# Relaxed-Responsibility Hierarchical Discrete VAEs

**Matthew Willetts**[1,2] MWILLETTS@TURING.AC.UK
**Xenia Miscouridou**[1,2] XMISCOURIDOU@TURING.AC.UK
**Stephen Roberts**[2,3] SJROB@ROBOTS.OX.AC.UK
**Chris Holmes**[1,2] CHOLMES@STATS.OX.AC.UK

[1] *Department of Statistics, University of Oxford*
[2] *Alan Turing Institute, London*
[3] *Oxford-Man Institute, University of Oxford*

## Abstract

Successfully training Variational Autoencoders (VAEs) with a hierarchy of discrete latent variables remains an area of active research. Leveraging insights from classical methods of inference we introduce *Relaxed-Responsibility Vector-Quantisation*, a novel way to parameterise discrete latent variables, a refinement of relaxed Vector-Quantisation. This enables a novel approach to hierarchical discrete variational autoencoder with numerous layers of latent variables that we train end-to-end. Unlike discrete VAEs with a single layer of latent variables, we can produce realistic-looking samples by ancestral sampling: it is not essential to train a second generative model over the learnt latent representations to then sample from and then decode. Further, we observe different layers of our model become associated with different aspects of the data.

## 1. Introduction

Probabilistic deep generative models have had significant and continuing success in learning continuous representations of data [1–5]. The learning of discrete representations has had recent successes [6–8] and remains an area research. Discrete representations are useful as they are compact and find application in various tasks such as compression and clustering. Advances in differentiable continuous relaxations of discrete probability distributions [9, 10] have contributed in model training containing discrete latent variables on high dimensional data using gradient-based methods [11]. However, learning rich hierarchical models with discrete latent variables for high-dimensional data remains an outstanding problem in the field [12, 13].

Here we propose an effective, scalable method for learning hierarchical discrete representations of data within a probabilistic unified framework. This work builds on both Vector-Quantised Variational Autoencoders (VQ–VAEs) [6] and their relaxation [11], developing a novel variety of hierarchical discrete Variational Autoencoders (VAEs). Previously developed hierarchical structures based around these building blocks have required various heuristics in model formulation and training [13]. With a new formulation of probabilistic Vector-Quantisation we train hierarchical discrete latent variable models end-to-end within a unified probabilistic framework. These models, which we call *Relaxed-Responsibility Vector-Quantised VAEs* or RRVQ–VAEs, have a hierarchical structure that lessens the need for post-hoc training of a density estimator over learnt embeddings to be used as a prior when sampling, as done in VQ–VAEs and their extensions [6, 7]. We find our models have superior performance when compared to VQ–VAE baselines in several ways. They achieve lower *bits per dim* (a scaled form of the ELBO) when evaluated using the initial prior, and their draws have higher fidelity as measured by Fréchet Inception Distance (FID) [14], when compared to VQ–VAE baselines in which sampling is done from the prior used in training. Further, this approach opens up new avenues for the building of hierarchical discrete VAEs. We consider this paper a step towards a unified probabilistic framework for specifying and training models of this type.

Our key contributions:

- A novel method of parameterising the discrete latent variables suitable for use in hierarchical discrete latent variable models, which we call *Relaxed-Responsibility Vector-Quantisation*.
- Using this method, we propose a novel variety of probabilistic hierarchical discrete VAE trained end-to-end that learn a clear hierarchy of representations and does not require a subsequently-trained prior.

## 2. Background: Vector Quantised Variational Autoencoders

The Vector-Quantised Variational Autoencoder (VQ–VAE) [6] is a density estimator for high dimensional data like audio, images and video. Instead of having continuous latent variables as in the vanilla VAE (see Appendix A), the latents $\mathbf{z}$ here are a set of $M$ discrete variables $\mathbf{z} = \{z^1, \dots, z^M\}$ each of dimensionality $K$. The joint $p_\theta(\mathbf{x}, \mathbf{z})$ factorises as for a vanilla VAE, but with $p(\mathbf{z}) = \prod_{m=1}^{M} \text{Cat}(z^m | \frac{1}{K})$.

The likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ does not depend directly on samples of $\mathbf{z}$. Rather the discrete vector $\mathbf{z}$ is used to index over a dictionary of $K$ embeddings, the codebook vectors $\mathbf{E} = \{\mathbf{E}^k\}$, each $\mathbf{E}^k \in \mathbb{R}^{d_E}$. For stochastic amortised variational inference in VQ–VAEs, introduce a recognition network $\mathbf{e}_\phi(\mathbf{x}) \in \mathbb{R}^{M \times d_e}$ outputing $M$ vectors in $\mathbb{R}^{d_e}$, the embedding space. The posterior $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{m=1}^{M} q_\phi(z^m|\mathbf{x})$ is then defined via a nearest neighbour vector lookup. For each latent $z^m$,

$$q_\phi\left(z^m = k|\mathbf{x}\right) = \begin{cases} 1 & \text{if } k = \text{argmin}_j \left|\mathbf{e}_\phi^m\left(\mathbf{x}\right) - \mathbf{E}^j\right|_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

This is a one-hot posterior: $q_\phi(\mathbf{z}|\mathbf{x})$ is deterministic. In a vanilla VAE we train the model by maximising the ELBO, $\mathcal{L}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q} \log p_\theta\left(\mathbf{x}|\mathbf{z}\right) - \text{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)||p(\mathbf{z})\right)$, over the dataset with respect to the generative and recognition model parameters. This standard training approach is not appropriate for this discrete model, for two reasons. Firstly, since it is not possible to differentiate through the vector lookup operation (due to the argmax) we cannot use differentiable samples to take gradients through Monte Carlo estimates of the expectations. Secondly, the one-hot posterior makes the KL term constant (equal to $M \log K$) so there is no regularisation on the posterior representations.

Thus, to train a VQ–VAE one introduces two additional terms to the objective: first, a vector quantisation loss to train the embeddings; and secondly a commitment loss to help control the output of the embedding network, weighted by a chosen hyperparameter $\beta$ [6].

