Boosting Edge-to-Cloud Data Transmission Efficiency with Semantic Transcoding

Murtadha Nisyif, Ahmed Refaey, and Sylvester Aboagye
University of Guelph, Guelph, Ontario, Canada.

***Abstract*—The evolution of network systems is driven by the increasing demand for higher data transmission volumes and reduced latency. This paper presents a semantic method to reduce latency in core network systems to accommodate the growing demand for high-speed data transmission. We minimize unnecessary transmissions and optimize bandwidth by prioritizing data relevance through semantic communication technologies. Our approach utilizes a transformer-based model from edge servers to the cloud to achieve minimal end-to-end latency. Experimental results demonstrate that our methods can reduce latency by up to 30%, offering a scalable solution that significantly enhances network performance and efficiency.**

***Index Terms*—MEC, 6G, Semantic communication.**

I. Introduction

The evolution of network systems is fundamentally driven by the escalating demand for higher data transmission volumes and reduced latency in data exchange [1]. This trend is primarily due to the proliferation of smart devices, the increasing complexity of applications, and the rise in connected devices. As a result, next-generation networks require faster data transmission rates to handle the enormous data traffic [1] [2]. For instance, the core network development aims to achieve a thousandfold increase in capacity and deliver data rates up to 10 Gbps per user. These advancements are necessary to support applications such as augmented reality (AR), virtual reality (VR), and the Internet of Things (IoT), which demand low latency and high reliability [1] [2]. Additionally, the continuous growth in global Internet traffic and the need for higher Quality of Service (QoS) have made it imperative to rely on AI-based systems. These systems are required to meet the ever-increasing user demands for higher data rates, reduced latency, and ensuring seamless data exchanges [3].

Edge computing has emerged as a solution to minimize latency by processing data closer to the end-user, thus enhancing real-time responsiveness and reducing the need for long-distance data transmission to be processed on the cloud [4]. Data processing and transmission can be further optimized by integrating AI and semantic transcoding within edge-to-cloud systems. AI algorithms dynamically manage network resources, predict traffic patterns, and enhance decision-making at the edge [5]. Simultaneously, semantic communication ensures efficient, context-aware data exchange between edge devices and cloud infrastructure, supporting latency-sensitive applications effectively [6].

For example, semantic communication systems leverage AI to understand and convey the meaning behind data, rather than just transmitting raw data. This approach significantly reduces the volume of data needing transmission, thereby lowering latency and enhancing efficiency [6]. Furthermore, AI at the edge can improve model training and deployment through federated learning techniques, allowing for local data processing while maintaining privacy and reducing bandwidth usage [7].

For instance, several research works have shown that by using semantic communication, edge servers can act as semantic relays, processing and interpreting data locally before sending only the necessary semantic information to other devices or servers [8] [7]. Indeed, this approach has been effectively utilized in various applications, where real-time data processing and low latency are critical [7]. By leveraging the capabilities of semantic communication, edge-to-cloud-based systems can operate more efficiently, making real-time decisions based on the interpreted data [5]. However, despite the significant advancements in semantic communication between edge devices, or edge servers to devices, there is a notable gap in the research regarding its application between the edge and the cloud. Current research works have yet to explore the potential benefits of utilizing semantic communication to enhance data exchange efficiency between edge and cloud data centers. This presents an opportunity to develop methods that can extend the benefits of semantic communication to the broader edge-to-cloud architecture, potentially impacting how large-scale data is processed and transmitted in distributed networks [9] [4].

Based on [10], this paper delves into the theoretical modeling and empirical analysis to develop a comprehensive latency measurement and optimization framework tailored for Edge-Cloud networks. By integrating latency management techniques, such as AI-driven resource allocation and semantic communication, we propose methods to boost network performance and contribute to the evolving landscape of network technology. These methods provide scalable solutions to meet the growing demands of the digital ecosystem. Through this work, we expect to enhance the end-to-end network performance that aligns with the QoS, reliability, and efficiency requirements [4] - [5].

The rest of the paper is outlined as follows: Section II details the related works in latency modeling and semantic communications. Section III introduces the proposed semantic-based optimized latency framework. Section IV outlines the tools utilized, the measured metrics and the data collection process. Section V lists experimental results, semantic models comparisons and data interpretation. Section VI presents concluding remarks and future steps.

II. Related Works

This section surveys the state-of-the-art in modeling end-to-end (E2E) networks, including recent advancements in semantic communication transcoding techniques. By analyzing these developments, we aim to elucidate prevailing trends, identify challenges, and highlight future research avenues to optimize E2E network performance through semantic transcoding.

