Conditional Denoising Diffusion Autoencoders for Wireless Semantic Communications

Mehdi Letafati Samad Ali Matti Latva-aho
Centre for Wireless Communications
University of Oulu
Oulu, Finland 90014
`{mehdi.letafati,samad.ali,matti.latva-aho}@oulu.fi`

Abstract

Semantic communication (SemCom) systems aim to learn the mapping from low-dimensional semantics to high-dimensional ground-truth. While this is more akin to a "domain translation" problem, existing frameworks typically emphasize on channel-adaptive neural encoding-decoding schemes, lacking full exploration of signal distribution. Moreover, such methods so far have employed autoencoder-based architectures, where the encoding is tightly coupled to a matched decoder, causing *scalability issues* in practice. To address these gaps, diffusion autoencoder models are proposed for wireless SemCom. The goal is to learn a "semantic-to-clean" mapping, from the semantic space to the ground-truth probability distribution. A neural encoder at semantic transmitter extracts the high-level semantics, and a conditional diffusion model (CDiff) at the semantic receiver exploits the source distribution for signal-space denoising, while the received semantic latents are incorporated as the conditioning input to "steer" the decoding process towards the semantics intended by the transmitter. It is analytically proved that the proposed decoder model is a *consistent estimator* of the ground-truth data. Furthermore, extensive simulations over CIFAR-10 and MNIST datasets are provided along with design insights, highlighting the performance compared to legacy autoencoders and variational autoencoders (VAE). Simulations are further extended to the multi-user SemCom, identifying the dominating factors in a more realistic setup.

1 Introduction

Semantic communications (SemCom) has been considered as one of the enablers for a truly AI-native communication framework for next-generations (nextG) of wireless systems [1]. In simple terms, SemCom paradigm provides means to convey the most relevant information, where bits are no longer the common currency between the application and physical layers. Recent advances in generative artificial intelligence (GenAI) and its capabilities in generating high-fidelity samples have boosted the interest in exploring its potentials for nextG systems [2, 3]. The evolution of diffusion models has contributed to the recent breakthroughs in non-language GenAI, with renowned solutions such as DALL.E 2 by OpenAI and Imagen by Google. Interestingly, recent studies have proven their applications in wireless AI as well [4]–[8]. Nevertheless, diffusion models lack any mechanism to extract semantic meaning. Our motivation in this paper is to smartly combine the capabilities of autoencoders in extracting decodable latents, with the capabilities of diffusion models in discovering the signal space distributions, to come up with an enhanced SemCom framework for nextG wireless.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI and ML for Next-Generation Wireless Communications and Networking (AI4NextG).

1.1 Related Works

Autoencoder-based architectures under the framework of deep joint source-channel coding (deep-JSCC), have been widely adopted for SemCom [9]. Semantic image transmission using diffusion models was proposed in [10] and [11]. A denoising diffusion probabilistic model (DDPM) was proposed to be appended to the deep-JSCC framework. However, this assumes a very simplistic view of generative models, in the sense that it simply considers generative models as an *additional module* to be added to a deep-JSCC scheme, only for further refinement, at the obvious cost of additional computation by adding one more AI processing block. Such simplistic view of using GenAI is not able to, for example, utilize generative modeling for replacing some of the legacy functionalities in deep-JSCC/SemCom framework. Moreover, the paper does not exploit the full capabilities of generative models, such as their conditioning mechanisms which can flexibly incorporate additional contextual information to guide the models in reconstruction with desired fidelity. Diffusion models as "channel denoisers" were studied in [12] and [13] by pre-pending the diffusion model to the deep-JSCC decoder. Nevertheless, such approach does not seem to be exploiting the full potential of diffusion model in learning complicated probability distributions, downgrading it to a simple channel denoiser. Furthermore, the it relies on the channel state information (CSI) knowledge at the receiver. This seems to be conceptually in contradiction with the purpose of channel denoising, which is supposed to learn the CSI distributions, not receiving the CSIs directly.

1.2 Contributions

Deep-JSCC-based SemCom frameworks with autoencoder architectures as backbone mainly focus on channel-adaptive encoding and decoding, lacking a powerful model for effectively learning signal distributions. Moreover, the decoder architecture is tightly dependent to the semantic encoder, which can cause *scalability issues* in practical deployments. Furthermore, autoencoders seem to lack sample quality in learning complicated distributions, lacking high-quality reconstruction performance. In this work, we propose diffusion autoencoder models for wireless SemCom. The idea is to use a neural encoder for extracting the high-level semantics, and a conditional diffusion model (CDiff) as the decoder to explore the source data distribution and reconstruct samples, while being guided by the semantic latents. In our scheme, the noisy semantics are incorporated as the conditioning input to the CDiff decoder. This way, semantic latents are distilled into the decoding process, which help "steer" the decoding process towards the semantics intended by the transmitter. For our CDiff decoder, we employ conditional DDPM, while the encoder is arbitrarily designed via prevalent convolution neural network (CNN)-based neural encoders. Our scheme offers advantages over generative models like variational autoencoders (VAE). While our scheme does not impose any constraint on the format of the semantic latent and allows the neural encoder to obtain the semantics as expressive as it can, VAEs typically cast a standard normal distribution on the prior distribution of the latent space, that can compromise the decodability quality of semantics.

With our diffusion-based decoder model at the receiver, the requirement to match the neural encoder and decoder pairs is dropped — we do not employ the matched decoder counterpart of the neural encoder. The reason is that the built-in conditioning mechanism of diffusion models is able to flexibly handle variable-length vectors corresponding to the semantic latents of different sizes, thanks to padding & masking conditioning mechanism. Thus, we not only enhance the semantic decoding performance, but also mitigate the *scalability issues* of existing deep-JSCC-based SemCom frameworks, saving computation and radio resources. Moreover, our decoder model works with a range of channel bandwidth ratios (CBR), and the CDiff decoder does not need to participate in end-to-end joint training with every single neural encoder. Once trained with an arbitrary encoder, the model is inherently capable of adapting to different neural encoders of different sizes through the diffusion conditioning. Our visions is that receivers will have a general-purpose semantic decoder, acting as a "foundation" decoder model in the future nextG systems.

To summarize, our contributions can be outlined as follows: i) We present a novel machine learning model for SemCom systems, which is based on conditional denoising diffusion models. We present the data pipelines for training and inference, as well as the sampling framework. ii) While generative models are *probabilist models* by nature, we provide formal guarantees, using probability theory, to statistically prove that the proposed decoder model is a *consistent estimator* of the ground-truth data. iii) We provide extensive simulation results over two different datasets, studying the effect of CBR, signal-to-noise ratio (SNR), and multi-user semantic interference. We highlight 55% improvement in terms of the learned perceptual image patch similarity (LPIPS) and 30% improvement in terms of structural similarity index measure (SSIM) compared to autoencoders with matched encoder-decoder architecture, as well as 50% improvement in LPIPS compared to VAE benchmark.