**Relaxed–VQ–VAEs** Instead of the deterministic posterior found in a vanilla VQ–VAE, a Gumbel-Softmax [9, 10] distribution can be used to specify a posterior distribution from which we can take differentiable samples [11] . This means that the posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is no longer a one-hot, deterministic distribution and the KL in $\mathcal{L}$ is no longer a fixed constant. Thus the VQ-related loss terms are no longer needed and the codebook can be learnt via gradient descent. One can choose the logits of the posteriors to be proportional to the square distance between the given embedding vector and each codebook vector [11]

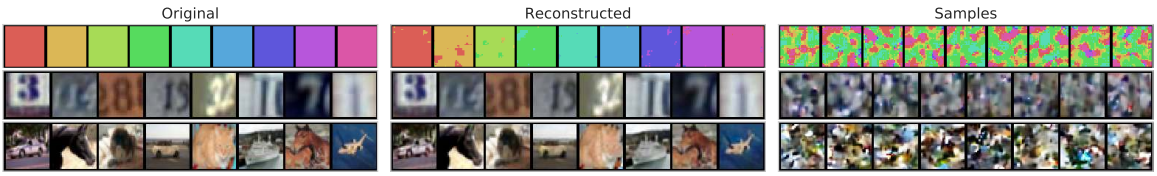
$$q(\mathbf{z}|\mathbf{x}) = \prod_{m=1}^{M} \text{Cat}\left(\mathbf{z}_m | \pi_\phi^m\left(\mathbf{x}\right)\right) \tag{2}$$

$$\pi_\phi^{m,k}(\mathbf{x}) \propto \exp\left(-\frac{1}{2}\left|\mathbf{e}_\phi^m(\mathbf{x}) - \mathbf{E}^k\right|_2^2\right). \tag{3}$$

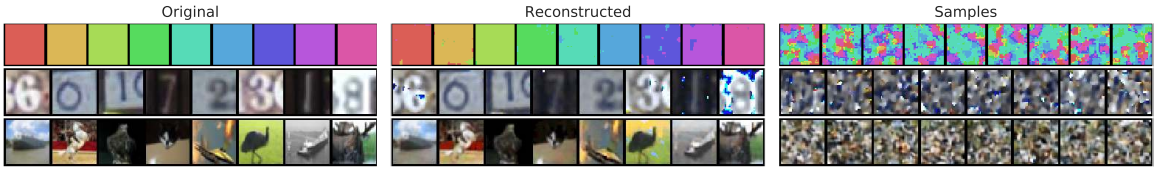
These *Relaxed*–VQ–VAEs have been shown to make better use of their latent variables than the deterministic base model, obtaining higher values of $\mathcal{L}$ [11]. Both Vanilla–VQ–VAEs and *Relaxed*–VQ–VAEs train relatively easily. By definition Vanilla–VQ–VAEs avoid posterior collapse [15–18] as the KL term in $\mathcal{L}$ is constant. For Relaxed–VQ–VAEs matching the posterior to the prior in the latent space is not possible in the general case as it would require the posterior embedding to be equidistant from all codebook vectors.

In this work we build on Relaxed–VQ–VAEs, not deterministic VQ–VAEs, both due to their demonstrated superior performance in maximising $\mathcal{L}$ and as they have truly probabilistic structure. In the rest of this paper, VQ–VAE is used both to refer generically to either Vanilla (ie, deterministic) VQ–VAEs and to Relaxed–VQ-VAEs, as their properties and behaviours are broadly similar. When needing to refer to them distinctly, we do so.

## 3. Sampling and Reconstructing in VQ–VAEs




(a) Relaxed–VQ–VAE




(b) Spatial–VAE

Figure 1: Here we demonstrate that the poor quality draws when sampling from a VQ–VAE's prior $p(\mathbf{z})$ is not from having discrete latents, but from the spatial arrangement of latent variables. We train (a) Relaxed–VQ–VAEs and (b) Spatial–VAEs (a VAE with continuous latents, but arranged spatially like a VQ–VAE) on (top) a toy dataset composed of 9 colour swatches, (middle) SVHN, (bottom) CIFAR-10. For each dataset, both models give good reconstructions (middle column) but ancestral samples from the prior $p(\mathbf{z})$ (right column) are very dissimilar to datapoints in the training set, even for the toy dataset – for which we do not see uniformly-coloured images, instead we see regions of each the different colours of the dataset. This shows that it is the method used to parameterise the model's latent variables that leads to this sampling phenomena, not being discrete vs continuous.

Here we focus on modelling square images, though the arguments we make can generalise to rectangular images, as well as to audio or video data. In VQ–VAEs, one uses convolutional neural networks to represent $p$ and $q$, laying out $\mathbf{z}$ as a square of side $\sqrt{M}$, mirroring the spatial structure of pixels in an image [6]. For audio one might choose a 1D structure, and 3D for video.

Interestingly, ancestral sampling from $p_\theta(\mathbf{z})$ in (relaxed or not) VQ–VAE models, gives draws that do not resemble the training data. Samples from this prior fail to capture the structure needed, correlations between the $M$ latents, necessary to produce realistic data when decoded. Meanwhile, even from early stages of training in VQ–VAEs the reconstructions of training data are of high fidelity. This is why in VQ–VAEs it is necessary to subsequently train a second density estimator, commonly

a large, powerful autoregressive model such as a PixelCNN [19, 20] over the latent representations to then sample from. This is followed in the two-layer extension of VQ–VAEs [7] too.

Conversely, in VAEs with continuous latent variables the reconstructions are generally found to be somewhat blurry, while samples tend to have some coherent structure. In a standard VAE with $p(\mathbf{z}) = \prod_{i=1}^{M} \mathcal{N}(z^m|0,1)$ the prior factorises over dimensions similar to how it does in a VQ–VAE, yet samples appear reasonable, which suggests that the reason is not only that.

We give an explanation for this phenomenon. It is not to do with discrete vs continuous latents at all, but rather with their neural parameterisation: In VQ–VAEs, convolutional neural networks are used to represent $p$ and $q$. With convolutionally-parameterised latents, each is tied spatially to be mostly concerned with a particular region of pixels in the input. This is unlike most implementations of vanilla VAEs, where the posterior's parameters, commonly the mean and diagonal covariance of a Gaussian, are output by MLPs. The learnt representations are thus intrinsically non-local, which in turn gives them the ability to learn easily the arrangement of parts and wholes in an image.

To demonstrate this, we train a simple *Spatial*–VAE where continuous-valued latent variables are arranged spatially, as in VQ–VAEs: $p_\theta(\mathbf{z}) = \prod_{m=1}^{M} \mathcal{N}(\mathbf{z}^m|\mathbf{0}, \mathbb{I})$ and $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{m=1}^{M} \mathcal{N}(z^m|\mu_\phi^m(\mathbf{x}), \sigma_\phi^m(\mathbf{x}))$, $\mathbf{z}^m \in \mathbb{R}^{16}$, with $p$ and $q$ convolutional networks each composed of 2 ResNet block with 32 channels, and the number of latents $M$ is the $1/4$ the number of pixels in the input. We also train an equivalent Relaxed–VQ–VAE, with embedding space dimensionality $d_e = K = 16$. We use SVHN, CIFAR-10, and (to make the effect most striking) a toy dataset containing images that are each uniform blocks of colours. See Fig 1 for the resulting reconstructions and samples for the three datasets for both models.

Embedding an image into the latent space for reconstruction is relatively easy. For the discrete model, the encoder learns to output embeddings $\mathbf{e}_\phi(\mathbf{x})$ to place high probability over the right codebook embedding for its local region of the image, and do so over the entire set of $M$ spatially-arranged latents. Similarly, the Spatial–VAE learns to place posterior probability over regions in the latent space appropriate for each latent position. However, when sampling from each model's prior, we end up with very mixed up generated images. Even for the toy dataset, the draws for both models are rainbow images where each patch of the image is separately given a random colour from the training set.