A. E2E Latency Modeling

Coll-Perales et al.'s comprehensive methodology for dissecting latency components in cellular core networks has significantly influenced this research [11]. Their E2E latency model, defining critical components and quantifying delays within network segments, has inspired our approach.

This work focuses on optimizing current core networks and meet the stringent latency requirements of next-generation networks. It addresses the need to understand how network components contribute to overall latency. Coll-Perales et al.'s model covers various components in different deployment scenarios, including both single and multiple Mobile Network Operators (MNOs), accounting for complexities in cross-MNO data transmissions. Table I showcases the latency components characterizing an E2E network application.

TABLE I: Latency components in E2E network

Parameter	Description
l_{E2E}	Total time of data transmitted from on end to another
l_{enc}	Time required to encode and process data at the edge server
l_{radio}	Time required to transmit data wirelessly
l_{TN}	Time for data to travel through the transport network
l_{CN}	Time to pass data through the core network infrastructure
$l_{\text{UPF-AS}}$	Time for data to traverse from the User Plane Function to the Application Server
l_{AS}	Time required to process data at the application server (cloud)
l_{pp}	Time delay when transmitting data between different Mobile Network Operators (MNOs)

The overall latency can be modelled as [11]:

$$l_{\text{E2E}} = l_{\text{RAN}} + l_{\text{CN}} + l_{\text{UPF-AS}} + l_{\text{AS}} \tag{1}$$

where l_{RAN} is further broken down into:

$$l_{\text{RAN}} = l_{\text{radio}} + l_{\text{TN}} \tag{2}$$

And for applications supported by different MNOs:

$$l_{\text{E2E}} = l_{\text{radio}} + l_{\text{TN}} + l_{\text{CN}} + l_{\text{UPF-AS}} + l_{\text{AS}} + l_{\text{pp}} \tag{3}$$

Their findings revealed that transport network and core network latencies are significant contributors to total latency, particularly in scenarios involving different MNOs where peering points notably increase delays.

It is worth noting that l_{radio} is irrelevant to our proposed model. Furthermore, the overall E2E network latency can be modelled as:

$$l_{\text{E2E}} = l_{\text{enc}} + l_{\text{TN}} + l_{\text{CN}} + l_{\text{UPF-AS}} + l_{\text{AS}} + l^{*}_{\text{pp}} \tag{4}$$

*when supporting different MNOs.

B. Semantic Communications

Semantic communications, an emerging paradigm, aims to enhance data transmission efficiency by focusing on the meaning of the data rather than merely its syntactic structure [12]. This approach leverages advancements in AI and NLP to understand and interpret the context and content of the transmitted information, significantly reducing the volume of data required while preserving its essential meaning [13] [14].

The integration of semantic understanding in communications marks a shift from traditional data transmission methods that prioritize syntax over meaning [12]. Recent advancements in Transformer-based architectures, such as the Swin Transformer [15], enhance semantic communication systems by focusing on data relevance. The Swin Transformer, with its hierarchical vision approach and shifted windowing scheme, achieves high efficiency and flexibility in tasks like image classification, object detection, and semantic segmentation. Attention mechanisms are critical in these models [16], enabling the prioritization of relevant data, which is crucial for bandwidth-limited systems. This preserves semantic integrity and optimizes bandwidth usage, improving the quality and efficiency of semantic communications.

The Wireless Image Transmission Transformer (WITT) and Swin Joint Source-Channel Coding (SwinJSCC) leverage the Swin Transformer to advance semantic communications. WITT captures long-range information for high-resolution image transmission, using a spatial modulation module to adapt to varying channel conditions [17]. SwinJSCC enhances this with Channel ModNet and Rate ModNet for dynamic adaptation to diverse channel conditions and transmission rates. It outperforms traditional systems like BPG with 5G LDPC, especially in high-resolution images, offering faster processing and superior quality. Furthermore, SwinJSCC comes in three sizes—small, base, and large—with varying parameters: 23.21 million, 33.07 million, and 52.07 million, respectively, while ensuring lower upto 319.5 GFLOPs costs compared to traditional DeepJSCC models 9851.6 GFLOPs [10].