2 Problem Formulation

Consider a SemCom system for data transmission from a source node equipped with a semantic encoder to a destination node equipped with a semantic decoder. Source data is denoted by $\mathbf{x}_0 \in \mathbb{R}^n$, following a non-trivial probability distribution $\mathbf{x}_0 \sim p_{\mathbf{x}_0}(\mathbf{x}_0)$. The semantic encoder, denoted by $\mathcal{E}(\cdot)$, is supposed to extract the high-level semantics of the source data, mapped to a semantic latent vector $\mathbf{x}_{\mathsf{sem}}$, i.e., $\mathbf{x}_{\mathsf{sem}} = \mathcal{E}(\mathbf{x}_0; \phi)$, where the encoder is parameterized by ϕ. Considering $\mathbf{x}_{\mathsf{sem}} \in \mathbb{R}^{2k}$, the channel input is treated as the equivalent k-dimensional complex-valued vector $\boldsymbol{z} \in \mathbb{C}^k$, normalized to average transmit power constraint P, i.e., $\frac{1}{k}\sum_{i=1}^{k} \mid z_i \mid^2 \leq P$. The source data dimension, n, is referred to as the source bandwidth, while the channel dimension k characterizes the channel bandwidth, and the bandwidth compression ratio (BCR) is defined as k/n with $k < n$. This can be seen as an indicator on the spectral or temporal radio resources available [9]. Communication channel is modeled as a transfer function $\eta(\cdot)$, where $\eta(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{n}$, with $\boldsymbol{n} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \boldsymbol{I}_k)$. Accordingly, a noisy distorted version of the semantic latents, denoted by $\mathbf{y}_{\mathsf{sem}}$, is received by the semantic decoder $\mathcal{D}(\cdot)$ parametrized by θ, which semantically reconstructs the data, using the received noisy semantic latent vectors, i.e., $\hat{\mathbf{x}}_\theta = \mathcal{D}(\mathbf{y}_{\mathsf{sem}}; \theta)$.

The problem is to obtain at the semantic decoder, high-quality semantic reconstructions $\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}(\hat{\mathbf{x}})$ close to the transmitter's ground-truth data $\mathbf{x}_0 \sim p_{\mathbf{x}_0}$ in the same signal space. The quality of the reconstruction is typically evaluated via distortion measures such as peak signal-to-noise ratio (PSNR). Nevertheless, the focus in SemCom frameworks is more on the perceptual reconstructions. This can be achieved by utilizing statistical distance measures (such as Kullback-Leibler (KL) divergence or f-divergence) between the generated probability distribution and the source probability distribution. Hence, the general objective function can be expressed as a weighted sum of the two objective functions, i.e., the distortion function $d_1(\mathbf{x}, \hat{\mathbf{x}})$ over the signal space, and the statistical divergence measures $d_2(P_X, P_{\hat{X}})$ over the probability distributions:

$$\underset{\phi,\theta}{\text{minimize}}\ \mathbb{E}_{\mathbf{x}_0 \sim p_{\mathbf{x}_0}} \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}|\mathbf{x}_0}} \left[d(\mathbf{x}_0, \hat{\mathbf{x}}_\theta) \right] = \lambda_1 \mathbb{E}_{p(\mathbf{x}_0, \hat{\mathbf{x}}_\theta)} \left[d_1(\mathbf{x}_0, \hat{\mathbf{x}}_\theta) \right] + \lambda_2 d_2(p_{\mathbf{x}_0}, p_{\hat{\mathbf{x}}}). \tag{1}$$

Solving this problem can be viewed as a domain translation type of problem, where the goal is to learn the semantic-to-clean mapping from noisy semantics to ground-truths. Semantic data transmission can be treated as a forward degradation process, which is typically assumed to be highly non-linear. Then the problem can be viewed as the *inverse problem,* with the received semantics treated as "measurements" which are gone through a typically non-invertible forward process. This can be approximated by a forward operator $\mathcal{F}(\cdot)$, where $\mathbf{y}_{\mathsf{sem}} = \mathcal{F}(\mathbf{x}_0) \approx \eta(\mathcal{E}(\mathbf{x}_0; \phi))$. Then the problem is to invert the forward process and semantically reconstruct the source data from the "observations", which can be realized by minimizing the following approximated loss function

$$\mathcal{L}(\phi, \theta) := \mathbb{E}\left[d\left(\mathbf{x}_0, \mathcal{D}\left(\eta\left(\mathcal{E}\left(\mathbf{x}_0; \phi\right)\right); \theta\right)\right)\right]. \tag{2}$$

Given the forward model and the measurements $\boldsymbol{y}_{\mathsf{sem}}$, this problem is solved via imposing a fidelity constraint and a proper prior on the expected ground truth [11]. Typically, the fidelity constraint is chosen to be MSE, and the prior knowledge can be realized via diffusion generative priors.

3 Solution

Fig. 1 depicts a schematic of the proposed solution. Semantic encoder extracts the semantics, which are then communicated with the receiver. The semantic latent vector is used for the conditioning of the CDiff decoder, starting the denoising from $\boldsymbol{x}_T$ as input. The intuition behind $\boldsymbol{x}_T$ as input here is that it inherently comes from the noisy (stochastic) version of the ground-truth which the decoder has learned during the forward diffusion training. While the autoencoder architecture has limitations in compressing stochastic

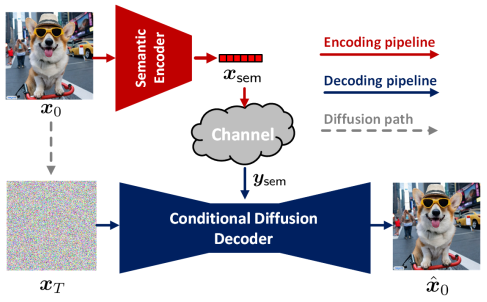


Figure 1: Communication pipeline of the proposed SemCom framework.

details, this is smartly handled by the diffusion model that learns to denoise and decode the semantics, while preserving the fine-grained details.

3.1 Semantic Encoder Model

Semantic encoder's functionality is modeled by a mapping from the source data to semantic latent, i.e., $\mathbf{x}_0 \to \mathbf{y}_{\textsf{sem}}$. The goal of the encoder is to summarize high-dimensional source information into a descriptive vector $\mathbf{y}_{\textsf{sem}} = \mathcal{E}(\mathbf{x}_0; \phi)$ of latent semantic representations such that it contains the necessary information to assist the decoder in the denoising and reconstruction process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_{\textsf{sem}})$. We do not assume any particular architecture for this encoder; because, thanks to the conditioning mechanism of diffusion models, we do not assume any matched encoder-decoder architecture, and the decoder model can work with varying-length latents. In this paper, since we assume that the source data-stream is image-type data, we employed CNN architecture, with details in Appendix C.

