Improved Variational Inference in Discrete VAEs using Error Correcting Codes

María Martínez-García[1,2] **Grace Villacrés**[3] **David Mitchell**[4] **Pablo M. Olmos**[2,5]

[1]Dept. of Computer Science, Saarland University, Saarbrücken, Germany
[2]Dept. of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, Spain
[3]Dept. of Signal Theory and Communications, Universidad Rey Juan Carlos, Fuenlabrada, Spain
[4]Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, USA
[5]Instituto de Investigación Sanitaria Gregorio Marañón, Madrid, Spain

Abstract

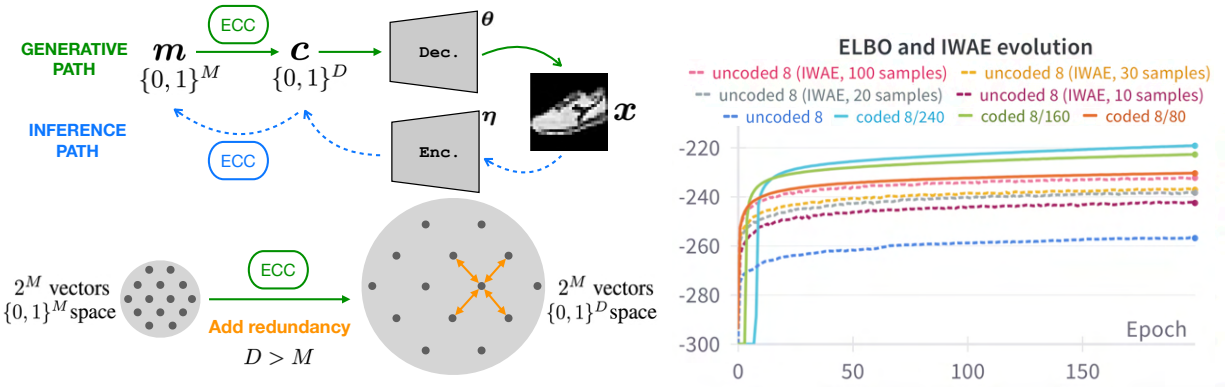
Despite advances in deep probabilistic models, learning discrete latent representations remains challenging. This work introduces a novel method to improve inference in discrete Variational Autoencoders by reframing the inference problem through a generative perspective. We conceptualize the model as a communication system, and propose to leverage Error Correcting Codes (ECCs) to introduce redundancy in latent representations, allowing the variational posterior to produce more accurate estimates and reduce the variational gap. We present a proof-of-concept using a Discrete Variational Autoencoder with binary latent variables and low-complexity repetition codes, extending it to a hierarchical structure for disentangling global and local data features. Our approach significantly improves generation quality, data reconstruction, and uncertainty calibration, outperforming the uncoded models even when trained with tighter bounds such as the Importance Weighted Autoencoder objective. We also outline the properties that ECCs should possess to be effectively utilized for improved discrete variational inference.

1 INTRODUCTION

Discrete latent space models represent data using a finite set of features, making them particularly well-suited for representing inherently discrete data modalities. Additionally, they provide benefits in terms of interpretability and computational efficiency compared to continuous representations [Van Den Oord et al., 2017, Jang et al., 2017, Vahdat et al., 2018a]. Consequently, recent advancements in deep generative models have increasingly embraced discrete latent representations [Razavi et al., 2019, Ramesh et al., 2021, Rombach et al., 2022].

However, model training and inference of low-dimensional discrete latent representations remains technically challenging due to non-differentiability, which often requires the use of approximations that can lead to unstable optimization [Jang et al., 2017, Kool et al., 2019]. Currently, Vector Quantized-Variational Autoencoders (VQ-VAEs) [Van Den Oord et al., 2017, Razavi et al., 2019] stand out as state-of-the-art discrete latent variable models. They rely on a non-probabilistic encoder trained with a straight-through estimator, which lacks uncertainty quantification in the latent space. In contrast, fully probabilistic discrete Variational Autoencoders (VAEs) [Kingma and Welling, 2014] employ continuous relaxations such as Gumbel-Softmax [Jang et al., 2017] or Concrete [Maddison et al., 2017] for differentiable sampling [Ramesh et al., 2021, Lievin et al., 2020]. However, these methods can be unstable, as gradient variance is highly sensitive to the temperature parameter. The Discrete Variational Autoencoder (DVAE) framework [Rolfe, 2016, Vahdat et al., 2018a,b] addresses this issue by augmenting binary latent representations with continuous variables, allowing for stable reparameterization after marginalizing the latent bits. In this setting, Boltzmann machines serve as discrete priors to enhance model flexibility, at the cost of degraded interpretability.

This work introduces a novel approach to improve variational inference in discrete latent variable models, particularly discrete VAEs, by offering an alternative perspective on the inference problem. Rather than viewing VAEs as lossy compression methods employed to minimize data reconstruction error, we adopt a generative perspective, where inference seeks to recover the latent vector that generated a given observation. This allows us to conceptualize the generative model as a communication system, where we reinterpret latent vectors as *messages* transmitted through a nonlinear *channel* given by the VAE decoder, and then recovered by the VAE encoder. Based on this analogy, we propose to incorporate Error Correcting Codes (ECCs) to introduce redundancy into latent representations before they are processed by the decoder to generate data.



(a) (b)

Figure 1: **ECC within a DVAE**. Fig. (a) illustrates the method and how adding redundancy transforms a dense M-dimensional latent space into a sparse D-dimensional one with 2^M valid vectors, enabling error correction in the variational posterior via minimum distance. Fig. (b) compares the ELBO and IWAE objectives for coded and uncoded models, where coding schemes with $M = 8$ and $D = 80, 160, 240$ introduce $D/8$ redundancy bits per latent bit.

Our approach builds on well-established techniques from digital communications and data storage, where information is *protected* using ECCs before transmission or storage to reduce the overall error rate during recovery [Moon, 2005]. Notably, Shannon's landmark work demonstrated that properly designed error correction schemes can achieve arbitrarily low error rates [Shannon, 1948]. We use ECCs to deterministically introduce controlled redundancy to the latent vectors, increasing their Hamming distance and creating a higher-dimensional but sparser space where only a subset of vectors are valid. As illustrated in Fig. 1a, this increased separation between latent codes allows for some errors in inference while still enabling recovery of the correct latent code by identifying the closest valid vector. Hence, this structure facilitates error correction during inference.

The integration of ECCs into discrete latent variable models is a novel approach that remains compatible with existing training methods for discrete generative models. Note that, while deep learning has been widely explored in digital communications [Guo et al., 2022, Wu et al., 2023, Shen et al., 2023, Ye et al., 2024, Chen et al., 2024], the use of ECCs as a design tool in machine learning remains largely unexplored. This work introduces a new research direction, leveraging communications theory to enhance deep generative models.

In our experiments, we use a simplified version of DVAE++ [Vahdat et al., 2018b] (referred to as *uncoded* DVAE) and show that the added redundancy enables a more accurate variational approximation to the true posterior, using simple independent priors. We highlight that error rates and variational gaps are linked through bounds derived from mismatch hypothesis testing, showing that minimizing the variational gap tightens the upper bound on the error rate [Schlüter et al., 2013, Yang et al., 2024]. Across different datasets, our results indicate that the DVAE with ECCs (Coded-DVAE) leads to reduced error rates in inference, resulting in smaller variational gaps. This improvement translates into superior generation quality, improved data reconstruction, and critically calibrated uncertainty in the latent space. Notably, Fig. 1b shows that Coded-DVAE achieves a significantly tighter training bounds even when the baseline is trained using the Importance Weighted Autoencoder (IWAE) objective [Burda et al., 2016].

Contributions. The primary contribution of this work is presenting a new perspective on the inference problem by framing the generative model as a communication system. We present proof-of-concept results that show how training discrete latent variable models can be improved by incorporating ECC techniques, an approach that, to the best of our knowledge, is novel in the existing literature. We then introduce a coded version of a discrete VAE, utilizing block repetition codes, and show that encoding and decoding can be done with linear complexity. Our results show that the Coded-DVAE model significantly reduces error rates during inference, leading to better data reconstruction, generation quality, and improved uncertainty calibration in the latent space. We further generalize this approach to other coding schemes and propose a hierarchical structure inspired by polar codes [Arikan, 2009], which effectively separates high-level information from finer details. [1]

2 OUR BASELINE: THE UNCODED DVAE

This section introduces a simplified version of the DVAE++ [Vahdat et al., 2018b] (*uncoded* DVAE), which serves as the foundational model upon which the subsequent aspects of our work are constructed. A key advantage of this framework is its stable training process, facilitated by reparame-

[1] Code: https://github.com/mariamartinezgarcia/codedVAE

terization through smoothing transformations, in contrast to more unstable methods like Gumbel-Softmax [Jang et al., 2017]. To ensure that any performance improvements in the Coded-VAE stem solely from the coding scheme rather than a complex prior, we adopt a simple independent prior.

Let $\boldsymbol{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N\}$ denote a collection of unlabeled data, where $\boldsymbol{x}_i \in \mathbb{R}^K$ represents a K-dimensional feature vector. While Rolfe [2016], Vahdat et al. [2018b] and Vahdat et al. [2018a] use Boltzmann machine priors, we consider a generative probabilistic model characterized by a simple low-dimensional binary latent variable $\boldsymbol{m} \in \{0,1\}^M$ comprising independent and identically distributed (i.i.d.) Bernoulli components $p(\boldsymbol{m}) = \prod_{j=1}^M p(m_j) = \prod_{j=1}^M \text{Ber}(\nu)$, where $\nu = p(m_j = 1)$. Since backpropagation through discrete samples is generally not possible, a smoothing transformation $p(\boldsymbol{z}|\boldsymbol{m}) = \prod_{j=1}^M p(z_j|m_j)$ of these binary variables is introduced. Following Vahdat et al. [2018b], we introduce truncated exponential distributions given by

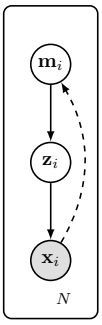


Figure 2: Graphical model of the uncoded DVAE.

$$p(z|m=1) = \frac{e^{\beta(z-1)}}{Z_\beta}, \quad p(z|m=0) = \frac{e^{-\beta z}}{Z_\beta}, \quad (1)$$

for $m \in \{0,1\}$, $z \in [0,1]$, and $Z_\beta = (1 - e^{-\beta})/\beta$. The parameter β serves as an inverse temperature term, similar to the one in the Gumbel-Softmax relaxation [Jang et al., 2017]. Given the simplicity of the defined binary prior, the complexity of the model is primarily determined by the likelihood function $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = p(f_{\boldsymbol{\theta}}(\boldsymbol{z}))$, the decoder function, where the likelihood is a Neural Network (NN) (referred to as the decoder) with parameter set $\boldsymbol{\theta}$.

2.1 VARIATIONAL FAMILY AND INFERENCE

Following Rolfe [2016], we assume an amortized variational family of the form $q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{m})$, where

$$q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}) = \prod_{j=1}^M \text{Ber}(m_j; q_j) = \prod_{j=1}^M \text{Ber}(m_j; g_{j,\boldsymbol{\eta}}(\boldsymbol{x})) \quad (2)$$

where $\text{Ber}(m_j; q_j)$ is a Bernoulli with parameter $q_j = q_{\boldsymbol{\eta}}(m_j = 1|\boldsymbol{x})$ and $g_{\boldsymbol{\eta}}(\boldsymbol{x})$ is a NN (referred to as the data encoder) with parameter set $\boldsymbol{\eta}$. Inference is achieved by maximizing the Evidence Lower Bound (ELBO), which can be expressed as

$$\log p(\boldsymbol{x}) \geq \int q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{m})}{q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z})} \right) d\boldsymbol{m} d\boldsymbol{z}$$
$$= \mathbb{E}_{q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \mathcal{D}_{KL}\big(q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})||p(\boldsymbol{m})\big), \quad (3)$$

where the first term corresponds to the reconstruction of the observed data and the second is the Kullback-Leibler (KL) Divergence between the variational family and the prior distribution, which acts as a regularization term. This can be computed in closed form as $\mathcal{D}_{KL}\big(q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})||p(\boldsymbol{m})\big) = \sum_{j=1}^M \left[q_j \log \frac{q_j}{\nu} + (1-q_j) \log \frac{1-q_j}{1-\nu} \right]$, with $\nu = p(m_j = 1)$. Since $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ does not depend on the binary latent variable $\boldsymbol{m}$, we can marginalize the posterior distribution as

$$q_{\boldsymbol{\eta}}(\boldsymbol{z}|\boldsymbol{x}) = \prod_{j=1}^M q_{\boldsymbol{\eta}}(z_j|\boldsymbol{x}),$$
$$q_\eta(z_j|\boldsymbol{x}) = \sum_{k=0}^1 q_\eta(m_j = k|\boldsymbol{x}) p(z_j|m_j = k). \quad (4)$$

As shown in Vahdat et al. [2018b], the corresponding inverse Cumulative Density Function (CDF) is given by

$$F^{-1}_{q_{\boldsymbol{\eta}}(z_j|\boldsymbol{x})}(\rho) = -\frac{1}{\beta} \log \left(\frac{-b + \sqrt{b^2 - 4c}}{2} \right), \quad (5)$$

where $b = \big(\rho + e^{-\beta}(q_j - \rho)\big)/(1 - q_j) - 1$ and $c = -[q_j e^{-\beta}]/(1 - q_j)$. Equation (5) defines a differentiable function that converts a sample ρ from an independent uniform distribution $\mathcal{U}(0,1)$ into a sample from $q_{\boldsymbol{\eta}}(\boldsymbol{z}|\boldsymbol{x})$. Thus, we can apply the reparameterization trick to sample from the latent variable $\boldsymbol{z}$ and optimize the ELBO.

3 ON THE VARIATIONAL GAP AND THE INFERENCE ERROR

While VAEs are typically viewed as lossy compression methods, this work introduces an alternative perspective on the inference problem from a generative viewpoint: we first sample a latent vector $\boldsymbol{m}$, generate an observation $\boldsymbol{x}$, and aim to minimize the error rate when recovering $\boldsymbol{m}$ from $\boldsymbol{x}$. This requires approximating the true, unknown posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{m}|\boldsymbol{x})$ with a sufficiently accurate proposed distribution $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$. The gap between $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{m}|\boldsymbol{x})$ is precisely the variational gap, which can be related to the error rate in inference, i.e. comparing the true $\boldsymbol{m}$ with samples drawn from $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$.

In statistical classification, particularly in the context of multiple hypothesis testing, classification error is the primary performance metric. Bayes' decision rule (also known as Bayes' test) minimizes classification error, assuming the true probability distributions governing the classification problem are known. In the case of model mismatch, e.g., when a posterior distribution is learned using variational inference, statistical bounds can be used to relate the classification error and the model estimation error [Schlüter et al., 2013, Yang et al., 2024]. In particular, for a generative model $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{m})$ with discrete latent vector $\boldsymbol{m}$, let $1 - A_{\boldsymbol{\theta}}$ be the error rate when estimating $\boldsymbol{m}$ from $\boldsymbol{x}$ using

the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{m}|\boldsymbol{x})$ and $1 - A_{\boldsymbol{\eta}}$ be the error rate using the variational approximate posterior $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$. Then, $\Delta \doteq A_{\boldsymbol{\theta}} - A_{\boldsymbol{\eta}}$ can be bounded as

$$0 \leq \Delta^2 \leq 1 - e^{-2\mathcal{D}_{KL}(q_{\boldsymbol{\eta}}(\boldsymbol{m},\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{m},\boldsymbol{x}))}, \tag{6}$$

where $\mathcal{D}_{KL}(q_{\boldsymbol{\eta}}(\boldsymbol{m},\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{m},\boldsymbol{x})) = \mathbb{E}_{p(\boldsymbol{x})}[\mathcal{D}_{KL}(q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{m}|\boldsymbol{x}))]$ is the variational gap. For fixed $\boldsymbol{\theta}$ (and thus for fixed $A_{\boldsymbol{\theta}}$), any improvement in $\boldsymbol{\eta}$ that reduces the expected variational gap also tightens an upper bound on $\Delta \doteq A_{\boldsymbol{\theta}} - A_{\boldsymbol{\eta}}$.

In fields where reliable data transmission or storage is crucial, introducing ECCs is a well-established approach to reduce the error rate when estimating a discrete source $\boldsymbol{m}$ transmitted through a noisy channel with output $\boldsymbol{x}$. Building on this, we propose using ECCs to protect $\boldsymbol{m}$ with controlled redundancy that the variational posterior $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ can leverage. This way, it is possible to reduce error rates in inference and refine the variational parameters $\boldsymbol{\eta}$ on the enhanced model, thus narrowing the variational gap. Notably, according to (6), a smaller variational gap results in a tighter upper bound on the error rates. In the following sections, we show that simple coding schemes enhance inference, improving generation, reconstruction, log-likelihoods (LLs), and uncertainty calibration in the latent space.

4 CODED-DVAE

This section extends the previously described DVAE, introducing an ECC over $\boldsymbol{m}$. We refer to this model as Coded-DVAE. The corresponding graphical model is shown in Fig. 3, where the diamond node represents a deterministic transformation. In ECCs, we augment the dimensionality of the binary latent space from M to D, introducing redundancy in a controlled and deterministic manner, where $R = M/D$ is the *coding rate*. An ECC is designed to maximize the separation between the 2^M possible codewords within the D-bit binary space, enabling efficient error correction by searching for the nearest valid codeword (see Fig. 1a). A random selection of codewords results in *random block codes* [Shannon, 1948], which are very robust and amenable to theoretical analysis. However, their lack of structure makes them impractical, as encoding and decoding require exhaustive codeword enumeration. Appendix N provides a detailed formulation of the encoding and decoding process for a random code within the Coded-DVAE.

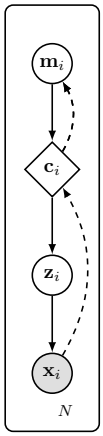


Figure 3: Graphical model of the Coded-DVAE.

We adopt a much simpler linear coding scheme, namely repetition codes, where each bit of the original message $\boldsymbol{m}$ (i.e., each information bit) is duplicated multiple times to form the encoded message $\boldsymbol{c}$. Intuitively, the more times an information bit is repeated, the better it is protected. Our experiments consider uniform (M, D) repetition codes where all bits are repeated L times, resulting in codewords of dimension $D = ML$ and a coding rate of $R = 1/L$. Note that repetition codes represent a special case of linear ECCs since each codeword can be computed by multiplying a binary vector $\boldsymbol{m}$ by an $M \times D$ *generator matrix* $\mathbf{G}$, such that $\boldsymbol{c} = \boldsymbol{m}^T\mathbf{G}$, where k-th row, with $k = 1, \ldots, M$, has entries equal to one at columns $L(k-1)+1, L(k-1)+2, \ldots, Lk$, and zero elsewhere. For example, for $M = 3$ and $L = 2$, the generator matrix of the $(3, 6)$ repetition code is

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{7}$$

The generative process of the Coded-DVAE is similar to the uncoded version and is illustrated in Fig. 3. We assume the same prior distribution $p(\boldsymbol{m})$, but in this case the samples $\boldsymbol{m}$ are deterministically encoded using $\mathbf{G}$. Consequently, the smoothing transformations are now defined over $\boldsymbol{c}$

$$p(\boldsymbol{z}|\boldsymbol{c}) = \prod_{j=1}^{D} p(z_j|c_j), \tag{8}$$

with $p(z_j|c_j)$ following Eq. (1). The likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ is again of the form $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = p(f_{\boldsymbol{\theta}}(\boldsymbol{z}))$. Compared to the uncoded case, the input dimensionality to the decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ is L times larger due to the introduced redundancy. Therefore, the overall architecture of the decoder (detailed in Appendix C) remains unchanged, except for the first Multilayer Perceptron (MLP) layer that processes the input $\boldsymbol{z}$. When using a repetition code with rate $R = 1/L$, the additional parameters of $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ amount to $(L-1) \times h$, where h is the dimension of the first hidden layer.

