Neural Distributed Source Coding
Jay Whang, Alliot Nagle, Anish Acharya, Hyeji Kim, Member, IEEE, and Alexandros G. Dimakis, Fellow, IEEE

Abstract—We consider the Distributed Source Coding (DSC) problem concerning the task of encoding an input in the absence of correlated side information that is only available to the decoder. Remarkably, Slepian and Wolf showed in 1973 that an encoder without access to the side information can asymptotically achieve the same compression rate as when the side information is available to it. This seminal result was later extended to lossy compression of distributed sources by Wyner, Ziv, Berger, and Tung. While there is vast prior work on this topic, practical DSC has been limited to synthetic datasets and specific correlation structures. Here we present a framework for lossy DSC that is agnostic to the correlation structure and can scale to high dimensions. Rather than relying on hand-crafted source modeling, our method utilizes a conditional Vector-Quantized Variational auto-encoder (VQ-VAE) to learn the distributed encoder and decoder. We evaluate our method on multiple datasets and show that our method can handle complex correlations and achieves state-of-the-art PSNR.

Index Terms—Distributed source coding, Berger-Tung inner bound, Slepian-Wolf coding, Wyner-Ziv coding, vector-quantized variational auto-encoder

I. INTRODUCTION

DATA compression plays an essential role in modern computer systems. From multimedia codecs running on consumer devices to cloud backups in large data centers, compression is a necessary component in any system that deals with high-volume or high-velocity sources. Applications such as multi-camera surveillance systems, IoT sensing, 3D scene capture and stereo imaging create distributed data streams with very large volume that are highly correlated. The central question is how each encoder terminal can efficiently compress its respective data so the decoder can recover the distributed data sources reliably.

This problem has been widely studied in the context of distributed source coding in information theory. For lossless distributed compression of discrete sources, Slepian and Wolf showed in 1973 that, surprisingly, the sum rate can be reduced to the joint entropy of the sources [1], i.e., the limit on the distributed lossless compression is equivalent to that on centralized compression, where the sources are jointly encoded. Cover provided proof of this surprising result using random binning techniques and extended this result to any pair of stationary ergodic sources [2]. For lossy distributed compression, Berger [3] and Tung [4] extended the Slepian-Wolf scheme to a compress-bin scheme, where typically the encoders perform joint encoding and binding, and the decoder performs joint typically decoding and symbol-by-symbol reconstruction. The Berger-Tung inner bound is not optimal in general [5]; however, it is shown to be optimal for several cases, e.g., two correlated Gaussian distributions with mean square error distortion [6], two correlated binary sources with log loss [7], the Gaussian CEO problems for two [8], [9], [10] or more [10], [11] sources.

We are interested in a special case of distributed source coding (DSC) – specifically the problem of compression with decoder side information. As depicted in Figure 1 (left), in this setting there are two correlated sources (input $x$ and side information $y$) that are physically separated. Both must be compressed and sent to a common decoder, but we assume that the side information $y$ is compressed in isolation and communicated to the decoder. The encoder has no access to the correlated side information $x$ when the correlated side information $y$ is available only at the decoder as shown in 1 (left). If side information is available to both the encoder and decoder as in 1 (right), it is well known that the side information can be utilized to improve the compression rate of $x$ [12, Ch. 11]. Surprisingly, an encoder that has no access to the correlated side information can asymptotically achieve the same compression rate as when side-information is available at both the encoder and the decoder, which is $H(x|y)$ [1]. In other words, distributed compression

![Distributed encoder](image1)

![Joint encoder](image2)
is asymptotically as efficient as joint compression. This is a remarkable result in classical information theory that defies intuition. For lossy compression, the situation is more nuanced. Wyner and Ziv extended the Slepian-Wolf scheme to include quantization, followed by random binning, for lossy compression [13]; this extension is a specific case of the Berger-Tung inner bound for distributed source coding [3], [4]. Whether the rate-distortion trade-off for the joint encoder is equivalent to that for the distributed encoder, however, depends on the distribution of the source and side information and the distortion metrics. For correlated binary sources with Hamming distortion, providing side information to both the encoder and decoder can reduce the compression rate. However, it does not offer any rate gain for correlated Gaussian sources with mean squared error distortion [14].

In 1999, Pradhan and Ramchandran advanced the insight from the Slepian-Wolf result [1] and introduced a constructive scheme for the distributed coding of i.i.d. correlated sources, such as Bernoulli sources, denoted by Distributed Coding Using Syndromes (DISCUS) [15]. The key underlying concept, as indicated by its name, is to utilize channel coding and syndrome decoding techniques, which efficiently replace random binning and joint typicality decoding. This idea can be readily extended to lossy distributed compression by first quantizing the source, then applying Slepian-Wolf coding as was proposed by [13]. Several constructive schemes that combine the DISCUS along with quantization are proposed, e.g., for correlated i.i.d. Gaussian sources, and are shown to be near-optimal as the message length approaches infinity [15], [16].

Despite these theoretical advancements, there are substantial challenges when it comes to designing practical distributed compression schemes for arbitrarily correlated sources. First, the joint distribution \( p(x, y) \) is required to design the encoder and decoder, but modeling high-dimensional joint distributions (e.g., for correlated images) is very challenging, especially without modern deep generative models. Second, designing a distributed compression scheme with reasonable run time beyond simple structures (e.g., correlated i.i.d. Bernoulli or Gaussian sources) remains an open question. The significant gap between the theory and practice of DSC has been well-acknowledged in the information theory community. Two decades after the Slepian-Wolf theorem, [17] writes: “despite the existence of potential applications, the conceptual importance of Slepian “Wolf coding has not been mirrored in practical data compression.” Constructive compression schemes have been designed only for very special cases, such as correlated Bernoulli sources [15], Gaussian sources [15], [16], and stereo images (as elaborated in II-D).

In this paper, we bridge this gap by leveraging the recent advances in deep generative modeling and show that it is possible to train an encoder and decoder for distributed compression of arbitrarily correlated high-dimensional sources. Specifically, our approach (denoted Neural DSC) parametrizes the encoder-decoder pair as a Vector-Quantized VAE (VQ-VAE) [18], so that: (a) we avoid needing a hand-crafted analytical model for the source distribution; (b) we can learn the complicated encoder and decoder mappings via neural networks with efficient inference; and (c) the discrete latent representation of VQ-VAE allows for a further rate improvement through post-hoc training of a latent prior Section (III-B). Our main contributions are as follows:

- We introduce Neural DSC, a compression scheme for lossy distributed source coding based on a particular type of VAE (VQ-VAE). The VQ-VAE architecture contains a codebook learned during training in addition to the encoder and decoder, to which the latent representations are quantized.
- We justify the use of the VAE by establishing a connection between DSC and a modified evidence lower bound (ELBO) used to train VAEs for our asymmetric encoder-decoder setup. We call this modified objective dELBO (distributed ELBO) and use it to obtain an objective for training our VQ-VAE.
- We show that Neural DSC performs favorably to existing techniques on stereo camera image compression, achieving state-of-the-art PSNR for rates above 0.1 bit per pixel (bpp); we observe up to a 21% rate reduction for the same PSNR against the best baseline we compare to. Moreover, our method achieves competitive performance in MS-SSIM [19], and we achieve these results with 4.2 – 7.6\( \times \) fewer parameters than the baselines.
- We further validate that our approach can adapt to complex correlations and other data modalities, such as gradients for distributed learning, and study how effectively the decoder uses side information. Our work is novel in this regard since prior work only considers stereo images in their experiments. We also provide a comparison between our Neural DSC and the theoretically optimal compressors on a synthetic dataset.