The poor quality of naive VQ–VAE draws is not intrinsically from having discrete latent variables, but from having discrete latent variables *that are arranged spatially having been parameterised using convolutional neural networks*. However, it is the choice to have spatial latent variables that provides high quality reconstructions. To get around this, one can train a powerful autoregressive model over samples from the aggregate posterior in $\mathbf{z}$. In Vanilla–VQ–VAEs the aggregate posterior is a sum of $\delta$ functions, so it resembles an empirical data distribution. Thus training a high-performance density estimator is reasonable, and provides realistic draws [6, 7]. In this manner of operation, the encoder-decoder networks can be viewed as methods for non-linear dimensionality-reduction, so that the density estimator can be trained in a lower-dimensional input space rather than trained on the raw data directly. While that is a proven and performative approach, our goal is how can we obtain some of the benefits of VQ–VAEs (high quality reconstructions, the desirable property of learning discrete representations, ease of training) without having a two-stage training process?

Here we develop ways to make discrete VAEs more expressive and flexible by adding hierarchical structure. This removes the two-stage training process, and also gives us the benefits of hierarchical representations – such as different layers learning different aspects of the data.

## 4. Relaxed-Responsibility Hierarchical Discrete VAEs

### 4.1 Hierarchical Discrete VAEs

To make a hierarchical discrete VAE, introduce $L$ layers of latent variables $\vec{\mathbf{z}} = \{\mathbf{z}_1, .., \mathbf{z}_L\}$. Note that $\mathbf{z}_\ell^m$ is the $m^{\text{th}}$ latent variable in the $\ell^{\text{th}}$ layer. We wish to have an autoregressive structure between layers. Reminiscent of the LVAE [3], but conditioning the data likelihood on all latents, we choose our generative model's factorisation to be

$$p_\theta\left(\mathbf{x}, \vec{\mathbf{z}}\right) = p_\theta\left(\mathbf{x}|\vec{\mathbf{z}}\right) p_\theta\left(\vec{\mathbf{z}}\right) = p_\theta(\mathbf{x}|\vec{\mathbf{z}}) p(\mathbf{z}_L) \prod_{\ell=1}^{L-1} p\left(\mathbf{z}_\ell|\mathbf{z}_{>\ell}\right) \tag{4}$$

where $p(\mathbf{z}_\ell|\mathbf{z}_{>\ell}) = \text{Cat}\left(\mathbf{z}_\ell|f_\ell^\theta(\mathbf{z}_{>\ell})\right)$ and $p_\theta(\mathbf{z}_L) = \text{Cat}\left(\mathbf{z}_L|\pi_\theta^L\right)$. Similarly, $q$ has an LVAE-like structure,

$$q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x}) \prod_{\ell=1}^{L-1} q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell}, \mathbf{x}). \tag{5}$$

The ELBO $\mathcal{L}(x)$ for this model is thus

$$\mathbb{E}_{\vec{\mathbf{z}}\sim q} \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) - \sum_{\ell=1}^{L-1} \mathbb{E}_{\mathbf{z}_{>\ell}\sim q} \text{KL}(q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell}, \mathbf{x})||p_\theta(\mathbf{z}_\ell|\mathbf{z}_{>\ell}) - \text{KL}\left(q_\phi(\mathbf{z}_L|\mathbf{x})||p(\mathbf{z}_L)\right). \tag{6}$$

This is directly analogous to hierarchical VAEs with continuous latent variables.

### 4.2 Relaxed-Responsibility Vector-Quantisation

Our first main contribution is a method of parameterising the generative model and approximate posterior for models containing vector-quantised discrete latents, which improves the ability of hierarchical models of this type to learn effectively. We call this method *Relaxed-Responsibility Vector-Quantisation* (RRVQ). We found that without these improvements models of this form had low performance and were often unstable during training, and that the two changes we propose are synergistic – working better together than either alone. For an ablation study of the changes we propose to $q$ and $p$, see Appendix B.

#### 4.2.1 Proposal for $q$

Vector-Quantisation has historic links to mixture models, mixtures of experts, and classical methods of inference. The exponential moving average method of updating the codebook in VQ–VAEs is closely linked to K-means [21]. Relaxed–VQ is linked to mean-field variational inference for a mixture of Gaussians: we can interpret the embedding codebook as recording the means of the cluster components, all having isotropic unit variance, and are a-priori equal in probability [22, Sec 10.2]. Eq (3) is equivalent to saying that the posterior at each position in $\mathbf{z}$ is equal to the cluster responsibilities for the embedding vector $\mathbf{e}_\phi(\mathbf{x})$ at that position.

We develop this link further, increasing the expressiveness of the parameterisation of the latents $\mathbf{z}_\ell$ in our hierarchical model, by relaxing the restriction that all components have unit isotropic covariance. We introduce a second codebook $\mathbf{E}_{\Sigma,\ell}$ for each layer, recording the diagonal covariance matrices of each component. The responsibilities then used for defining $\pi_{\phi,\ell}(\mathbf{e}_{\phi,\ell})$ are

$$\pi_{\phi,\ell}^{m,k}(\mathbf{e}_{\phi,\ell}) \propto \frac{1}{\sqrt{(2\pi)^{d_e}\, \mathbf{E}_{\ell,\Sigma}^k}} \exp\left(-\frac{1}{2\mathbf{E}_{\ell,\Sigma}^k} \left|\mathbf{e}_{\phi,\ell}^m - \mathbf{E}_{\mu,\ell}^k\right|_2^2\right), \tag{7}$$

where $m$ indexes over the latent positions, $k$ over the codebook entries, $\mathbf{e}_{\phi,\ell} \in \mathbb{R}^{M \times d_e}$, $\mathbf{E}_{\mu,\ell}$ is the codebook of means for the $\ell^{\text{th}}$ layer and $\mathbf{e}_{\phi,\ell}$ is the embedding-space output of a network taking the appropriate inputs for the current layer, as written in Eq (5).

Viewing VQ as a mixture-of-experts model [23], where each codebook embedding mean is a local expert, we can view this extension as allowing the neighbourhoods of different experts to be more diffuse or more concentrated. By learning $\mathbf{E}_{\Sigma,\ell}$, codebook embeddings with large associated diagonal covariance will have their means used preferentially when the model outputs embeddings $\mathbf{e}_{\phi,\ell}$ are far away from the codebook means, and those with small diagonal covariance will dominate at short ranges, being highly confident of being the appropriate expert when $\mathbf{e}_{\phi,\ell}$ is close.