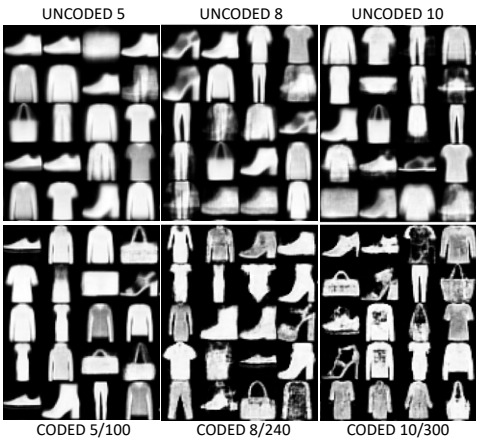
4.1 VARIATIONAL FAMILY AND INFERENCE

Since $\boldsymbol{c}$ is a deterministic transformation of $\boldsymbol{m}$, its randomness is entirely determined by $\boldsymbol{m}$. Therefore, following the recognition model described in Fig. 3, we assume a variational family factorizing as

$$q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{c}), \tag{9}$$

where $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}) = \prod_{j=1}^{M} q_{\boldsymbol{\eta}}(m_j|\boldsymbol{x})$ is computed in two steps. First, we construct an encoder $g_{\boldsymbol{\eta}}(\boldsymbol{x})$ similar to the one in (2), which estimates $\boldsymbol{c}$ from $\boldsymbol{x}$

$$q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) = \prod_{j=1}^{D} q^u_{\boldsymbol{\eta}}(c_j|\boldsymbol{x}) = \prod_{j=1}^{D} \text{Ber}(c_j; g_{j,\boldsymbol{\eta}}(\boldsymbol{x})), \tag{10}$$



Model	BER	BER MAP	WER	WER MAP	LL test
uncoded 5	0.051	0.051	0.195	0.190	-267.703
coded 5/50	0.011	$1.960{\cdot}10^{-4}$	0.046	$9.800{\cdot}10^{-4}$	-241.882
coded 5/80	0.008	$4.000{\cdot}10^{-6}$	0.039	$2.000{\cdot}10^{-5}$	-232.992
coded 5/100	0.010	$2.120{\cdot}10^{-4}$	0.049	$9.600{\cdot}10^{-4}$	-241.404
uncoded 8	0.089	0.088	0.384	0.376	-249.880
coded 8/80	0.021	0.003	0.144	0.021	-232.992
coded 8/160	0.027	0.001	0.189	0.009	-235.819
coded 8/240	0.037	0.004	0.231	0.023	-238.459
uncoded 10	0.142	0.144	0.622	0.644	-244.997
coded 10/100	0.040	0.005	0.321	0.036	-230.772
coded 10/200	0.044	0.006	0.341	0.039	-234.849
coded 10/300	0.045	0.002	0.349	0.014	-238.647

Figure 4: **Evaluation of generation in FMNIST.** Uncurated randomly generated samples (left) and evaluation metrics (right), including BER and WER sampled from the variational posterior, BER and WER using the MAP value, and test LL.

where the superscript u indicates that this posterior does not utilize the ECC (*uncoded*). Now, we leverage the known redundancy introduced by the ECC to constrain the solution of $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$, given that each bit from $\boldsymbol{m}$ has been repeated L times to create $\boldsymbol{c}$. To do so, we follow a *soft decoding* approach, where the marginal posteriors of the information bits are derived from the marginal posteriors of the encoded bits, exploiting the code's known structure. In the case of repetition codes, we compute the all-are-zero and the all-are-ones products of probabilities of the bits in $\boldsymbol{c}$ that are copies of the same message bit and re-normalize as

$$\begin{aligned} q(m_k = 1|\boldsymbol{x}) &= \frac{1}{Z} \prod_{j=L(k-1)+1}^{Lk} q^u_{\boldsymbol{\eta}}(c_j = 1|\boldsymbol{x}), \\ q(m_k = 0|\boldsymbol{x}) &= \frac{1}{Z} \prod_{j=L(k-1)+1}^{Lk} q^u_{\boldsymbol{\eta}}(c_j = 0|\boldsymbol{x}), \end{aligned} \tag{11}$$

for $k = 1, \dots, M$ and Z is the normalization constant. For the $M = 3$, $L = 2$ toy example in (7)

$$\begin{aligned} q_{\boldsymbol{\eta}}(m_1 = 1|\boldsymbol{x}) &\propto q^u_{\boldsymbol{\eta}}(c_1 = 1|\boldsymbol{x}) q^u_{\boldsymbol{\eta}}(c_2 = 1|\boldsymbol{x}), \\ q_{\boldsymbol{\eta}}(m_2 = 1|\boldsymbol{x}) &\propto q^u_{\boldsymbol{\eta}}(c_3 = 1|\boldsymbol{x}) q^u_{\boldsymbol{\eta}}(c_4 = 1|\boldsymbol{x}), \\ q_{\boldsymbol{\eta}}(m_3 = 1|\boldsymbol{x}) &\propto q^u_{\boldsymbol{\eta}}(c_5 = 1|\boldsymbol{x}) q^u_{\boldsymbol{\eta}}(c_6 = 1|\boldsymbol{x}). \end{aligned}$$

This approach can be seen as a soft majority voting strategy. All operations in (11) are differentiable and implemented in the log domain. We consider the same architecture for the encoder $g_{\boldsymbol{\eta}}(\boldsymbol{x})$ in (10) as in the uncoded case, with the only difference being the final MLP layer. The additional overhead in the coded case requires $(L-1) \times h'$ parameters in the last layer, where h' is the output dimension. In Appendix K.2, we conduct an ablation study on the number of trainable parameters to show that performance gains do not stem from the increased parameter count.

4.2 EFFICIENT REPARAMETERIZATION

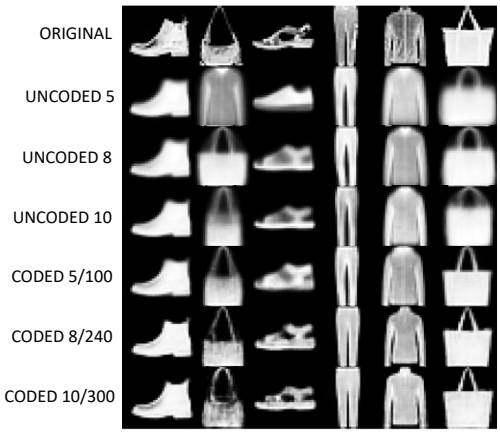
Given the variational family in (9), the ELBO remains unchanged from (3). However, the reparameterization trick in (5) assumes independent bits, which does not hold for $\boldsymbol{c}$. To address this issue efficiently during training, we adopt a *soft encoding* approach. We compute a marginal probability for each bit in $\boldsymbol{c}$, leveraging the ECC structure and the marginal posterior probabilities of the information bits in $\boldsymbol{m}$. For a repetition code, this involves simply replicating the posterior probabilities $q_{\boldsymbol{\eta}}(m_k|\boldsymbol{x}), k = 1, \dots, M$, for each copy of the same information bit[2]. Hence, we treat the bits in $\boldsymbol{c}$ as independent but distributed according to $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$. The training algorithm can be found in Appendix B.

When marginalizing $\boldsymbol{c}$ using the *soft encoding* marginals, we disregard potential correlations between the coded bits. Consequently, sampling from the marginals may produce an invalid codeword. However, since we do not sample the coded bits during training but instead propagate their marginal probabilities, we consider this method to have minimal negative impact. In fact, it can be seen as a form of probabilistic dropout, which enhances robustness during training. Since encoded words may contain inconsistent bits, the decoder learns to utilize the correlated inputs from repeated bits rather than disregarding them. Remark that when sampling from the generative model in test time, we use hard bits encoded into valid codewords, yielding visually appealing samples, indicating effective training.

5 RESULTS

This section empirically evaluates the Coded-DVAE with repetition codes and its uncoded counterpart on reconstruc-

[2]For the example in (7), this results in $p_{\boldsymbol{\theta}}(\boldsymbol{c}|\boldsymbol{x}) \approx q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) = \text{Ber}(c_1; q_1)\text{Ber}(c_2; q_1)\text{Ber}(c_3; q_2)\text{Ber}(c_4; q_2)\text{Ber}(c_5; q_3)\text{Ber}(c_6; q_3)$.



Model	PSNR	Acc	Conf. Acc	Entropy
uncoded 5	14.477	0.536	0.536	0.237
coded 5/50	16.241	0.647	0.700	1.899
coded 5/80	16.624	0.688	0.748	2.180
coded 5/100	16.702	0.700	0.757	2.256
uncoded 8	15.598	0.594	0.595	0.467
coded 8/80	17.318	0.750	0.816	2.905
coded 8/160	17.713	0.783	0.831	3.637
coded 8/240	17.861	0.799	**0.893**	4.000
uncoded 10	16.000	0.644	0.648	0.659
coded 10/100	17.694	0.790	0.850	3.879
coded 10/200	18.009	0.814	0.871	4.609
coded 10/300	**18.111**	**0.817**	0.870	5.076

Figure 5: **Reconstruction performance in FMNIST**. The figure at the left shows an example of reconstructed test images obtained with different model configurations. Observe that more details are visualized as we increase the number of bits in the latent space and decrease the coding rate. The table at the right includes reconstruction metrics. Acc is the semantic accuracy, and Conf. Acc the confident semantic accuracy. Entropy is the average entropy of $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ in the test set.

tion and generation tasks. In particular, we present results for MNIST [Deng, 2012], FMNIST [Xiao et al., 2017], CIFAR10 [Krizhevsky et al., 2009], and Tiny ImageNet [Le and Yang, 2015]. Additionally, we compare the coded model to the uncoded DVAE trained with the IWAE objective (see Fig. 1 and Appendix H). All experiments use a fixed architecture (see Appendix C), modifying the encoder's output and decoder's input to accommodate the repetition code. We identify the models based on the number of information bits for uncoded models and the code rate for coded models.

5.1 GENERATION

We first evaluate the models using the image generation task. Fig. 4 presents examples of randomly generated images using various model configurations on FMNIST. Additional results for other datasets are provided in the Supplementary Material. All models produce more detailed and diverse images as the number of information bits increases. However, when the number of latent vectors becomes too high for the dataset's complexity, some codebook entries remain unspecialized during training. This results in generation artifacts, where images contain overlapping features from different object classes. A visual inspection of Fig. 4 suggests that such artifacts occur more frequently in the uncoded model.

Error metrics. The effectiveness of the ECC-based inference can be assessed by generating images and measuring errors in recovering $\boldsymbol{m}$ by either drawing one sample from the variational posterior $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ or using the Maximum a Posteriori (MAP) estimate. As shown in the table included in Fig. 4, the coded models significantly reduce both Bit Error Rate (BER) and Word Error Rate (WER) compared to the uncoded case for the same number of latent bits. Note also that the error rates grow with the number of latent bits, which is expected due to the increased complexity of the inference process.

Log-likelihood (LL). Aditionally, we estimated the data LL using importance sampling with 300 samples per observation. As shown in the table included in Fig. 4, coded models consistently outperform their uncoded counterparts on both train and test sets, aligning with the rest of the results. LL values generally improve with more information bits, reflecting increased model flexibility. However, reducing the code rate does not enhance LL, suggesting possible decoder overfitting, where reconstruction improves, but LL declines. This may be due to the use of feed-forward networks at the decoder's input, which might not fully capture the correlations in coded words (see Appendix L for details).

5.2 RECONSTRUCTION

We also assess the model's performance in image reconstruction. In the table in Fig. 5, we report the Peak Signal-To-Noise Ratio (PSNR) of reconstructions on the FMNIST test set, calculated as

$$\text{PSNR} = 20 \log \Big(\frac{\max(x_{i,j})}{\text{RMSE}(\boldsymbol{x}, \boldsymbol{x}')} \Big), \tag{12}$$

where $\max(x_{i,j})$ is the highest pixel value, RMSE represents the Root Mean Square Error, and $\boldsymbol{x}'$ the reconstructed image. Results for the other datasets can be found in the Supplementary Material. In all cases, coded models outperform their uncoded counterparts in reconstruction quality, as indicated by higher PSNR values, a trend also visible in Fig. 5. As the number of information bits increases, PSNR improves, reflecting greater model flexibility to capture data

structure. We also observe improved PSNR with reduced code rates, as more redundancy is added. However, this increased redundancy does not enhance the model's flexibility, which is determined by the number of information bits.

Semantic Accuracy. Since the PSNR operates at the pixel level, it does not account for the *semantic* errors committed by the model. To address this, we evaluate reconstruction accuracy in Fig. 5, checking whether the model reconstructs images within the correct semantic class (type of clothing in FMNIST). For this, we trained an image classifier for each dataset and compared the predicted labels of the reconstructed images with the ground truth labels. We also report a *confident* reconstruction accuracy, which is calculated by considering only the images projected into a latent vector with posterior probability above 0.4 [3]. Results indicate that incorporating an ECC enables the model to produce latent spaces that better capture the semantics of the data.

Posterior Uncertainty Calibration. Finally, we report the average entropy of the variational posterior $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ over the test set in the table included in Fig. 5. The low entropy in the uncoded models indicates low uncertainty when the model maps data points into the latent space, which could be beneficial if the model consistently assigns high probability to the correct latent vectors. However, the semantic accuracy results show that this is not the case for the uncoded model, as it often projects images into vectors corresponding to the wrong semantic class with high confidence.

Coded models, in contrast, improve semantic accuracy and exhibit higher entropy. This indicates that (i) the Coded-DVAE recognizes multiple latent vectors that might be related to a given semantic class, and (ii) the model's posterior

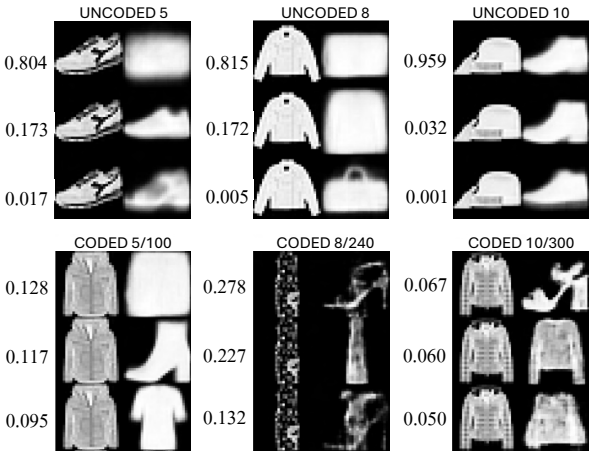


Figure 6: **Example of erroneous reconstructions in FMNIST**. The first column in each image shows the original images, while the second column displays the reconstructions. The $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$ probability is indicated in each row.

[3] We do not count errors when the MAP of $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}) < 0.4$.

Figure 7: **Reconstruction results** for CIFAR10 (top) and Tiny ImageNet (bottom).

demonstrates significant uncertainty (high entropy) for images where the class is not well identified. This is illustrated in Fig. 6, where we show images with class reconstruction errors caused by the MAP latent word from $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$. The reconstructions of the three most probable latent vectors, along with their respective probabilities, are displayed. Notably, the uncoded model shows high confidence regardless of the reconstruction result, while the coded posterior shows greater uncertainty. Note that this does not imply an uninformative distribution, as information bits remain recoverable. Additionally, increasing the number of latent bits does not lead to an excessive increase in entropy, despite the exponential growth in the number of vectors.

5.3 RESULTS ON CIFAR10 AND IMAGENET

Since MNIST-like datasets are relatively simple, it is challenging to fully assess the benefits of introducing ECCs in our model. To address this, we present additional results on more complex datasets, CIFAR10 and Tiny ImageNet, which feature colored images with more intricate shapes, patterns, and greater diversity. We trained both uncoded and coded models with different configurations to better understand the impact of the ECC. For context, the DVAE++ model [Vahdat et al., 2018b] required 128 binary latent variables to achieve state-of-the-art performance on these datasets, using a more complex model with Boltzmann Machine priors. In Fig. 7, we show reconstruction examples, and in Fig. 8, we present randomly generated images. Additional results are available in Appendices F and G. The results align with those from previous sections, but the performance difference is even more pronounced in this case. We find that the uncoded DVAE struggles to decouple spatial information in the images, while the Coded-DVAE shows

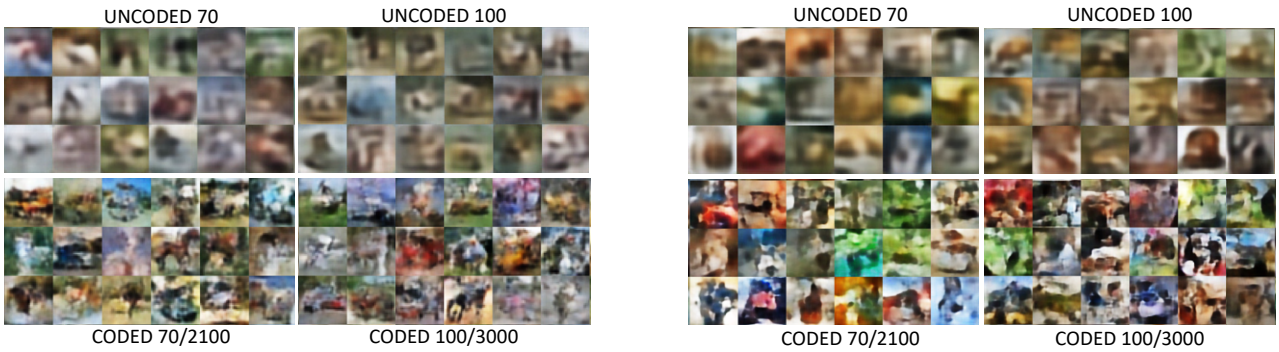


Figure 8: **Generation results** for CIFAR10 (left) and Tiny ImageNet (right).

particular promise for learning low-dimensional discrete latent representations, even for complex datasets. It is important to note that we used a simple architecture to isolate the gains achieved purely through the introduction of ECCs in the latent space, keeping a simple independent prior.

6 BEYOND REPETITION CODES

We have presented compelling proof-of-concept results that incorporating ECCs, like repetition codes, into DVAEs can improve performance. We believe this opens a new path for designing latent probabilistic models with discrete latent variables. Although a detailed analysis of the joint design of ECC and encoder-decoder networks is beyond the scope of this work, in Appendix I we outline key properties that any ECCs must satisfy to be integrated within this framework.