II. BACKGROUND

In this section, we introduce the background necessary to introduce our method. First, we provide a motivating example to provide some intuition on DSC and follow it up by briefly providing some background on ELBO, VAE, and VQ-VAE. Finally, we discuss some related work for learning-based DSC.

A. Intuition Behind DSC

To provide some intuition behind distributed source coding, we describe a simple example that illustrates how side information known only to the decoder can be as useful as side information known to both the encoder and decoder [15]. Let \( x \) and \( y \) be uniformly random 3-bit sources that differ by at most one bit. Clearly, losslessly compressing \( x \) requires 3 bits. However, if \( y \) is known to both encoder and decoder, then \( x \) can be transmitted using 2 bits instead. This is because the encoder can send the difference between \( x \) and \( y \), which is uniform in \( \{000, 001, 010, 100\} \). Thus, joint compression uses 2 bits.

Now, if the side information \( y \) is available only at the decoder, Slepian-Wolf theorem suggests that the encoder can still transmit \( x \) using only 2 bits. How could this be possible? The key idea is to group 8 possible values of \( x \) into 4 bins, each containing two bit-strings with maximal Hamming distance: 

\[
B_0 = \{000, 111\}, \quad B_1 = \{001, 110\}, \quad B_2 = \{010, 101\}, \quad B_3 = \{011, 000\}
\]
{011, 100}. Then the encoder simply transmits the bin index \( m \in \{0, 1, 2, 3\} \) for the bin containing \( x \). The decoder can produce the reconstruction \( \hat{x} \) based on the bin index \( m \) and \( y \), precisely, \( \hat{x} = \arg \max_{m \in E_m} P(x \mid y) \). Since \( x \) and \( y \) are off by at most one bit and the Hamming distance between the bit strings in each bin is 3, the decoder can recover \( x \) without error given \( y \). In other words, the side information allows the decoder to correctly choose between the two candidates in the bin specified by the encoder.

### B. ELBO & VAE

A variational auto-encoder (VAE) [20] is a special type of auto-encoder that regularizes the latent space such that the latent variable \( z \) is a Gaussian random variable. Typically, the training objective of auto-encoders is to minimize the reconstruction loss. However, the training objective of the VAE is to recover the true distribution \( p(x) \) from which the training data is assumed to have been sampled from.

While the decoder \( p(x \mid z) \) allows us to trivially compute \( p(x, z) = p(x \mid z)p(z) \) under the assumption that \( z \) is a Gaussian random variable, we still cannot directly compute \( p(x) = \int p(x \mid z)p(z) \) due to the intractable marginalization over \( z \). Thus, the VAE training objective instead seeks to maximize the evidence lower bound (ELBO). ELBO is a lower bound on the log-likelihood \( p(x) \) of the training data that depends on an encoder distribution \( q(z \mid x) \), often also called the variational posterior. The tightness of this lower bound depends on the choice of \( q \), with equality achieved when \( q \) exactly recovers the true posterior, i.e., \( q(z \mid x) = p(z \mid x) \).

VAE training is done by jointly training both the decoder distribution \( p(x \mid z) \) and the encoder distribution \( q(z \mid x) \) to maximize the ELBO, which consists of a distortion (reconstruction) term and a rate term:

\[
\log p(x) \geq \text{ELBO}(x) = \mathbb{E}_{q(z \mid x)}[\log p(x \mid z)] - D_{KL}(q(z \mid x) \parallel p(z)) \tag{1}
\]

Intuitively, the distortion term represents the reconstruction loss, and the rate term regularizes the latent space such that the posterior \( q(z \mid x) \) is close to the prior \( p(z) \). Note again that \( p(z) \) is zero mean isotropic Gaussian. As a special case, when \( p(x \mid z) \) is assumed to be Gaussian, the distortion term becomes the \( \ell_2 \) reconstruction loss. A derivation of ELBO and a detailed discussion around VAEs can be found in [21].

VAEs are commonly used for data compression; one such example is bits back coding [22]. Some earlier works [23], [24] have demonstrated the utility of VAEs for image compression, and these works form the backbone of recent works in the DSC setting [25], [26]; we discuss them in Section II-D.

### C. Vector-Quantized VAE

Vector-Quantized VAE (VQ-VAE) [18] is a specific type of VAE [20], [27] with a discrete latent variable, even when the input is continuous. Because the latent code is discrete and has fixed size, VQ-VAEs are a natural fit for lossy compression. Indeed, many existing works have explored its use in various compression tasks, ranging from music generation to high-resolution image synthesis [28], [29], [30], [31], [32].

A VQ-VAE consists of three components: an encoder, a decoder, and a codebook. The main difference between VQ-VAE and a regular VAE is that the output of the encoder is quantized to the nearest vector in the codebook. During training, all three components are jointly optimized. During inference, the encoder compresses the input into a lower-dimensional representation of vectors which are quantized to the latent vectors in the codebook. These quantized representations are passed through the decoder, which reconstructs the image. Once a VQ-VAE is trained, it is common to use an autoregressive model, such asPixelCNN [33] or transformers [34], to achieve a lower rate for a fixed distortion value. The autoregressive model learns the joint distribution for the ordered set of latent codebook vectors used to represent each input to the VQ-VAE; this model can be used to lower the compression rate of the VQ-VAE via arithmetic coding [35].

### D. Related Work

The authors of DSIN [36] propose a method to perform DSC for stereo camera image pairs with a high spatial correlation. Due to the large spatial similarity between the images, one of them can serve as the side information for the other. The key component of their method is the “Side Information Finder”, a module that finds similar image patches between the side information and the reconstructed signal produced by a pre-trained autoencoder. Since the two images have many approximately overlapping patches, the reconstruction is further improved by copying over matching image patches from the side information to the reconstruction.

While this leads to a considerable improvement in compression performance, this method is only applicable when the input and side information have large spatial overlap. On the other hand, our method is applicable to any correlated sources. We empirically validate this using data sources with substantially more complex correlation (see Fig. 11, right).

NDIC [25] leverages a (regular) VAE for DSC of image data; the proposed architecture is designed to explicitly model the common information between the input and side information. Intuitively, the goal is to guide the encoder to compress only the portion that is not recoverable from the side information. Once trained, the encoder simply discards the common information and only transmits the residual, hoping the decoder can reconstruct the common information from the side information. NDIC-CAM [26] further adapts this architecture by encoding and decoding the side information in parallel to the input image decoder network and utilizing cross-attention modules between the two decoder networks. They report that the cross-attention modules allow for better alignment between the latent representations of the side information and the input image, thereby enabling better utilization of the side information over NDIC.