In the context of edge-cloud computing, where direct channels with negligible noise are assumed, a crucial aspect is optimizing end-to-end performance [10]:

$$(\phi, \theta) = \arg\min_{\phi,\theta} \mathbb{E}x \sim p_x \mathbb{E}\hat{y} \sim p_{\hat{y}|x}[d(x, \hat{x})], \tag{5}$$

where ϕ and θ are the encoder and decoder parameters. This optimization helps maximize computational resources and bandwidth efficiency, ensuring high-quality transmission and scalability. By leveraging this approach, SwinJSCC enhances real-time performance, making it ideal for various edge computing applications

In our implementation, assuming a noise-free channel, the distortion measure $d(x, \hat{x})$ is still crucial. It ensures that the encoder and decoder minimize approximation errors and maintain high image fidelity. By optimizing this measure, we guarantee that the reconstructed images closely match the originals, achieving robust and accurate performance while focusing on end-to-end (E2E) latency performance.

III. Semantic Edge-Cloud Computing Model

The proposed system model (Fig. 1) employs transformer models at both the edge and cloud servers, following the architectural principles of the WITT and SwinJSCC frameworks [17], [10]. These models are selected for their efficiency in semantic communications, critical for reducing latency in E2E communications.

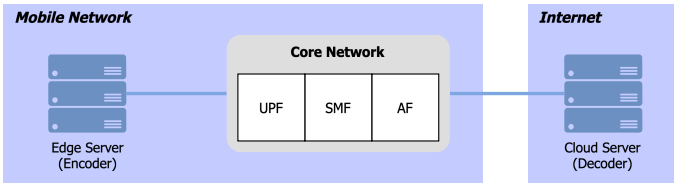


Fig. 1: Overview of the proposed system model.

At the edge server, the encoder employs a transformer-based architecture adept at semantic analysis and data prioritization, ensuring that only the most relevant data is transmitted. This mechanism minimizes unnecessary network traffic and improves latency as it will be shown in the results section. The model's ability to discern semantic importance aligns with WITT methodologies for image data, adapted for diverse data types in various applications.

Conversely, the decoder reconstructs the semantic content from the compressed data stream in the cloud using an optimized transformer architecture. While WITT and SwinJSCC focus on image data, their underlying principles apply to various data types.

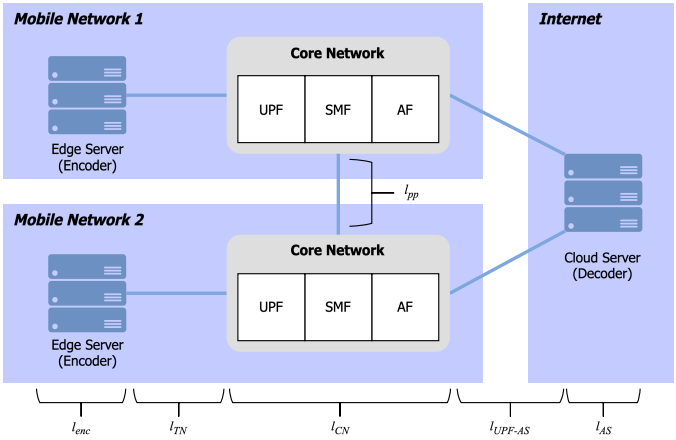


Fig. 2: All possible latency components affecting E2E data transmission

The system model illustrated in Fig. 2 showcases several latency components in an end-to-end network. It depicts a network architecture designed to facilitate communication between at least two edge servers over different Mobile Network Operators (MNOs), connecting to a centralized cloud server responsible for semantically decoding data from the edge servers. Each cellular network features an edge server that functions as an encoder, optimizing the data before it is sent through the cellular core network.

Table II provides a concise overview of the critical network latency assumptions used to evaluate communication delays within our experimental setup. Each entry represents a distinct aspect of latency within the network architecture, supported by references from the relevant literature. These assumed values, derived from the referenced papers, will form the basis for our benchmarking and evaluation.

TABLE II: Network Latency Assumptions

Description	Parameter	Value/Range	Reference
Core Network	l_{CN}	$\sim$ 24 ms	[2]
Cloud-based (E2E)	l_{E2E}	[53.8, 150] ms	[11]
UPF to AS	$l_{\text{UPF-AS}}$	$\sim$ 20 ms	[18]
Edge / Cloud latencies	l_{AS} & l_{enc}	13 ms resp.	[10]
Local Peering Point	l_{pp}	0.431 ms	[11]

IV. Experimental Setup

This section outlines the experimental setup used to evaluate the performance of our models. It includes details on the hardware setup, evaluation metrics, implementation details, and the data collection process.