Drawing inspiration from polar codes [Arikan, 2009], we introduce a hierarchical Coded-DVAE with two layers. The graphical model is shown in Fig. 9. In this model, the latent bits $\boldsymbol{m}_1$ are encoded using a repetition code in the first layer, producing $\boldsymbol{c}_1$ and $\boldsymbol{z}_1$. Simultaneously, the vector $\boldsymbol{m}_2$ is linearly combined with $\boldsymbol{m}_1$ using modulo 2 operations $\left(\boldsymbol{m}_1 \oplus \boldsymbol{m}_2\right)$ and then encoded using a repetition code, yielding $\boldsymbol{c}_2$ and $\boldsymbol{z}_2$. Both soft vectors are concatenated and fed to the decoder to generate $\boldsymbol{x}$. Since $\boldsymbol{m}_1$ appears in both generative branches, it receives stronger protection. Inference follows a similar approach to the Coded-DVAE, incorporating the linear combination of $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ used in the second branch. This hierarchical structure allows the model to separate high-level information from finer details, as we show in the results presented in Appendix J. The bits in $\boldsymbol{m}_1$ offer stronger protection, allowing the model to encode the semantic information (type of clothing in FMNIST). As a result, during inference, it is more likely to project the image into a latent vector $\boldsymbol{m}_1$ that represents the correct semantic class. In contrast, $\boldsymbol{m}_2$ provides weaker protection and encodes image details that are less crucial for reconstruction. Fig. 9 demonstrates that by keeping $\boldsymbol{m}_1$ fixed (semantic information) and sampling $\boldsymbol{m}_2$ randomly (fine details) we can generate diverse images within the same semantic class.

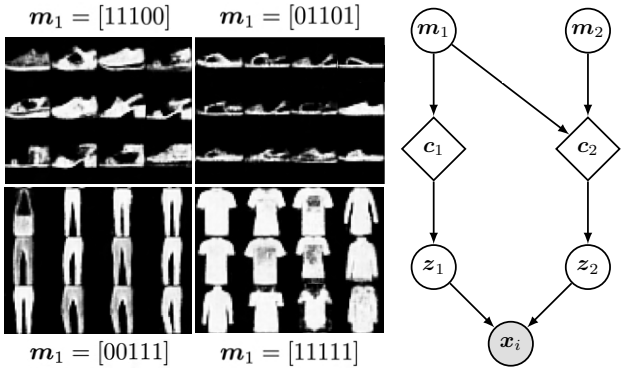


Figure 9: **Hierarchical Coded-DVAE**. Graphical model (right) and generated images with a 5/100 code per branch (left). Here, $\boldsymbol{m}_1$ is fixed, and $\boldsymbol{m}_2$ is randomly sampled.

7 CONCLUSION

This paper presents the first proof-of-concept demonstration that safeguarding latent information with ECCs within deep generative models holds promise for enhancing overall performance. By integrating redundancy into the latent space, the variational family can effectively refine the inference network's output according to the structure of the ECC. Our findings underscore the efficacy of simple and efficient ECCs, like repetition codes, showcasing remarkable improvements over a lightweight version of the DVAE introduced in Vahdat et al. [2018b].

Furthermore, our work reveals numerous avenues for future research. Firstly, investigating architectures capable of efficiently utilizing the correlations and structure introduced by the ECCs, in contrast to the feed-forward networks employed in this study. We also contemplate exploring more complex and robust coding schemes, conducting theoretical analyses aligned with Shannon's channel capacity and mutual information to determine the fundamental parameters of the ECC needed to achieve reliable inference, exploring different modulations, and integrating these concepts into state-of-the-art models based on discrete representations.

Acknowledgements

María Martínez-García was supported by the Generación de Conocimiento grant PID2022-142506NA-I00. Grace Villacrés was supported by the Comunidad de Madrid within the 2023-2026 agreement with Universidad Rey Juan Carlos for the granting of direct subsidies for the promotion, encouragement of research and technology transfer, line of Action A Emerging Doctors, under Project OrdeNGN (Ref. F1177). David Mitchell was supported by the National Science Foundation under Grant No. CCF-2145917 and he acknowledges travel support from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 951847. Pablo M. Olmos was supported by the Comunidad de Madrid IND2022/TIC-23550, IDEA-CM project (TEC-2024/COM-89), the ELLIS Unit Madrid (European Laboratory for Learning and Intelligent Systems), the 2024 Leonardo Grant for Scientific Research and Cultural Creation from the BBVA Foundation, and by projects MICIU/AEI/10.13039/501100011033/FEDER and UE (PID2021-123182OB-I00; EPiCENTER).

We would like to thank Isabel Valera for her valuable feedback on earlier drafts of the manuscript, as well as for the discussions that helped us improve the presentation of our method to the machine learning community.

References

Erdal Arikan. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, 2009.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. In *4th International Conference on Learning Representations (ICLR)*, 2016.

Kuan-Fu Chen, Ming-Chun Lee, Chia-Hung Lin, Wan-Chi Yeh, and Ta-Sung Lee. Multi-Fault and Severity Diagnosis for Self-Organizing Networks Using Deep Supervised Learning and Unsupervised Transfer Learning. *IEEE Transactions on Wireless Communications*, 23(1):141–157, 2024.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2022.

Kamélia Daudel, Joe Benton, Yuyang Shi, and Arnaud Doucet. Alpha-divergence variational inference meets importance weighted auto-encoders: Methodology and asymptotics. *Journal of Machine Learning Research*, 24 (243):1–83, 2023.

Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.

Jiajia Guo, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. Overview of Deep Learning-Based CSI Feedback in Massive MIMO Systems. *IEEE Transactions on Communications*, 70(12):8017–8045, 2022.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE Samples, Get a Baseline for Free! In *ICLR 2019 Deep Reinforcement Learning meets Structured Prediction Workshop*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. 2009.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Ya Le and Xuan Yang. Tiny ImageNet Visual Recognition Challenge. *CS 231N*, 7(7):3, 2015.

Valentin Victor David Julien Lievin, Andrea Dittadi, Lars Maaløe, and Ole Winther. Towards Hierarchical Discrete Variational Autoencoders. In *2nd Symposium on Advances in Approximate Bayesian Inference*. International Machine Learning Society (IMLS), 2020.

C Maddison, A Mnih, and Y Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, USA, 2005. ISBN 0471648000.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya

Sutskever. Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. *Advances in Neural Information Processing Systems*, 32, 2019.

Jason Tyler Rolfe. Discrete Variational Autoencoders. In *4th International Conference on Learning Representations (ICLR)*, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Tim Salimans and David A Knowles. On Using Control Variates with Stochastic Approximation for Variational Bayes and its Connection to Stochastic Linear Regression. *arXiv preprint arXiv:1401.1022*, 2014.

Ralf Schlüter, Markus Nussbaum-Thom, Eugen Beck, Tamer Alkhouli, and Hermann Ney. Novel tight classification error bounds under mismatch conditions based on f-divergence. In *2013 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2013.

Viktoria Schuster and Anders Krogh. The Deep Generative Decoder: MAP estimation of representations improves modelling of single-cell RNA data. *Bioinformatics*, 39 (9):btad497, 2023.

Claude Elwood Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.

Wenhan Shen, Zhijin Qin, and Arumugam Nallanathan. Deep Learning for Super-Resolution Channel Estimation in Reconfigurable Intelligent Surface Aided Systems. *IEEE Transactions on Communications*, 71(3): 1491–1503, 2023.

Jakub Tomczak and Max Welling. Vae with a vampprior. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 09–11 Apr 2018.

J Townsend, T Bird, and D Barber. Practical Lossless Compression with Latent Variables using Bits Back Coding. In *7th International Conference on Learning Representations (ICLR)*, volume 7, 2019.

Arash Vahdat, Evgeny Andriyash, and William Macready. DVAE#: Discrete Variational Autoencoders with Relaxed Boltzmann Priors. *Advances in Neural Information Processing Systems*, 31, 2018a.

Arash Vahdat, William Macready, Zhengbing Bian, Amir Khoshaman, and Evgeny Andriyash. DVAE++: Discrete Variational Autoencoders with Overlapping Transformations. In *International Conference on Machine Learning*, pages 5035–5044. PMLR, 2018b.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical Quantized Autoencoders. *Advances in Neural Information Processing Systems*, 33:4524–4535, 2020.

Yongzhi Wu, Filip Lemic, Chong Han, and Zhi Chen. Sensing Integrated DFT-Spread OFDM Waveform and Deep Learning-Powered Receiver Design for Terahertz Integrated Sensing and Communication Systems. *IEEE Transactions on Communications*, 71(1):595–610, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset For Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zijian Yang, Vahe Eminyan, Ralf Schlüter, and Hermann Ney. Refined statistical bounds for classification error mismatches with constrained bayes error. In *2024 IEEE Information Theory Workshop (ITW)*, pages 283–288. IEEE, 2024.

Xiaowen Ye, Qian Zhou, and Liqun Fu. Deep Reinforcement Learning-Based Scheduling for NR-U/WiGig Coexistence in Unlicensed mmWave Bands. *IEEE Transactions on Wireless Communications*, 23(1):58–73, 2024.

Improved Variational Inference in Discrete VAEs using Error Correcting Codes (Supplementary Material)

María Martínez-García[1,2] **Grace Villacrés**[3] **David Mitchell**[4] **Pablo M. Olmos**[2,5]

[1]Dept. of Computer Science, Saarland University, Saarbrücken, Germany
[2]Dept. of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, Spain
[3]Dept. of Signal Theory and Communications, Universidad Rey Juan Carlos, Fuenlabrada, Spain
[4]Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, USA
[5]Instituto de Investigación Sanitaria Gregorio Marañón, Madrid, Spain

The following Appendices offer further details on the model architecture, implementation, and experimental setup. They also include additional results on the FMNIST [Xiao et al., 2017], MNIST [Deng, 2012], CIFAR10 [Krizhevsky et al., 2009], and Tiny ImageNet [Le and Yang, 2015] datasets, along with comparisons to uncoded models trained with the IWAE objective [Burda et al., 2016], and a description of the hierarchical Coded-DVAE. Given the length of the material, we have included a Table of Contents for easier navigation.

A UNCODED TRAINING ALGORITHM

The following pseudo-code describes the training process for the uncoded DVAE. It's important to note that the main difference from the training of the Coded-DVAE lies in the fact that the encoder directly outputs $q^u_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}_i)$, which is used to sample $\boldsymbol{z}$. Therefore, we skip the soft decoding and coding steps.

Algorithm 1 Training the model with *uncoded* inference.

Input: training data $\boldsymbol{x}_i$.
repeat
  $q^u_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}_i) \leftarrow$ forward encoder $g_{\boldsymbol{\eta}}(\boldsymbol{x}_i)$
  $\boldsymbol{z} \leftarrow$ sample from (5)
  $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \leftarrow$ forward decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$
  Compute ELBO according to (3)
  $\boldsymbol{\theta}, \boldsymbol{\eta} \leftarrow Update(ELBO)$
until convergence

B CODED TRAINING ALGORITHM

The following pseudo-code describes the training process for the Coded-DVAE. Here, we utilize soft decoding to leverage the added redundancy and retrieve the marginal posteriors of the information bits $\boldsymbol{m}$, correcting potential errors in $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i)$. We then apply the soft encoding technique to incorporate the structure of the code and sample $\boldsymbol{z}$ using the reparameterization trick as described in (5).

Algorithm 2 Training the Coded-DVAE with repetition codes.

Input: training data $\boldsymbol{x}_i$, matrix $\mathbf{G}$.
repeat
  $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i) \leftarrow$ forward encoder $g_{\boldsymbol{\eta}}(\boldsymbol{x}_i)$
  $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}_i) \leftarrow$ soft decoding by aggregating $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i)$ according to (11)
  $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i) \leftarrow$ repeat posterior bit probabilities $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x}_i)$ according to $\mathbf{G}$
  $\boldsymbol{z} \leftarrow$ sample from (5)
  $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \leftarrow$ forward decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$
  Compute ELBO according to (3)
  $\boldsymbol{\theta}, \boldsymbol{\eta} \leftarrow Update(ELBO)$
until convergence

C ARCHITECTURE

In this section, we detail the architecture used to obtain the experimental results with FMNIST and MNIST (28x28 gray-scale images). Note that, across the experiments, we only modify the output layer of the encoder and the input layer of the decoder to adapt to the different configurations of the model. This modification leads to a minimal alteration in the total number of parameters. In Section K.2, we conduct an ablation study on the number of trainable parameters to show that the enhancement in performance is not attributed to the increased dimensionality introduced by redundancy.

For the additional CIFAR10 experiments, we change the input of the encoder and the output of the decoder to process the 32x32 color images. For the Tiny ImageNet experiments, we do the same to process 64x64 color images. The rest of the architecture remains unchanged.

These architectures are comprehensively described in the following subsections.

C.1 ENCODER

The encoder NN consists of 3 convolutional layers followed by two fully connected layers. We employed Leaky ReLU as the intermediate activation function and a Sigmoid as the output activation, since the encoder outputs bit probabilities. The

full architecture is detailed in Fig. 10.

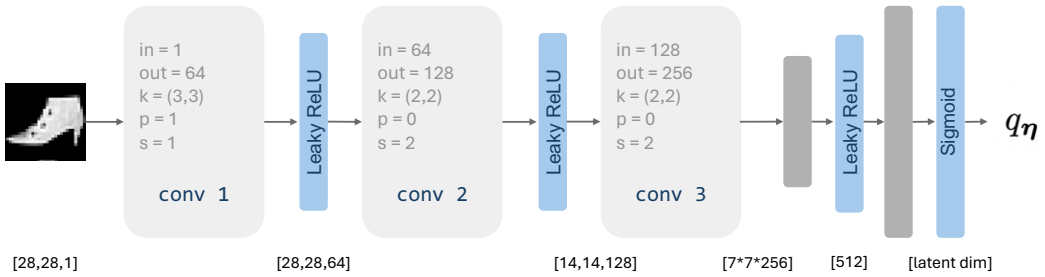


Figure 10: Block diagram of the **encoder architecture** for FMNIST and MNIST.

C.2 DECODER

The decoder architecture is inspired by the one proposed in Schuster and Krogh [2023]. It is composed of two fully connected layers, followed by transposed convolutional layers with residual connections and Squeeze-and-Excitation (SE) layers [Hu et al., 2018]. We employed Leaky ReLU as the intermediate activation function and a Sigmoid as output activation, given that we consider datasets with gray-scale images. The complete architecture is detailed in Fig. 11.

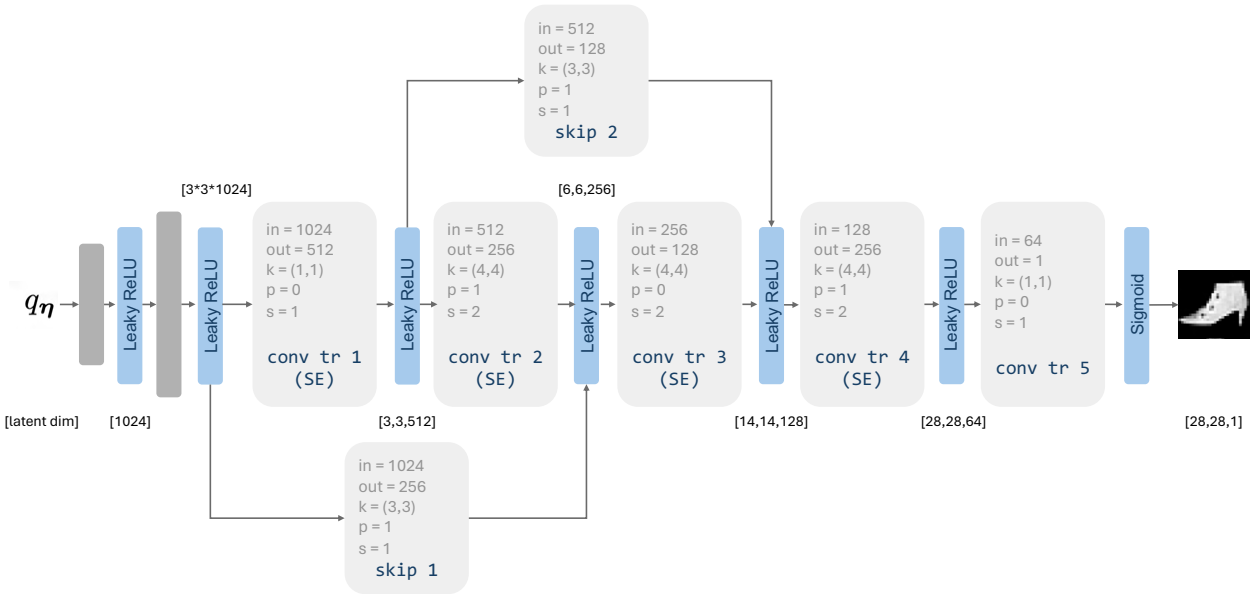


Figure 11: Block diagram of the **decoder architecture** for FMNIST and MNIST.

D FMNIST RESULTS

In this section, we present supplementary results obtained with the FMNIST dataset.

D.1 TRAINING

We present the evolution of the ELBO and its terms throughout the training process. The models were trained for 200 epochs using an Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. Fig. 12 displays the results for configurations with 5 information bits, Fig. 13 for 8 information bits, and Fig. 14 for 10 information bits. The colors in all plots represent the various code rates.

Across all cases, coded models achieve superior bounds. The main differences in the ELBO come from the different performances in reconstruction. As we have observed in the different experiments, coded models are capable of generating more detailed images and accurate reconstructions. Introducing the repetition code also leads to smaller KL values, indicating that the posterior latent features are potentially disentangled and less correlated.

We observe that, as we decrease the code rate, we obtain better bounds in general. Adding redundancy does not increase the model's flexibility, since the information bits determine the number of latent vectors. However, the introduction of ECCs in the model allows for latent spaces that better capture the structure of the images while employing the same number of latent vectors.

Figure 12: Evolution of the ELBO during training with 5 information bits on FMNIST.

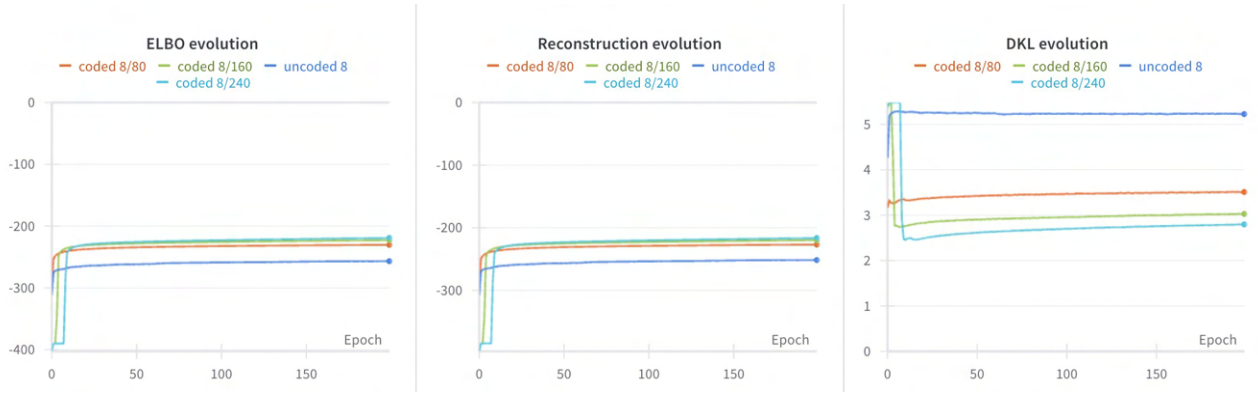


Figure 13: Evolution of the ELBO during training with 8 information bits on FMNIST.