LDMIC [37] presents a DSC framework for multi-view images, where \( K \) overlapping images (from multi-view cameras, for example) are compressed independently but decompressed jointly. Their key architectural contribution is...
their joint context transfer module; this module is used in the
decoder to reconstruct image \( k \in \{1, \ldots, K \} \) based on the
compressed representations of the remaining \( \{1, \ldots, K \} \setminus \{k\} \)
images. They consider a setting where the side information is
lossily compressed before being presented to the decoder. This
is different from our setting, where we (along with the other
methods) consider a situation where the side information is
losslessly presented to the decoder.

Finally, a recent study has shown that neural distributed
compressors learn to bin their source data [38]. In addition
to their provided theory, the authors empirically observe the
emergence of binning for Gaussian and Laplacian data. Our
work differs in that (1) we study the DSC with complex
 correlations between the encoder input and the decoder side
information, and (2) we consider vector input data, whereas
this other work considers scalar input data.

III. NEURAL DISTRIBUTED SOURCE CODING

In this section, we derive our ELBO bound for the DSC
setting. We discuss the connection of our distributed ELBO
bound to the optimization objective of NDIC-CAM [26],
the state-of-the-art baseline we compare our work to. We use our
distributed ELBO bound to motivate a new loss for the VQ-
VAE architecture. Figure 2 provides a high-level overview of
our method, showing how side information is incorporated into
the VQ-VAE architecture and how arithmetic coding is used to
achieve improved compression rates.

A. ELBO and Distributed Source Coding

VAEs have been extensively used for neural compression
[23], [24], [39], as its training objective, ELBO, can be interpreted as
a sum of rate and distortion of a lossy compressor as shown in
Equation (1). For example, when the decoder \( p(x | z) \) is Gaussian, the first term turns into the
widely-used \( \ell_2 \) distortion.

However, this interpretation breaks down in our asymmetric
encoder-decoder setup where only the decoder has access to
the side information \( y \). In this setup, we are interested in
modelling the conditional likelihood \( p(x | y) \) as opposed to
the likelihood \( p(x) \) for the symmetric (vanilla VAE) case. We
can show that the corresponding objective in our setup, which
we call distributed ELBO (or dELBO for short), is in fact a
lower bound to the conditional log-likelihood log \( p(x | y) \):

Proposition 1: Let \( x, y, z \) be random variables following the
generative process \( y \rightarrow x \leftarrow z \), i.e., \( z \) is the latent variable
that is independent of \( y \) and \( p(x, y, z) = p(x | y, z)p(y)p(z) \).
Then for any choice of posterior \( q(z | x) \) valid under \( p \) (i.e., \( \text{supp } q(z | x) \subseteq \text{supp } p(z) \) for all \( x \)), we have

\[
\log p(x | y) \geq \mathbb{E}_q[\log p(x | y, z)] - KL[q(z | x) \| p(z)]
\]

\( \Delta \) dELBO \( (x, y) \).

Proof:

\[
\log p(x | y) = \mathbb{E}_{z \sim q(z | x)}[\log p(x | y)] = \mathbb{E}_q[\log p(x, y, z) - \log p(z | x, y) - \log p(y)]
\]

\[
= \mathbb{E}_q[\log p(x, z) + \log p(y) - \log p(z | x, y) - \log p(y)]
\]

where (A) follows from our assumption that \( y \) and \( z \) are
marginally independent. Note that the dELBO rate term,
\( D_{KL}(q(z | x) \| p(z)) \), matches the rate term in ELBO.

The importance of this connection between ELBO and
distributed source coding is twofold. First, it shows that there
is a tractable surrogate objective that minimizes the optimal
conditional compression rate \( -\log p(x | y) \), which is also the
asymptotically optimal rate for distributed coders [1], [13].
Second, and perhaps more importantly, this scheme also allows
us to obtain a practically useful encoder-decoder pair whose
rate-distortion we can control as we shall see later.

Remark 1: We point out that dELBO is a special case of
the loss objective proposed in Equation (1) of the NDIC-
CAM work [26]. Beyond the dELBO terms, their loss
objective includes terms for the rate and distortion of the
side information \( y \), controlled by hyperparameter \( \alpha \), and the
rate of their proposed common information, controlled by
hyperparameter \( \beta \). However, they empirically show that setting
\( \alpha = \beta = 0 \) yields the best results, which is the case that
recovers our dELBO objective. Specifically, their loss function
is the following:

\[
L = R_x + \lambda D_x + \alpha (R_y + \lambda D_y) + \beta R_w,
\]

where \( R_x, D_x, R_y, D_y \) are the rate and distortion terms for the
input and side-information, respectively, and \( R_w \) is the rate of the
common information, which is a nonlinear transform of the
side-information. Our dELBO objective is recovered for
\( \alpha = \beta = 0 \), since \( R_x, D_x \) both depend on \( x \) given \( y \).

B. Our Method

1) Notation and Setup: We let \( x, y, \) and \( c_m \) denote the
original message, correlated side information, and compressed
message, respectively. The encoder \( f \) maps the given message

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
to a low-dimensional vector \( f(x) \), which is then quantized to
the nearest codebook vector:

\[
c_m = \arg \min_{c \in C} \| c - f(x) \|,
\]

where the table of code vectors \( C \) is jointly trained with
the encoder \( f \) and the decoder \( g \). Since the decoder also has access
to the codebook, only the index \( m \) of the compressed message
is transmitted, which is accomplished in a lossless manner.
The decoder \( g \) in turn tries to reconstruct the input signal using
the compressed message and side info: \( \hat{x} = g(c_m; y) \). Note that
the encoder only accepts \( x \) as input, while the decoder accepts both \( c_m \) and the side information \( y \).

We refer to the number of bits required to transmit \( c_m \) as the
rate, and the reconstruction performance (measured by \( \ell_2 \) error
or MS-SSIM) as the distortion. For our image compression
experiments, the rate is reported as bits-per-pixel (bpp) as
done in standard practice. bpp measures the average number of
bits needed to compress a pixel for a given image. Therefore,
the total number of bits required to transmit \( c_m \) (as an RGB
image), with \( n \) pixels and bpp \( r \), is \( n \times r \).

2) Training Objective: We train the distributed encoder-decoder pair \( (f, g) \) along with the codebook \( C \) as a conditional
VQ-VAE, where only \( g \) is conditioned on the side information \( y \). In particular, we iterate the updates of the distributed
encoder-decoder pair \( (f, g) \) and the codebook \( C \); we first
discuss the loss function for updating \( (f, g) \) for a given
codebook \( C \), followed by the method for updating \( C \) for a given
\( (f, g) \).

\[
L(f, g, C) = ||g(c_m; y) - x||^2 
+ \beta ||f(x) - \text{stop}_\text{grad}(c_m)||^2, \tag{3}
\]

where the \text{stop}_\text{grad} operator acts as the identity function
in the forward direction but prohibits the propagation of
gradients during training. This loss can be interpreted as a
weighted sum of a log-likelihood term under a Gaussian
decoder (distortion) and a vector quantization term that tries to
bring encoder output close to existing code vectors, which in
turn has an impact on the resulting rate. This training objective
is derived from dELBO in a similar fashion as the original
VQ-VAE objective was derived from ELBO [18], which we
detail here. Recall that the dELBO objective we derived for a
VAE is

\[
d\text{ELBO}(x, y) = -D_{\text{KL}}(q(z \mid x) \parallel p(z)) + \mathbb{E}_q[\log p(x \mid y, z)].
\]