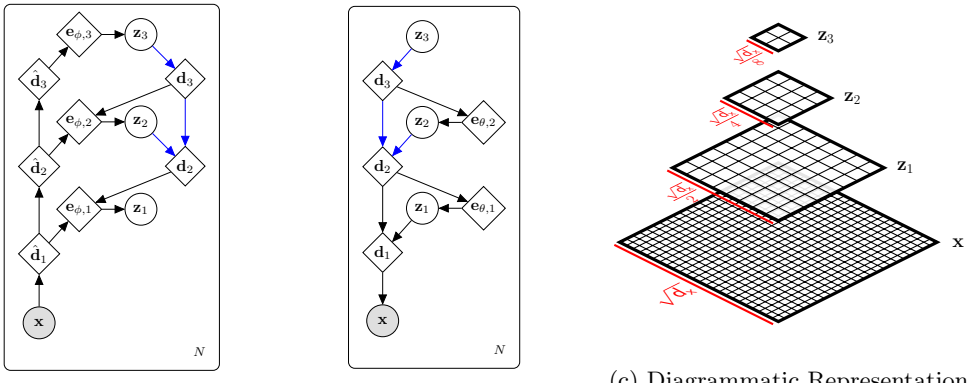
#### 4.2.2 Proposal for $p$

One obvious approach is to parameterise the (log) probabilities of $p_\theta(\mathbf{z}_\ell|\mathbf{z}_{>\ell})$ directly by a deep net. However, we found training to be unstable in hierarchical discrete VQ–VAEs that directly parameterised these conditional probabilities. What is a reasonable, flexible form for the generative model that will maintain stability of training while giving preserving performance?

Single-layer VQ–VAEs obtain realistic data by sampling from a discrete distribution parameterised using distances between a neural network's outputs and each of the embedding vectors, the equations for $q$, Eqs (3), now (7) if we are learning the covariances. Thus we choose to parameterise the conditional distributions in $p_\theta(\vec{\mathbf{z}})$ using a vector in the embedding space, along the same lines as for $q$, and using the same codebooks as for $q$. So, in the generative model each conditional distribution in $p_\theta(\vec{\mathbf{z}})$, rather than directly receiving the parameters needed to define it directly as the output from a deep net, we choose to have the probabilities parameterised using responsibilities given an embedding $\mathbf{e}_{\theta,\ell} \in \mathbb{R}^{M \times d_e}$ output by a deep net

$$\pi^{m,k}_{\theta,\ell}\left(\mathbf{e}_{\theta,\ell}\right) \propto \frac{1}{\sqrt{(2\pi)^{d_e}\,\mathbf{E}^k_{\ell,\Sigma}}} \exp\left(-\frac{1}{2\mathbf{E}^k_{\ell,\Sigma}}\left|\mathbf{e}^m_{\theta,\ell} - \mathbf{E}^k_{\mu,\ell}\right|^2_2\right). \tag{8}$$

### 4.3 Overall Model




(a) Variational Posterior (b) Gen. Model (c) Diagrammatic Representation

Figure 2: RRVQ–VAE with $L = 3$, (a) the variational posterior and (b) generative model, as defined in Eq (6). Blue arrows indicate shared networks. For simplicity the codebooks are not represented. (c) is a diagrammatic representation of the model, showing the spatial arrangement of latents, whose multiplicity we decrease by a factor of 4 each layer.

By combining Relaxed-Responsibility VQ with a hierarchical discrete VAE structure, we obtain our proposed model, a Relaxed-Responsibility Vector Quantised VAE (RRVQ–VAE). See Fig 2 for a graphical representation of this model. There is a deterministic chain in the inference network, the representations $\{\hat{\mathbf{d}}_\ell\}$. Similarly, there is a deterministic downwards chain of representations $\{\mathbf{d}_\ell\}$ in the generative model. These representations enable the conditional structure given in Eqs (4-5): that in the generative model we have an autoregressive structure over layers, and similarly that in the posterior each layer of latents is conditioned both on $\mathbf{x}$ and on those above it in the hierarchy. We choose to have a progressively smaller number of latent variables per layer as we ascend the hierarchy. If we continue decreasing the number until the top-most latent is a single discrete variable, it is reasonable for us to place a uniform categorical prior over it. Following continuous VAE models, including Ladder-VAEs [3], ResNet-VAEs [24] and BIVA [5], we enforce weight sharing between the generative and inference networks, indicted by blue arrows.

## 5. Related Work

VQ–VAEs have been extended to the two-layer case [7], with large, powerful autoregressive models subsequently trained as priors to then sample from, producing draws competitive with the state of the art when combined with a classifier-based accept-reject algorithm. Various recent works have worked towards hierarchical discrete VAEs that eschew the training of priors as auxiliary models. One recent paper trains layers of discrete latent variables in various hierarchical arrangements on MNIST and Fashion-MNIST [12], building on variational memory addressing methods [25]. In Hierarchical Quantised Autoencoders [13], much like in the original VQ–VAE paper, a sequential training pipeline is proposed. Here Relaxed–VQ–VAEs are trained one at a time, one after the other, with the first trained on the dataset and each subsequent sub-model trained on sampled values of the latents from the one below. This gives a Markovian structure, both in the generative and inference networks.

Methods have been developed to perform bits-back coding [26] using the learnt representations of VAEs [27], including for hierarchical VAEs [28]. In these methods the latents are continuous during training, with the space then subsequently discretised into buckets. Recently flow-based models [29–31] have been extended to handle discrete variables [32, 33].

As discussed in Section 4.2, Vector-Quantisation has close link to mixture models and mixtures of experts [23, 34]. Historically it has been known that stochastic relaxations of Vector-Quantisation offer various benefits compared to deterministic assignment, and that these methods are equivalent to certain classes of mixture models [34].

Vector-Quantisation can be thought of as inference on a Voronoi partition [35, Chpt 5]. In our model, our distributions are the responsibilities from a mixture model with learnt variances, so the deterministic version of RRVQ would result in Mahalanobis-distance Voronoi partitions.

## 6. Experiments

We train our model for image reconstruction on CIFAR-10, SVHN and CelebA, with $L = 5$ layers of latent variables. The models for CIFAR-10 and SVHN have identical specification, with some small changes for CelebA due to the different image size. We implement these models using fully convolutional networks composed of ResNet blocks. We use ResNets containing 4 blocks in the networks: that output $\mathbf{d}$ or $\hat{\mathbf{d}}$ representations; that map from $\mathbf{d}_{\ell+1}$ to $\mathbf{e}_{\theta,\ell}$ and from $\hat{\mathbf{d}}_{\ell+1}$ to $\mathbf{e}_{\phi,\ell}$; the decoder network that maps from $\mathbf{d}_1$ to $p_\theta(\mathbf{x}|\cdot)$.

We decrease the number of latent variables per layer as we ascend the hierarchy, as represented in Fig 2(c), such that for all datasets our latents $\vec{\mathbf{z}}$ are in 5 layers of size $\mathbf{M} = \{16\times16, 8\times8, 4\times4, 2\times2, 1\times1\}$. $\{\mathbf{E}_{\mu,\ell}, \mathbf{E}_{\Sigma,\ell}\}$. For CIFAR-10 and SVHN these each containing $K = 256$ codebook values $\in \mathbb{R}^{d_e}$, $d_e = 128$, per layer. For CelebA, we taper the number of embeddings per layer so $\mathbf{K} = \{128, 64, 32, 16, 8\}$, $d_e = 32$, and have networks layer-to-layer with fewer channels, for reasons of compute capacity. For CIFAR-10 and SVHN are compressing our data to $\sum \mathbf{M} = 341$ bytes, a compression factor of $\approx 9$. For CelebaA we compress into $\approx 284$ bytes, a factor of $\approx 43$.