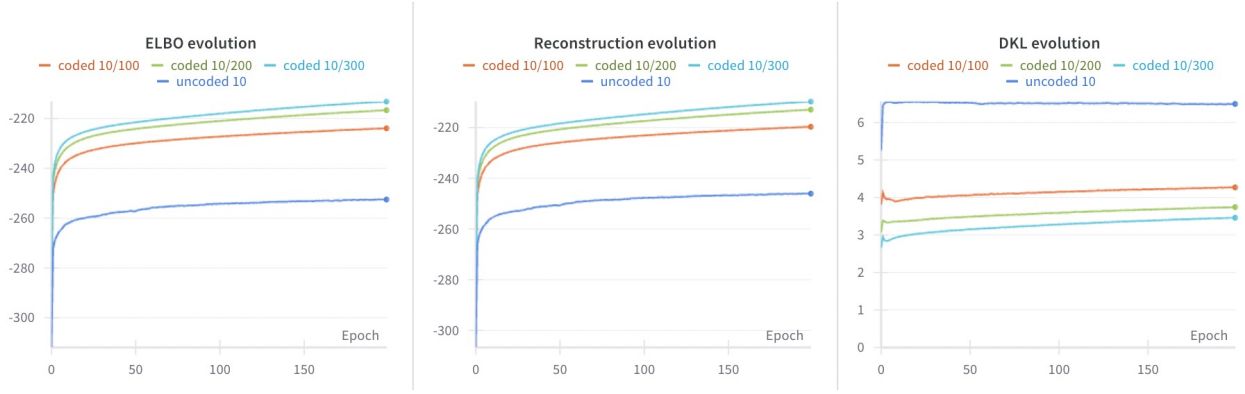


Figure 14: Evolution of the ELBO during training with 10 information bits on FMNIST.

D.2 RECONSTRUCTION AND GENERATION

In this section, we augment the results presented in the main text, including outcomes obtained with the training dataset in Table 1. We include again the results obtained in the test to facilitate comparison. Additionally, we assess reconstruction quality using the Structural Similarity Index Measure (SSIM) to provide a more comprehensive evaluation. The results remain consistent across the two data partitions, and the analysis conducted for the test set also applies to training data.

In all the cases, the coded models yield higher PSNR and SSIM values than their uncoded counterparts, indicating a superior performance in reconstruction. We observe a general improvement in PSNR and SSIM as we increase the number of information bits (i.e., as we augment the latent dimensionality of the model) and decrease the code rate (i.e., as we introduce more redundancy).

As we discussed in the main text, the reconstruction metrics such as the PSNR and SSIM do not account for the semantic errors committed by the model. Therefore, we additionally report the semantic accuracy and the *confident* semantic accuracy. While the reconstruction accuracy is computed across the entire dataset partitions, for the confident accuracy, we only consider those images projected into a latent vector with a probability exceeding 0.4. We observe that coded models better capture the semantics of the images while employing the same number of latent vectors, significantly outperforming the uncoded models in terms of accuracy in all the cases.

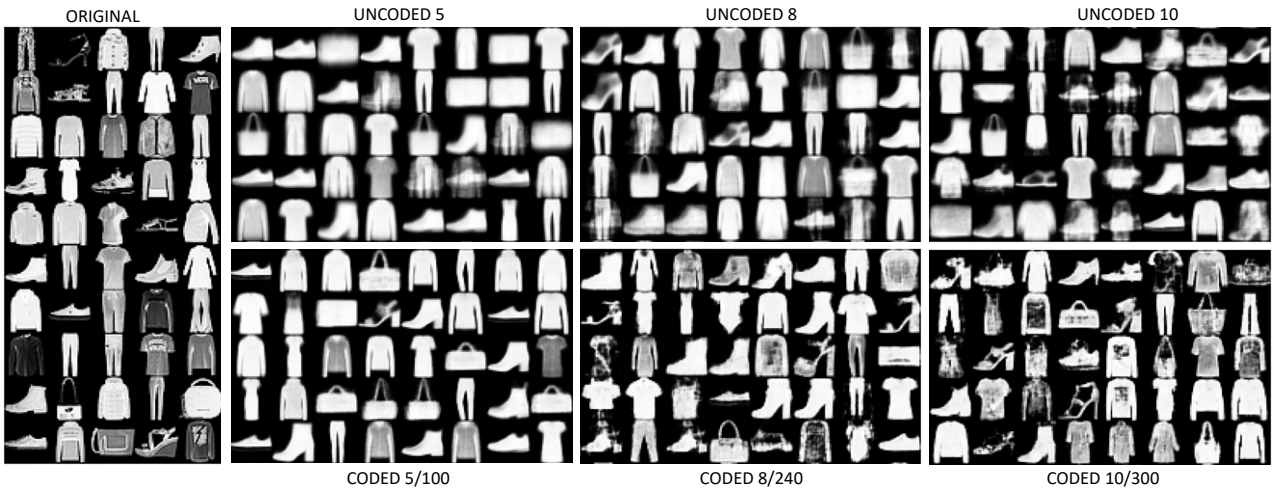


Figure 15: Example of randomly generated, uncurated images using different model configurations.

Table 1: Evaluation of reconstruction performance in FMNIST.

Model	PSNR (train)	SSIM (train)	Acc (train)	Conf. Acc. (train)	PSNR (test)	SSIM (test)	Acc (test)	Conf. Acc. (test)
uncoded 5	14.490	0.438	0.541	0.541	14.477	0.437	0.536	0.536
coded 5/50	16.375	0.559	0.656	0.702	16.241	0.551	0.647	0.700
coded 5/80	16.824	0.585	0.694	0.751	16.624	0.574	0.688	0.748
coded 5/100	17.001	0.596	0.708	0.760	16.702	0.580	0.700	0.757
uncoded 8	15.644	0.513	0.601	0.602	15.598	0.509	0.594	0.595
coded 8/80	17.877	0.647	0.769	0.842	17.318	0.619	0.750	0.816
coded 8/160	18.828	0.690	0.807	0.878	17.713	0.641	0.783	0.831
coded 8/240	19.345	0.713	0.831	0.921	17.861	0.653	0.799	**0.893**
uncoded 10	16.053	0.542	0.650	0.652	16.000	0.536	0.644	0.648
coded 10/100	18.827	0.690	0.813	0.885	17.694	0.639	0.790	0.850
coded 10/200	19.937	0.735	0.846	0.897	18.009	0.659	0.814	0.871
coded 10/300	**20.529**	**0.754**	**0.855**	**0.907**	**18.111**	**0.662**	**0.817**	0.870

In Fig. 15, we include additional examples of randomly generated images using different model configurations. We observe that coded models can generate more detailed and diverse images than their uncoded counterparts.

E MNIST RESULTS

In this section, we report the results obtained with the MNIST dataset.

E.1 TRAINING

We present the evolution of the ELBO and its terms throughout the training process. The models were trained for 100 epochs using an Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. Fig. 16 displays the results for configurations with 5 information bits, Fig. 17 for 8 information bits, and Fig. 18 for 10 information bits. The colors in all plots represent the various code rates.

The results are consistent with the ones obtained for FMNIST. Across all the configurations, coded models achieve superior bounds. The main differences in the ELBO come from the different performances in reconstruction. As we have observed

Figure 16: Evolution of the ELBO during training with 5 information bits on MNIST.

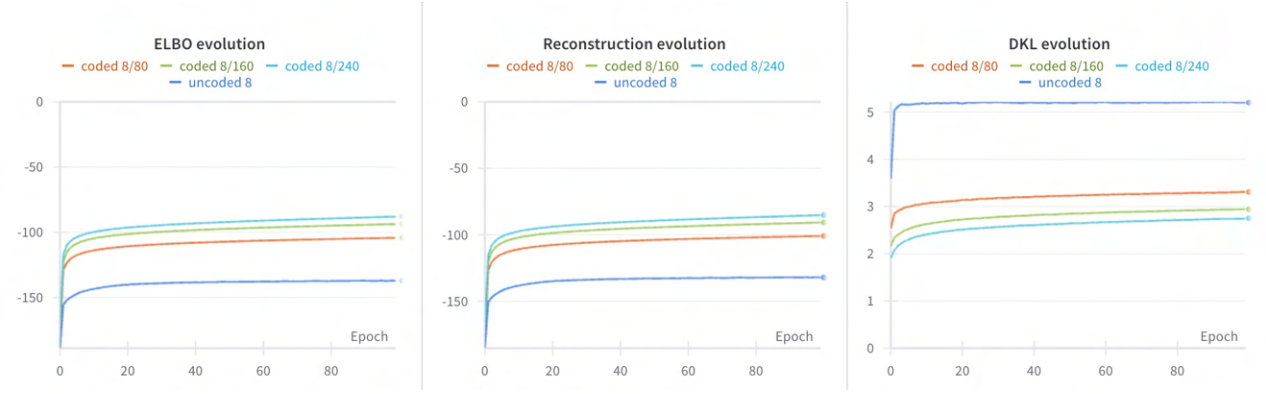


Figure 17: Evolution of the ELBO during training with 8 information bits on MNIST.

Figure 18: Evolution of the ELBO during training with 10 information bits on MNIST.

across the different experiments, coded models are capable of better capturing the structure of the data, generating more detailed images and accurate reconstructions.

We observe that, as we decrease the code rate, we obtain better bounds in general. Adding redundancy does not increase the model's flexibility, since the information bits determine the number of latent vectors. However, the introduction of ECCs in the model allows for latent spaces that better capture the structure of the images while employing the same number of latent vectors.

E.2 RECONSTRUCTION

We first evaluate the model's performance in reconstructing data by examining its *uncoded* and *coded* versions across different configurations, varying the number of information bits and code rates. All the results obtained with MNIST are consistent with those presented in the main text for FMNIST.

In Table 2 we quantify the quality of the reconstructions measuring the PSNR and the SSIM in both training and test sets. In all the cases, coded models yield higher PSNR and SSIM values, indicating a superior performance in reconstruction. This improvement is also evident through visual inspection of Fig. 19, where the coded models better capture the details in the images. Similar to the results observed on FMNIST, we find that both PSNR and SSIM generally improve as the number of

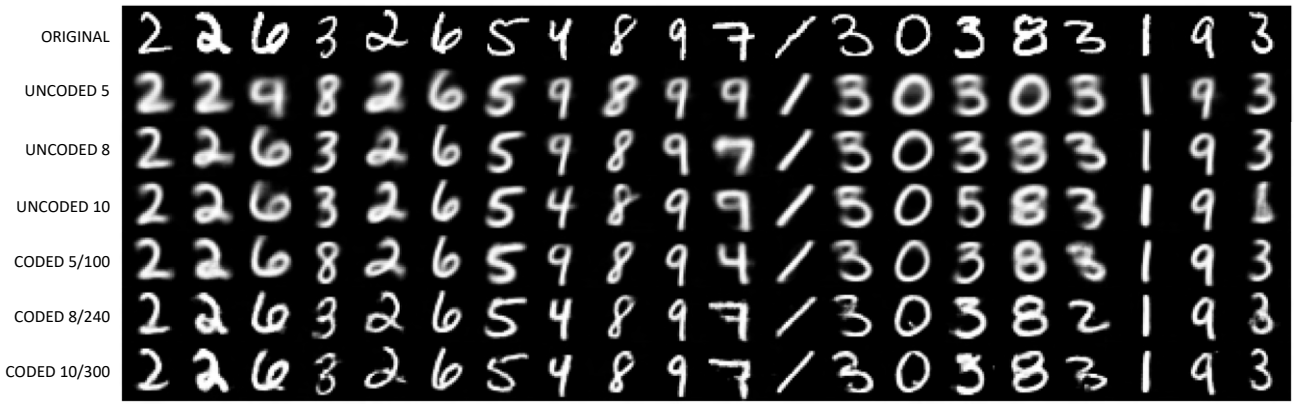


Figure 19: Example of reconstructed test images obtained with different model configurations. Observe that more details are visualized as we increase bits in the latent space and decrease the coding rate.

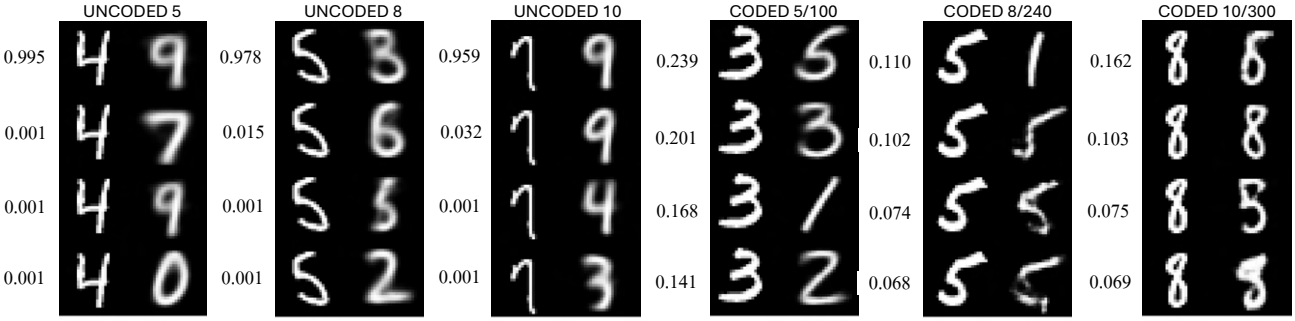


Figure 20: Example of erroneous reconstructions in MNIST using the 4 most-probable words (a posterior). The first column in each image shows the original input to the model, while the second column displays the reconstructions. The a posteriori probability of the word used for reconstruction is indicated in each row.

Table 2: Evaluation of reconstruction performance in MNIST.

Model	PSNR (train)	SSIM (train)	Acc (train)	Conf. Acc. (train)	PSNR (test)	SSIM (test)	Acc (test)	Conf. Acc. (test)	Entropy
uncoded 5	13.491	0.434	0.702	0.703	13.483	0.430	0.701	0.702	0.277
coded 5/50	14.983	0.601	0.887	0.923	14.888	0.598	0.887	0.920	2.073
coded 5/80	15.436	0.641	0.899	0.936	15.263	0.634	0.895	0.929	2.237
coded 5/100	15.590	0.652	0.905	0.931	15.352	0.641	0.898	0.924	2.382
uncoded 8	14.530	0.558	0.860	0.864	14.490	0.558	0.860	0.868	0.513
coded 8/80	16.878	0.739	0.937	0.964	16.042	0.699	0.912	0.947	3.105
coded 8/160	18.108	0.802	0.957	0.974	16.497	0.736	0.927	0.951	3.645
coded 8/240	19.984	0.838	0.967	0.978	16.688	0.752	0.936	0.957	3.881
uncoded 10	14.879	0.591	0.888	0.891	14.816	0.589	0.887	0.890	0.636
coded 10/100	17.584	0.777	0.945	0.972	16.795	0.744	0.928	**0.968**	4.080
coded 10/200	20.060	0.875	0.973	0.977	16.863	0.765	0.932	0.944	4.411
coded 10/300	**21.083**	**0.902**	**0.979**	**0.984**	**17.114**	**0.781**	**0.941**	0.945	4.810

information bits increases and the code rate decreases.

As we discussed in the main text, reconstruction metrics such as PSNR and SSIM do not account for the *semantic* errors committed by the model. Therefore, we additionally evaluate the reconstruction accuracy, ensuring that the model

successfully reconstructs images within the same class as the original ones. We also provide a *confident* reconstruction accuracy, for which we do not count errors when the MAP value of $q(\boldsymbol{m}|\boldsymbol{x})$ is below 0.4. In light of the results, we argue that introducing ECCs in the model allows for latent spaces that better capture the semantics of the images while employing the same number of latent vectors, outperforming the uncoded models in all the cases.

We also report the average entropy of the variational posterior over the test set in Table 2. If we consider the entropy together with the semantic accuracy, we can argue that coded VAE is aware that multiple vectors might be related to the same image class, and that the posterior shows larger uncertainties for images for which the model has not properly identified the class. We illustrate this argument in Fig. 20, where we show some images selected so that the MAP latent word of $q(\boldsymbol{m}|\boldsymbol{x})$ induces class reconstruction errors. We show the reconstruction of the 4 most probable latent vectors and their corresponding probabilities. Observe that the uncoded model is confident no matter the reconstruction outcome, while in the coded posterior, the uncertainty is much larger.

Table 4 shows the log-likelihood values obtained for the MNIST dataset with various model configurations. Coded models consistently outperform their uncoded counterparts for both the training and test sets, consistent with the findings observed using the FMNIST dataset.

E.3 GENERATION

In this section, we evaluate the model in the image generation task. In Fig. 21, we show examples of randomly generated images using different model configurations in MNIST. These results are consistent with the ones obtained in reconstruction, and with the ones obtained for FMNIST, where we observe that the coded models can generate more detailed and diverse images.

The improved inference provided by the repetition code can also be tested by generating images using the generative model and counting errors using the MAP solution of the variational posterior distribution, as well as sampling from the approximate posterior distribution. Table 3 reports the BER and WER for MNIST. Remarkably, for the same number of latent bits, coded models reduce both the BER and WER w.r.t. the uncoded case. Note also that the error rates grow with the number of latent bits, but this is expected due to the increased complexity of the inference process.

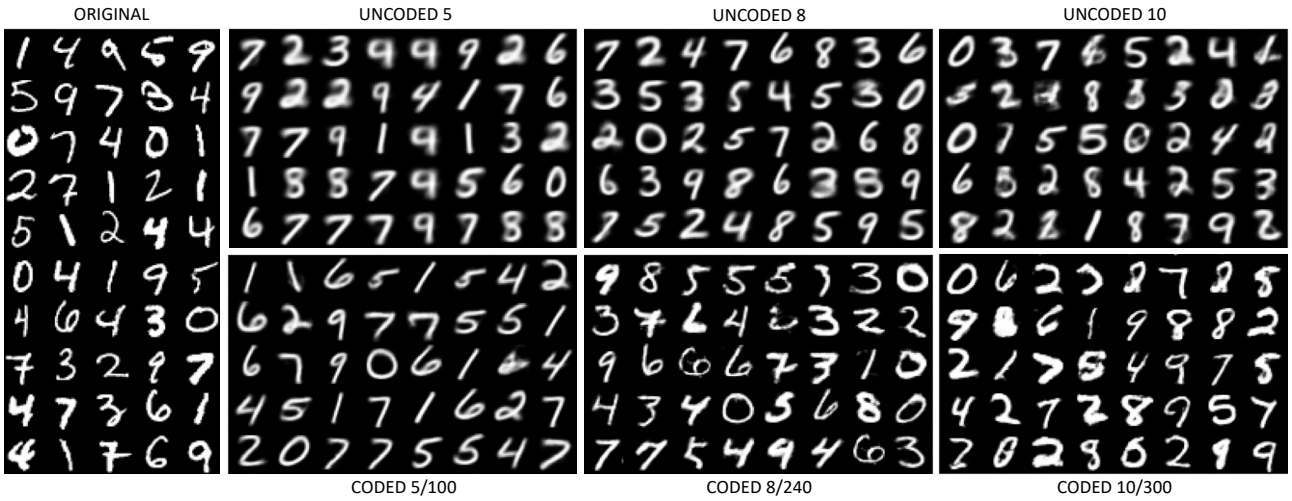


Figure 21: Example of randomly generated, uncurated images using different model configurations.