A. Setup and Evaluation metrics

The experiments were conducted on a high-performance computing system running Ubuntu 22.04.4 LTS x86. The hardware configuration included an Intel Xeon E5-2667 V3 operating at 3.196 GHz, 24 GB of RAM and a 150GB SSD. The implementation utilized libraries like PyTorch 1.9.0 with Python 3.8.19. Additional libraries included psutil for system monitoring, threading for concurrent execution, pandas for data manipulation, and seaborn for data visualization.

System performance metrics focused on computational costs, specifically monitoring CPU and memory utilization over time, to evaluate the resource efficiency of the models during encoding and decoding processes.

B. Data Collection

For data collection, the models transmitted three random images each from the Kodak and CLIC21 datasets. This approach ensured a diverse set of images to evaluate the performance of the encoding and decoding processes.

1) Reading the CPU Times: The logger reads CPU times from the `/proc/stat` file on Linux systems, which contains cumulative time statistics for different CPU states. Table III lists the different CPU states logged in the `/proc/stat` file:

TABLE III: Description of different CPU states in `/proc/stat` file

CPU States	Description
user	Time spent in user-mode processes (excluding nice)
nice	Time spent in lower-priority user processes
system	Time spent in kernel-mode processes
idle	Time the CPU is not executing any process
I/O wait	Time waiting for I/O operations
irq/softirq	Time handling hardware/software interrupts

The created logger reads the cumulative CPU times at two points in time separated by a specified interval. By calculating the difference between these two readings, it determines the time spent in each state during that interval.

The percentage of CPU utilization is calculated as:

$$\text{CPU Usage (\%)} = \frac{\text{Active Time}}{\text{Total Time}} \times 100 \tag{6}$$

where, Active Time = User + Nice + System + Interrupts times, and Total Time = Active Time + Idle Time + I/O Wait.

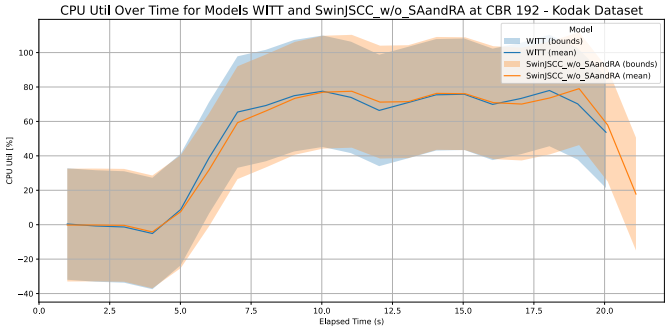
Also, regarding memory utilization, it is calculated by reading the memory statistics provided by the `psutil` Python library, which uses data from system files like `/proc/meminfo`. The relevant memory statistics include the total system memory available, the memory currently used by all running processes, and the memory available for use without swapping.

Memory utilization in gigabytes (GB) is calculated as:

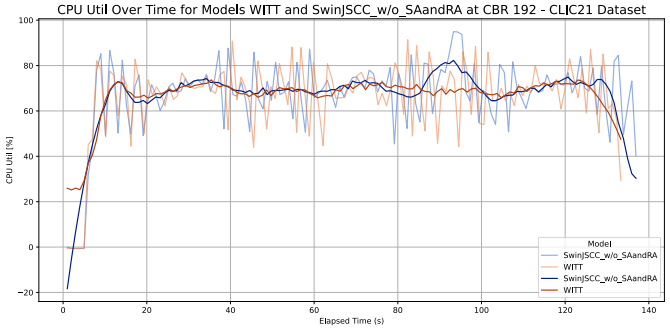
$$\text{Memory Usage (GB)} = \frac{\text{Used Memory (Bytes)}}{1024^3} \tag{7}$$

V. Experimental Analysis

This section presents the CPU and memory utilization results for the WITT and SwinJSCC models transcoding three random images from both Kodak and CLIC21 datasets. The performance trends for both the WITT and SwinJSCC models, as visualized in the provided plots, offer valuable insights into each model's performance under operational stress.