Down- and up-sampling is achieved using strided convolutions within the networks that cross between adjacent layers. Each layer has its own pair of codebooks, We used per-pixel 256-way categorical distributions for $p_\theta(\mathbf{x}|\cdot)$, so our model consists entirely of discrete components (see Appendix C for more discussion of this). Training is done using Adamax [36] and we keep $\tau$, the temperature of the Gumbel-Softmax distribution, fixed at 0.5 during training, as has been done for single-layer Relaxed–VQ–VAEs [11]. For further model description and implementation details, see Appendix D.

Given our model specification we would not expect sampling quality to be competitive with single-layer VQ–VAE models with large, post-trained, autoregressive priors. However, these are to our knowledge the first results for hierarchical probabilistic deep generative models with discrete latent variables trained in a unified manner on these datasets. In addition to the results we present here, we also include results for smaller, narrower but deeper models with 16 and 32 layers of latent variables in Appendix E.

### 6.1 Numerical Results

We show in Table 1 numerical results from when our model, benchmarked against Relaxed–VQ–VAEs. We measure the bits per dim (bpd) for the training and test set (using non-relaxed categorical distributions). We also evaluate the Fréchet Inception Distance (FID) score [14]. This measures how close the embeddings are of two set of images in the hidden space of a standard pre-trained deep net. For each model we compare the its reconstructions and its samples to its training dataset, giving us the reconstruction FID (rFID) and sample FID (sFID). Our baseline is a non-hierarchical Relaxed–VQ–VAE, the bottom layer of our hierarchical model trained with a set of uniform categorical priors. These results show clearly the benefits our approach brings to VQ–VAEs, in particular the improvements in the values reached of test and train bits-per-dim and sample quality (as measured by sFID).

Table 1: Comparison of our model, RRVQ–VAE, to Relaxed–VQ–VAEs in bits per dim (bpd) for train and test sets, FID score of reconstructions (rFID) and of samples (sFID). For all, lower is better.

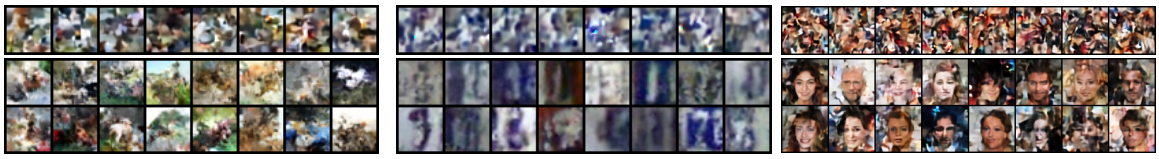
| Model | Dataset | Test bpd | Train bpd | rFID | sFID |
|---|---|---|---|---|---|
| Relaxed–VQ–VAE | CIFAR-10 | 5.33 | 5.33 | **8.4** | 194.3 |
| RRVQ–VAE, $L = 5$ | CIFAR-10 | **4.65** | **4.4** | 18.0 | **92.5** |
| Relaxed–VQ–VAE | SVHN | 4.11 | 4.20 | **15.0** | 200.3 |
| RRVQ–VAE, $L = 5$ | SVHN | **3.02** | **2.96** | 18.6 | **78.7** |
| Relaxed–VQ–VAE | CelebA | 5.31 | 5.31 | 87.4 | 355.4 |
| RRVQ–VAE, $L = 5$ | CelebA | **4.92** | **4.92** | **68.0** | **103.6** |

### 6.2 Visual Analysis of Reconstructions and Samples

Fig 3 shows reconstructions and Fig 4 ancestral samples for our models and baselines. Our RRVQ–VAE model achieves reasonable sampling quality without having to train a second post-hoc model over the learnt embeddings. Fig 5 demonstrates the effect each hierarchical layer has on the final draws. Within each layer, we sample repeatedly (plotted along each row) conditioned on sampled value of all the layers above (each set of these values getting a different row). We then propagate deterministically down through the layers below by taking the mode. The latent structure is interpretable, with the layers showing a degree of separation in their purpose, for instance that $\ell = 2$ describes digit identity.

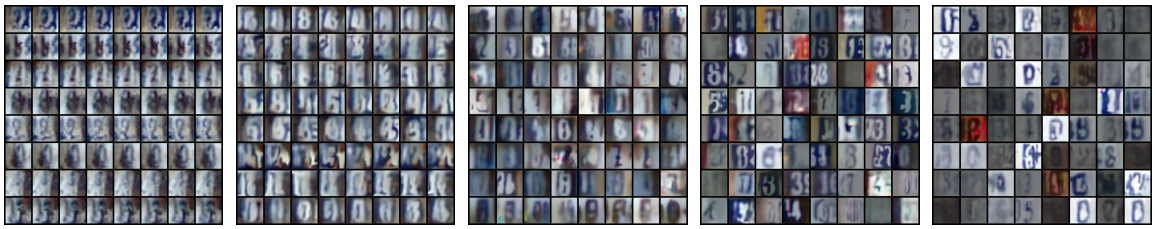
(a) CIFAR-10 (b) SVHN (c) CelebA

Figure 3: Reconstructions: we demonstrate our approach still provides high quality reconstructions, for CIFAR-10, SVHN and CelebA. Top row shows original images, middle row reconstructions from single-layer Relaxed–VQ–VAE baseline and bottom row reconstructions from our $L = 5$ model.



(a) CIFAR-10 (b) SVHN (c) CelebA

Figure 4: Sampling: we perform ancestral sampling for single-layer Relaxed–VQ–VAE baselines (top row) and our $L = 5$ models (lower two rows), for CIFAR-10, SVHN and CelebA.



(a) $\ell = 1$ (b) $\ell = 2$ (c) $\ell = 3$ (d) $\ell = 4$ (e) $\ell = 5$

Figure 5: Layerwise sampling in 5 layer RRVQ–VAE trained on SVHN. Note that layer $\ell = 2$ seems to represent digit identity: resampling in this layer changes digit identity while keeping the rest of the image roughly the same.

## 7. Conclusion

We have presented a novel parameterisation for stochastic Vector Quantisation, Relaxed-Responsibility Vector Quantisation. RRVQ learns a codebook of variances alongside the codebook of means, using the responsibilities under the Gaussian mixture model represented by those quantities to define discrete distributions, both for inference and the forward model. We then use this is as a building block to develop a novel variety of hierarchical discrete VAE, Relaxed-Responsibility Vector-Quantised VAEs. To our knowledge, RRVQ–VAEs are the first probabilistic deep generative models with hierarchies of discrete latent variables to be trained end-to-end on the datasets studied.