Table 3: Evaluation of the BER, WER in MNIST.

Model	BER	BER MAP	WER	WER MAP
uncoded 5	0.002	0.001	0.008	0.004
coded 5/50	0.007	0.000	0.034	0.000
coded 5/80	0.004	0.000	0.021	0.000
coded 5/100	0.009	0.000	0.045	0.000
uncoded 8	0.015	0.013	0.071	0.057
coded 8/80	0.020	$9.750{\cdot}10^{-5}$	0.147	$7.800{\cdot}10^{-4}$
coded 8/160	0.021	$2.500{\cdot}10^{-5}$	0.160	$2.000{\cdot}10^{-4}$
coded 8/240	0.023	$7.800{\cdot}10^{-4}$	0.167	0.006
uncoded 10	0.057	0.052	0.373	0.353
coded 10/100	0.030	$2.040{\cdot}10^{-4}$	0.258	0.002
coded 10/200	0.034	$3.260{\cdot}10^{-4}$	0.282	0.003
coded 10/300	0.041	$9.600{\cdot}10^{-4}$	0.331	0.009

Table 4: Evaluation of the LL in MNIST.

Model	LL (train)	LL (test)
uncoded 5	-149.049	-148.997
coded 5/50	-117.979	-119.094
coded 5/80	-114.911	-116.639
coded 5/100	-115.189	-117.200
uncoded 8	-127.079	-127.555
coded 8/80	-96.554	-104.692
coded 8/160	-96.014	-107.436
coded 8/240	-97.316	-111.312
uncoded 10	-120.594	-121.332
coded 10/100	-92.545	-99.373
coded 10/200	-86.072	-106.249
coded 10/300	-88.904	-110.799

F CIFAR10 RESULTS

In this section, we provide additional results using the CIFAR10 dataset with different model configurations.

F.1 TRAINING

We present the evolution of the ELBO and its terms throughout the training process. The models were trained for 300 epochs using Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. Fig. 22 displays the results for configurations with 70 information bits, Fig. 23 for 100 information bits, and Fig. 24 for 130 information bits. The colors in all plots represent the various code rates.

Across all the configurations, coded models achieve superior bounds. The main differences in the ELBO come from the different performances in reconstruction. As we have observed across the different experiments, coded models are capable of better capturing the structure of the data, generating more detailed images and accurate reconstructions.

In the coded case, we do not observe significant differences in the obtained bounds as we increase the number of information bits and reduce the code rate. However, the difference is notable if we compare the coded and uncoded models. Adding redundancy does not increase the model's flexibility, since the information bits determine the number of latent vectors.

Figure 22: Evolution of the ELBO during training with 70 information bits on CIFAR10.

Figure 23: Evolution of the ELBO during training with 100 information bits on CIFAR10.

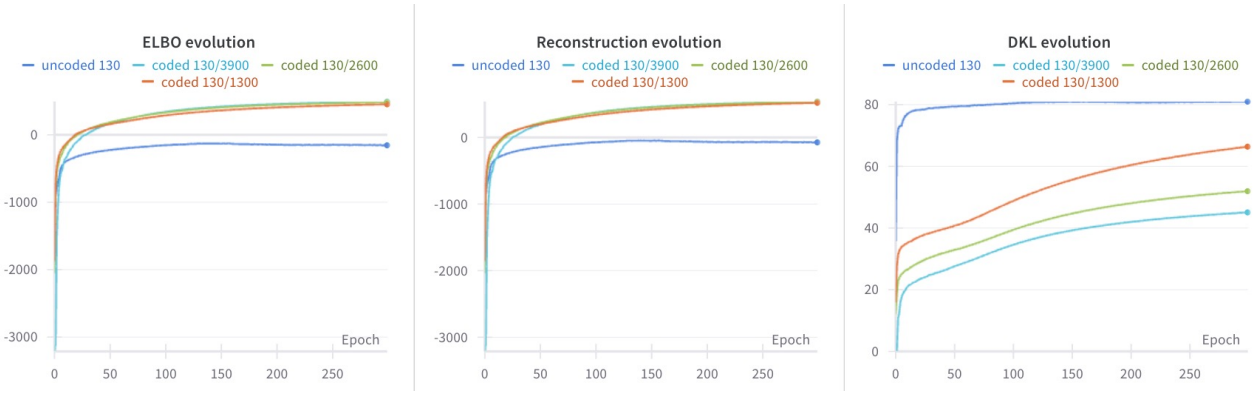


Figure 24: Evolution of the ELBO during training with 130 information bits on CIFAR10.

However, the introduction of ECCs in the model allows for latent spaces that better capture the structure of the images while employing the same number of latent vectors.

It's important to note that we are currently using feed-forward networks at the decoder's input. However, this approach may not be suitable for the correlations present in our coded words. Utilizing an architecture capable of effectively leveraging these correlations among the coded bits could potentially enable us to better exploit the introduced redundancy.

F.2 RECONSTRUCTION AND GENERATION

We first evaluate the model's performance in reconstructing data by examining its *uncoded* and *coded* versions across different configurations, varying the number of information bits and code rates.

In Table 5 we quantify the quality of the reconstructions measuring the PSNR and the SSIM in both training and test sets. Coded models yield higher PSNR and SSIM values in train, and similar values in test, although the coded models with lower rates outperform the rest of the configurations. However, the improvement in reconstruction is evident through visual inspection of Fig. 25, where the coded models better capture the details in the images. We observe that the coded model produces images that better resemble the structure of the dataset, while the uncoded DVAE cannot decouple spatial information from the images and project it in the latent space.

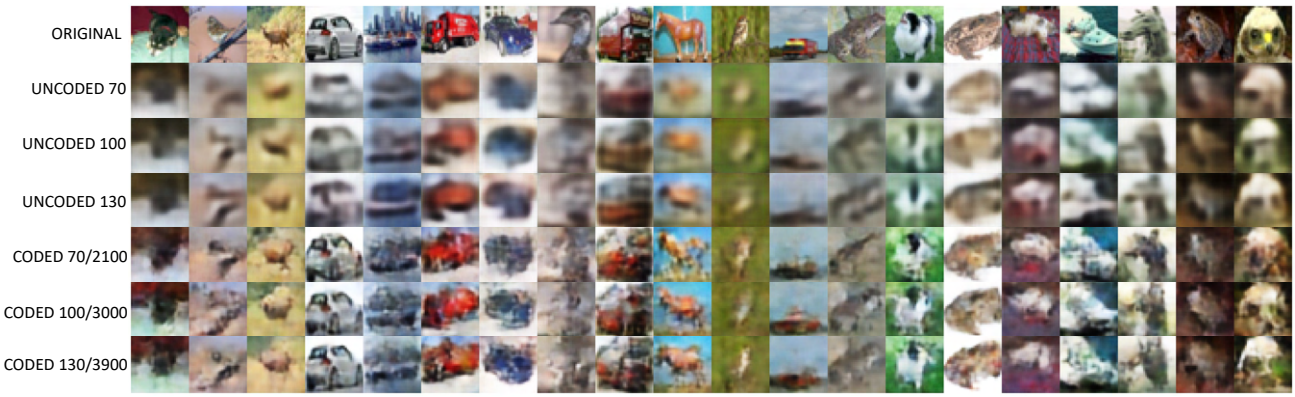


Figure 25: Example of reconstructed test images obtained with different model configurations. Observe that more details are visualized as we increase bits in the latent space and introduce redundancy.

Figure 26: Example of randomly generated, uncurated images using different model configurations.

We hypothesize that, to adequately model complex images, transitioning to a hierarchical structure may be necessary. This would allow for explicit modeling of both global and local information. However, despite employing this rather simple model, we observe that coded configurations outperform their uncoded counterparts in capturing colors and textures.

We also evaluate the model in the image generation task. In Fig. 26, we show examples of randomly generated images using different model configurations in CIFAR10. These results are consistent with the ones obtained in reconstruction, as we observe that the coded models can generate more detailed and diverse images. Additionally, we obtained the Fréchet Inception Distance (FID) score using the test set and 10k generated samples. For this, we used the implementation available at `https://github.com/mseitzer/pytorch-fid`. We can observe that the coded models significantly reduced the FID score in all the cases compared to their uncoded counterparts. For the coded models, we do not observe a clear influence of the code rate on the quality of the generations.

Table 5: Reconstruction metrics in CIFAR10 with different model configurations.

Model	PSNR (train)	SSIM (train)	PSNR (test)	SSIM (test)
uncoded 70	17.985	0.340	17.596	0.324
coded 70/700	23.790	0.709	17.731	0.323
coded 70/1400	24.555	0.748	18.008	0.357
coded 70/2100	25.551	0.748	18.401	0.385
uncoded 100	18.509	0.381	18.334	0.370
coded 100/1000	24.754	0.754	18.229	0.375
coded 100/2000	24.866	0.761	18.927	0.432
coded 100/3000	25.646	0.793	18.920	0.426
uncoded 130	18.951	0.419	18.758	.408
coded 130/1300	25.007	0.767	18.887	0.426
coded 130/2600	25.460	0.784	**19.416**	**0.464**
coded 130/3900	**25.515**	**0.785**	19.292	0.456

Table 6: Evaluation of the BER, WER, and FID in CIFAR10 with different model configurations.

Model	BER	BER MAP	WER	WER MAP	FID
uncoded 70	0.162	0.162	1.000	0.999	177.524
coded 70/700	0.101	0.081	1.000	0.988	104.977
coded 70/1400	0.088	0.044	0.999	0.917	104.078
coded 70/2100	0.090	0.029	0.999	0.811	102.795
uncoded 100	0.182	0.179	1.000	1.000	172.063
coded 100/1000	0.123	0.104	1.000	0.999	107.887
coded 100/2000	0.114	0.060	1.000	0.990	101.182
coded 100/3000	0.138	0.077	1.000	0.997	107.287
uncoded 130	0.197	0.194	1.000	1.000	164.138
coded 130/1300	0.144	0.115	1.000	0.999	109.905
coded 130/2600	0.164	0.102	1.000	1.000	110.250
coded 130/3900	0.185	0.132	1.000	1.000	108.561

G TINY IMAGENET RESULTS

In this section, we provide additional results using the Tiny ImageNet dataset with different model configurations.

G.1 TRAINING

We present the evolution of the ELBO and its terms throughout the training process. The models were trained for 300 epochs using an Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. Fig. 27 displays the results for configurations with 70 information bits, Fig. 28 for 100 information bits, and Fig. 29 for 130 information bits. The colors in all plots represent the various code rates.

As in the rest of the datasets, coded models achieve superior bounds across all the configurations. The main differences in the ELBO come from the different performances in reconstruction. As we have observed across the different experiments, coded models are capable of better capturing the structure of the data, generating more detailed images and accurate reconstructions.

Figure 27: Evolution of the ELBO during training for the configurations with 70 information bits.

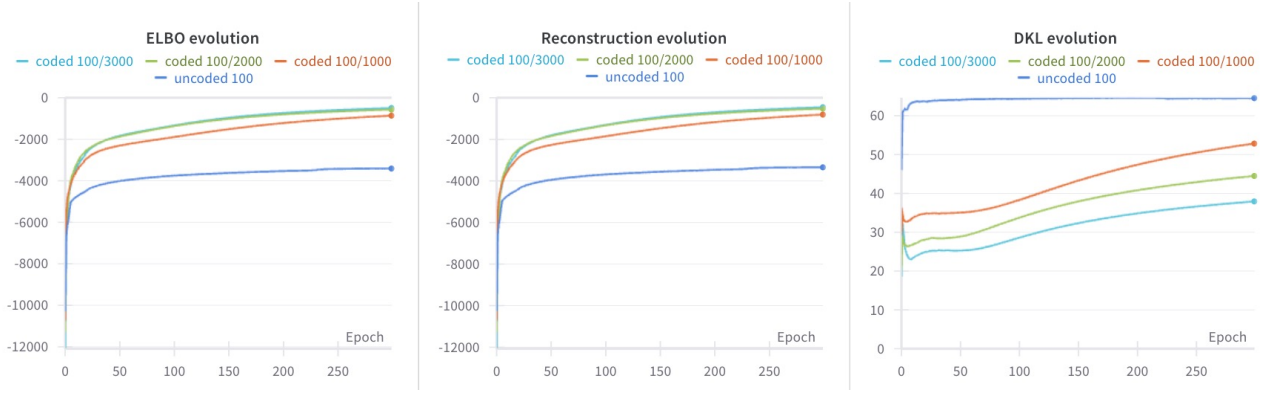


Figure 28: Evolution of the ELBO during training for the configurations with 100 information bits.

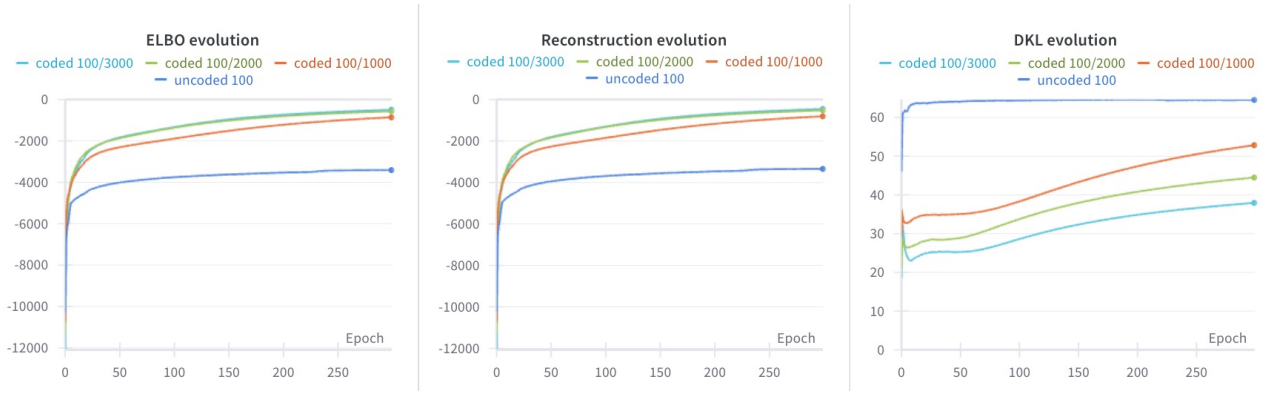


Figure 29: Evolution of the ELBO during training for the configurations with 130 information bits.

In the coded case, we do not observe significant differences in the obtained bounds as we increase the number of information bits and reduce the code rate. However, the difference is notable if we compare the coded and uncoded models. Adding redundancy does not increase the model's flexibility, since the information bits determine the number of latent vectors. However, the introduction of ECCs in the model allows for latent spaces that better capture the structure of the images while employing the same number of latent vectors.

It's important to note that we are currently using feed-forward networks at the decoder's input. However, this approach may not be suitable for the correlations present in our coded words. Utilizing an architecture capable of effectively leveraging these correlations among the coded bits could potentially enable us to better exploit the introduced redundancy.

G.2 RECONSTRUCTION AND GENERATION

We first evaluate the model's performance in reconstructing data by examining its *uncoded* and *coded* versions across different configurations, varying the number of information bits and code rates.

In Table 7, we quantify the quality of the reconstructions measuring the PSNR and the SSIM in both training and test sets. Coded models yield higher PSNR and SSIM values in train, and similar values in test, although coded models with lower rates outperform the rest of the configurations. However, the improvement in reconstruction is evident through

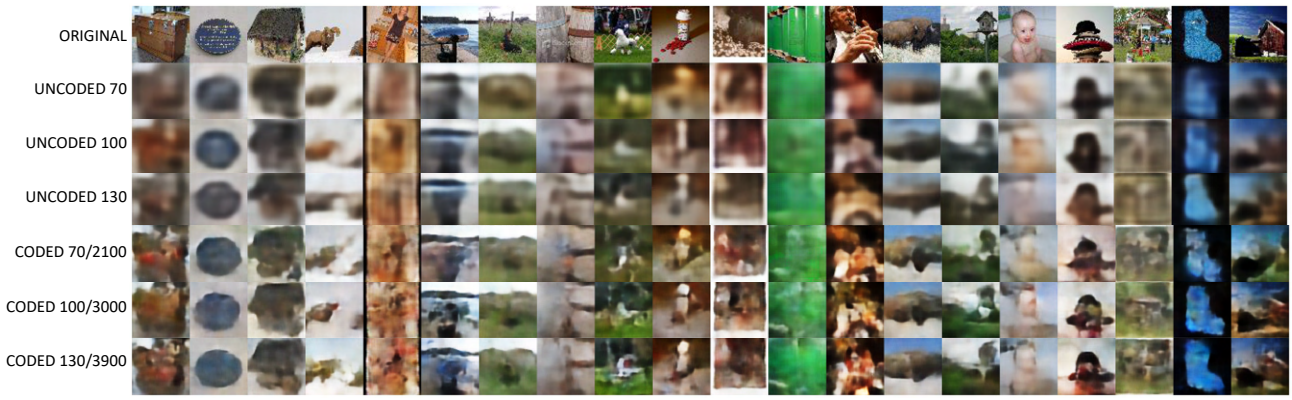


Figure 30: Example of reconstructed test images obtained with different model configurations. Observe that more details are visualized as we increase the bits in the latent space and introduce redundancy.

Figure 31: Example of randomly generated, uncurated images using different model configurations.

visual inspection of Fig. 30, where the coded models better capture the details in the images. We observe that the coded model produces images that better resemble the structure of the dataset, while the uncoded DVAE cannot decouple spatial information from the images and project it in the latent space.

We hypothesize that, to adequately model complex images, transitioning to a hierarchical structure may be necessary. This would allow for the explicit modeling of both global and local information. However, despite employing this rather simple model, we observe that coded configurations outperform their uncoded counterparts in capturing colors and textures.

We also evaluate the model in the image generation task. In Fig. 31, we show examples of randomly generated images using different model configurations in Tiny ImageNet. These results are consistent with the ones obtained in reconstruction, where we observe that the coded models can generate more detailed and diverse images. Additionally, we obtained the FID score using the test set and 10k generated samples. For this, we used the implementation available at `https://github.com/mseitzer/pytorch-fid`. We can observe that the coded models significantly reduced the FID score in all the cases compared to their uncoded counterparts.

For the coded models, we do not observe a clear influence of the code rate on the quality of the generations in CIFAR-10. However, in Tiny ImageNet, smaller code rates produce worse FID scores. We hypothesize that this may be due to the presence of artifacts in the generated images. Our experiments indicate that coded models with lower rates attempt to model fine details in images, which can lead to artifacts in generation.

Table 7: Reconstruction metrics in Tiny ImageNet with different model configurations.

Model	PSNR (train)	SSIM (train)	PSNR (test)	SSIM (test)
uncoded 70	15.598	0.207	15.402	0.196
coded 70/700	18.156	0.367	15.158	0.191
coded 70/1400	18.396	0.383	15.419	0.203
coded 70/2100	18.228	0.369	15.789	0.223
uncoded 100	16.012	0.229	15.774	0.215
coded 100/1000	18.298	0.378	15.677	0.218
coded 100/2000	18.647	0.404	15.892	0.232
coded 100/3000	18.729	0.408	16.167	0.248
uncoded 130	16.278	0.243	16.009	0.228
coded 130/1300	18.719	0.409	15.900	0.233
coded 130/2600	18.818	**0.433**	**16.329**	**0.259**
coded 130/3900	**19.020**	0.415	16.288	**0.259**

Table 8: Evaluation of the BER, WER, and FID in Tiny ImageNet with different model configurations.