For the VQ-VAE objective, first, we assume that \( p(z) \) is uniform.
Consequently, the KL divergence \( D_{\text{KL}}(q(z \mid x) \parallel p(z)) \) is
therefore constant with respect to the encoder parameters and
can be removed from the objective. In particular, when \( p(z) \) is
uniform, then \( D_{\text{KL}}(q(z \mid x) \parallel p(z)) = \sum_{z \in \text{supp } q(z \mid x)} p(z \mid
x) \log \frac{q(z \mid x)}{p(z)} = \log |C| \) since \( q(z \mid x) \) is deterministic (a one-
hot encoded distribution) when \( x \) is known. This leaves us
with the \( \mathbb{E}_q[\log p(x \mid y, z)] \) distortion term, which, under
the standard assumption that \( p(x \mid y, z) \) is Gaussian, becomes
the MSE loss. This corresponds to the first term in our loss
function shown in Equation (3). In practice, the distortion
does not need to be restricted to MSE and can instead be any loss that measures the distortion between two images, such as MS-SSIM. Next, we include the second term in our loss function shown in Equation (3), known as the commitment loss, which forces the encoder to learn to use the codebook and “commit” to outputting vectors that are close to the codebook vectors.

Note that our loss does not include a term that reflects codebook updates. Instead, to update the codebook \( C \) for a given \((f, g)\), we follow the strategy of the VQ-VAE paper where the codebook is updated using exponential moving averages (EMA). Intuitively, this approach is similar to a K-means update where the cluster centers (the codebook vectors in our case) are updated according to the average of the nearest vectors. Optimally, each codebook vector \( c_i \) shall be the average of the \( n_i \) closest output vectors from the encoder across all inputs \( x \). However, this approach is not well-suited for us, as we only have access to mini-batches. Instead, we update each codebook vector \( c_i \) as the exponential moving averages of the cluster centers of mini-batch outputs; specifically, we update the \( i \)-th codebook vector to be

\[
\hat{c}_i(t) = \alpha_i(t) \hat{c}_i(t-1) + (1 - \gamma) \sum_j \delta_{ij}^{(t)} \text{and} \quad \beta_i(t) = \gamma \beta_i(t-1) + (1 - \gamma)n_i(t).
\]

Here, \( \{\delta_{ij}^{(t)}\}_{j=1}^{N} \) are the \( n_i(t) \) encoder outputs in the current \( t \)-th mini-batch which are closest to codebook vector \( c_i^{(t)} \). \( \gamma \in [0, 1] \) is a hyperparameter; we follow the original VQ-VAE work and set it to \( \gamma = 0.99 \). At initialization, prior to any training, \( \alpha_i(0) \) is uniform and \( D_{KL}(q(z \mid x) \parallel p(z)) \) is therefore constant with respect to the encoder parameters and can be removed from the objective, we add the commitment loss (the second term in Equation (3)), and the codebook \( C \) is jointly updated as an exponential moving average of code vectors following the original VQ-VAE work.

3) Why VQ-VAE?: Although much of the existing literature has focused on VAEs as the backbone for neural compression [23], [24], [39], [40], [41], we intentionally chose VQ-VAE instead for a few reasons. First, VQ-VAE offers explicit control over the rate through the latent dimension and codebook size—unlike VAEs, for which we can only estimate the rate after training the model. VQ-VAE also guarantees an upper bound on the resulting rate regardless of the input, which is often desired in practice (e.g., a communication channel with a strict maximum bandwidth limit). Finally, a rate improvement can be achieved for a trained VQ-VAE by fitting a latent prior model to the VQ-VAE’s discrete latent space; we demonstrate such a rate improvement in Figure 3a. The downside of using a VQ-VAE is that training is less stable than a VAE, but this did not pose much issue during our experiments. These stability issues are remedied through the use of the commitment loss term and the EMA codebook updates, which we discuss further in Section III-B2.

4) Use of a Latent Prior: Our VQ-VAE architecture achieves comparable performance to other methods in the DSC setting. However, the aggregate posterior learned by VQ-VAE is often far from uniform in practice. This allows for a further rate improvement by losslessly compressing the latent codes by fitting a discrete prior over them. Following [18], we use an autoregressive prior based on the Transformer [34], which can be readily used with arithmetic coding [35] to obtain a lossless compressor that closely matches the model’s negative log-likelihood.

Specifically, we use the decoder-only Transformer trained with a maximum-likelihood objective on discrete latent codes. This differs from the original VQ-VAE work [18], which utilized a PixelCNN prior. The Transformer has been shown to achieve better performance compared to PixelCNN variants [29], [42]. In Figure 3, we provide our VQ-VAE results with (solid blue line) and without (dashed blue line) the latent prior. We provide VQ-VAE results with (solid blue line) and without (dashed blue line) the latent prior. When we use the latent prior, we outperform all other PSNR methods, and match the best method on MS-SSIM.
IV. EXPERIMENTS

We demonstrate the efficacy of our framework through a diverse set of experiments. In our stereo image compression experiments in Section IV-A, we match or exceed the performance of all baselines. Then in Section IV-B, we methodically investigate how well VQ-VAEs handle complex correlations between the input message and side information in the distributed, joint, and separate cases. Finally, in Section IV-C, we provide a proof of concept experiment to show how our framework can be used in distributed training.

A. Stereo Image Compression

1) Setup: We first evaluate our method on stereo image compression. Following [36], we construct a dataset consisting of pairs of images obtained from the KITTI Stereo 2012 and 2015 [43], [44]. Each pair of images is taken by two cameras at a slightly different angle and share spatial similarity (see Figure 11, left). The goal is to compress one of the images in each pair, treating the other image as side information only the decoder can access. The performance of a compressor is evaluated by its rate-distortion points, where the rate is measured in terms of bits-per-pixel and distortion is measured in terms of Peak Signal-to-Noise Ratio (PSNR) or Structural Similarity (MS-SSIM) [19]. These rate and distortion measures are commonly reported in the image compression literature. More specifically, for an image \( I \), its reconstruction has PSNR\(_I\) := \( 10 \log_{10}(\text{MAX}_I^2 / \text{MSE}_I) \), where \( \text{MAX}_I \) is the maximum possible pixel value (which is 255 for an 8-bit representation) and \( \text{MSE}_I \) is the Mean-Squared Error (MSE) between each pixel in the reconstruction and \( I \). The PSNR metric has the same semantic meaning as the MSE metric, but is expressed on a decibel scale. The MS-SSIM metric is a similarity metric between two images; in our case, we measure the similarity between a ground-truth image and its reconstruction. The exact definition is deferred here due to its complexity compared to PSNR; however, the key idea is that it quantifies the perceptual change in structural information (where pixels rely on their neighbors to create specific structures) between two images. In contrast, PSNR assesses the absolute change between the images. The MS-SSIM metric tends to agree with human perception of distortion better than PSNR [36], [45].

We train a conditional VQ-VAE for each target rate as described in Section III-B. We then evaluate its distortion averaged over the test set and plot the rate-distortion point. The exact details of how the dataset is constructed, as well as the model hyperparameters, are included in the Appendix (Section A-A).