3.2 Diffusion-Based Semantic Decoder Model

The main focus of the proposed scheme is on the design of the decoder model. The CDiff, realized via a conditional DDPM, first goes through a forward diffusion process, i.e., the Markov chain of $(\mathbf{x}_0; \mathbf{y}_{\textsf{sem}})$ — $(\mathbf{x}_t; \mathbf{y}_{\textsf{sem}})$ — $\mathbf{x}_T$, purposefully diffusing the ground-truth samples by adding noise. That is, the decoder first aims to learn the finer-grained details of source information, which are not expressed by $\mathbf{y}_{\textsf{sem}}$ due to the limited capacity of the neural encoder. Then in a reverse process, the CDiff decoder de-maps back to the ground-truth probability distribution space, starting from $\mathbf{x}_T$ and guided by the semantics $\mathbf{y}_{\textsf{sem}}$. I.e., $(\mathbf{y}_{\textsf{sem}}; \mathbf{x}_T)$ — $(\mathbf{y}_{\textsf{sem}}; \mathbf{x}_t)$ — $\mathbf{x}_0$. The decoding transition probability can be modeled by a conditional probability distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_{\textsf{sem}})$. The proposed CDiff decoder is expected to estimate the reverse transition probability distribution that match the true distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x})$. The reverse process of "generative decoding" can then be modeled as

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}) \approx q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0), \overset{(a)}{\approx} q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}; t)\right), \tag{3}$$

$$p_\theta(\boldsymbol{x}, \boldsymbol{x}_{1:T}|\mathbf{y}_{\textsf{sem}}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}), \tag{4}$$

where the true distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x})$ is approximated in (a) by replacing the unknown ground-truth $\mathbf{x}_0$ with its corresponding estimate $\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}; t)$.

Forward process & training Having the semantic information to guide the decoder via conditioning, the forward diffusion process can be modeled via the following transition probability

$$q(\mathbf{x}_t|\mathbf{x}_0, \widetilde{\mathbf{y}}_{\textsf{sem}}) \sim \mathcal{N}\Big((1 - w_t)\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + w_t\sqrt{\bar{\alpha}_t}\widetilde{\mathbf{y}}_{\textsf{sem}}, \delta_t \boldsymbol{I}\Big), \tag{5}$$

where $\delta_t := (1 - \bar{\alpha}_t) - w_t^2\bar{\alpha}_t$, and $\widetilde{\mathbf{y}}_{\textsf{sem}}$ is the padded-and-reshaped version of the conditional information, to have the same dimensions as the source samples, to enable distilling the semantics into the learning process. This is a generalization of the vanilla forward diffusion process, where a weighted sum (parameterized by w_t) of the ground-truth and the conditioning is considered. Typically, w_t start from 0 and gradually increases to $w_T \approx 1$ via a pre-defined scheduling. Intuitively, this guides the model during each step of training, to first focus on diffused ground-truth samples, explore, and learn their fine-grained details; and then, while the diffused samples loose their expressiveness due to high volume of noise, the model switches its attention to learning from the semantic latents.

Training is then carried out by optimizing the following loss function.

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\substack{t\sim\textsf{Unif}[T] \\ \mathbf{x}_0, \mathbf{y}_{\textsf{sem}} \sim p(\mathbf{x}_0, \mathbf{y}_{\textsf{sem}}) \\ \mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0, \mathbf{y}_{\textsf{sem}})}} \Big[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}; t) - \mathbf{x}_0\|_2^2 \Big]. \tag{6}$$

Sampling process Each step of the decoding (so-called "sampling") is viewed as an interpolation between the denoised samples $\mathbf{x}_t$, the conditional information regarding the source data (semantic latents in our case) $\mathbf{y}_{\textsf{sem}}$, and the predicted noise $\boldsymbol{\epsilon}_\theta(\cdot, t)$:

$$\mathbf{x}_{t-1} = \psi_x\mathbf{x}_t + \psi_y\mathbf{y}_{\textsf{sem}} - \psi_\epsilon\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{y}_{\textsf{sem}}, t) + \sqrt{\delta_t}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, I) \tag{7}$$

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}, t)$ is the noise predictor model obtained by reparameterization of $\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t)$:

$$\epsilon_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}, t) := \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t))}{\sqrt{1-\bar{\alpha}_t}}. \tag{8}$$

The coefficients ψ_x, ψ_y, and ψ_ϵ can be estimated by solving the evidence lower bound (ELBO) optimization criterion on the objective function following [16]. In this work, we use the coefficients as derived in [14], where $\psi_x = \frac{\delta_{t-1}(1-\lambda_t)}{\delta_t(1-\lambda_{t-1})}\sqrt{\alpha}_t + (1-\lambda_{t-1})\frac{\delta_{t|t-1}}{\delta_t\sqrt{\alpha}_t}$, $\psi_y = (\lambda_{t-1}\delta_t - \frac{\lambda_t(1-\lambda_t)}{1-\lambda_{t-1}}\alpha_t\delta_{t-1})\frac{\sqrt{\bar{\alpha}_{t-1}}}{\delta_t}$, $\psi_\epsilon = (1-\lambda_{t-1})\frac{\delta_{t|t-1}\sqrt{1-\bar{\alpha}_t}}{\delta_t\sqrt{\alpha_t}}$, and $\delta_{t|t-1}{=}\delta_t - \left(\frac{1-\lambda_t}{1-\lambda_{t-1}}\right)^2\alpha_t\delta_{t-1}$.

3.3 Theoretical Analysis

Theorem 1. *Assume θ_n^* be a minimizer of an n-sample Monte Carlo approximation of $\mathcal{L}(\theta, \phi)$ with respect to the decoder model θ. The conditional diffusion decoder $\hat{\mathbf{x}}_{\theta_n^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t)$ is a consistent estimator of the ground-truth, i.e., for sufficiently large number of Monte Carlo samples, we have*

$$\hat{\mathbf{x}}_{\theta_n^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) \xrightarrow{P} \mathbf{x}_0. \tag{9}$$

Proof. Please see Appendix B □

4 Experiments

4.1 Results over MNIST

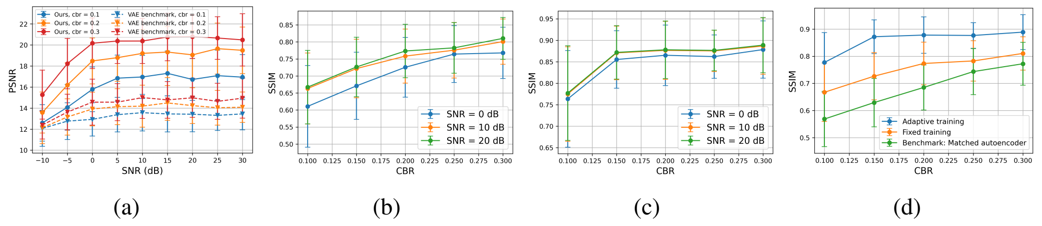


Figure 2: Effects of SNR and CBR on SemCom performance for different training strategies over MNIST.

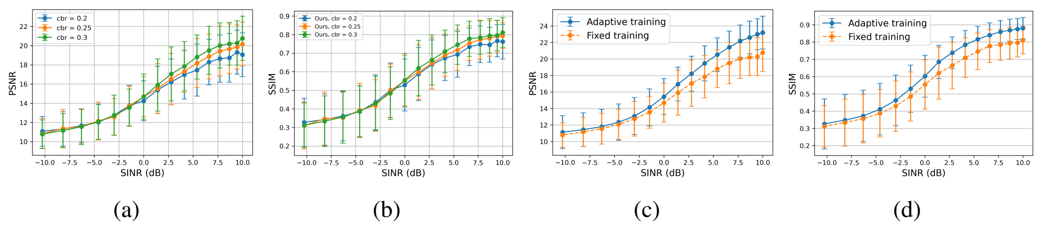


Figure 3: Effects of multi-user interference (in terms of SINR) on SemCom performance over MNIST.

