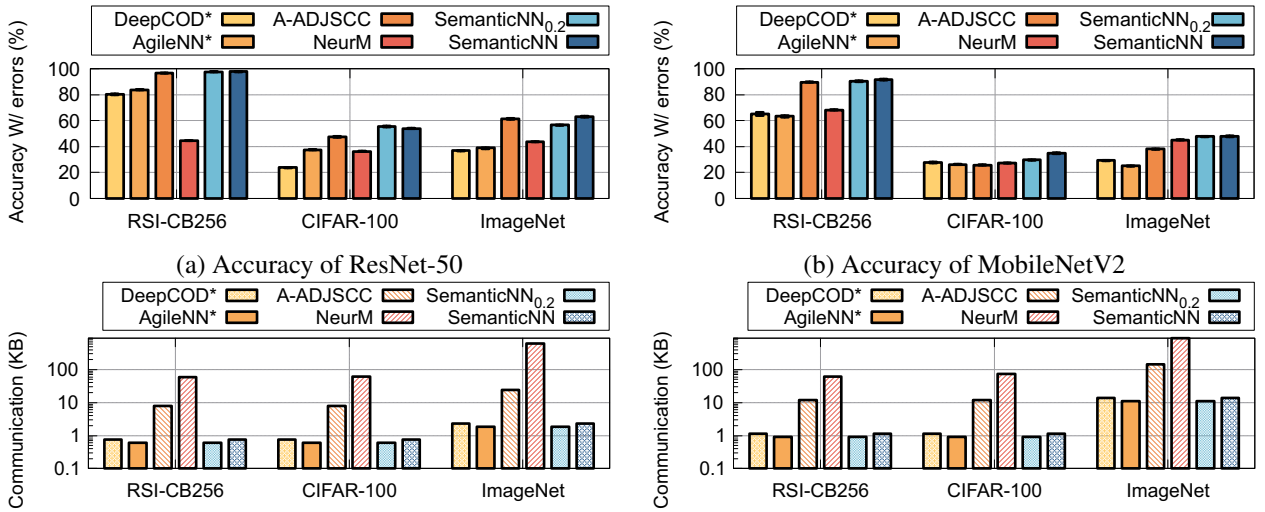
also added on Equation 2:

$$\mathcal{L}_{\text{XAI}} = \lambda \cdot \frac{1}{\sum_i \max(\phi_i, 0)} + (1-\lambda) \cdot \frac{M}{|\{i : \phi_i > 0\}|} - r, \quad (3)$$

where M is the total number of offloaded features, $\sum_i \max(\phi_i, 0)$ is the sum of positive attribution scores, $|\{i : \phi_i > 0\}|$ is the number of features with positive attributions, and r is a scaling factor that ensures the reward term is on a comparable scale.

SemanticNN aims to offload all balanced quantified features. Additionally, it also enables partial offloading (i.e., offloading features of a specified ratio to the edges) to adapt to constraint communication resources. We utilize pixel-level feature slicing as depicted in Algorithm 1. For all channels, we cut off the corresponding percentage of pixels. For slice ratio, it is best not to exceed 50%, and as Figure 13 in the Appendix reveals, around 50% features are important. Correspondingly, at the receiver end, SemanticNN employs the image optimization method known as *ReflectionPad2d* (Lin 2023) for recovering from the untransmitted pixels, which is more effective compared to traditional zero padding.

Besides, the most important component is the guidance model in XAI, which is used to characterize the importance of features. Indeed, different models assess the significance of features within the same input in varying ways. Therefore, instead of relying on another more potent model to ascertain input importance (i.e., edge-side NN 1 in Figure 5), we leverage the edge-side NN itself. With Equation 3 added in L_2, the entire NNs (i.e., original task models and SemanticNN) engaged in the end-to-end training process possess their interpretations of the input samples.