2) Results: We compare our methods to existing distributed image compression methods DSIN [36], NDIC [25], and NDIC-CAM [26]. While the authors of NDIC-CAM did not report PSNR in their paper, we used their code to train their model with an MSE objective for a fair comparison. We report the \( \alpha, \beta, \lambda \) hyperparameters we used to train their model in the Appendix (Section C-B).

In Figure 3(b), we see that our method remains competitive compared to NDIC-CAM, which currently achieves the best results for MS-SSIM. However, in Figure 3(a), our method outperforms all baselines in PSNR for rates above 0.1 bpp and is competitive with NDIC-CAM for rates below 0.1 bpp. These results suggest that our approach is amenable to training with different distortion functions while maintaining state-of-the-art reconstruction performance. In Figure 12 in the Appendix, we show Figure 3 but with LDMIC [37] included. The setting of LDMIC is different than the remaining methods (including ours) because \( y \) is presented to the decoder after being lossily compressed, whereas the other methods have access to \( y \) directly. Therefore, we defer the result to the Appendix.

In addition, our model is significantly more parameter-efficient compared to NDIC and NDIC-CAM. Table 1 shows the parameter count for the models used to generate the points in Figure 3(a). As shown, our models are smaller by a factor of up to approximately 8\( \times \).

B. Handling Complex Correlation

1) Setup: To further investigate how well our method can handle complex correlations between the input and side information, we evaluate our method on a challenging distributed compression setup. First, we create a dataset of correlated images from 256 \( \times \) 256 CelebA-HQ dataset [46] containing images of celebrity faces. Each image is vertically split into top and bottom halves, where the top half is used as the input, and the bottom half is used as side information (see Figure 11, right). Following [47], we use 27000/3000 split between train/test data.

While there is clearly some correlation between the top and bottom halves of an image of a human face, modeling this correlation (e.g., the conditional distribution over the top half of a face given the bottom half) is highly nontrivial. This experiment is thus designed to show our model’s ability to leverage this correlation to improve compression.

2) Baseline VQ-VAEs: To analyze the gains from distributed compression, we train three different variants of VQ-VAEs: distributed (our method), joint, and separate. In the joint model, both the encoder and decoder have access to the side information, as depicted in Figure 1. This serves as a proxy to the intractable theoretical rate-distortion bound established by [13] for lossy distributed compression. We expect this to be the upper limit on the performance of our method. The separate model is positioned at the other extreme, where neither the encoder nor the decoder uses the side information. This serves as the lower limit on the performance of our

| \text{Parameter Count (M)} | \text{Factor} |
|---------------------------|-------------|
| Ours                      | 3.9 – 7.4   | 1.0-1.9\( \times \) |
| NDIC (Ball\( \text{\texttt{e17}} \)) | 16.3       | 4.2\( \times \) |
| NDIC (Ball\( \text{\texttt{e18}} \)) | 25.0       | 6.4\( \times \) |
| NDIC-CAM                  | 29.7       | 7.6\( \times \) |
method. To ensure a fair comparison among these variants, they have identical architecture and number of parameters for the autoencoder portion and only vary in the way they handle the side information. Architectural details and network hyperparameters are provided in the Appendix (Section A-A).

3) Results: In Figure 4, we see that the distributed VQ-VAE achieves nearly identical performance as the joint VQ-VAE, further proving the viability of our method as a distributed coding scheme.

4) Effect of Side Information: It is possible that the distributed VQ-VAE learns to ignore the side information, effectively collapsing to a separate VQ-VAE. We investigate whether the distributed decoder actually uses the side information by intentionally providing incorrect input.

We show in Figure 5 that the side information plays a significant role in the quality of reconstruction. For example, providing side information with a different face leads to the reconstruction having a matching skin tone that is different from the original. As expected, side information has no effect for the separate encoder.

C. Communication Constrained Distributed Optimization

1) Background: Here, we consider an interesting proof of concept that applies our method to distributed training of a neural network $h_\theta$ with parameters $\theta \in \mathbb{R}^d$. With the increasing size of deep learning models, an important bottleneck in distributed training is the repeated communication of gradients between workers and the parameter server. Many gradient compression techniques have been developed to alleviate this issue, and here we demonstrate that our approach can be used and it performs well. To alleviate this issue, many gradient compression techniques have been developed.

A key observation we make is that the gradients coming from different workers are correlated. This suggests that individual workers may be able to compress their gradients via distributed source coding without having to communicate with each other. We compare this approach to a representative subset of baselines, which we describe in Section IV-C5. While our approach is not ready for practical deployment, we observe a vastly improved training performance, suggesting a promising direction for future research.

2) Experimental Setup: We focus on a simple setup where two workers $j = 1, 2$ are assigned a subset $X_j \in \mathcal{X}$ of the training data. The workers then compute gradients locally using SGD-like iterations and communicate the gradients back to a central parameter server. At each iteration $t$, the correlation between the client gradients $g^1_t$ and $g^2_t$ can be exploited to improve compression performance further. Concretely, we treat $g^1_t$ as side information, train a distributed encoder for $g^2_t$, and show that the cost of communicating $g^2_t$ can be substantially reduced.

The neural network $h_\theta$ being trained using compressed gradients is a small convolutional network for MNIST digit classification [48]. We partition the MNIST training data $\mathcal{X}$ into two equal subsets: $\mathcal{X}_{\text{pre}}$ and $\mathcal{X}_{\text{train}}$. $\mathcal{X}_{\text{pre}}$ is used to train the VQ-VAE encoder and $\mathcal{X}_{\text{train}}$ is used to train $h_\theta$ using the trained encoder as gradient compressor. We measure the performance of our method using two metrics: (a) rate-distortion of the distributed compressor, and (b) classification accuracy of the model $h_\theta$ trained using gradients compressed by our VQ-VAE-compressed gradients.

3) Generating Training Data for VQ-VAE Gradient Compressors Encoder: We train $h_\theta$ for $T$ steps using the Adam optimizer across two nodes over $\mathcal{X}_{\text{pre}}$. This generates a sequence of $T$ gradient pairs $\{g^1_t, g^2_t\}_{t=1}^T$ which can be used to train the VQ-VAE gradient compressors encoders.

However, applying our method to this setting naively leads to a suboptimal VQ-VAE as these gradient pairs are highly correlated across training steps. Noticing that $g^1_t$ and $g^2_t$ are conditionally independent given the initial model weights and the time step $t$, we train $h_\theta$ for multiple runs with different initialization for $\theta$, while randomly sampling a subset of gradients from each run. Thus, we generate a dataset of tuples $(t, g^1_t, g^2_t)$ sampled from multiple independent runs. We also update the architecture of the VQ-VAE so that both the encoder and decoder are conditioned on $t$. This way, the gradients become i.i.d. samples over random runs and time steps.