RRVQ–VAEs are highly expressive; their hierarchy of representations separate out different aspects of the data. The capacity and flexibility of the models is demonstrated by the fact that they do not require training of a secondary generative model over the latents in order to produce samples. We hope that this work inspires further research into discrete hierarchical variational autoencoders.

## References


[1] Diederik P Kingma and Max Welling. Auto-encoding Variational Bayes. In *ICLR*, 2014.

[2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*, 2014.

[3] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder Variational Autoencoders. In *NeurIPS*, 2016.

[4] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning Hierarchical Features from Generative Models. In *ICML*, 2017.

[5] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. *NeurIPS*, 2019.

[6] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. *NeurIPS*, 2017.

[7] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. *NeurIPS*, 2019.

[8] Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. SOM-VAE: Interpretable discrete representation learning on time series. In *ICLR*, 2019.

[9] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 2017.

[10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

[11] Casper Kaae Sønderby, Ben Poole, and Andry Mnih. Continuous Relaxation Training of Discrete Latent Variable Image Models. In *NeurIPS Bayesian Deep Learning Workshop*, 2017.

[12] Valentin Liévin, Andrea Dittadi, Lars Maaløe, and Ole Winther. Towards Hierarchical Discrete Variational Autoencoders. In *Advances in Approximate Bayesian Inference*, 2019.

[13] Will Williams, Sam Ringer, Tom Ash, John Hughes, David MacLeod, and Jamie Dougherty. Hierarchical Quantized Autoencoders. Technical report, 2020.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[15] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

[16] Ali Razavi, Oriol Vinyals, Aäron Van Den Oord, and Ben Poole. Preventing posterior collapse with $\delta$-VAES. In *ICLR*, 2019.

[17] Bin Dai and David Wipf. Diagnosing and enhancing VAE models. In *ICLR*, 2019.

[18] James Lucas, George Tucker, Roger Grosse, and Mohammad Norouzi. Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. In *NeurIPS*, 2019.

[19] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *NeurIPS*, 2016.

[20] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCnn with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[21] J MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[22] Christopher M Bishop. *Pattern Recognition and Machine Learning*. New York, 2006.

[23] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

[24] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. In *NeurIPS*, 2016.

[25] Jörg Bornschein, Andriy Mnih, Daniel Zoran, and Danilo J Rezende. Variational Memory Addressing in Generative Models. In *NeurIPS*, 2017.

[26] Brendan J. Frey and Geoffrey E. Hinton. Free energy coding. In *Data Compression Conference*, pages 73–81, 1996.

[27] James Townsend, Tom Bird, and David Barber. Practical Lossless Compression with Latent Variables using Bits Back Coding. In *ICLR*, 2019.

[28] James Townsend, Thomas Bird, Julius Kunze, and David Barber. HiLLoC: Lossless Image Compression with Hierarchical Latent Variable Models. In *ICLR*, 2020.

[29] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. In *ICLR*, 2015.

[30] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *ICLR*, 2017.

[31] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. Technical report, DeepMind, London, UK, 2019.

[32] Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. Integer Discrete Flows and Lossless Compression. In *NeurIPS*, 2019.

[33] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In *NeurIPS*, 2019.

[34] Geoffrey E Hinton and RIchard S Zemel. Autoencoders, Minimum Description Length, and Helmholtz Free Energy. In *NeurIPS*, 1994.

[35] J. R. Sack and J. Urrutia, editors. *Handbook of Computational Geometry*. North-Holland Publishing Co., NLD, 2000.

[36] Diederik P Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015.

## Appendix A. Relaxed Responsibility Vector Quantisation from a Mixture Model

To gain more insight into VQ-derived models, we can take a hierarchical discrete VAE and within it promote the embedding outputs $\vec{\mathbf{e}} = \{\mathbf{e}_1, .., \mathbf{e}_L\}$ to probabilistic variables. In doing this we obtain a hierarchical Gaussian mixture model, where each layer is itself a set of Gaussian mixture latent variables:

$$p_\theta(\mathbf{x}, \vec{\mathbf{z}}, \vec{\mathbf{e}}) = p_\theta(\mathbf{x}|\vec{\mathbf{e}})p_\theta(\vec{\mathbf{e}}, \vec{\mathbf{z}}) = p_\theta(\mathbf{x}|\vec{\mathbf{e}}) \prod_{\ell=1}^{L-1} [p(\mathbf{e}_\ell|\mathbf{z}_\ell)p_\theta(\mathbf{z}_\ell|\mathbf{e}_{>\ell})]p(\mathbf{z}_L) \tag{A.9}$$

where

$$p(\mathbf{e}_\ell|\mathbf{z}_\ell) = \prod_{m=1}^{M} \mathcal{N}(\mathbf{e}_\ell^m|\boldsymbol{\mu} = \mathbf{E}_{\mu,\ell}\mathbf{z}_\ell^m, \boldsymbol{\Sigma} = \mathbf{E}_{\Sigma,\ell}\mathbf{z}_\ell^m) \tag{A.10}$$

and

$$p_\theta(\mathbf{z}_\ell|\mathbf{e}_{>\ell}) = \prod_{m=1}^{M} \mathrm{Cat}(\mathbf{z}_m|\pi_\theta^m(\mathbf{e}_{>\ell})). \tag{A.11}$$

The posterior is given by

$$q_\phi(\vec{\mathbf{z}}, \vec{\mathbf{e}}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x})q_\phi(\mathbf{e}_L|\mathbf{z}_L) \prod_{\ell=1}^{L-1} q_\phi(\mathbf{z}_\ell|\mathbf{e}_{>\ell}, \mathbf{x})q_\phi(\mathbf{e}_\ell|\mathbf{z}_\ell). \tag{A.12}$$

We can obtain our model as a restricted version of this. Our intent is to bottleneck our representations through a set of discrete latent variables. Thus we choose $q_\phi(\mathbf{e}_\ell|\mathbf{z}_\ell) = p_\theta(\mathbf{e}_\ell|\mathbf{z}_\ell) = \delta(\mathbf{e}_\ell - \mathbf{E}_{\mu,\ell}\mathbf{z}_\ell)$, where $\delta(\cdot)$ is the Dirac delta function. This gives us an ELBO of the form

$$\begin{aligned}\mathcal{L}(\mathbf{x}) = &\mathbb{E}_{\vec{\mathbf{z}}\sim q} \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) \\ &- \sum_{\ell=1}^{L-1} \mathbb{E}_{\mathbf{z}_{>\ell}\sim q} \mathrm{KL}(q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell}, \mathbf{x})||p_\theta(\mathbf{z}_\ell|\mathbf{z}_{>\ell})) \\ &- \mathrm{KL}(q_\phi(\mathbf{z}_L|\mathbf{x})||p(\mathbf{z}_L)),\end{aligned} \tag{A.13}$$

where we have changed the likelihood to depend on $\vec{\mathbf{z}}$, as the $\vec{\mathbf{e}}$ it depended on is now deterministic given $\vec{\mathbf{z}}$. This mirrors our original notation, where one writes $p_\theta(\mathbf{x}|\mathbf{z})$ and $\mathbf{z}$ implicitly looks-up the codebook embeddings inside the likelihood. If we then we choose Eqs (7, 8) to parameterise the inference and generative models of each $\mathbf{z}$, we thus obtain our RRVQ–VAE.