Model	BER	BER MAP	WER	WER MAP	FID
uncoded 70	0.143	0.140	1.000	0.999	265.474
coded 70/700	0.096	0.072	0.998	0.978	171.993
coded 70/1400	0.104	0.066	1.000	0.972	170.496
coded 70/2100	0.096	0.034	0.998	0.832	176.245
uncoded 100	0.164	0.162	1.000	1.000	234.358
coded 100/1000	0.099	0.074	1.000	0.996	153.743
coded 100/2000	0.097	0.053	1.000	0.981	162.889
coded 100/3000	0.098	0.035	1.000	0.925	163.049
uncoded 130	0.200	0.198	1.000	1.000	219.003
coded 130/1300	0.129	0.107	1.000	0.999	165.064
coded 130/2600	0.114	0.060	1.000	0.996	164.759
coded 130/3900	0.128	0.070	1.000	0.999	170.603

H IWAE RESULTS

One could draw a parallel between the Coded-DVAE with repetition codes and the well-known IWAE [Burda et al., 2016], but the two approaches are fundamentally different. In the IWAE, independent samples are drawn from the variational posterior and propagated independently through the generative model to obtain a tighter variational bound on the marginal log-likelihood. In our method, we jointly propagate the output of the ECC encoder through the generative model, obtaining a single prediction and exploiting the introduced known correlations in the variational approximation of the posterior. In the case of repetition codes, the ECC encoder outputs are repeated bits, or repeated probabilities in the case of soft encoding. However, our approach extends beyond repetition codes, opening a new field for improved inference in discrete latent variable models.

In our approach, we specifically utilize the redundancy introduced by the repetition code to correct potential errors made by the encoder through a soft decoding approach. This results in a more accurate approximation of the posterior $p(\boldsymbol{m}|\boldsymbol{x})$ and an improved proposal for sampling, resulting in improved performance over uncoded models, even those trained with the tighter IWAE bound. To compute the IWAE objective in the uncoded case, we draw samples from the posterior using the reparameterization trick described in Eq. (5), and compute the importance weights as $\boldsymbol{w} = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\eta}}(\boldsymbol{z}|\boldsymbol{x})}$. Here, the prior and posterior over $\boldsymbol{z}$ are obtained by marginalizing out $\boldsymbol{m}$ in $p(\boldsymbol{z}, \boldsymbol{m}) = p(\boldsymbol{m})p(\boldsymbol{z}|\boldsymbol{m})$ and $q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{m})$, respectively.

Additionally, we implemented two extensions of the IWAE bound to confirm that the observed improvements are not due to a known shortcoming of the reparametrized gradient estimator of the IWAE bound. Following Daudel et al. [2023], we implemented the VR-IWAE generalization, which introduces a hyperparameter $\alpha \in [0, 1)$ into the objective function. This formulation interpolates between the IWAE bound (recovered when $\alpha = 0$) and the ELBO (recovered when $\alpha = 1$). For our experiments, we selected a midpoint value of $\alpha = 0.5$. Additionally, we implemented the doubly-reparametrized (DR) gradient estimator associated with the VR-IWAE objective, as described in Daudel et al. [2023]. We trained the models on FMNIST using 10, 20, and 30 posterior samples, matching the number of repetitions used in our coded models. We included an additional experiment training an uncoded model with the IWAE bound considering 100 samples.

From the results, outlined in Table 9, we observe that training uncoded models with tighter bounds (even if we use bounds that are meant to become tighter as the number of samples increases) does not lead to substantial improvements in performance. These models consistently underperform relative to their coded counterparts, which are trained using the ELBO as the objective function. For a fair comparison, we use the same neural network architecture for the uncoded and the coded posterior (differing only in the final dense layer to match the respective output dimensionalities), the same network architecture for the decoder (differing only in the first dense layer to match the respective input dimensionalities), and the same simple independent prior.

Fig. 32 shows the evolution of the different objectives, with coded models obtaining better bounds than the uncoded IWAE

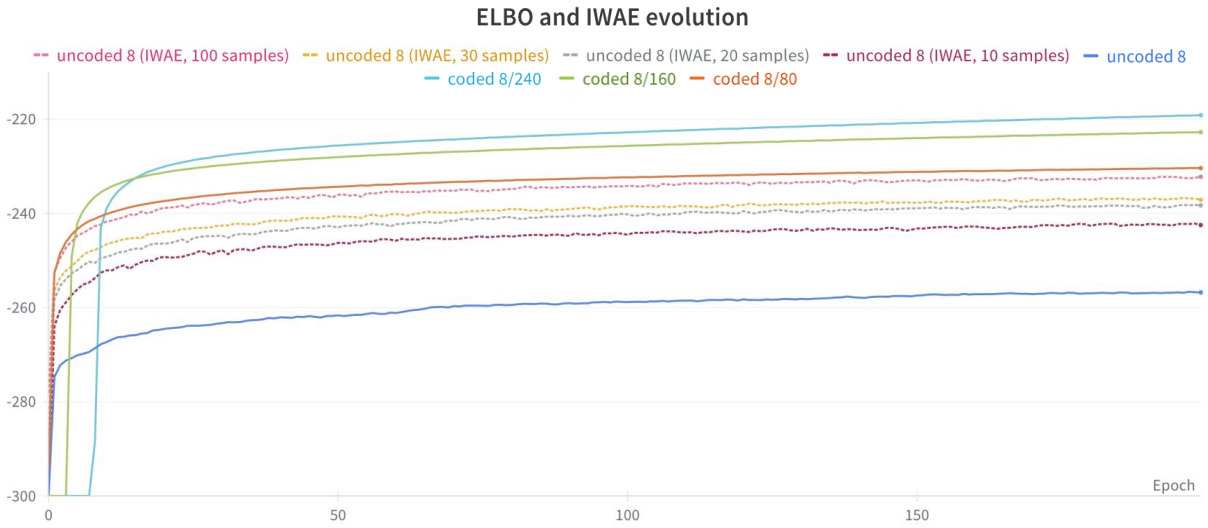


Figure 32: Evolution of the ELBO and the IWAE objectives for various configurations. Observe that the IWAE provides a tighter bound than the ELBO in the uncoded setting. However, coded models obtain even better bounds using the same number of samples/repetitions.

Table 9: Comparison of test metrics between our method and the uncoded DVAE trained with the IWAE objective.

Model	BER	WER	Entropy	Acc.	Conf. Acc.	PSNR	SSIM
uncoded 8	0.089	0.384	0.467	0.594	0.595	15.598	0.509
uncoded 8 IWAE 10 samples	0.063	0.372	1.309	0.617	0.640	14.282	0.470
uncoded 8 IWAE 20 samples	0.075	0.447	1.391	0.634	0.651	14.237	0.460
uncoded 8 IWAE 30 samples	0.074	0.438	1.564	0.619	0.641	13.757	0.457
uncoded 8 IWAE 100 samples	0.107	0.583	2.274	0.596	0.663	12.996	0.433
uncoded 8 IWAE ($\alpha = 0.5$, 10 samples)	0.059	0.353	0.929	0.630	0.638	14.582	0.488
uncoded 8 IWAE ($\alpha = 0.5$, 20 samples)	0.067	0.388	1.066	0.638	0.644	14.432	0.479
uncoded 8 IWAE ($\alpha = 0.5$, 30 samples)	0.053	0.349	1.165	0.630	0.644	14.178	0.472
uncoded 8 IWAE (DR, $\alpha = 0.5$, 10 samples)	0.052	0.312	1.004	0.623	0.629	14.542	0.483
uncoded 8 IWAE (DR, $\alpha = 0.5$, 20 samples)	0.056	0.353	1.041	0.600	0.614	14.231	0.463
uncoded 8 IWAE (DR, $\alpha = 0.5$, 30 samples)	0.057	0.352	1.068	0.602	0.607	14.016	0.449
coded 8/80	0.021	0.144	2.905	0.750	0.816	17.318	0.619
coded 8/160	0.027	0.189	3.637	0.783	0.831	17.713	0.641
coded 8/240	0.037	0.231	4.000	0.799	0.893	17.861	0.653

models, even when considering 100 samples. In Table 9, we present test metrics, with coded models again overperforming their uncoded counterparts trained with the IWAE objective. We observe that some metrics slightly decline augmenting the number of samples, likely due to overfitting, since we applied a common early stopping point.

I BEYOND REPETITION CODES

We have presented compelling proof-of-concept results that incorporating ECCs, like repetition codes, into DVAEs can improve performance. We believe this opens a new path for designing latent probabilistic models with discrete latent variables. Although a detailed analysis of the joint design of ECC and encoder-decoder networks is beyond the scope of this work, we will outline key properties that any ECCs must satisfy to be integrated within this framework.

- **Scalable hard encoding $(\boldsymbol{m} \to \boldsymbol{c})$.** Our model requires hard encoding for generation once the model is trained. This process should have linear complexity in M.

- **Scalable soft encoding** $\big(p(\boldsymbol{m}) \rightarrow p(\boldsymbol{c})\big)$**.** Soft encoding is required during training for reparameterization. This process should also have linear complexity in M.
- **Scalable soft decoding** $\big(p(\boldsymbol{c}) \rightarrow p(\boldsymbol{m})\big)$**.** Our model employs soft-in soft-out (SISO) decoding during inference. This process should again be linearly complex in M.
- **Differentiability.** Both encoding and decoding processes must be differentiable w.r.t. the inputs to enable gradient computation and backpropagation.

Since Shannon's landmark work [Shannon, 1948], researchers have been developing feasible ECC schemes that meet (or approach) the aforementioned properties for ever-increasing values of M and R. This is driven by the high throughput demands in digital communications and limited power constraints in the case of wireless communications. In this regard, state-of-the-art ECC schemes such as Low Density Parity Check (LDPC) codes [Gallager, 1962] can be designed to meet the above criteria, leveraging their linear algebraic structure. Similarly, encoding and decoding algorithms for polar codes [Arikan, 2009] have complexity of $\mathcal{O}(D \log D)$, approaching this target. Additionally, in digital communications, soft decision decoding is known to provide savings of up to 2-3 dB of signal energy over hard decision decoding (binary input). For this reason, significant effort has been also made in the communications field to develop efficient and powerful SISO decoders, such as the sum-product algorithm (SPA) [Kschischang et al., 2001] for LDPC codes.

J HIERARCHICAL CODED-DVAE RESULTS

Inspired by polar codes [Arikan, 2009], we present a hierarchical Coded-DVAE with two layers of latent bits, as illustrated in Fig. 33. In this model, the latent bits $\boldsymbol{m}_1$ are encoded using a repetition code in the first layer, producing $\boldsymbol{c}_1$ and $\boldsymbol{z}_1$. Concurrently, the bits in the second layer, $\boldsymbol{m}_2$, are linearly combined with $\boldsymbol{m}_1$ following $\boldsymbol{m}_{1,2} = \boldsymbol{m}_1 \oplus \boldsymbol{m}_2$, considering a binary field or Galois field. The resulting vector is then encoded with another repetition code to produce $\boldsymbol{c}_2$, which is subsequently modulated into $\boldsymbol{z}_2$. Finally, both $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are concatenated and passed through the decoder network to generate $\boldsymbol{x}$. The model provides stronger protection for $\boldsymbol{m}_1$, as it appears in both branches of the generative model. Inference follows a similar approach to the one employed in the Coded-DVAE, incorporating the linear combination of $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ used in the second branch.

We adopt the same variational family as used in the standard Coded-VAE; however, in this case, we incorporate both hierarchical levels, leading to

$$q_{\boldsymbol{\eta}}(\boldsymbol{m}, \boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x}) q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x}) p(\boldsymbol{z}_1|\boldsymbol{c}_1) p(\boldsymbol{z}_2|\boldsymbol{c}_2), \tag{13}$$

where $q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x})$ is calculated following the same approach as in the Coded-DVAE with repetition codes, computing the all-are-zero and all-are-ones products of probabilities of the bits in $\boldsymbol{c}_1$ that are copies of the same message bit. The posterior

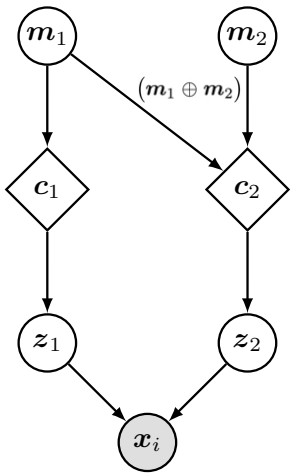


Figure 33: Graphical model of the hierarchical Coded VAE with two layers.

$q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x})$ considering both the encoder's output and the inferred posterior distribution $q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x})$. Note that in this case, the decoder outputs the probabilities for both $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, with $\boldsymbol{c}_2$ being the encoded version of the linear combination $\boldsymbol{m}_{1,2} = \boldsymbol{m}_1 \oplus \boldsymbol{m}_2$. Consequently, we first obtain $q_{\boldsymbol{\eta}}(\boldsymbol{m}_{1,2}|\boldsymbol{x})$ following the same approach as in the Coded-DVAE with repetition codes, and determine $q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x})$ as

$$q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x}) = \prod_{j=1}^{M} \text{Ber}(p_j), \tag{14}$$

$$p_j = q_{\boldsymbol{\eta}}(m_{1,2,j} = 1|\boldsymbol{x})q_{\boldsymbol{\eta}}(m_{1,j} = 0|\boldsymbol{x}) + q_{\boldsymbol{\eta}}(m_{1,2,j} = 0|\boldsymbol{x})q_{\boldsymbol{\eta}}(m_{1,j} = 1|\boldsymbol{x}). \tag{15}$$

After obtaining $q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x})$ and $q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x})$, we recalculate the posterior bit probabilities for the linear combination $q'_{\boldsymbol{\eta}}(\boldsymbol{m}_{1,2}|\boldsymbol{x})$ as

$$q'_{\boldsymbol{\eta}}(\boldsymbol{m}_{1,2}|\boldsymbol{x}) = \prod_{j=1}^{M} \text{Ber}(q_j), \tag{16}$$

$$q_j = q_{\boldsymbol{\eta}}(m_{1,j} = 1|\boldsymbol{x})q_{\boldsymbol{\eta}}(m_{2,j} = 0|\boldsymbol{x}) + q_{\boldsymbol{\eta}}(m_{1,j} = 0|\boldsymbol{x})q_{\boldsymbol{\eta}}(m_{2,j} = 1|\boldsymbol{x}). \tag{17}$$

Next, we apply the soft encoding approach to incorporate the repetition code structure at both levels of the hierarchy. The posterior probabilities $q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x})$ are repeated to obtain $q_{\boldsymbol{\eta}}(\boldsymbol{c}_1|\boldsymbol{x})$, and the posterior probabilities $q_{\boldsymbol{\eta}}(\boldsymbol{m}_{1,2}|\boldsymbol{x})$ are repeated to produce $q_{\boldsymbol{\eta}}(\boldsymbol{c}_2|\boldsymbol{x})$. Utilizing the reparameterization trick from Eq. (5), we sample $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, concatenate them to form $\boldsymbol{z}$, and pass this through the decoder to generate $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. The model is trained by maximizing the ELBO, given by

$$\text{ELBO} = \mathbb{E}_{q_{\boldsymbol{\eta}}(\boldsymbol{m},\boldsymbol{z}|\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \mathcal{D}_{KL}\big(q_{\boldsymbol{\eta}}(\boldsymbol{m}_1|\boldsymbol{x})||p(\boldsymbol{m}_1)\big) - \mathcal{D}_{KL}\big(q_{\boldsymbol{\eta}}(\boldsymbol{m}_2|\boldsymbol{x})||p(\boldsymbol{m}_2)\big), \tag{18}$$

where both $p(\boldsymbol{m}_1)$ and $p(\boldsymbol{m}_2)$ are assumed to be independent Bernoulli distributions with bit probabilities of 0.5, consistent with the other scenarios.

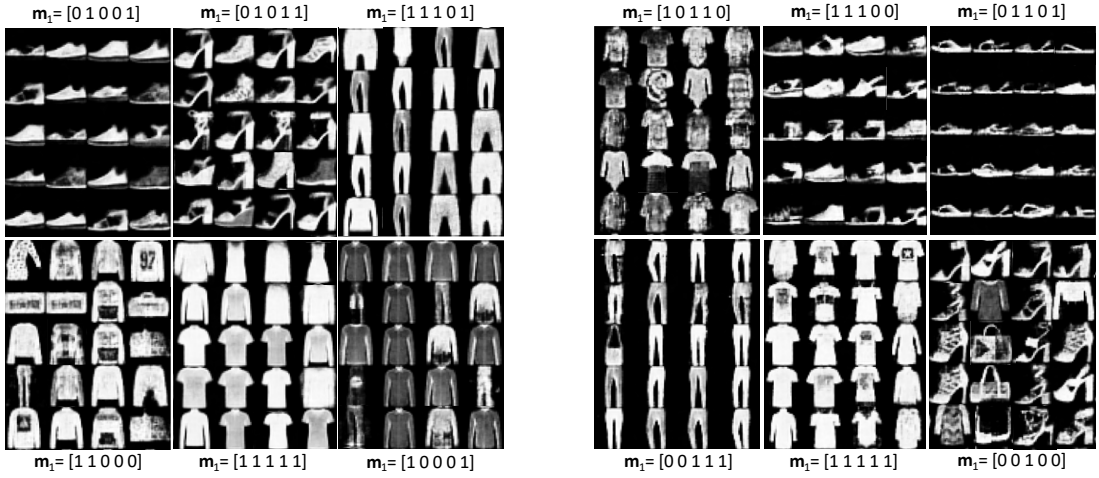
We obtained results on the FMNIST dataset using a model with 5 information bits per branch and repetition rates of $R = 1/10$ and $R = 1/20$. In this case, we applied the same code rate to both branches, although varying code rates could be used to control the level of protection at each hierarchy level. Tables 10 and 11 present the metrics obtained for the different configurations. Specifically, Table 10 shows the overall metrics obtained with this structure, and Table 11 compares the error metrics across the two hierarchy levels. As expected, $\mathbf{m}_2$ shows poorer error metrics compared to $\mathbf{m}_1$, since the model provides more redundancy to $\mathbf{m}_1$ incorporating it in both branches. Although the overall metrics and generation quality are similar to those of the Coded-DVAE with 10 information bits (see tables in Figs. 5 and 4), the introduced hierarchy results in a more interpretable latent space. In this setup, $\mathbf{m}_1$ captures global features (such as clothing types in the FMNIST dataset), while $\mathbf{m}_2$ controls individual features, as we can observe in Fig. 34, where we show examples of the model's generative outputs for fixed $\mathbf{m}_1$ and random samples of $\mathbf{m}_2$.

Table 10: Comparison of the obtained metrics for the Coded-DVAE with polar codes with different configurations, which we refer to as 'hierarchical Coded-DVAE'.

Model	**BER**	**WER**	**Acc**	**Conf. Acc**	**PSNR**
hier. 5/50	0.099	0.400	0.753	0.800	17.130
hier. 5/100	0.050	0.330	0.784	0.870	17.513

Table 11: Comparison of the obtained error metrics in the different hierarchy levels.