4) Pre-Training VQ-VAE Gradient Compressors Encoders: We first compare the performance of three different VQ-VAE variants (distributed, joint, and separate, as depicted in Figure 1) over the course of a single training run of $h_\theta$. As shown in Figure 6(a), we observe a substantial improvement in $\ell_2$ distortion for the distributed VQ-VAE, compared to the separate model. As training progresses, the gap between
nearly match the performance of its joint counterpart in terms of sufficiently large distributed VQ-V AE compression. We observe that the distributed encoder leads to comparable accuracy as the joint encoder, but using about only half of the communication cost. Shaded regions represent standard error. (c) Distributed training performance with compressed gradients using a distributed encoder in comparison to other baseline methods in terms of test set accuracy. Note that standard error is very small and barely visible, and may require zooming in on an electronic copy of this manuscript.

V. DISCUSSION AND ANALYSIS

In this section, we further provide a discussion and analysis of our framework. Namely, we first analyze the effect of reconstruction diversity as we vary side information. Next, we perform a sanity check to see whether our approach is indeed performing DSC by comparing it with the optimal distributed compressor. Finally, we compare the performance of our approach with DISCUS [15] in a setting with a simpler correlation structure between the source and side information.

A. Role of Side Information

In Section II-A, we saw that grouping symbols with maximal distance from each other in the same group allows the decoder to determine the correct symbol using the side information. Here, we investigate to what extent our models perform such a grouping (also known as binning). If approximate binning were occurring, the same codeword (i.e., bin index) would be decoded into different images for different side information. In other words, a distributed compressor should have high reconstruction diversity for a single codeword as we vary the side information. We refer to this as bin diversity.

On the other hand, a model that does not perform binning should decode a single codeword to similar symbols regardless of what side information is given to the decoder. To test this hypothesis, we train two different models: Ours was trained with the correct side information as was done for other experiments, and Uncorrelated SI was trained by replacing the side information with random ones from other irrelevant samples in the dataset, thus making the input and side
information completely independent. We expect this model to ignore side information and achieve lower bin diversity.

To quantitatively measure diversity, we used average pairwise distance with respect to $\ell_2$ norm and LPIPS distance [56], which has been used in the literature [57], [58] as a measure of sample diversity. Both metrics were computed and averaged over all images in the CelebA-HQ test dataset.

As shown in Table II, the VQ-VAE trained with correlated side information exhibits much higher bin diversity in both metrics. In other words, the images within each bin are much farther from each other compared to the other model, suggesting that some form of approximate binning is happening. The uncorrelated VQ-VAE has particularly low bin diversity with respect to the LPIPS distance, meaning there is very little perceptual difference regardless of what side information is used.

### B. Performance of the Learned Compressor on Synthetic Sources

Another natural question is whether our learned distributed compressor is performing DSC and strictly better than the joint compressor. We investigate this using a synthetic data source studied in [15] with two correlated Gaussian sources: $Y = X + N$, $X \sim N(0, 1)$, and $N \sim N(0, 0.1^2)$. The asymptotically optimal (i.e., in the limit of compressing infinitely many symbols together) rate-distortion curve of these sources is known analytically, so we can check how our model compares to the theoretical limit.

Figure 7 shows that both learned methods significantly outperform the asymptotically optimal encoder without side information (SI) for low rates. This is a concrete evidence that our learned encoder is actually performing DSC (as opposed to simply achieving a very good compression rate for single source coding). Moreover, the distributed compression performance remains very close to that of joint compression depicted in Figure 1, showing the efficacy of our practical distributed coding scheme.

At higher rates, however, the value of side information quickly diminishes, and the learned methods perform worse than even the encoder without SI. This is expected as the optimal curves presented in this figure are asymptotic in the limit of jointly compressing infinitely many symbols, which is clearly not the case for the learned methods that compress each symbol one at a time.

We also note that a recent work by Ozyìlkàn, Ballé, and Erkip has shown that the distortion of learning-driven DSC can be further improved by explicitly optimizing the entropy and tailoring the design to scalar sources such as Gaussian and Laplacian [38]. Interestingly, they show that neural distributed compressors bin their source data without explicitly training for it and can nearly attain the asymptotic Wyner-Ziv rate-distortion bound for scalar Gaussian and Laplacian sources. In contrast, our architectural design and training methodologies are tailored for high-dimensional data sources with complex distributions (but still utilize side information for Gaussian sources as shown in Figure 7).

### C. Comparison With Non-Learning Baselines

1) Setup: We consider a simple but systematic correlation structure: correlated i.i.d. Bernoulli sources [15]. Specifically, we consider a pair of source sequence and side information $(x_i, y_i)_{i=1}^{648}$, where $y_i \oplus x_i$ is an i.i.d. Bernoulli random variable.

We compare the performance of our Neural DSC approach with the Distributed Source Coding Using Syndromes (DISCUS), a non-learning based constructive scheme that is designed specifically for this setup and shown to be near optimal [15]. Our experimental results are shown in Figure 8. As expected, our neural DSC performs worse than DISCUS (labeled as LDPC); however, the performance of distributed neural DSC is comparable with the performance of joint source coding, which serves as a sanity check.

2) Interpretation: Unlike other experiments, here we focus on a systematic correlation structure, where the problem at hand is learning a complex algorithm; there is no complexity in the data distribution or correlation structure. In Figure 8,
we plot the block error rate of our method in the joint and distributed cases and compare it with a DISCUS baseline. We implemented the DISCUS [15] using a rate 3/4 Low-Density Parity Check (LDPC) code, specified in IEEE 802.11 [59], [60], which maps 486 message bits to a length-648 codeword. The parity-check matrix is created using the ldpcQuasiCyclicMatrix function, with the block size 27 and prototype matrix P provided in Appendix D.

These results show that even in such scenarios, our method learns a nontrivial compression algorithm (i.e., a compression scheme without side information achieves the block error rate $\approx 1$). We note, however, that our neural DSC results in higher error rates than the DISCUS algorithm. This observation suggests that learning DISCUS, or an alternative method that outperforms DISCUS, is a highly non-trivial task. This is perhaps not surprising given that learning channel codes using neural networks is, in general, highly challenging [61], [62], [63] and typically requires a tailored selection of neural architectures and training methodologies, which is beyond the scope of this paper.

VI. CONCLUSION

We presented Neural DSC, a framework for distributed compression of correlated sources. Our method is built on the power of modern deep generative models, namely VQ-VAEs, which are excellent data-driven models to represent high-dimensional distributions using compressed discrete latent codes. Our training objective is justified in part by the connection we establish between distributed source coding and the modified ELBO for asymmetric VAEs.

Empirically, we show that our method achieves the new state-of-the-art performance in terms of PSNR on stereo image compression. Our approach remains competitive with the current best results for MS-SSIM. We also show that our method is able to leverage complex correlations far beyond spatial similarity for better compression performance, approaching the joint compression rate. Finally, we show that our method is not limited to images and show a promising proof of concept for compressing gradients for distributed training.

We believe our work provides encouraging initial results for practical data-driven DSC. We hope to see further developments that will bring insights on DSC pioneered by Berger, Tung, Slepian, and Wolf, and further enriched by decades of research in information theory, to practical applications – with the aid of learning and a data-driven approach.

Some specific future directions include exploring a hybrid approach that combines the DISCUS algorithm with neural non-linear representation learning would be interesting. Such an integration could harness both methodologies’ strengths while simplifying the overall complexity. Additionally, our research could be extended to scenarios involving more than two distributed encoders collaborating to compress correlated data for a single decoder. Furthermore, we encourage exploring scenarios where the decoder’s objective shifts from reconstructing the raw correlated sources to reconstructing a function derived from these correlated sources. This scope expansion will open new avenues for applying DSC to a broader range of real-world problems.