Fig. 2 illustrates the reconstruction performance of our scheme over a wide range of channel SNRs and CBR values. In Fig. 2-(a) we start with studying the very common reconstruction performance in terms of the PSNR. Notably, our scheme reasonably shows the capability to improve its performance when the CBR or SNR is increased, while the VAE benchmark does not show any notable performance in response to the improvement in channel bandwidth or SNR. We further remark that the scheme has been trained over SNR values between -10 dB to 10 dB, and a fixed CBR of 0.3 (for fixed-training strategy) while showcasing promising results over a wide range of channel SNRs and CBR values. Please see Appendix D for the details on training and hyperparameters. In Figs. 2-(b) to 2-(d), the effect of CBR is studied. Fig. 2-(b) evaluates the performance of the model when trained for a fixed CBR (0.3), while Fig. 2-(c) evaluates the performance when the model is trained over a set of defined CBRs (see Appendix D for the details). Fig. 2-(d) compares the results with the matched autoencoder architecture, highlighting outperforming of our scheme especially for radio resource limited scenarios (i.e., low CBRs, thus low available bandwidth). Comparing Figs. 2-(b) and 2-(c), it can be observed that training the model over a range of CBRs can make the model more robust (less variations) against changes in channel SNR and CBR. These curves provide important design guidelines: One can choose among different available CBRs, depending on how much physical resources are available, and how accurate the reconstruction quality is aimed to be. For example, only 10% SSIM improvement can be achieved if we have three times more communication resources in terms of channel bandwidth (CBR of 0.1 versus 0.3). One can trade-off whether the SemCom performance is important, or it is preferred to save radio resources (channel bandwidth) by $1/3$.

In Fig. 3, we extend the simulations for the scenario of multi-user SemCom. This is simulated by having our CDiff decoder receive a convex combination of semantic latents from two neural encoders each independently transmitting their own i.i.d data samples over the same channel. The figure studies the effect of multi-user interference measured in terms of the signal-to-interference-plus-noise ratio (SINR) on the reconstruction performance in terms of PSNR and SSIM. While CBR seems to be the most important factor in most of deepJSCC/SemCom papers, this is the case when the typical point-to-point single user scenario is considered. However, in a practical system, there might be multiple encoders for multiple intended users, whose latent vectors (semantic meanings) might interfere. Fig. 3 indicates that the interference signals seem to be the dominating factor in a practical multi-user setup, not the CBR. This is because the PSNR/SSIM metrics vary with SINR, (not with the change in CBR). This can be a very important practical insight for SemCom system designs.

4.2 Results over CIFAR-10

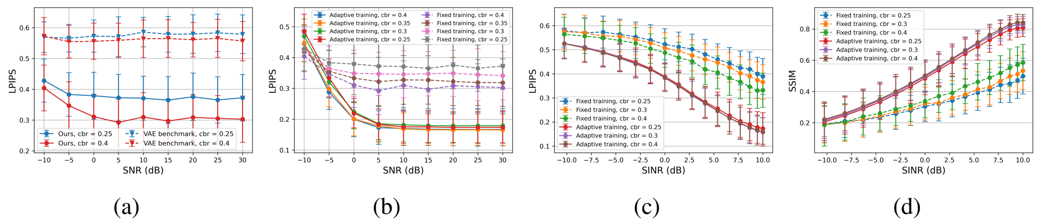


Figure 4: Effects of SNR and SINR on SemCom performance for different training strategies over CIFAR-10.

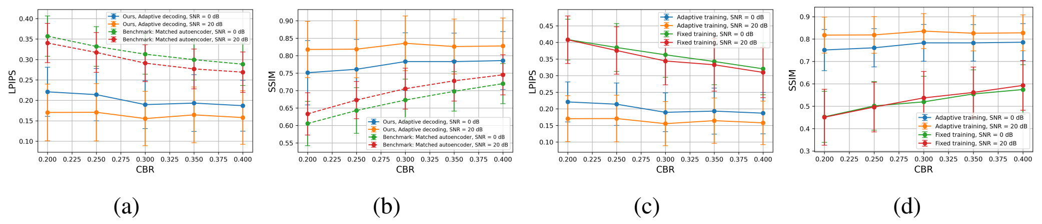


Figure 5: Effects of CBR on SemCom performance over CIFAR-10.

Fig. 4 studies the performance with respect to LPIPS and SSIM metrics, over a wide range of SNRs and interference levels. Fig. 4-(a) highlights about 50% improvement in LPIPS achieved from our scheme compared to the VAE benchmark. This can be explained due to the fact that VAEs do not maintain any ‘fine-grained” denoising mechanism to explore over probability distributions. Moreover, VAEs typically cast a standard normal distribution on the prior distribution of the latent space, which can compromise the decodability quality of semantics, while our scheme does not impose any constraint, allowing the neural encoder to obtain the semantics as expressive as it can. Fig. 4-(b) indicates the effect of adaptive training (over a set of CBRs provided to the neural encoder) in achieving a robust performance during inference, with respect to changes in channel SNR and CBR values. Figs. 4-(c) and 4-(d) highlight the effect SINR, indicating the dominating factor of interference in a more realistic multi-user SemCom transmission, as the LPIPS/SSIM metrics show more significant variations with changes in SINR level, than with changes in the selected CBR.

Fig. 5 compares our scheme with the matched autoencoder benchmark. In addition to the scalability issues the autoencoders have in a practical setup (discussed earlier), our scheme showcases more than 55% improvement in terms of LPIPS and around 30% improvement in terms of SSIM according to Figs. 5-(a) and 5-(b), respectively. Moreover, Figs. 5-(c) and 5-(d) highlight the resource-utility trade-off curve. The figures indicate one does not need to select high CBRs when trained via adaptive CBR strategy, since almost the same LPIPS/SSIM can be achieved with lower CBRs (e.g., 0.2), thus saving radio resources and bandwidth. Further visual experiments are provided in Appendix E.

5 Conclusions

Conditional diffusion autoencoders have been proposed for the nextG wireless SemCom systems. The proposed scheme utilizes a conditional DDPM framework for the semantic decoder, and is capable of flexibly decoding semantic latents without the need for the decoder architecture to be coupled to the encoder architecture, while also capable of learning the fine-grained details through diffusion learning. Theoretical analysis along with extensive simulations have been carried out to study the semantic reconstruction performance, as well as providing insights and practical guidelines.

Acknowledgments

This work was supported by the Research Council of Finland (former Academy of Finland) 6G Flagship Programme under Grant 346208, and in part by the 6GARROW project which has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101192194 and from the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (No. RS-2024-00435652). The authors wish to also acknowledge CSC - IT Center for Science, Finland, for computational resources.