## Appendix B. Ablation Study

How does our approach compare to other possible hierarchical extensions of Relaxed–VQ–VAEs? Here in Table G.4 we show the test and train bpd obtained under various hierarchical discrete VAEs. All of these have the ELBO Eq (6), but vary in how we parameterise $p$ and $q$.

Table B.2: Ablation study of our approach, for $L = 5$ models on CIFAR-10 and SVHN. We can have: the generative model log probabilities directly output by a net (*Direct-Cat in p*) or parameterised using responsibilties in the embedding space (*Embed-Cat in p*), and we can learn a codebook of diagonal covariances for the responsibilities ($\sigma$ *learnt*) or have them all fixed to one ($\sigma = 1$). RRVQ is when we have *Embed-Cat in p* and $\sigma$ *learnt*. Note that Direct-Cat with $\sigma$ learnt is unstable during training for SVHN.

| Model | Dataset | Test bpd | Train bpd |
|---|---|---|---|
| Direct-Cat in $p$, $\sigma = 1$ | CIFAR-10 | 5.05 | 5.00 |
| Direct-Cat in $p$, $\sigma$ learnt | CIFAR-10 | 5.10 | 5.08 |
| Embed-Cat in $p$, $\sigma = 1$ | CIFAR-10 | 5.11 | 5.06 |
| RRVQ–VAE | CIFAR-10 | **4.65** | **4.4** |
| Direct-Cat in $p$, $\sigma = 1$ | SVHN | 3.32 | 3.44 |
| Direct-Cat in $p$, $\sigma$ learnt | SVHN | – | – |
| Embed-Cat in $p$, $\sigma = 1$ | SVHN | 3.41 | 3.51 |
| RRVQ–VAE | SVHN | **3.02** | **2.96** |

**Direct-Cat in $p$, $\sigma = 1$** This is the simplest approach. One simply chooses to define the posteriors using a standard Relaxed–VQ–VAE parameterisation, Eq (3), and in generative model have the log probabilities of $p_\theta(\mathbf{z}_\ell|\mathbf{z}_{\ell+1})$ directly parameterised by a neural net.

**Direct-Cat in $p$, $\sigma$ learnt** Next you could learn the posterior diagonal covariances, so we have Eq (7) for the posteriors, but keep the direct parameterisation of log probabilities in the generative model. This approach is highly unstable for SVHN, and for CIFAR-10 leads to a small reduction in performance.

**Embed-Cat in $p$, $\sigma = 1$** Alternately, you can use an embedding parameterisation in the generative model, Eq (8), but keep fixed $\mathbf{E}_\Sigma = 1$, so we have the same standard Relaxed–VQ parameterised posteriors as in *Direct-Cat in p, $\sigma = 1$*. This approach also does not lead to increases in performance.

**Embed-Cat in $p$, $\sigma$ learnt: RRVQ** This is our proposed approach. There is a synergistic property here, that the these two changes together lead to improved performance of the models.

## Appendix C. Interpreting Discrete Hierarchical VAEs as Learning a Series of Reconstructions

We can expand each KL in Eq (6) as a cross entropy and an entropy: $\mathrm{KL}(q||p) = \mathcal{H}(q||p) - \mathcal{H}(q)$. The ELBO for this model can then be written as

$$\begin{aligned}\mathcal{L}(\mathbf{x}) = &\mathbb{E}_{\vec{\mathbf{z}}\sim q}[\mathcal{H}(q(\mathbf{x})||p_\theta(\mathbf{x}|\vec{\mathbf{z}}))] \\ &- \sum_{\ell=1}^{L-1} \mathbb{E}_{\mathbf{z}_{>\ell}\sim q}[\mathcal{H}(q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell},\mathbf{x})||p_\theta(\mathbf{z}_\ell|\mathbf{z}_{>\ell})) - \mathcal{H}(q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell},\mathbf{x}))] \\ &- \mathcal{H}(q_\phi(\mathbf{z}_L|\mathbf{x})||p(\mathbf{z}_L)) + \mathcal{H}(q_\phi(\mathbf{z}_L|\mathbf{x})),\end{aligned} \tag{C.14}$$

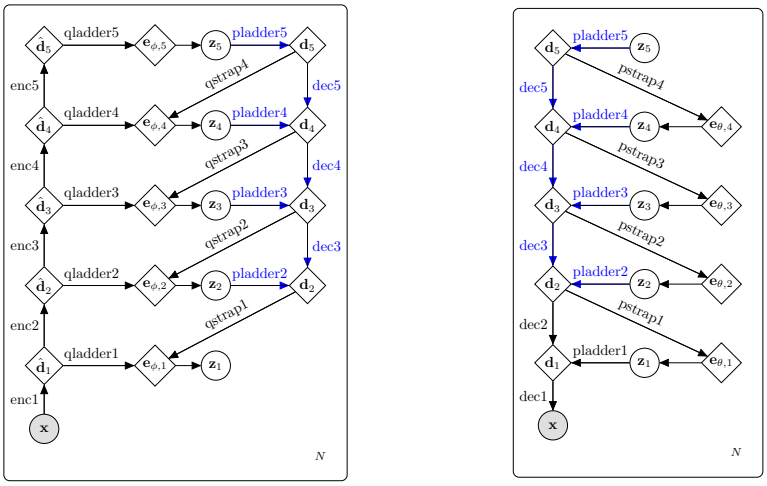
where $q(\mathbf{x})$ is the per-datapoint empirical distribution (we view a datapoint as a set sub-pixels) of one-hot discrete distributions.

If the likelihood $p_\theta(\mathbf{x}|\vec{\mathbf{z}})$ is itself a set of categorical distributions, then in a hierarchical discrete VAE the latent layers and the likelihood term all provide to the ELBO cross-entropy terms between discrete distributions, with then the entropy of each latent posterior acting as regularisers. If that is that case, then during training we are, in effect, requiring our model to build a series of representations $\vec{\mathbf{z}}$, all of which are scored under local objectives of the same form as how we score the reconstruction of our datapoint under our likelihood.

One might expect that the embedding of images when plotted as an image looks somewhat like a compressed version of the input data, up to the arbitrary indexing of the discrete latents. This is a weak effect, but can be seen somewhat in Fig C.6 below for two datapoints from CelebA. For each, the background is being encoded mostly using a single codebook index, which means that the person can be seen segmented out spatially in the latent representation in the first layer.

Figure C.6: For two input images from CelebA we plot them and their $\mathbf{z}_1$ representations from a RRVQ–VAE, colouring the indexes using the norm of the corresponding codebook mean.