APPENDIX A

ARCHITECTURAL DETAILS

A. VQ-VAE Architecture

For our image compression experiments, we used a convolutional VQ-VAE architecture similar to the one used in [31] with residual connections.

Both the input and the side information to this network have the shape $3 \times 128 \times 256$ (i.e., vertical halves of a full $3 \times 256 \times 256$ image). The encoder scales down the input image by a factor of $4 \times$ or $8 \times$ both vertically and horizontally, producing latent variables of shape $32 \times 64$ and $16 \times 32$, respectively. Each dimension of the latent variable is allotted different numbers of bits (i.e., codebook bits), which range from 1 to 8 in our experiments. The decoder conversely takes a discrete latent variable and upscales it by a factor of $4 \times$ or $8 \times$ to produce a reconstruction of the original shape.

A detailed specification of the architecture is provided in Figure 9. The notation $(\text{Tconv}) \ "\text{Conv} \ A \times B \ (C \rightarrow D)"$ represents a 2D (transposed) convolution with kernel size $A$ and stride $B$ with input and output channels $C$ and $D$, respectively. The boxes “Residual Block $(A \rightarrow B)$” represent a two-layer residual network. We used GELU (Gaussian Error Linear Unit) activation for all layers except for the very last convolution of the decoder, for which we used sigmoid. $\oplus$ represents the channel-wise concatenation of the image representations. The exact number of channels may differ for different rate-distortion points, and we refer the reader to the supplementary code submission for full details.

We used a small network (denoted “SI Net”) shared by the encoder and decoder to preprocess the side information. Whenever the encoder or decoder does not receive side
information (for distributed and separate VQ-VAEs), we simply replace the output of SI Net with a zero tensor of the same shape. On the decoder side, the output of the SI Net is passed through the Cond Net; the architecture of this network is nearly identical to the encoder. The architecture of the SI Net and the Cond Net are detailed in Figure 10.

We note that all three VQ-VAE variants have the same architecture and number of parameters for the auto-encoder portion (the portion that the horizontal line goes through in Figure 9). Thus, there is no architectural advantage among the VQ-VAE variants.

### B. Specifying the Target Rate

The shape and the number of codebook bits determine the total rate. This is different from neural compression models that use a regular VAE, which controls the rate-distortion by reweighting the training objective (ELBO). As mentioned in Section III-B, the fact that we have a hard constraint on the rate is useful when working with a hard communication limit.

There are several ways to achieve a desired target rate. For example, an 8× downscaling encoder with 4 codebook bits produces a compressed message of size 16 × 32 × 4 = 2048 bits. The same rate can be achieved using a 4× downscaling encoder with 1 codebook bit: 32 × 64 × 1 = 2048 bits. This flexibility is what allows us to have a fairly granular control over the target rate at the cost of hyperparameter choices.

#### Hyperparameters for KITTI Stereo Experiment

Due to the above choices, we use differently sized networks for each rate-distortion curve for our stereo image compression experiments. While the model definition is included in the code, we include the relevant hyperparameter information in Table III below.

#### Hyperparameters for CelebA-HQ Experiment

For this experiment, we used the single 8× downscaling encoder with codebook bits {1, 2, 3, 4}, resulting in the total rates of [512, 1024, 1536, 2048].

#### Distributed optimization

For gradient compression VQ-VAEs, we followed the same architecture as the image compression experiments but replaced all convolutional layers with fully-connected layers. Full specification of the network is available at https://github.com/acnagle/neural-dsc.

## Appendix B

### Stereo Image Dataset

For our stereo image compression experiment, we follow the setup of [25]. First, we construct training/test datasets using the files specified in the official repository for [36] (https://github.com/ayziksha/DSIN/tree/master/src/data_paths). Then we apply the same preprocessing steps of [25], where we first take the center crop of size 370 × 740, then resize it to 128 × 256 using PyTorch transformations. This results in the training split containing 1576 pairs of stereo images and the test split containing 790 pairs. In Figure 12, we show the performance of LDMIC [37] with two sources against all other methods. For LDMIC, we vary the compression rate for
FIG. 12. Recreation of Figure 6 from our paper with results from LDMIC [37] on KITTI Stereo. We measure the distortion with PSNR (a) and MS-SSIM (b).

TABLE IV
DIFFERENT VQ-VAE MODELS AND THE ASSOCIATED HYPERPARAMETERS FOR THEIR LATENT PRIOR MODEL COUNTERPARTS FOR THE STEREO IMAGE COMPRESSION EXPERIMENT. EACH ROW REPRESENTS A SINGLE RATE-DISTORTION POINT IN FIGURE 3(A)

| Downscaling Factor | Codebook Size (bits) | Rate (bpp) with Latent Prior | # Attn Heads Per Block | # Blocks | Batch Size | Parameter count |
|-------------------|----------------------|------------------------------|------------------------|----------|------------|-----------------|
| 8x                | 1                    | 0.0110                       | 4                      | 4        | 128        | 859,394         |
| 8x                | 2                    | 0.0233                       | 4                      | 4        | 128        | 859,908         |
| 8x                | 3                    | 0.0364                       | 4                      | 4        | 128        | 860,936         |
| 8x                | 8                    | 0.0978                       | 4                      | 4        | 16         | 924,672         |
| 4x                | 3                    | 0.1262                       | 4                      | 4        | 16         | 1,057,544       |
| 4x                | 6                    | 0.2545                       | 4                      | 4        | 16         | 1,071,936       |
| 2x                | 3                    | 0.3869                       | 2                      | 2        | 5          | 1,447,432       |
| 2x                | 4                    | 0.5395                       | 2                      | 2        | 5          | 1,449,488       |

one of the two views, denoted by $x$, (Rate (bpp) on the x-axis) and fix the compression rate for the other one, denoted by $y$, as 24 bits-per-pixel so that LDMIC can losslessly compress the second view $y$. In principle, the LDMIC for two image views should recover our setup. However, the results in Figure 12 demonstrate that the performance of LDMIC when configured in this way is generally much worse than our approach and other neural distributed source coding schemes designed for compression with side information.

APPENDIX C
TRAINING DETAILS

We trained our models for up to 1000 epochs on a DGX machine for stereo image compression experiments. Some training runs were early stopped because the validation performance started to plateau. For CelebA-HQ experiments, we trained the VQ-VAEs for a total of 20 epochs distributed over two Nvidia GTX 2080 GPUs. We trained the fully connected VQ-VAEs for 500 epochs on a single GPU for gradient compression experiments.

We evaluated validation loss after each epoch in all cases and observed no overfitting. This leads us to believe that it may be possible to further improve the performance of our method by training a larger network, which we leave for future work.

The latent prior models were implemented as an autoregressive transformer decoder network. All latent prior models were trained with an initial learning rate of 3e-4 with a cosine annealing learning rate decay. Training lasted 100 epochs and the learning rate had a linear warmup period for the first 2000 steps of training. The Adam optimizer was used with parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e - 8$. The hyperparameters for each latent prior model are provided in Table IV. The models were trained on an RTX 3090 GPU, and due to memory constraints some of the latent prior models had to be trained with fewer blocks, fewer attention heads, and a smaller batch size.