References

[1] D. Gündüz et al., "Beyond transmitting bits: Context, semantics, and task-oriented communications," *IEEE Journal on Selected Areas in Communications,* vol. 41, no. 1, pp. 5–41, Jan. 2023.

[2] N. Van Huynh et al., "Generative AI for physical layer communications: A survey," *IEEE Transactions on Cognitive Communications and Networking,* vol. 10, no. 3, pp. 706–728, June 2024.

[3] E. Erdemir, T. -Y. Tung, P. L. Dragotti, and D. Gündüz, "Generative joint source-channel coding for semantic image transmission," *IEEE Journal on Selected Areas in Communications,* vol. 41, no. 8, pp. 2645–2657, Aug. 2023.

[4] T. Jayashankar, G.C. Lee, A. Lancho, A. Weiss, Y. Polyanskiy, and G. Wornell, "Score-based source separation with applications to digital communication signals," *Advances in Neural Information Processing Systems 36 (NeurIPS 2023),* Dec. 2023, vol 36, pp. 5092–125.

[5] M. Arvinte and J. I. Tamir, "MIMO channel estimation using score-based generative models," *ICLR Workshop on Deep Generative Models for Highly Structured Data (ICLR 2022),* Mar. 2022.

[6] M. Letafati, S. Ali, M. Latva-aho, "Denoising diffusion probabilistic models for hardware-impaired communications," *2024 IEEE Wireless Communications and Networking Conference (WCNC),* Dubai, United Arab Emirates, 2024, pp. 1-6.

[7] M. Letafati, S. Ali, and M. Latva-Aho, "Diffusion model-aided data reconstruction in cell-free massive MIMO downlink: A computation-aware approach," *IEEE Wireless Communications Letters,* vol. 13, no. 11, pp. 3162–3166, Nov. 2024.

[8] M. Letafati, S. Ali, and M. Latva-Aho, "Conditional denoising diffusion probabilistic models for data reconstruction enhancement in wireless communications," *IEEE Transactions on Machine Learning in Communications and Networking,* vol. 3, pp. 133–146, 2025.

[9] J. Xu, T. -Y. Tung, B. Ai, W. Chen, Y. Sun, and D. Gündüz, "Deep joint source-channel coding for semantic communications," *IEEE Communications Magazine,* vol. 61, no. 11, pp. 42–48, Nov. 2023.

[10] S. F. Yilmaz, X. Niu, B. Bai, W. Han, L. Deng, and D. Gündüz, "High perceptual quality wireless image delivery with denoising diffusion models," *2024 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* Vancouver, BC, Canada, 2024, pp. 1–5.

[11] J. Chen, D. You, D. Gündüz, and P. L. Dragotti, "CommIN: Semantic image communications as an inverse problem with INN-guided diffusion models," *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Seoul, Korea, Republic of, 2024, pp. 6675–6679.

[12] T. Wu, Z. Chen, D. He, L. Qian, Y. Xu, M. Tao, and W. Zhang, "CDDM: channel denoising diffusion models for wireless semantic communications," *IEEE Transactions on Wireless Communications,* vol. 23, no. 9, pp. 11168–11183, Sep. 2024.

[13] Y. Zeng, X. He, X. Chen, H. Tong, Z. Yang, and Y. Guo, "DMCE: Diffusion model channel enhancer for multi-user semantic communication systems," *ICC 2024 - IEEE International Conference on Communications,* Denver, CO, USA, 2024, pp. 855-860.

[14] Y. -J. Lu, Z. -Q. Wang, S. Watanabe, A. Richard, C. Yu, and Y. Tsao, "Conditional diffusion probabilistic model for speech enhancement," *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Singapore, Singapore, 2022, pp. 7402-7406.

[15] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 586–595, 2018.

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems,* vol. 33, pp. 6840–6851, 2020.

[17] S. M. Ross, *A First Course in Probability,* 9th ed. Boston, MA, USA: Pearson, 2012.

[18] W. K. Newey and D. McFadden, "Large sample estimation and hypothesis testing," in *Handbook of Econometrics,* vol. 4, R. F. Engle and D. L. McFadden, Eds. Amsterdam, The Netherlands: Elsevier, 1994, pp. 2111–2245.

[19] Z. Liu, H. Mao, C.Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 11976–11986.

[20] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in Neural Information Processing Systems 33 (NeurIPS 2020),* Dec. 2020.

A Preliminaries on conditional DDPM

Denoising diffusion probabilistic models (DDPM) are capable of learning, over iterative steps, to reverse a stochastic diffusion process over which the ground-truth data is transformed into an unstructured noisy distribution. The diffusion process is typically defined via a Gaussian kernel that gradually perturbs an input sample $\mathbf{x}_0 \sim p_{\mathbf{x}_0}(\mathbf{x}_0)$ following a noise variance scheduling over T steps. This can be formulated by

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{\bar{\sigma}_t}\boldsymbol{\epsilon}, \quad \mathbf{x}_0 \sim p_{\mathbf{x}}(\mathbf{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{10}$$

where ϵ denotes an isotropic Gaussian, and α_t and σ_t are defined according to a noise schedule with diffusion signal-to-noise ratio (SNR) of $\mathsf{SNR} = \frac{\bar{\alpha}_t}{\bar{\sigma}_t}$. According to the literature [16], each step of the diffusion process is designed as expressed below, such that $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon, \quad \mathbf{x}_0 \sim p_{\mathbf{x}}(\mathbf{x}_0), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{11}$$

DDPMs train a parameterized neural network $\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{c}; t)$ to learn the reverse of the diffusion process, where the inputs are the noisy sample $\mathbf{x}_t$, the time-step t, and any conditioning input, $\mathbf{c}$, (e.g. a label, a vector, or a discrete code) as side-information related to the ground-truth. The objective is to estimate and generate ground-truth samples over fine-grained denoising steps. This is typically defined as a mean squared error (MSE) loss between the ground-truth and its noisy versions $\mathbf{x}_t$ averaged over the denoising steps $t \in [T]$:

$$\mathcal{L}(\theta) := \mathbb{E}_{t,(\mathbf{x}_0,\mathbf{c}),\boldsymbol{\epsilon}} \left\| \hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{c}; t) - \mathbf{x}_0 \right\|_2^2. \tag{12}$$

B Proofs

Theorem 1: *Assume θ_n^* be a minimizer of an n-sample Monte Carlo approximation of $\mathcal{L}(\theta, \phi)$ with respect to the decoder model θ. The conditional diffusion decoder $\hat{\mathbf{x}}_{\theta_n^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t)$ is a consistent estimator of the ground-truth, i.e., for sufficiently large number of Monte Carlo samples, we have*

$$\hat{\mathbf{x}}_{\theta_n^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) \xrightarrow{P} \mathbf{x}_0. \tag{13}$$

Proof. We first make the following assumptions:

- **Assumption 1.** Assume that the space of parameters Θ, where $\theta \in \Theta$, and the data space $\mathcal{X}$, where $\mathbf{x}_0 \in \mathcal{X}$, are compact.