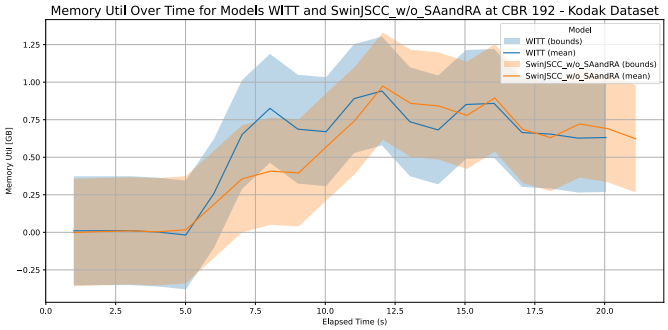


(a) CPU Utilization [%] vs Time [s] for Kodak dataset

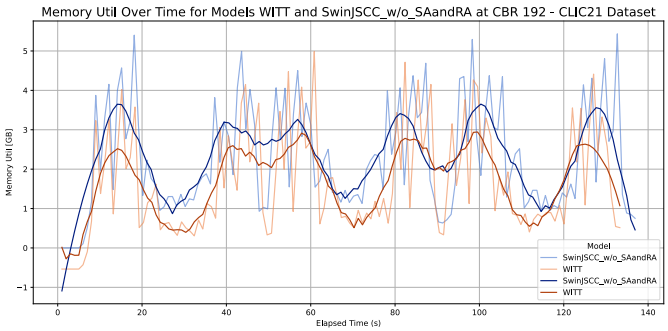


(b) CPU Utilization [%] vs Time [s] for CLIC21 dataset

Fig. 3: Comparison of CPU Utilization for WITT and SwinJSCC models transmitting images from Kodak and CLIC21 datasets at CBR 1/8.



(a) Memory Utilization [GB] vs Time [s] for Kodak dataset



(b) Memory Utilization [GB] vs Time [s] for CLIC21 dataset

Fig. 4: Comparison of Memory Utilization for WITT and SwinJSCC models transmitting images from Kodak and CLIC21 datasets at CBR 1/8.

A. CPU Utilization

The CPU utilization plots for the WITT and SwinJSCC models reveal significant fluctuations in computational demands, with both models showing relatively similar performance. For the CLIC21 dataset, both models frequently hit peaks over 85% CPU utilization (see Fig. 3b). These spikes result from complex operations like window partitioning, cyclic shifting, and adaptive modulation based on SNR values within the SwinTransformerBlock. The dynamic and conditional nature of these operations contributes to the variability in CPU usage. In contrast, while processing the Kodak dataset, both models exhibit slightly higher CPU utilization, about 5% more than with the CLIC21 dataset (see Fig. 3a). The models in the Kodak dataset also tends to be relatively smoother compared to the CLIC21 dataset, this can be due to the Kodak dataset's simplicity. Despite this simplicity, both models still hit highs of over 80% CPU utilization during processing, indicating the intensive nature of their operations.

While Kodak processing is more stable, the CLIC21 dataset shows more fluctuations due to its complexity and the number of features to be processed. In both datasets, WITT processes images faster than SwinJSCC, with an interval of up to 3 seconds. The average processing time per image from the Kodak dataset is around 5-6 seconds, while it takes around 45 seconds for each image in the CLIC21 dataset. This substantial difference in processing time highlights the impact of image

complexity on computational efficiency.

It is also evident that CPU activity is intense during the encoding and decoding phases, as observed from the intervals where CPU utilization peaks. These observations underscore the importance of optimizing both models for better handling of complex datasets to achieve more consistent and efficient performance (see Fig. 3b).

B. Memory Utilization

The memory utilization plots for the WITT and SwinJSCC models reveal distinct patterns when processing images from the Kodak and CLIC21 datasets (see Fig.4). For the Kodak dataset, the average memory utilization of the WITT model remains relatively stable (see Fig.4a). This stability can be attributed to efficient memory handling within the SwinTransformerBlock and the model's structured processing.

The complexity of the datasets plays a significant role in memory usage. Throughout the entire Kodak dataset processing, the maximum memory utilization for WITT did not exceed 1.5 GB, while for the CLIC21 dataset, it averaged around 4.5 GB, demonstrating the higher memory demands for more complex data (see Fig. 4b). Despite SwinJSCC having similar memory utilization to WITT when processing the Kodak dataset, it consumes up to 40% more memory on average compared to the WITT memory consumption (approximately 1 GB more) and exhibits more intense fluctuations.

SwinJSCC uses approximately 40% more memory than the WITT model, with a maximum of up to 5 GB of total memory usage, highlighting its higher memory demands during transcoding operations. However, the adaptability of the SwinJSCC attention mechanism, including the use of window-based attention and a shifted window scheme, allows it to dynamically adjust to varying data complexities. This adaptability enables the SwinJSCC model to handle complex patterns and dependencies effectively, making up for its higher memory consumption.