Model	**BER** m_1	**WER** m_1	**BER** m_2	**WER** m_2
hier. 5/50	0.079	0.259	0.119	0.362
hier. 5/100	0.026	0.110	0.075	0.287



(a) (b)

Figure 34: Examples of generated images using the hierarchical Coded-DVAE with (a) a 5/50 repetition code in each branch, and (b) a 5/100 repetition code in each branch. In all the examples provided, $\boldsymbol{m}_1$ was fixed while $\boldsymbol{m}_2$ was randomly sampled.

K ABLATION STUDY

In this section, we conduct ablation studies on the hyperparameter β of the model, responsible for regulating the decay of exponentials in the smoothing transformation, as well as on the number of trainable parameters in the models.

K.1 ABLATION STUDY ON THE HYPERPARAMETER β

Across all experiments, we have consistently configured the hyperparameter β, which controls the decay of exponentials in the smoothing transformation, to a value of 15. To illustrate its impact on the overall performance of the model, we conducted an ablation study on the value of this hyperparameter for both uncoded and coded cases.

The smoothing distribution employed for the reparameterization trick consists of two overlapping exponentials. The hyperparameter β functions as a temperature term, regulating the decay of the distributions and, consequently, influencing the degree of overlapping. A lower β value results in more overlapped tails, while a higher value leads to less overlapped distributions. A priori, we would like these distributions to be separated, allowing us to retrieve the true value of the bit and effectively use the latent structure of the model.

K.1.1 Coded model

We first evaluate the influence of the parameter β in coded models. We take as a reference the coded model with 8 information bits and a rate $R = 1/30$, and train it using $\beta = 5, 10, 15, 20$. We assess the performance of the model in reconstruction and

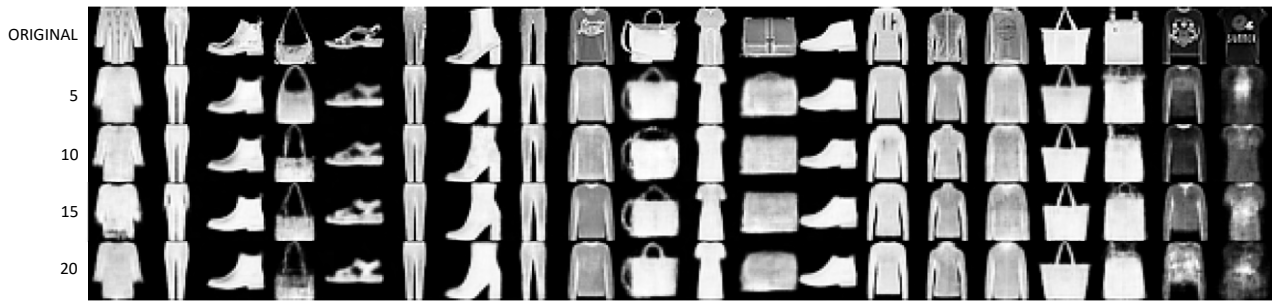


Figure 35: Example of reconstructed images obtained with different values of β using the coded model with an 8/240 code.

Table 12: Evaluation of reconstruction performance in FMNIST with different values of β using the coded model with an 8/240 code.

Model	PSNR (train)	Acc (train)	Conf. Acc. (train)	PSNR (test)	Acc (test)	Conf. Acc. (test)	Entropy
5	18.344	0.791	0.895	17.614	0.766	0.849	4.025
10	19.106	0.822	0.904	17.737	0.793	0.872	4.023
15	19.345	0.831	0.921	17.861	0.799	0.893	4.000
20	18.797	0.809	0.887	17.837	0.787	0.877	3.810

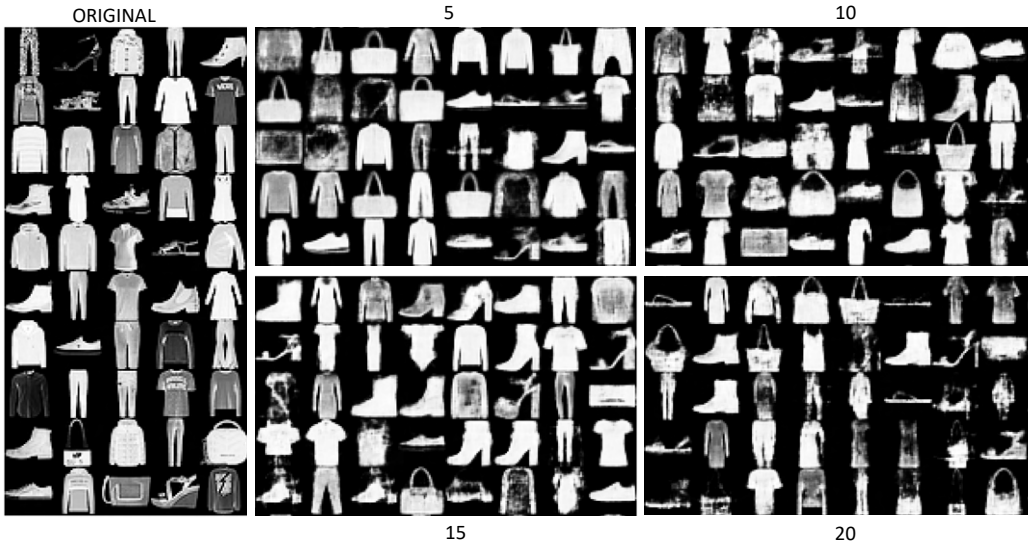


Figure 36: Example of randomly generated, uncurated images with different values of β using the coded model with an 8/240 code.

Table 13: Evaluation of the BER and WER in FMNIST with different values of β using the coded model with an 8/240 code.

Beta	BER	WER
5	0.150	0.726
10	0.080	0.480
15	0.037	0.231
20	0.065	0.399

Table 14: Evaluation of the log-likelihood (LL) in FMNIST with different values of β using the coded model with an 8/240 code.

Beta	LL (train)	LL (test)
5	-228.448	-234.629
10	-229.379	-237.495
15	-231.679	-238.459
20	-229.627	-235.927

generation tasks. We observe the model is fairly robust, achieving similar performance across configurations in most metrics.

In Fig. 35 we show examples of reconstructed images using the different configurations to assess reconstruction through visual examination, and Table 12 contains the associated reconstruction metrics. All the configurations achieve similar performances, although the models trained with $\beta = 10$ and $\beta = 15$ seem to be the best configurations for this scenario. Larger values may result in unstable training and inferior performance.

Next, we evaluate the model in the image generation task. Fig. 36 contains examples of randomly generated images using the different configurations. Table 13 reports the obtained BER and WER, and Table 14 the estimated log-likelihood of the different values of β. The model trained with $\beta = 15$ stands out in terms of error metrics, although achieves similar log-likelihood values as the model trained with $\beta = 10$. Again, these two configurations appear to be the most suitable in this scenario.

K.1.2 Uncoded model

We first evaluate the influence of the parameter β in uncoded models. We take as a reference the coded model with 8 information bits and train it using $\beta = 5, 10, 15, 20$. We assess the performance of the model in reconstruction and generation tasks. We observe that the uncoded model is also robust, achieving similar performance across configurations.

In Fig. 37 we show examples of reconstructed images using the different configurations to assess reconstruction through visual examination, and Table 15 contains the associated reconstruction metrics. All the configurations achieve similar performances, although the models trained with $\beta = 10$ and $\beta = 15$ seem to be the best configurations for this scenario. Larger values may result in unstable training and inferior performance, as we can clearly observe in this case.

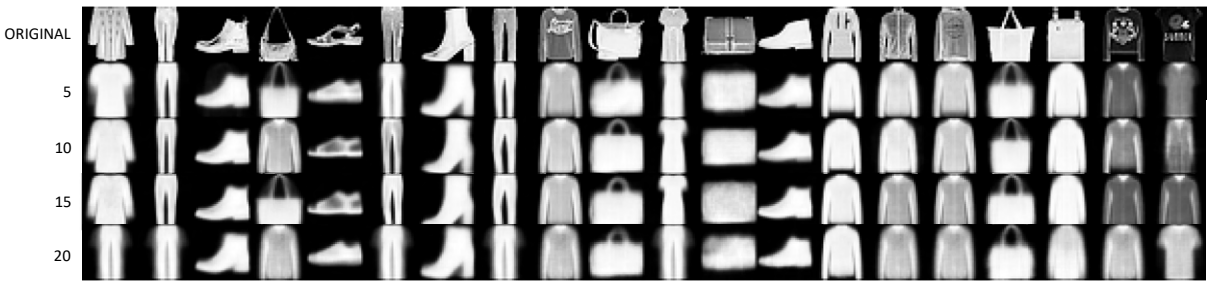


Figure 37: Example of reconstructed images obtained with different values of β using an uncoded model 8 information bits.

Table 15: Evaluation of reconstruction performance in FMNIST with different values of β using an uncoded model 8 information bits.

Model	PSNR (train)	Acc (train)	Conf. Acc. (train)	PSNR (test)	Acc (test)	Conf. Acc. (test)	Entropy
5	14.239	0.503	0.501	14.237	0.503	0.491	0.231
10	15.624	0.606	0.603	15.571	0.598	0.598	0.357
15	15.644	0.601	0.602	15.598	0.594	0.595	0.467
20	13.717	0.464	0.466	13.743	0.460	0.462	0.383

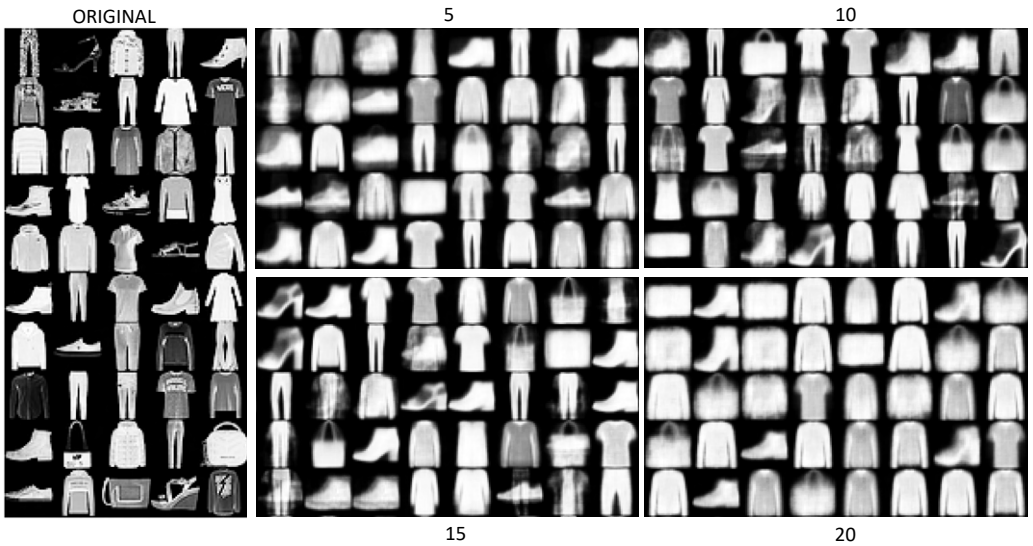


Figure 38: Example of randomly generated, uncurated images with different values of β using an uncoded model 8 information bits.

Table 16: Evaluation of the BER and WER in FMNIST with different values of β using an uncoded model 8 information bits.

Beta	BER	WER
5	0.203	0.852
10	0.086	0.384
15	0.089	0.384
20	0.278	0.939

Table 17: Evaluation of the log-likelihood (LL) in FMNIST with different values of β using an uncoded model 8 information bits.

Model	LL (train)	LL (test)
5	-256.431	-257.983
10	-247.507	-249.460
15	-247.964	-249.880
20	-272.460	-273.554

Next, we evaluate the model in the image generation task. Fig. 38 contains examples of randomly generated images using the different configurations. Table 16 reports the obtained BER and WER, and Table 17 the estimated log-likelihood of the different values of β. The models trained with $\beta = 15$ and $\beta = 10$ clearly outperform the other two in this task, generating more diverse and detailed images, and obtaining better error metrics and log-likelihood values.

K.2 ABLATION STUDY ON THE NUMBER OF TRAINABLE PARAMETERS

A consistent architecture was employed across all experiments, which is detailed in Section C. However, since the introduction of the code alters the dimensionality of the latent space, it is necessary to adjust the encoder's output and the decoder's input. This results in an augmentation of the trainable parameters in the coded cases compared to their uncoded counterparts.

Given that a higher number of parameters usually results in better performance, we conducted an ablation study on the model's trainable parameters to confirm that the improved performance introduced by the coded models is not due to this factor. We adjusted the hidden dimensions of the encoder and decoder architectures to ensure both configurations (coded and uncoded) have roughly the same number of trainable parameters. We have conducted the ablation study using the uncoded model with 8 bits and the coded 8/240 model trained on FMNIST.

We adjusted the encoder's last hidden dimension and the decoder's first hidden dimension to equalize the parameter count between the uncoded and coded models. This adjustment was straightforward since the last layers of the encoder and the first layers of the decoder are feed-forward layers. We kept the latent dimension of the model unchanged, ensuring that the modification solely pertained to the neural network architecture.

Table 18: Parameter count.

Model	# encoder parameters	# decoder parameters
uncoded 8	6,592,008	19,341,185
uncoded 8 adjusted	6,717,538	19,581,035
coded 8/240	6,711,024	19,578,753
coded 8/240 adjusted	6,583,174	19,332,871

K.2.1 Evaluation

This section provides an empirical evaluation of the models trained with the adjusted parameter count, demonstrating that the enhanced performance observed in coded models does not result from an augmented number of trainable parameters. We found that the performance of the original and adjusted models is very similar, meaning that the conclusions drawn in the main text hold even in this scenario.

We first evaluate the reconstruction performance, measuring the PSNR and reconstruction accuracy in both train and test sets, which are included in Table 19. Then, we assess generation measuring the BER and WER, reported in Table 20. Finally, we compute the log-likelihood for train and test sets, shown in Table 21.

In terms of reconstruction, both the original and adjusted models exhibit very similar performance, observed in both reconstruction quality and accuracy. However, the adjusted models show slightly inferior results than the original ones. For the coded model, this might be attributed to reduced flexibility when decreasing the number of parameters. As for

the uncoded model, the increased complexity while maintaining the same low-dimensional latent vector may not provide enough expressiveness to leverage the added flexibility in the architecture, potentially causing the observed decrease in performance. We observe the same behavior when we evaluate the BER and WER.

Analyzing the results, especially the log-likelihood values shown in Table 21, we can argue that increasing the flexibility in the architecture does not necessarily lead to improved performance in this scenario. The coded models exhibit similar performance with both the original and adjusted parameter counts, consistently outperforming the uncoded models. These results indicate that the performance enhancement is attributed to the introduction of ECCs in the latent space, rather than differences in the architecture required to handle the introduced redundancy.

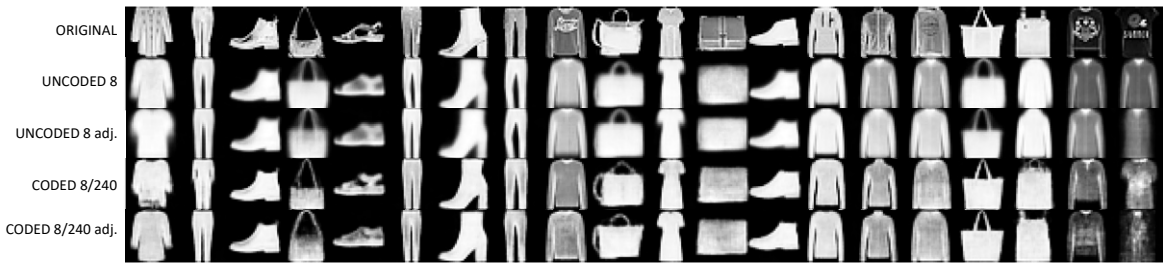


Figure 39: Example of reconstructed images obtained with different configurations.

Table 19: Evaluation of reconstruction performance in FMNIST with the adjusted parameter count.

Model	PSNR (train)	Acc (train)	Conf. Acc. (train)	PSNR (test)	Acc (test)	Conf. Acc. (test)	Entropy
uncoded 8	15.644	0.601	0.602	15.598	0.594	0.595	0.659
uncoded 8 adj.	15.530	0.586	0.586	15.491	0.581	0.580	0.449
coded 8/240	19.345	0.831	0.921	17.861	0.799	0.893	4.609
coded 8/240 adj.	19.383	0.828	0.883	17.771	0.792	0.828	3.952

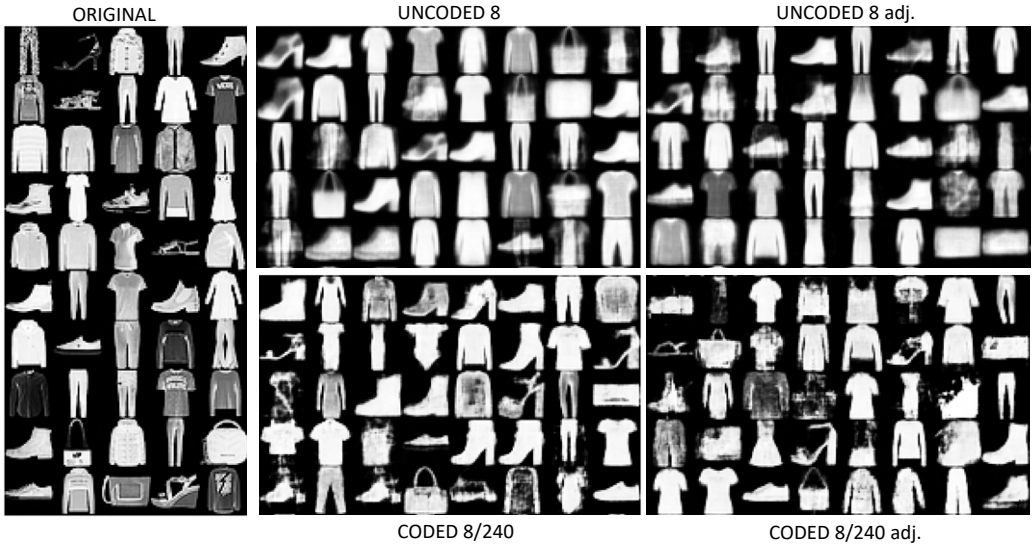


Figure 40: Example of randomly generated, uncurated images using different model configurations.

Table 20: Evaluation of the BER and WER in FMNIST with the adjusted parameter count.

Model	BER	WER
uncoded 8	0.089	0.384
uncoded 8 adj.	0.125	0.561
coded 8/240	0.037	0.231
coded 8/240 adj.	0.064	0.399

Table 21: Evaluation of the log-likelihood (LL) in FMNIST with the adjusted parameter count.