- **Assumption 2.** Assume that there exists a unique $\theta^* \in \Theta$ such that $\hat{\mathbf{x}}_{\theta^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) = \mathbf{x}_0$.

Inspecting the loss function in (6), and using conditional independence and the law of total expectation [17, Ch. 7, Proposition 5.1.], we can write

$$\mathbb{E}_{\substack{t\sim\mathsf{Unif}[T]\\ \mathbf{x}_0,\mathbf{y}_{\mathsf{sem}}\sim p(\mathbf{x}_0,\mathbf{y}_{\mathsf{sem}})\\ \mathbf{x}_t\sim q(\mathbf{x}_t|\mathbf{x}_0,\mathbf{y}_{\mathsf{sem}})}} \left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) - \mathbf{x}_0\|_2^2 \right] \tag{14}$$

$$= \mathbb{E}_{\substack{t\sim\mathsf{Unif}[T]\\ \mathbf{x}_0,\mathbf{x}_t,\mathbf{y}_{\mathsf{sem}}\sim p(\mathbf{x}_0,\mathbf{x}_t,\mathbf{y}_{\mathsf{sem}})}} \left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) - \mathbf{x}_0\|_2^2 \right]. \tag{15}$$

Now define $\nu := (t, \mathbf{x}_0, \mathbf{x}_t, \mathbf{y}_{\mathsf{sem}})$ and $q(\nu) := p(t, \mathbf{x}_0, \mathbf{x}_t, \mathbf{y}_{\mathsf{sem}})$ as the joint probability distribution. Also define $f(\nu, \theta) := \|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) - \mathbf{x}_0\|_2^2$. Since $t \sim \mathsf{Unif}[T]$ is independent of $(x_0, x_t, y_{\mathsf{sim}}) \sim p(\mathbf{x}_0, \mathbf{x}_t, \mathbf{y}_{\mathsf{sem}})$, (15) can be rewritten as

$$\mathcal{L}(\theta) = \mathbb{E}_{\nu\sim q(\nu)}[f(\nu, \theta)] \tag{16}$$

Therefore, by uniform law of large numbers [18, Lemma 2.4], the Monte Carlo approximation of (14), i.e., $\mathcal{L}^{(n)}(\theta) = \frac{1}{n}\sum_{i=1}^n f(\nu_i, \theta)$ converges uniformly in probability to $\mathcal{L}(\theta) = \mathbb{E}_{\nu\sim q(\nu)}[f(\nu, \theta)]$.

Now consider the following Lemma:

Lemma 1 (Consistency of extremum estimators)**.** *[18, Theorem 2.1]*
Let Θ be compact and consider a family of functions $\mathcal{L}^{(n)} : \Theta \to \mathbb{R}$. Moreover, suppose there exists a function $\mathcal{L} : \Theta \to \mathbb{R}$ such that

- *$\mathcal{L}(\theta)$ is uniquely minimized at θ^*.*
- *$\mathcal{L}(\theta)$ is continuous.*
- *$\mathcal{L}^{(n)}(\theta)$ converges uniformly in probability to $\mathcal{L}(\theta)$.*

Then

$$\theta_n^* := \underset{\theta\in\Theta}{\arg\min}\, \mathcal{L}^{(n)}(\theta) \xrightarrow{P} \theta^*.$$

Let θ^* be the minimizer of $\mathcal{L}(\theta)$ with respect to θ, then by Lemma 1, it indicates that $\theta_n^* \xrightarrow{P} \theta^*$, and by Assumption 2, it implies that $\hat{\mathbf{x}}_{\theta^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) = \mathbf{x}_0$. Therefore, $\hat{\mathbf{x}}_{\theta_n^*}(\mathbf{x}_t, \mathbf{y}_{\mathsf{sem}}; t) \xrightarrow{P} \mathbf{x}_0$, and the proof is completed. □

C Simulation Setup and neural network architectures

Simulations are carried out using PyTorch run on an NVIDIA V100 GPU for AI training and inference. Evaluations are carried out over MNIST[1] (resized to 32×32) and CIFAR-10[2] datasets. MNIST dataset is split into 60,000 training images and 10,000 testing images of hand-written digits, and CIFAR-10 dataset consists of 60000 colored images of dimension $32 \times 32 \times 3$ pixels (height, width, channels), with separated training and evaluation sets containing 50000 and 10000 images, respectively. We ignore CIFAR-10 classes as our goal in this work is semantic reconstruction, not data classification. Evaluation metrics to measure the quality of semantic reconstruction over MNIST are PSNR and SSIM, while for CIFAR-10, SSIM and LPIPS metrics are considered. LPIPS (learned perceptual image patch similarity) measures the perceptual similarity between images by evaluating how close the representations of the images are in high-dimensional feature space. LPIPS correlates more strongly with human perception of image quality than metrics like PSNR and SSIM, which are more sensitive to pixel-level differences. Lower LPIPS value indicates higher perceptual similarity between the original data and the regenerated version at the decoder. Our plots include error bars. We have used the `errorbar` function from `pyplot` in `matplotlib` library of Python, which takes the statistical mean and standard deviation as input and plot the error bars via `plt.errorbar(vals, mean_vals, yerr = std_vals)`.

[1]`http://yann.lecun.com/exdb/mnist`

[2]`https://www.cs.toronto.edu/~kriz/cifar.html`

Encoder's neural architecture The encoder takes images of input size $\texttt{channels} \times 32 \times 32$, with $\texttt{channel}$ equal to 1 for MNIST and 3 for CIFAR-10 dataset respectively, and maps them to a vector of complex channel symbols, with the number of symbols determined by the channel bandwidth ratio (CBR). Under MNIST dataset, the neural encoder consists of three convolutional layers with kernel size 3×3, stride 2, and padding 1, having 8, 16, and 32 output channels, respectively. Each convolutional layer is followed by a ReLU activation. Similarly, under CIFAR-10 dataset, the neural encoder consists of four convolutional layers with 64, 128, 256, and 256 output channels, respectively, followed by batch normalization and a ReLU activation. The convolutional stack progressively downsamples the input to a $32 \times 4 \times 4$ feature map over MNIST ($256 \times 4 \times 4$ over CIFAR-10), which is then flattened and projected through a fully connected layer to produce a latent vector of length $2N_c$, where N_c is the number of complex symbols (with real and imaginary parts stored separately), i.e., $\texttt{latent\_dim} = 2 * \texttt{int}(\texttt{self.input\_dim} * \texttt{cbr})$. Finally, the latent vector is normalized to have unit ℓ_2-norm to satisfy the average power constraint.

For the benchmarks, the encoder architecture follow the same architecture as the one employed for our scheme, and the decoder counterpart mirrors the encoding functionality for the autoencoder benchmark (matched decoder). While for the VAE benchmark, flattened features are further passed through two parallel linear layers which respectively estimate the mean μ and log-variance $\log \sigma^2$ of the posterior distribution over latent variables. VAE sampling is then performed using the standard reparameterization trick, where $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. A normalization step ensures the power normalization for the sampled latent vector before transmission, emulating practical communication constraints. VAE training is carried out with the Adam optimizer (learning rate 0.001), jointly training the encoder and decoder parameters with a loss function that combines (with equal weights) a pixel-wise MSE reconstruction loss with the KL divergence regularizer, ensuring the latent space follows a Gaussian prior.