(a) Accuracy of ResNet-50

(b) Accuracy of MobileNetV2

(c) Offloaded features of ResNet-50

(d) Offloaded features of MobileNetV2

Figure 6: Overall performance of image classification. SemanticNN$_{0.2}$ and SemanticNN represent two versions, where SemanticNN$_{0.2}$ offloads 20% less amount of data than that of SemanticNN. SemanticNN achieves the best top-1 accuracy and less communication overhead with transmission errors across two pre-trained models and three datasets. We repeated 6 times, and the corresponding error bar indicates the stable task performance.

w/ E Performance	Image	Feature	SemanticNN
RSI-CB256 (%)	54.76	59.28	**97.92**
CIFAR-100 (%)	45.43	29.32	**53.96**
Data (KB)	48	64	**0.75**
ImageNet-200 (%)	44.24	32.84	**62.93**
Data (KB)	588	748	**2.3**
COCO (mAP)	0.28	0.48	**0.55**
VOC2007 (mAP)	0.44	**0.81**	0.80
VHR (mAP)	0.01	0.68	**0.81**
Data (MB)	4.69	1.17	**0.037**

Table 1: SemanticNN achieves outperforming classification performance with transmission errors while the amount of offloading data is only 0.39-1.56% of the original image and 0.31-1.17% of features in size. It also achieves satisfactory object detection performance while offloading only 0.79% of the original image.

5 Evaluation

5.1 Datasets and Setup

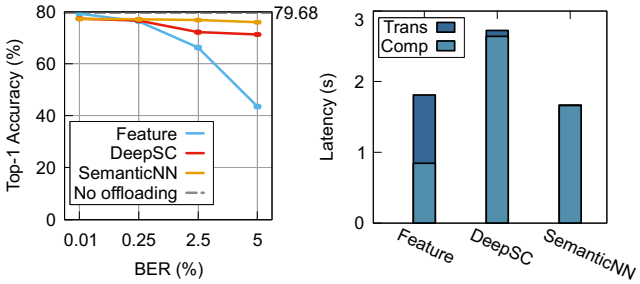
We implement SemanticNN with *PyTorch*. All models are trained on NVIDIA RTX-3090Ti GPUs and tested on STM32. We manually inject bit-flips to the offloaded features for evaluation under various BERs, acknowledging the complexity of controlling BER in real-world scenarios. According to real-world cases (Appendix E), we set the BER range to be 0.01-5% randomly. Due to space limitations, we have added the details of our experimental setup, datasets, metrics, and baselines in Appendix B.

5.2 Image Classification Performance

We compared the SemanticNN based on two base models with all baselines shown in Figure 6 for top-1 accuracy with transmission errors and communication overhead (i.e., features per image to be offloaded).

First, A-ADJSCC achieves slightly lower accuracy compared with two versions of SemanticNN, and outperforms the first two compressed baselines significantly. Since A-ADJSCC needs more than 10× the amount for offloading due to the absence of quantization and coding strategies, compared with SemanticNN and the first two compressed baselines. NeuroMessenger has a retransmission mechanism without quantization and coding strategies, resulting in the most communication overhead.

In split computing scenarios in IoT applications, the amount of offloaded data could be an important performance metric since the communication resource could be very constrained. In split computing, the amount of offloaded data is a vital metric due to potentially limited communication resources. The first two compressed baselines, i.e., DeepCOD* and AgileNN*, offload a similar amount of data compared with two versions of SemanticNN, which is only about 0.39% to 2.34% of the original image size. SemanticNN significantly outperforms the two baselines by 7.11% to 30.88% across various pre-trained models and datasets, showing the error resilience inference ability of SemanticNN. AgileNN is the best existing approach in terms of this metric, by proposing an XAI-based feature importance manipulation scheme. To compare the performance of AgileNN and SemanticNN fairly, SemanticNN$_{0.2}$ in Figure 6 is a version of SemanticNN which ensures the same amount of offloaded data compared with that of AgileNN. From the figure, we can see that although the accuracy of



(a) Under various BERs (b) Latency

Figure 7: CNN-based SemanticNN achieves better performance compared to transformer-based structures.

Params (KB)		Transmitted data (KB)	
DeepSC	SemanticNN	DeepSC	SemanticNN
9165	1875	98	24.5

Table 2: SemanticNN utilizes a more lightweight CNN structure and achieves efficient data transmission compared to Transformer-based DeepSC.

$\text{SemanticNN}_{0.2}$ is slightly lower than the accuracy of SemanticNN, $\text{SemanticNN}_{0.2}$ still outperforms AgileNN* significantly with the same amount of offloaded data. Despite the streamlined inference efficiency of MobileNetV2, it falls short of the accuracy of ResNet-50, although it sustains a competitive compression rate.