Model	LL (train)	LL (test)
uncoded 8	-247.964	-249.880
uncoded 8 adj.	-250.543	-252.408
coded 8/240	-231.679	-238.459
coded 8/240 adj.	-229.283	-238.302

L EVALUATING LOG-LIKELIHOOD USING THE SOFT-ENCODING MODEL

The reparameterization trick introduced in (5) requires that bits are independent, a condition not met in $\boldsymbol{c}$ once the code's structure is introduced. To address this issue, during training, we employ a *soft encoding strategy*. We assume the bits in $\boldsymbol{c}$ are independent and equally distributed according to $q_{\boldsymbol{\eta}}(\boldsymbol{m}|\boldsymbol{x})$. Therefore, instead of directly repeating sampled bits in $\boldsymbol{m}$ to obtain $\boldsymbol{c}$ following $\boldsymbol{c} = \boldsymbol{m}^T\mathbf{G}$, we repeat the posterior probabilities for the copies of the same bit and sample $\boldsymbol{z}$ using the reparameterization trick in (5). Remark that, despite the soft encoding assumption during training, the generative results in Fig. 4, 8, 15, 21, 26, 26, and 31 are obtained through *hard encoding*. Namely, we sampled an information word $\boldsymbol{m}$ and obtained $\boldsymbol{c}$ by repeating its bits.

While the hard-encoding images are visually appealing, to evaluate the Coded-DVAE LL for a given image $\boldsymbol{x}$, we have to leverage the soft-encoding model since it is unrealistic for a sample from the soft-encoding model to have equally repeated bits. In the soft-coded model, we sample bit probabilities from a prior distribution that we model through the product of M independent Beta distributions, and we use a proposal distribution model similar to the Vamp-prior in Tomczak and Welling [2018]. Namely, a mixture model with components given by $q(\boldsymbol{m}|\boldsymbol{x})$ for different training points.

In the main text and Section K, we report the LL values for different model configurations trained on the FMNIST dataset. Table 4 presents the results obtained for the MNIST dataset. Due to their simplicity, these datasets do not require high-dimensional latent spaces, and competitive results can be achieved with just 8 or 10 information bits. However, for more complex datasets like CIFAR10 or Tiny ImageNet, high-dimensional latent spaces are necessary to capture spatial information, colors, and textures. For these high-dimensional datasets, we were unable to obtain valid log-likelihood estimates because the number of samples needed for importance sampling to converge was too large. However, given the difference in the level of detail in the reconstructed and generated images between coded and uncoded models (see Sections F and G), coded models are expected to better approximate the true marginal likelihood of the data.

M CONNECTION TO PREVIOUS WORKS ON VAES AS SOURCE CODING METHODS

Effective learning of low-dimensional discrete latent representations from complex data is a technically challenging task. In this work, we propose a novel method to improve inference in discrete VAEs within a fully probabilistic framework, introducing a new perspective on the inference problem. While previous studies have analyzed VAEs using rate-distortion (RD) theory [Chen et al., 2022, Townsend et al., 2019, Van Den Oord et al., 2017], our approach stems from a different yet complementary perspective that remains compatible with these works.

In the literature, VAEs are often interpreted as lossy compression models (e.g., VAEs as source coding methods), but our contribution is best understood from a generative perspective. We conceptualize inference via $q(\boldsymbol{m}|\boldsymbol{x})$ as a decoding process, where the goal is to recover the discrete latent variable from the observed data. We sample a latent vector $\boldsymbol{m}$, generate a data point $\boldsymbol{x}$ (e.g., an image), and aim to minimize the error rate in recovering $\boldsymbol{m}$ from $\boldsymbol{x}$. Achieving this requires the variational approximation $q(\boldsymbol{m}|\boldsymbol{x})$ to closely match the true posterior $p(\boldsymbol{m}|\boldsymbol{x})$.We demonstrate that incorporating redundancy into the generative process reduces errors in estimating $\boldsymbol{m}$, leading to a more accurate latent posterior approximation. This improvement directly enhances our model's overall performance, as confirmed by our experimental results. Additionally, error rates and variational gaps are related through bounds derived from mismatch hypothesis testing. These bounds demonstrate that minimizing the variational gap by maximizing the ELBO effectively tightens an upper bound on the error rate [Schlüter et al., 2013, Yang et al., 2024].

In RD theory, the primary focus is on compression within the latent space, typically analyzed from an encoder/decoder and reconstruction (distortion) perspective. RD theory establishes theoretical limits on achievable compression rates and

describes how practical models may diverge from these limits. Using RD practical compression methods, Townsend et al. [2019] demonstrates how asymmetric numeral systems (ANS) can be integrated with a VAE to improve its compression rate by jointly encoding sequences of data points, bringing performance closer to RD theoretical limits. Similarly, Chen et al. [2022] shows that a complex prior distribution in a VAE using an autoregressive invertible flow narrows the gap between the approximate and the true posterior distribution, thereby enhancing the overall performance of the VAE. We note that even using an independent prior, the hierarchical code structure outlined in Section J naturally decouples information across different latent spaces at various conceptual levels. Specifically, the most relevant information (class label) is captured by the most protected latent space, while the other space captures fine-grained features. This effect is not straightforward to enforce through direct design of a more complex prior.

Our method complements all these efforts by introducing redundancy in the generative pathway to enhance variational inference, leading to more accurate latent posterior approximations. Notably, even when using a complex prior distribution, ECCs can still be leveraged to enhance inference and overall model performance. In this work, we demonstrate that our approach yields more robust models even with a simple, fixed independent prior, as evidenced by improved LL, generation quality, and reconstruction metrics. Moreover, in Section J, we show that integrating a hierarchical ECC with the same independent prior leads to even greater performance gains.

It is also important to discuss the VQ-VAE [Van Den Oord et al., 2017], one of the most relevant works related to our approach, which is well-known for effectively learning compressed discrete representations of complex data. We believe that highlighting key differences between VQ-VAEs and our method can provide useful insights. A key distinction lies in the structure and dimensionality of the latent space. In image modeling, VQ-VAEs typically employ a latent matrix where indices correspond to codewords in a codebook. This design allows different codewords to capture specific patches of the original image, improving reconstruction and generation (the latter through an autoregressive prior on the latent representations). However, representing each data point as a grid of embeddings complicates interpretability. In contrast, our method encodes the entire image into the latent space, rather than partitioning it into patches. Another major difference is the nature of the encoder. VQ-VAEs rely on a non-probabilistic encoder that maps inputs to the nearest latent code using a distance-based metric. This deterministic mapping limits the model's ability to capture and quantify uncertainty in the latent space. Our approach, on the other hand, operates within a fully probabilistic framework, allowing for uncertainty quantification in the learned latent representations. Additionally, our method is fully differentiable, enabling seamless gradient computation and backpropagation for end-to-end training. Our approach also allows us to leverage the reparameterization trick introduced in DVAE++ [Vahdat et al., 2018b], which we found to be more stable than continuous relaxations like the Gumbel-Softmax used in VQ-VAEs with stochastic quantization [Williams et al., 2020].

N VARIATIONAL INFERENCE AT CODEWORD LEVEL

Here, we present an alternative variational family that is valid for any ECC, including random codes. We assume that we have a deterministic mapping of the form $\boldsymbol{c} = \mathcal{C}(\boldsymbol{m})$. We assume a variational family of the form

$$q_{\boldsymbol{\eta}}(\boldsymbol{c}, \boldsymbol{z}|\boldsymbol{x}) = q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{c}), \tag{19}$$

$$q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) \propto p(\boldsymbol{c})q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}), \tag{20}$$

$$q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) = \prod_{j=1}^{M/R} \text{Ber}(g_{j,\boldsymbol{\eta}}(\boldsymbol{x})), \tag{21}$$

where $g_{\boldsymbol{\eta}}(\boldsymbol{x})$ represents the output of the encoder with a parameter set denoted as $\boldsymbol{\eta}$. Note that $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$ corresponds to the *uncoded* posterior, which we subsequently constrain using the prior distribution $p(\boldsymbol{c})$ over the code words to obtain the *coded* posterior $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$. Then, the *coded* posterior distribution can be defined as a categorical distribution over the set of codewords $\mathcal{C}(\boldsymbol{m})$, which is given by

$$\begin{aligned} q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) &= \text{Cat}\Big(\Big[\frac{1}{W}q^u_{\boldsymbol{\eta}}(\boldsymbol{c}_1|\boldsymbol{x}), \ldots, \frac{1}{W}q^u_{\boldsymbol{\eta}}(\boldsymbol{c}_{2^M}|\boldsymbol{x})\Big]\Big) \\ &= \frac{1}{W}\prod_{\boldsymbol{c}_i \in \mathcal{C}(\boldsymbol{m})} \prod_{j=1}^{M/R} g_{j,\boldsymbol{\eta}}(\boldsymbol{x})^{c_{i,j}}(1 - g_{j,\boldsymbol{\eta}}(\boldsymbol{x}))^{(1-c_{i,j})}, \end{aligned} \tag{22}$$

where $W = \sum_{\boldsymbol{c}_i \in \mathcal{C}(\boldsymbol{m})} q^u_{\boldsymbol{\eta}}(\boldsymbol{c}_i|\boldsymbol{x})$ is a constant for normalization.

Inference is done by maximizing the ELBO, which can be expressed as

$$\begin{aligned} \text{ELBO} &= \int q_{\boldsymbol{\eta}}(\boldsymbol{c}, \boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{c})}{q_{\boldsymbol{\eta}}(\boldsymbol{c}, \boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{c} d\boldsymbol{z} \\ &= \mathbb{E}_{q_{\boldsymbol{\eta}}(\boldsymbol{c}, \boldsymbol{z}|\boldsymbol{x})} \log \left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}|\boldsymbol{c}) p(\boldsymbol{c})}{q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) p(\boldsymbol{z}|\boldsymbol{c})} \right) \\ &= \mathbb{E}_{q_{\boldsymbol{\eta}}(\boldsymbol{c}, \boldsymbol{z}|\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \mathcal{D}_{KL}\big(q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}) || p(\boldsymbol{c})\big). \end{aligned} \tag{23}$$

In this case, due to the inability to compute the KL Divergence in closed form, we approximate it via Monte Carlo, sampling from the categorical distribution $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$. The reconstruction term also needs to be approximated via Monte Carlo. Since the use of channel coding introduces structural dependencies among the components of the vectors $\boldsymbol{c}$, we can no longer assume their independence. Consequently, the formulation of the smoothing transformation as independent mixtures introduced in the DVAE is no longer applicable. Hence, this approach involves sampling $\boldsymbol{c}'$ from the categorical distribution $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$ and subsequently applying the transformation over the sampled word. Thus, we obtain a smooth transformation for each sample $\boldsymbol{c}'$ using the inverse CDFs of $p(z_j|c_j = 0)$ and $p(z_j|c_j = 1)$, which are given by

$$F^{-1}_{p(z_j|\boldsymbol{c}'_j=0)}(\rho) = -\frac{1}{\beta} \log\big(1 - \rho(1 - e^{-\beta})\big), \tag{24}$$

$$F^{-1}_{p(z_j|\boldsymbol{c}'_j=1)}(\rho) = \frac{1}{\beta} \log\big(\rho(1 - e^{-\beta}) + e^{-\beta}\big) + 1. \tag{25}$$

These are differentiable functions that convert samples ρ from a uniform distribution $\mathcal{U}(0, 1)$ into a sample from $q_\eta(\boldsymbol{z}, \boldsymbol{c} = \boldsymbol{c}'|\boldsymbol{x})$ following

$$\begin{aligned} &q_\eta(\boldsymbol{z}, \boldsymbol{c} = \boldsymbol{c}'|\boldsymbol{x}) = \\ &= \prod_{j=1}^{M/R} \Big[F^{-1}_{p(z_j|\boldsymbol{c}'_j=1)}(\rho)^{c'_j} + F^{-1}_{p(z_j|\boldsymbol{c}'_j=0)}(\rho)^{(1-c'_j)} \Big]. \end{aligned} \tag{26}$$

Thus, we can apply the reparameterization trick to obtain samples from the latent variable $\boldsymbol{z}$ and optimize the reconstruction term of the ELBO with respect to the parameters $\boldsymbol{\theta}$ of the decoder.

The KL Divergence term is approximated via Monte Carlo, drawing samples from $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x})$. Since it is not possible to backpropagate through discrete variables, we approximate the gradients with respect to the parameters of the encoder using the REINFORCE leave-one-out estimator [Salimans and Knowles, 2014, Kool et al., 2019], given by

$$\begin{aligned} &\widehat{g}_{LOO} = \\ &= \frac{1}{S-1} \left[\sum_{s=1}^{S} f_{\boldsymbol{\eta}}\big(\boldsymbol{z}^{(s)}, \boldsymbol{c}^{(s)}\big) \bigtriangledown_{\boldsymbol{\eta}} \log q_{\boldsymbol{\eta}}\big(\boldsymbol{c}^{(s)}|\boldsymbol{x}\big) - \overline{f}_{\boldsymbol{\eta}} \sum_{s=1}^{S} \bigtriangledown_{\boldsymbol{\eta}} \log q_{\boldsymbol{\eta}}\big(\boldsymbol{c}^{(s)}|\boldsymbol{x}\big) \right], \end{aligned} \tag{27}$$

where

$$f_{\boldsymbol{\eta}}(\boldsymbol{z}^{(s)}, \boldsymbol{c}^{(s)}) = \log \left(\frac{q_{\boldsymbol{\eta}}(\boldsymbol{c}^{(s)}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(s)}) p(\boldsymbol{c}^{(s)})} \right), \tag{28}$$

$$\overline{f_{\boldsymbol{\eta}}} = \frac{1}{S} \sum_{s=1}^{S} f_{\boldsymbol{\eta}}(\boldsymbol{z}^{(s)}, \boldsymbol{c}^{(s)}). \tag{29}$$

Defining a distribution over the codebook can seem intuitive, but scalability becomes challenging as the size of the codebook increases. The reason for this is that the posterior distribution must be evaluated for all codewords during both inference and test time. However, it can still provide a bound, enabling the utilization of more complex codes with theoretical guarantees that can outperform the previously proposed repetition codes.

Algorithm 3 Training the model with inference at codeword level.

Input: training data $\boldsymbol{x}_i$, codebook.
repeat
  $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i) \leftarrow$ forward encoder $g_{\boldsymbol{\eta}}(\boldsymbol{x}_i)$
  $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i) \leftarrow$ evaluate $q^u_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i)$ over the codebook and normalize
  $\tilde{\boldsymbol{c}} \leftarrow$ sample from $q_{\boldsymbol{\eta}}(\boldsymbol{c}|\boldsymbol{x}_i)$
  $\boldsymbol{z} \leftarrow$ modulate $\tilde{\boldsymbol{c}}$
  $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \leftarrow$ forward decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$
  Compute ELBO according to (23)
  Compute encoder's gradients according to (27)
  $\boldsymbol{\theta}, \boldsymbol{\eta} \leftarrow Update(ELBO)$
until convergence

O COMPUTATIONAL RESOURCES

All the experiments in this paper were conducted on a single GPU. Depending on availability, we used either a Titan X Pascal with 10GB of RAM, a Nvidia GeForce GTX with 10GB of RAM, or a Nvidia GeForce RTX 4090 with 24GB of RAM. Since training times varied significantly based on the hardware used, we were unable to provide comparable training times.

P CODED-DVAE SCHEME

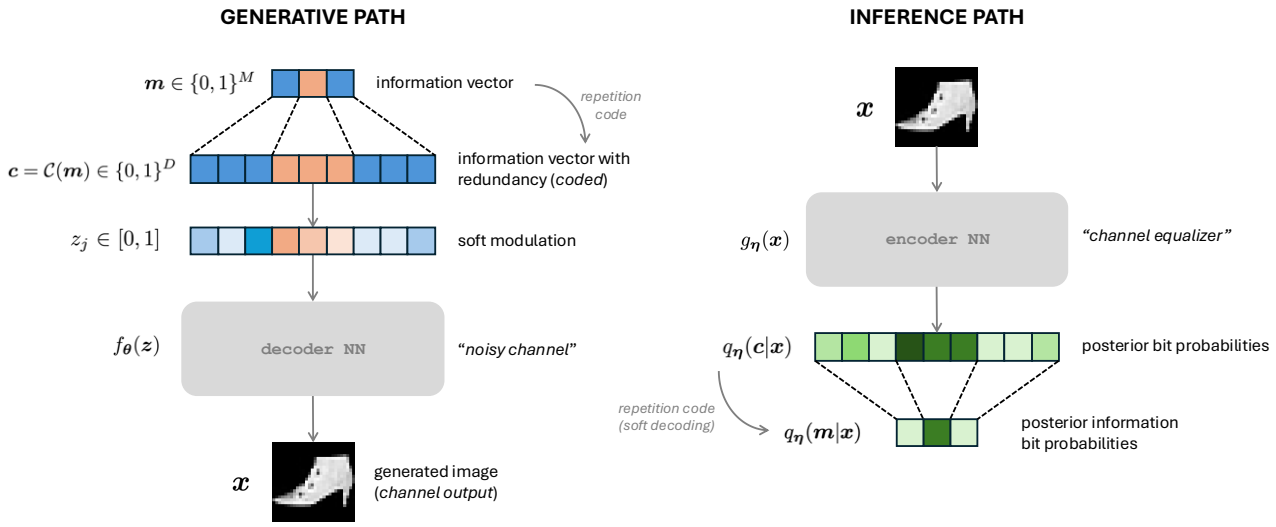


Figure 41: Graphic representation of the Coded-DVAE, illustrating both the generative and inference paths.

Q HIERARCHICAL CODED-DVAE SCHEME

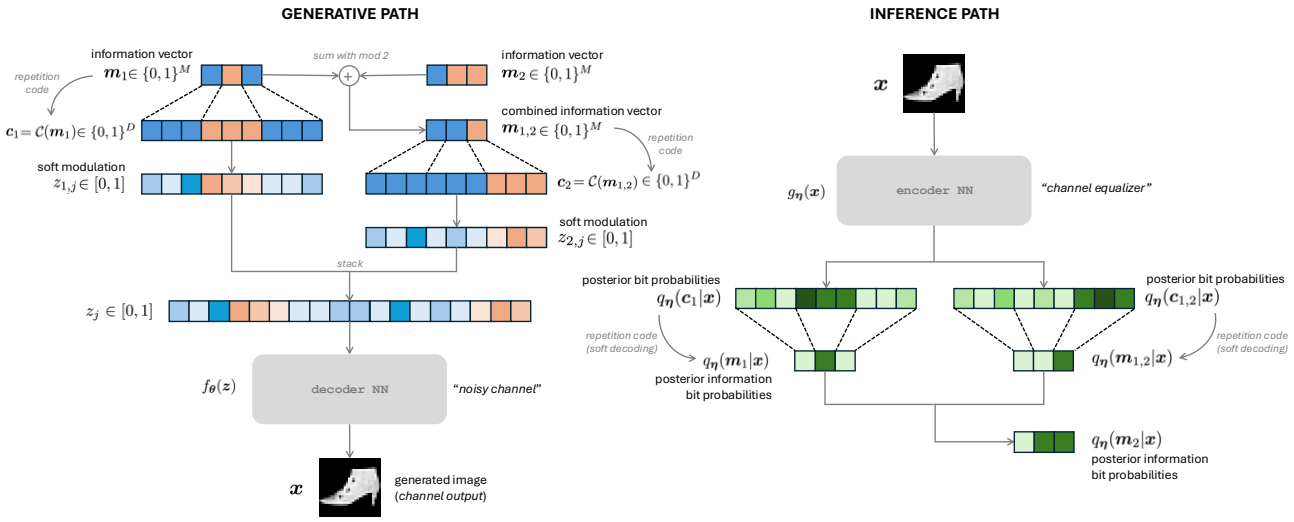


Figure 42: Graphic representation of the hierarchical Coded-DVAE, illustrating both the generative and inference paths.