Diffusion decoder architecture The implementation of our diffusion model is inherited from the seminal DDPM framework [16][3] implemented it in PyTorch (the original code was in TensorFlow). We further incorporate the conditioning mechanism into this framework, where the conditioning mechanism is developed upon [14][4] and adopted to the image generation task (the original code is for speech signal generation).

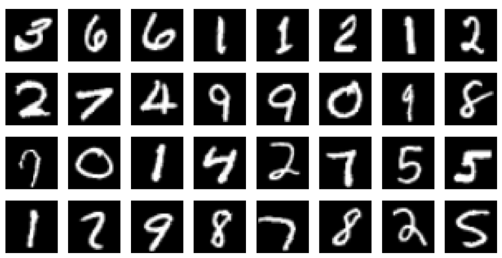
The core learning model as the backbone of our decoder is a time-conditioned U-Net architecture, integrated into a SemCom data pipeline, and optimized to operate jointly with neural encoders. The U-Net uses a base spatial feature dimension $\texttt{dim} = 32$ with four resolution stages determined by the multiplier tuple $(1, 2, 4, 8)$. The input to the U-Net is the diffused original samples $\boldsymbol{x}_t$ concatenated with the conditioning input (noisy semantic latent vector), where the semantics are (padded and) reshaped into a $(\texttt{channels}, 32, 32)$ tensor to match the image spatial dimensions, ensuring that semantic conditioning information can be injected directly into the spatial feature maps during training and denoising. The network begins with an initial convolution that takes two-channel input (original channels concatenated with the conditional tensor) and maps it to an initial dimension $\texttt{init\_dim} = (\texttt{dim}//3) * 2$ From there, each stage of the encoder consists of two ConvNeXt-style residual blocks [19] with depthwise convolution of kernel size 7, followed by two pointwise convolutions, GELU activation, and group normalization (with a single group) with learnable skip connections. ConvNeXt blocks are extended with a conditioning pathway that incorporates time-step embeddings (from the diffusion process) through a sinusoidal position embedding projected by an MLP to dimension of `time_dim = dim * 4 = 128`, which is added to the block's activation. This enables the network to adapt its representation across different noise levels in the generative process. Within each resolution stage, a linear attention module (single-head dot-product attention with linear complexity) is wrapped in a pre-norm residual structure, i.e., the feature maps are normalized, and processed by the attention operation, and finally the original input is added back to the attention output. Spatial resolution is reduced between U-Net stages across the encoder path by a downsampling operation via a regular convolution layer (kernel size 4, stride 2, padding 1), enabling the network to progressively increase its receptive field and capture higher-level semantic features until the bottleneck is reached. At the bottleneck, two ConvNeXt blocks are applied with a full multi-head attention layer in between, capturing both local and global context at the coarsest spatial resolution. The decoder path mirrors the encoder in reverse order. Each upsampling stage concatenates the skip connection from its symmetric encoder stage, then processes the tensors with two ConvNeXt blocks, a linear attention layer, and an upsampling layer. The final stage employs a ConvNeXt block followed by a 1×1 convolution, defaulting back to the original channel dimensions $\texttt{out\_dim} = 1$ for MNIST and $\texttt{out\_dim} = 3$ for CIFAR-10, respectively, so as to having output samples with the same spatial dimension as the input tensor.

D Training details

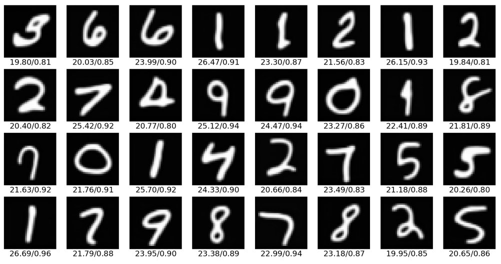
Training of our scheme is carried out in two different scenarios: i) Fixed training (for 10 epochs over MNIST and 50 epochs over CIFAR-10 dataset), where a fixed CBR of 0.3 over MNIST and 0.4 over CIFAR-10 is set for the neural encoder, and the joint training of the neural encoder and diffusion decoder is done for a fixed

[3]`https://github.com/hojonathanho/diffusion`

[4]`https://github.com/neillu23/CDiffuSE`



(a) Original samples



(b) Reconstructed samples

Figure 6: Data visualization over MNIST for test SNR 0 dB. PSNR/SSIM values are shown below each sample.

CBR. Nevertheless, the evaluations are still carried out under different CBRs to highlight the scalability of our approach, i.e, no need for re-training the pipeline for different CBRs. Rather, once trained with an arbitrary neural encoder (mimicked here as a neural encoder with CBR of 0.3 or 0.4), the CDiff decoder is inherently capable of adapting to different neural encoders of different types/sizes via the padding-and-masking condition mechanism. In the fixed training scenario, to test how well the model generalizes to lossy links with CBRs lower than the training CBR (or scenarios where less radio resources are available in inference compared to the training stage), stochastic masking (aka soft masking) is applied to the neural encoder's output (followed by the re-normalization to preserve average transmit power constraint) before sending the semantic latents over the wireless channel. In stochastic masking, we randomly drop or scale down parts of the neural encoder output probabilistically (instead of hard-zeroing as in hard masking). Intuitively, this basically allows us to simulate a channel scenario where each symbol has a chance to get through independently. ii) Adaptive training (for 20 epochs over MNIST and 50 epochs over CIFAR-10 dataset), where the neural encoder is provided with a selectable set of channel bandwidth ratio (CBR) from a predefined set $\{\texttt{CBR1}, \texttt{CBR2}, \dots\}$. The feature extraction stage is shared across all CBR settings and the same neural architecture is assumed, where for each CBR value, a dedicated fully connected projection layer maps this shared feature vector to a latent vector of length $2N_c$, where N_c is the number of complex symbols for the selected CBR. We defined the set of CBRs for the adaptive neural encoder as $\texttt{cbr\_list} = [0.2, 0.25, 0.3, 0.35, 0.4, 0.45]$, and the joint training is done via randomly choosing from this set in each epoch.

We set the number of denoising steps to 200, and use adaptive moment estimation (Adam) optimizer with learning rate $\lambda = 10^{-3}$. To stabilize the training process, exponential moving average (EMA) method is implemented [20] for model update in each epoch, where a copy of the previous weights is kept, and the weighted average between the previous and new version of the weights are calculated as the model update. For the conditioning weights $0 < w_t < 1$, we consider a linear scheduling. For the variance scheduling β_t, we also consider a linear scheduling, starting from $\beta_0 = 0.0001$ to $\beta_T = 0.0095$. The values were simply obtained out of trial-and-error. The SNR of communication links is defined as:

$$\Gamma = 10 \log_{10} \frac{P}{\sigma^2} \ \text{(dB)}. \tag{17}$$

This basically represents the ratio of the average power of the encoded data-stream of the latent space (i.e., the channel input) to the average AWGN noise power. Without loss of generality, we set the average signal power to $P = 1$ for all experiments, while varying the SNR by setting the noise variance σ^2. Training SNR is chosen uniformly from the range of $-10 \leq \Gamma^{\texttt{train}} \leq 10$ dB during training, while the evaluations are carried out over a wider range $-10 \leq \Gamma^{\texttt{test}} \leq 30$ dB to highlight the generalization of our scheme.