The other scenario is offloading without compressing images or features, as shown at the top of Table 1. SemanticNN outperforms these two baselines facing transmission errors. From the results, we can see that SemanticNN performs at least comparable with them, offloading only 0.39-1.56% of the original image and 0.31-1.17% of features in size. Moreover, considering realistic BER variations, real-world case studies further validate its effectiveness in Appendix E.

5.3 Object Detection Performance

We further evaluated SemanticNN using YOLOv3 on three varied datasets. Since the baselines used in the image classification scenario did not include open-source implementation for object detection tasks, we only compare SemanticNN with basic methods, i.e., offloading the original image or feature map directly without any compression. The bottom of Table 1 shows that SemanticNN also achieves satisfactory results with transmission errors. Specifically, with only offloading data, which is only 0.79% of the image size or 3.16% of the feature size, Semantic achieves much higher accuracy with various transmission errors. Thus, SemanticNN demonstrates its versatility by being amenable to a broad spectrum of AI applications.

5.4 Impact of Model Structures

Transformers vs. CNN. A large amount of research focused on semantic communication relies on *Transformers* to enhance the extraction of semantic information from transmitted data. Thus, we conducted a performance comparison with DeepSC, which employs a Transformer structure with two encoders and two decoders, across various BERs. Continuing with the split computing of ResNet-50 on ImageNet-200, Figure 7 (a) illustrates the performance under various BER conditions: 0.01%, 0.25%, 2.5%, and 5%. With increasing BERs, SemanticNN achieves comparable performance with DeepSC. However, for MobileNetV2 on CIFAR-100, Figure 7 (b) shows that SemanticNN significantly minimizes transmission latency. Table 2 shows the parameters of DeepSC and SemanticNN. Moreover, as for the overhead of weak edge devices, SemanticNN can achieve efficient data transmission with a more lightweight structure and less memory usage measured in Appendix C.

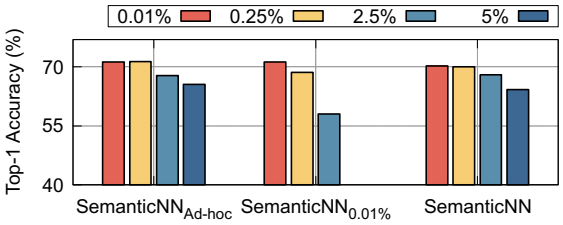


Figure 8: SemanticNN trained with attention under dynamic changes in BERs, achieves comparable accuracy as training SemanticNN at a specific BER.

Attention in decoder. $\text{SemanticNN}_{Ad-hoc}$ in Figure 8 represents training and testing at a specific BER. $\text{SemanticNN}_{0.01\%}$ represents testing at other BERs with a model trained at 0.01% BER. SemanticNN is trained at dynamically varying BERs and tested at a specific BER. Results show that SemanticNN with attention modules can adapt to varying BERs and achieve comparable accuracy as training at a specific BER.

6 Conclusions

In this paper, we present *SemanticNN*, a semantic codec designed to achieve error-resilient *Semantic Neural Network* offloading to achieve compressive transmission correctness at a semantic level for extremely weak devices. To achieve this, we carefully design the model structure of SemanticNN, including the SQ-based encoder and BER-aware decoder. We then propose the novel *Feature-augmentation Learning* to enhance the performance of SemanticNN in the presence of transmission errors. We further propose the *XAI-based Asymmetry Compensation* method to aid in error-resilient semantic extraction and feature map fine-tuning in the weak encoder.

We conduct extensive evaluations of SemanticNN, demonstrating that it drastically reduces the offloaded features by 56.82-344.83× and significantly outperforms compressed offloading approaches by 7.11-30.88% and error resilient works by 1.28-53.42% with transmission errors in image classification. Finally, the case study confirms the broad applicability of SemanticNN in real-world scenarios, enabling error-resilient AI in poor communication scenarios.

Acknowledgments

We thank all the reviewers for their valuable comments and helpful suggestions. This work is supported by the National Natural Science Foundation of China under Grant No. 62272407 and No. 92582114, the "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province under Grant No. 2023C01033, the Major Project of the Zhejiang Provincial Natural Science Foundation under Grant No. D26F020015, and the Young Elite Scientists Sponsorship Program of Zhejiang Province under Grant No. 20225E21842.