These observations underscore the differences in memory utilization efficiency between the WITT and SwinJSCC models. While both models exhibit high computational demands, WITT manages memory usage more effectively, resulting in more stable consumption patterns. Nevertheless, the adaptability of the SwinJSCC model's attention mechanism provides a significant advantage in handling diverse and complex datasets.

VI. Conclusion

In this paper, we introduce a semantic method to reduce latency in core network systems, optimizing bandwidth and minimizing unnecessary transmissions through semantic communication technologies. Utilizing a transformer-based model for edge-to-cloud data transmission, in our approach significantly reduces transmission volume and lowers latency, supporting next-generation applications like AR, VR, and IoT. This research explores semantic transcoding within Edge-Cloud computing paradigms, demonstrating potential improvements in handling complex datasets. Future work will focus on parallelizing encoding and decoding with GPUs, expected to vastly improve processing speed and efficiency, while refining algorithms to maintain low latency in resource-constrained environments.

References

[1] S. K. Sharma, I. Woungang, A. Anpalagan, and S. Chatzinotas, "Toward tactile internet in beyond 5g era: Recent advances, current issues, and future directions," *IEEE Access*, vol. 8, pp. 56 948–56 991, 2020.

[2] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5g: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[3] DataReportal, "Internet use in 2024 — datareportal – global digital insights," 2024. [Online]. Available: https://datareportal.com/reports/digital-2024-global-overview-report

[4] S. Tang, L. Chen, K. He, J. Xia, L. Fan, and A. Nallanathan, "Computational intelligence and deep learning for next-generation edge-enabled industrial iot," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2881–2893, 2023.

[5] C. Liang, H. Du, Y. Sun, D. Niyato, J. Kang, D. Zhao, and M. A. Imran, "Generative ai-driven semantic communication networks: Architecture, technologies and applications," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2401.00124

[6] E. Figetakis, Y. Bello, A. Refaey, and A. Shami, "Decentralized semantic traffic control in avs using rl and dqn for dynamic roadblocks," 2024. [Online]. Available: https://arxiv.org/abs/2406.18741

[7] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah, "Edge learning for b5g networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 9–39, 2023.

[8] J. Guo, H. Chen, B. Song, Y. Chi, C. Yuen, F. R. Yu, G. Y. Li, and D. Niyato, "Distributed task-oriented communication networks with multimodal semantic relay and edge intelligence," *IEEE Communications Magazine*, pp. 1–7, 2024.

[9] E. Figetakis and A. Refaey, "Autonomous mec selection in federated next-gen networks via deep reinforcement learning," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 2045–2050.

[10] K. Yang, S. Wang, J. Dai, X. Qin, K. Niu, and P. Zhang, "Swinjscc: Taming swin transformer for deep joint source-channel coding," *arXiv*, 2023.

[11] B. Coll-Perales, M. C. Lucas-Estañ, T. Shimizu, J. Gozalvez, T. Higuchi, S. Avedisov, O. Altintas, and M. Sepulcre, "End-to-end v2x latency modeling and analysis in 5g networks," in *Proceedings of the International Conference on Communications*. IEEE, 2023, pp. 1–15, available online; accessed April 15, 2024.

[12] Z. Qin, X. Tao, J. Lu, W. Tong, and G. Y. Li, "Semantic communications: Principles and challenges," *arXiv*, 2022.

[13] Y. Liu, X. Wang, Z. Ning, M. Zhou, L. Guo, and B. Jedari, "A survey on semantic communications: technologies, solutions, applications and challenges," *Digital Communications and Networks*, 2023.

[14] W. Yang, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. Cao, and K. B. Letaief, "Semantic communication meets edge intelligence," *IEEE Wireless Communications*, vol. 29, no. 5, pp. 28–35, 2022.

[15] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV 2021*, October 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/swin-transformer-hierarchical-vision-transformer-using-shifted-windows/

[16] M. Mortaheb, E. Karakaya, M. A. A. Khojastepour, and S. Ulukus, "Transformer-aided semantic communications," *arXiv*, 2024.

[17] K. Yang, S. Wang, J. Dai, K. Tan, K. Niu, and P. Zhang, "WITT: A wireless image transmission transformer for semantic communications," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[18] M. Candela, V. Luconi, and A. Vecchio, "Impact of the covid-19 pandemic on the internet latency: A large-scale study," *Computer Networks*, vol. 182, p. 107495, 2020.