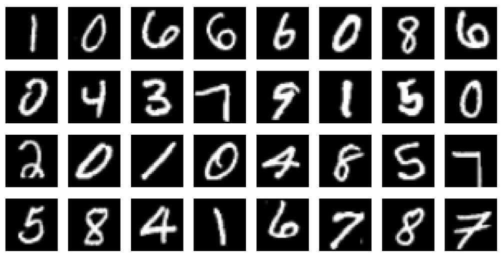
E Further experiments

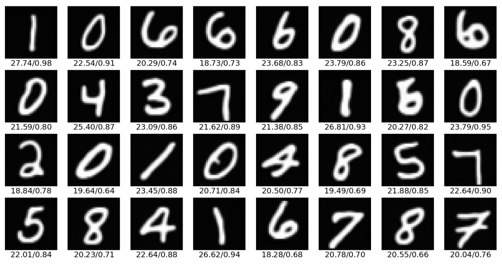
Figs. 6, 7, 8, and 9 further provide the visualizations on the reconstructed samples at our CDiff decoder, where the performance metrics are shown below each sample.

F Limitations

We would like to point out the limitations of our work, in terms of the assumptions, datasets used, computation efficiency, and practical implications.

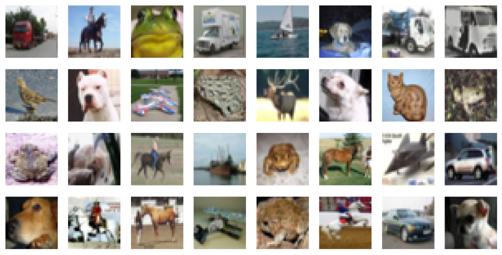
Performance under non-stationary fading channels While in this work, we modeled the channel as an additive Gaussian noise (AWGN channel), the performance of SemCom systems under non-stationary wireless

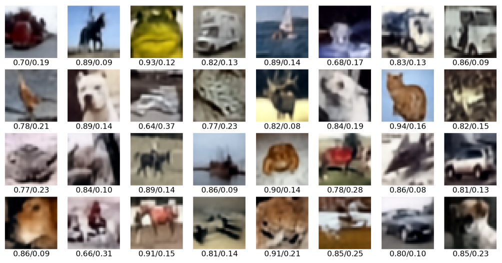

(a) Original samples



(b) Reconstructed samples

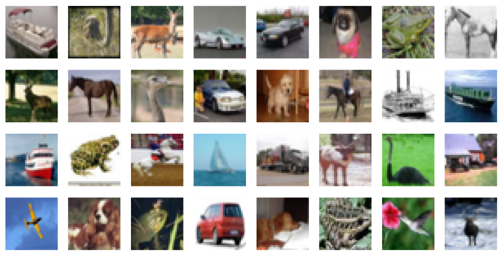
Figure 7: Data visualization for multi-user scenario over MNIST. A convex combination with coefficients of 0.8 and 0.2 are assumed for the source semantics and the interfering latents (equivalent $\texttt{SINR} \approx -2.1$ dB). Test SNR is set to 0 dB. PSNR/SSIM values are shown below each sample.

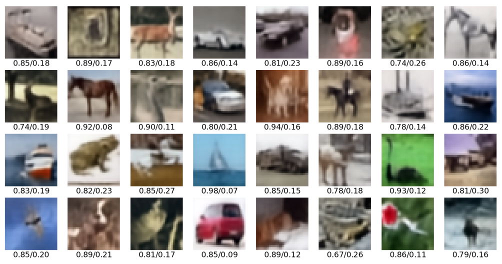

(a) Original samples



(b) Reconstructed samples

Figure 8: Data visualization over CIFAR-10 for test SNR 10 dB. SSIM/LPIPS values are shown below each sample.


(a) Original samples



(b) Reconstructed samples

Figure 9: Data visualization for SemCom reconstitution under multi-user interference over CIFAR-10. A convex combination with coefficients 0.9 and 0.1 are assumed for the source semantics and the interfering latents respectively (equivalent $\texttt{SINR} \approx 16$ dB). Test SNR is set to 20 dB. SSIM/LPIPS values are shown below each sample.

channel models needs careful consideration. One approach to solve this issue would be to re-train or fine-tune the systems (joint end-to-end fine-tuning) over *real-time* wireless channels. However, another challenge arises here: the channel coherence time of wireless channels in practice are relatively larger than the rate at which data samples are processed for training. Hence, only a few channel realizations might be observed over every batch, which may not be favorable in terms of generalization to a wide range of channel realizations. Another simple approach might be to rely on traditional channel equalizers before feeding the semantic latents to the decoder.

Out-of-distribution (OOD) performance OOD performance limitation of our scheme should be revisited with respect to both the wireless channel models (discussed above), and also the datasets over which the models are trained. SemCom systems so far have shown to be typically dataset-dependent. This can be seen as one major challenge with such systems in practical deployments. Still, the fact that datasets are shared among the communication parties can be seen as a reasonable interpretation: In the context of SemCom, the transmitter and receiver share a common *knowledge base* (KB) to realize a common "mutual understanding" of the semantic meanings. This is typically realized as a dataset shared between both sides. In practice, transmitter (e.g., the terminal device equipped with neural encoder) can signal the dataset identifier (ID) to the receiver (e.g., the cellular gNodeB equipped with a foundation decoder model), and the receiver can activate its corresponding model (among pool of models) pre-trained on the corresponding dateset ID; and then the communication gets started after handshaking on the KB.

Sampling speed and computation efficiency A practical limitation of our work is that DDPMs may not be computation efficient due to their iterative denoising. Nevertheless, there is an active line of research within the AI community, exploring ways to accelerate the generation process, and we envision that more computation-efficient diffusion models will evolve in the near future. Moreover, this can be viewed in another way: SemCom framework makes the communication systems radio resource-efficient, saving the transmit power, physical resource blocks, and also the bandwidth, due to sending only the minimal relevant information rather than the entire datastream. Notably, this is achieved by moving the processings from PHY to AI. The AI processing can be done at cloud, over-the-top (OTT) server from device perspective, or core network from gNodeB perspective, in a practical nextG cellular system, thanks to richer compute resources available in those network entities. Hence, from communication perspective, Semcom is actually making the systems more efficient, particularly for resource-constrained wireless systems.

Definitions, specification, and standardizations Concepts and definitions we used in this work for semantic latents, conditioning mechanism, neural encoding/decoding functionality, etc., may not be the same as the ones which will be specified in practical SemCom deployments in nextG. For a wide adoption of SemCom, proper definitions for semantic information, semantic noise, semantic interference, semantic error rate, knowledge base, etc., should be specified and agreed upon cellular partners to be standardized with respect to signaling interfaces, data, and control information elements to ensure interoperability in real-world setups.