References

2023. STM32H750 Overview. https://www.st.com/en/microcontrollers-microprocessors/stm32h750-value-line.html/.

Bajić, I. V. 2021. Latent space inpainting for loss-resilient collaborative object detection. In *ICC 2021-IEEE International Conference on Communications*, 1–6. IEEE.

Becke, M.; Dreibholz, T.; Adhari, H.; and Rathgeb, E. P. 2012. On the fairness of transport protocols in a multi-path environment. In *2012 IEEE International Conference on Communications (ICC)*, 2666–2672. IEEE.

Benazir, A.; Xu, Z.; and Lin, F. X. 2024. Speech Understanding on Tiny Devices with A Learning Cache. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 425–437.

Bin, K.; Park, J.; Park, C.; Kim, S.; and Lee, K. 2024. CoActo: CoActive Neural Network Inference Offloading with Fine-grained and Concurrent Execution. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 412–424.

Bourtsoulatze, E.; Kurka, D. B.; and Gündüz, D. 2019. Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking*, 5(3): 567–579.

Bragilevsky, L.; and Bajić, I. V. 2020. Tensor completion methods for collaborative intelligence. *IEEE Access*, 8: 41162–41174.

Breda, J.; Springston, M.; Mariakakis, A.; and Patel, S. 2023. FeverPhone: Accessible Core-Body Temperature Sensing for Fever Monitoring Using Commodity Smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(1): 1–23.

Dai, J.; Wang, S.; Tan, K.; Si, Z.; Qin, X.; Niu, K.; and Zhang, P. 2022. Nonlinear transform source-channel coding for semantic communications. *IEEE Journal on Selected Areas in Communications*, 40(8): 2300–2316.

Dhondea, A.; Cohen, R. A.; and Bajić, I. V. 2021. CALTeC: Content-adaptive linear tensor completion for collaborative intelligence. In *2021 IEEE International Conference on Image Processing (ICIP)*, 2179–2183. IEEE.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*.

Han, T.; Yang, Q.; Shi, Z.; He, S.; and Zhang, Z. 2022. Semantic-aware speech to text transmission with redundancy removal. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, 717–722. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Hojjat, A.; Haberer, J.; Zainab, T.; and Landsiedel, O. 2024. LimitNet: Progressive, Content-Aware Image Offloading for Extremely Weak Devices & Networks. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 519–533.

Huang, K.; and Gao, W. 2022. Real-time neural network inference on extremely weak devices: agile offloading with explainable AI. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 200–213.

Huang, Z.; Dong, F.; Shen, D.; Zhang, J.; Wang, H.; Cai, G.; and He, Q. 2021. Enabling Low Latency Edge Intelligence based on Multi-exit DNNs in the Wild. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 729–739. IEEE.

Itahara, S.; Nishio, T.; Koda, Y.; and Yamamoto, K. 2022. Communication-oriented model fine-tuning for packet-loss resilient distributed inference under highly lossy IoT networks. *IEEE Access*, 10: 14969–14979.

Jiang, S.; Lin, Z.; Li, Y.; Shu, Y.; and Liu, Y. 2021. Flexible high-resolution object detection on edge devices with tunable latency. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 559–572.

Kallimani, R.; Pai, K.; Raghuwanshi, P.; Iyer, S.; and López, O. L. 2024. TinyML: Tools, applications, challenges, and future research directions. *Multimedia Tools and Applications*, 83(10): 29015–29045.

Kang, D.; Guibas, J.; Bailis, P. D.; Hashimoto, T.; and Zaharia, M. 2022. TASTI: semantic indexes for machine learning-based queries over unstructured data. In *Proceedings of the 2022 International Conference on Management of Data*, 1934–1947.

Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T.; Mars, J.; and Tang, L. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1): 615–629.

Laskaridis, S.; Venieris, S. I.; Almeida, M.; Leontiadis, I.; and Lane, N. D. 2020. SPINN: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*, 1–15.

Lee, C.-H.; Lin, J.-W.; Chen, P.-H.; and Chang, Y.-C. 2019. Deep learning-constructed joint transmission-recognition for Internet of Things. *IEEE Access*, 7: 76547–76561.