A. NDIC Training Details

For the CelebA-HQ compression experiment, we trained NDIC [25] using the official code released by the authors. We used the “Balle18” [24] backbone and trained the model for 10 epochs ($\approx 300K$ examples). We used the model checkpoint with the best validation set performance for evaluation. While [25] report training with a batch size of 1 for 500K steps, we chose to use a batch size of 20. This was done for several reasons. First, using a batch size of 1 is very inefficient as we do not benefit from GPU parallelism. It also leads to high variance in the gradient, often leading to slower convergence. In our training, the loss plateaued well before reaching 300K total examples.
TABLE V
HYPERPARAMETERS USED TO TRAIN NDIC-CAM [26] ON THE PSNR OBJECTIVE. EACH ROW REPRESENTS A SINGLE RATE-DISTORTION POINT IN FIGURE 3(A)

| α   | β   | λ        | Rate (bpp) | PSNR  |
|-----|-----|----------|------------|-------|
| 0   | 0   | 3 x 10^-5 | 0.00116    | 17.533|
| 0   | 0   | 6 x 10^-5 | 0.00408    | 19.644|
| 0   | 0   | 1.1 x 10^-4 | 0.00814   | 20.749|
| 0   | 0   | 1.8 x 10^-4 | 0.01649    | 21.558|
| 0   | 0   | 3 x 10^-4  | 0.02505    | 22.227|
| 0   | 0   | 8 x 10^-4  | 0.07297    | 23.882|
| 0   | 0   | 2 x 10^-3  | 0.18738    | 25.662|
| 0   | 0   | 3 x 10^-5  | 0.25674    | 26.752|
| 0   | 0   | 5 x 10^-3  | 0.38164    | 28.002|
| 0   | 0   | 8 x 10^-3  | 0.48933    | 28.963|

B. NDIC-CAM Training Details
We report the hyperparameters used to train the NDIC-CAM method on the KITTI Stereo dataset while optimizing for PSNR. All models were trained for 1000 epochs, at which point the models were well-converged. The hyperparameters and their corresponding results are shown in Table V. The results shown for MS-SSIM are reported from the NDIC-CAM paper [26].

C. LDMIC Training Details
We train all models following the hyperparameters used in the paper which proposes the LDMIC framework. For MSE distortion, we train a model for each \( \lambda \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\} \) with an initial learning rate \( 10^{-4} \) and train for 400 epochs while decaying the learning rate by a factor of two for every 100 epochs of training. We use these models’ weights to initialize the models’ weights trained with MS-SSIM distortion and train for an additional 400 epochs with an initial learning rate \( 5 \times 10^{-5} \). We use a batch size of 32 for all training runs.

D. Neural DSC Training Details for i.i.d. Bernoulli Experiments
A schematic of the VQ-VAEs trained in the joint and distributed settings is shown in Figure 13. A primary difference between this architecture and the architecture for the other experiments is that the output of the decoder of this architecture has a learnable parameter \( \alpha \), which is used to interpolate the reconstructed sequence and the side information linearly, i.e., the output of the network is \((1 - \alpha)\hat{x} + \alpha y\), where \( \hat{x} \) is the reconstruction and \( y \) is the side information. This enables the VQ-VAE architecture to find a tradeoff between the reconstruction and the side information as the final reconstruction of the Bernoulli sequence. Our VQ-VAE architectures were trained on i.i.d. Bernoulli sequences of length 648, and the block error rate was computed on a fixed validation set of 10,000 i.i.d. Bernoulli sequences. All models were trained with the same hyperparameters: 128 batch size, 3e-4 learning rate, 0.15 commitment cost (for the VQ-VAE loss), and 10 epochs. The Adam optimizer was used with parameters \( \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8} \).

APPENDIX D
DISCUS DETAILS
For the experiment on the correlated i.i.d. Bernoulli sources Figure 8, we used the following prototype matrix \( P \) for the quasi-cyclic LDPC code.

\[
P^T = \begin{bmatrix}
16 & 25 & 25 & 9 & 24 & 2 \\
17 & 12 & 18 & 7 & 5 & 2 \\
22 & 12 & 26 & 0 & 26 & 19 \\
24 & 3 & 16 & 1 & 7 & 14 \\
9 & 3 & 22 & 17 & 1 & 24 \\
3 & 26 & 23 & -1 & -1 & 1 \\
14 & 6 & 9 & -1 & -1 & 15 \\
-1 & 21 & -1 & 7 & 15 & 19 \\
4 & -1 & 0 & 3 & 24 & -1 \\
2 & 15 & -1 & -1 & 15 & 21 \\
7 & 22 & 4 & 3 & -1 & -1 \\
-1 & -1 & -1 & 23 & 8 & 2 \\
26 & 15 & 4 & -1 & -1 & -1 \\
-1 & -1 & 16 & 13 & 24 & 2 \\
4 & 2 & 8 & -1 & -1 & -1 \\
-1 & -1 & 12 & -1 & 13 & 3 \\
21 & -1 & 11 & 21 & -1 & -1 \\
-1 & 16 & -1 & -1 & 11 & 2 \\
1 & -1 & -1 & 0 & -1 & 1 \\
0 & 0 & -1 & -1 & -1 & -1 \\
-1 & 0 & 0 & -1 & -1 & -1 \\
-1 & -1 & 0 & 0 & -1 & -1 \\
-1 & -1 & -1 & 0 & 0 & -1 \\
-1 & -1 & -1 & 0 & 0 & -1
\end{bmatrix}
\]
[54] D. Basu, D. Data, C. Karakus, and S. Diggavi, “Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 14668–14679.

[55] S. Shi et al., “A distributed synchronous SGD algorithm with global top-k sparsification for low bandwidth networks,” in Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS), 2019, pp. 2238–2247.

[56] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 586–595.

[57] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, “VEEGAN: Reducing mode collapse in GANs using implicit variational learning,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 3310–3320.

[58] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, “Unrolled generative adversarial networks,” 2016, arXiv:1611.02163.

[59] “IEEE Draft Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standards, P802.11-REVmd/D1.0, Feb. 2018.

[60] “LDPC encoding and LDPC quasicyclic Codeword.” 2021. Accessed: Jan. 25, 2023. [Online]. Available: https://www.mathworks.com/help/comm/ref/ldpcencode.html

[61] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, “Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 1–19. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/228499b55310264a8ea0e27b6e7c6ab6-Paper.pdf

[62] A. V. Makkuva, X. Liu, M. V. Jamali, H. Mahdavifar, S. Oh, and P. Viswanath, “KO codes: Inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning,” in Proc. 38th Int. Conf. Mach. Learn., 2021, pp. 7368–7378. [Online]. Available: https://proceedings.mlr.press/v139/makkuva21a.html

[63] M. V. Jamali, H. Saber, H. Hatami, and J. H. Bae, “ProductAE: Toward training larger channel codes based on neural product codes,” in Proc. IEEE Int. Conf. Commun., 2021, pp. 3898–3903. [Online]. Available: https://api.semanticscholar.org/CorpusID: 238582716

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.