Li, H.; Hu, C.; Jiang, J.; Wang, Z.; Wen, Y.; and Zhu, W. 2018. Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution. In *2018 IEEE 24th international conference on parallel and distributed systems (ICPADS)*, 671–678. IEEE.

Lin, E. 2023. Comparative Analysis of Pix2Pix and CycleGAN for Image-to-Image Translation. *Highlights in Science, Engineering and Technology*, 39: 915–925.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1529–1538.

Ling, N.; Wang, K.; He, Y.; Xing, G.; and Xie, D. 2021. Rt-mdl: Supporting real-time mixed deep learning tasks on edge platforms. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 1–14.

MacKay, D. J. 2005. Fountain codes. *IEE Proceedings-Communications*, 152(6): 1062–1068.

Maj, M.; Rymarczyk, T.; Cieplak, T.; and Pliszczuk, D. 2022. Deep learning model optimization for faster inference using multi-task learning for embedded systems. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 892–893.

Nogales, C.; Grim, B.; Kamstra, M.; Campbell, B.; Ewing, A.; Hance, R.; Griffin, J.; and Parke, S. 2018. MakerSat-0: 3D-printed polymer degradation first data from orbit.

Quy, V. K.; Hau, N. V.; Anh, D. V.; Quy, N. M.; Ban, N. T.; Lanza, S.; Randazzo, G.; and Muzirafuti, A. 2022. IoT-enabled smart agriculture: architecture, applications, and challenges. *Applied Sciences*, 12(7): 3396.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Sau, B. B.; and Balasubramanian, V. N. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Shao, J.; Mao, Y.; and Zhang, J. 2021. Learning task-oriented communication for edge inference: An information bottleneck approach. *IEEE Journal on Selected Areas in Communications*, 40(1): 197–211.

Shao, J.; and Zhang, J. 2020. Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 1–6. IEEE.

Sundararajan, J. K.; Shah, D.; and Médard, M. 2008. ARQ for network coding. In *2008 IEEE International Symposium on Information Theory*, 1651–1655. IEEE.

Svensson, A. 2007. An introduction to adaptive QAM modulation schemes for known and predicted channels. *Proceedings of the IEEE*, 95(12): 2322–2336.

Teerapittayanon, S.; McDanel, B.; and Kung, H.-T. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2464–2469. IEEE.

Wang, S.; and Zhang, X. 2022. NeuroMessenger: Towards Error Tolerant Distributed Machine Learning Over Edge Networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 2058–2067. IEEE.

Weng, Z.; Qin, Z.; Tao, X.; Pan, C.; Liu, G.; and Li, G. Y. 2023. Deep learning enabled semantic communications with speech recognition and synthesis. *IEEE Transactions on Wireless Communications*.

Wicker, S. B.; and Bhargava, V. K. 1999. *Reed-Solomon codes and their applications*. John Wiley & Sons.

Xie, H.; Qin, Z.; Li, G. Y.; and Juang, B.-H. 2021. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 69: 2663–2675.

Xu, J.; Ai, B.; Chen, W.; Yang, A.; Sun, P.; and Rodrigues, M. 2021. Wireless image transmission using deep source channel coding with attention modules. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(4): 2315–2328.

Xu, M.; Zhu, M.; Liu, Y.; Lin, F. X.; and Liu, X. 2018. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*, 129–144.

Yao, S.; Li, J.; Liu, D.; Wang, T.; Liu, S.; Shao, H.; and Abdelzaher, T. 2020. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 476–488.

Yuan, M.; Zhang, L.; He, F.; Tong, X.; and Li, X.-Y. 2022. Infi: end-to-end learnable input filter for resource-efficient mobile-centric inference. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 228–241.

Yun, S.; Kim, H.; and Tan, K. 2010. Towards zero retransmission overhead: A symbol level network coding approach to retransmission. *IEEE transactions on mobile computing*, 10(8): 1083–1095.

Zhang, Y.; Gu, T.; and Zhang, X. 2021. MDLdroidLite: a release-and-inhibit control approach to resource-efficient deep neural networks on mobile devices. *IEEE Transactions on Mobile Computing*.

Zhuo, S.; Chen, H.; Ramakrishnan, R. K.; Chen, T.; Feng, C.; Lin, Y.; Zhang, P.; and Shen, L. 2022. An empirical study of low precision quantization for tinyml. *arXiv preprint arXiv:2203.05492*.