## Appendix D. Details of Model Architecture




(a) Variational Posterior

(b) Generative Model

Figure D.7: RRVQ–VAE with $L = 5$ as in our experiments. (a) the variational posterior and (b) generative model, as defined in Eq (6). Blue arrows indicate shared networks. For simplicity the codebooks are not represented. Each labelled arrow corresponds to a network, described below.

The building block of our networks is a ResNet block composed of: ReLU $\rightarrow$ convolution with $3 \times 3$ filters $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ convolution with $1 \times 1$ filters $\rightarrow$ BatchNorm$\rightarrow$ gated convolution with $1 \times 1$ filters. In gated convolutions we output twice the number of channels as we need, and use the second half of the channels to gate the first half: taking the second half's sigmoid and multiplying that with the activations of the first half. All ResNet block's internal convolutions have 32 filters. ResNet blocks are used in sequences in each of our networks. Before each series of ResNet blocks we apply 2 initial convolutional layers and at the end we apply a final convolutional layer. These convolutional layers all have 128 filters

Now we describe the structure of each variety of network inside our model: enc_, dec_, qladder_, pladder_, qstrap_ and pstrap_. See Figure D.7 for reference.

First the chains of encoder and decoder networks, enc_ and dec_. Each of these networks is composed of 4 ResNet blocks, with the first initial convolutional layer performing down/up sampling by having a stride of 2 (giving an scaling in spatial size of factor of 4 from each application of one of these nets).

The networks qladder_ map from the backbone of encoders to the embedding space, and pladder_ map from the embedding space to the backbone of decoders. Each of these are composed of 4 ResNet blocks, with the initial convolutions having stride 1.

The networks qstrap_ and pstrap_, like dec_, carry out upsampling using an initial convolutional layer with a stride of 2 and are composed of 4 ResNet blocks. They output in the embedding space. The embeddings used to define each posterior distribution are the sum of the outputs of that layer's qstrap_ and qladder_ networks, and the embedding used for the generative model is simply the output of pstrap_.

We train using AdaMax with an initial learning rate that we decay on plateau, multiplying by 0.8 when there has been no decrease in the test set ELBO for 20 epochs, down to a minimum of $5 \times 10^{-5}$. The initial learning rate is $1 \times 10^{-4}$ for CIFAR-10 and $5 \times 10^{-4}$ for SVHN and CelebA. We train with for 500,000 steps. We used Azure VMs with NVIDIA M60 GPUs to train our models – using a single M60 to train a model takes $\approx 1$ week.

## Appendix E. Deeper, Lightweight Models

We also trained smaller, more light weight models with more layers of latent variables. In these models we expand each level of the model, each set of latents of a particular spatial extent, to contain not a single pair of mean and covariance codebooks but multiple pairs. This approach is reminiscent of that used in vanilla Relaxed–VQ–VAEs, where multiple codebooks can be used inside a single-layer Relaxed–VQ-VAE. To be clear, we are adding extra sub-layers of latents within each 'block' layer – the autoregressive structure of the model remains over the blocks, with the newly-multiplied sub-layers conditionally independent of each other. We choose to have 4 or 8 such sublayers per layer, and to have 4 blocks of layers: we have an extra stride at the base of the hierarchy. Training on CIFAR-10 and SVHN, this results in latents of size $\mathbf{M} = \{4 \times 8 \times 8, 4 \times 4 \times 4, 4 \times 2 \times 2, 4 \times 1 \times 1\}$ with 4 books per block and $\mathbf{M} = \{8 \times 8 \times 8, 8 \times 4 \times 4, 8 \times 2 \times 2, 8 \times 1 \times 1\}$ with 8 books per block.

We include these results to show that with very simple design choices our approach can be scaled up to more layers of latent variables. As these models are smaller in capacity than those shown in the main paper they do not show superior performance nor would be expect them to.

Table E.3: bits-per-dim for deeper models

| Model | Dataset | Test bpd | Train bpd |
|---|---|---|---|
| RRVQ–VAE, $L = 16$ | CIFAR-10 | 5.20 | 5.12 |
| RRVQ–VAE, $L = 32$ | CIFAR-10 | 5.20 | 5.07 |
| RRVQ–VAE, $L = 16$ | SVHN | 3.71 | 3.86 |
| RRVQ–VAE, $L = 32$ | SVHN | 3.56 | 3.62 |


Figure E.8: Lightweight 32 latent layer models trained on CIFAR-10 and SVHN, showing original inputs, reconstructions and samples


Figure E.9: Lightweight 16 latent layer models trained on CIFAR-10 and SVHN, showing original inputs, reconstructions and samples

## Appendix F. Progressive Training For Faster Training

Hierarchical discrete VAEs with Markovian structure have been proposed, and are trained in parts sequentially. First the parts of the model concerned with just $\mathbf{x}$ and $\mathbf{z}_1$ is trained, then those parameters are fixed and the parts of the model for $\mathbf{z}_1$ and $\mathbf{z}_2$ are now trained, and so on.

Here we employ a method of progressive training, where we start training only the bottom-most latent $\mathbf{z}_1$ and then include higher layers as training continues. We carry our this manner of training to reduce wall-clock time. Training the higher layers of the network does require that the lower layers have already learnt useful representations. Not having to train the entire network at this early stage, when training the upper layers does not achieve much, speeds up training. To be completely clear, this method is not necessary for RRVQ–VAEs to train stably or to learn to use their model capacity, and its use does not appear to lead to improvements in the performance of the resultant models. Instead, it is a useful trick for speeding up experiments.

## Appendix G. Relaxation Bias

Recent work has highlighted the potential issue of *relaxation bias* in hierarchical discrete VAEs. During training the model does not have true categorical latents but Gumbel-Softmax relaxations thereof. Maximising the ELBO under this relaxation is not exactly equivalent to maximising the ELBO under a Categorical: the relaxation induces a bias. This bias leads to the possibility that models learn poorly under the true task we care about, learning a model with Categorical distributions.

We calculate this for CIFAR-10 and SVHN, over test and train sets, for our model and for the ablations we studied above. Our model does if anything a little worse than the ablations in the final value of this bias. How to best handle this bias remains an area for future research.

Table G.4: Relaxation Gap for different ablations and for our model, all with $L = 5$. Again, Direct-Cat with $\sigma$ learnt is unstable for SVHN.

| Model | Dataset | Test relaxation bias in bpd | Train relaxation bias in bpd |
|---|---|---|---|
| Direct-Cat in $p$, $\sigma = 1$ | CIFAR-10 | 0.39 | 0.40 |
| Direct-Cat in $p$, $\sigma$ learnt | CIFAR-10 | 0.44 | 0.45 |
| Embed-Cat in $p$, $\sigma = 1$ | CIFAR-10 | **0.36** | **0.38** |
| RRVQ–VAE | CIFAR-10 | 0.47 | 0.47 |
| Direct-Cat in $p$, $\sigma = 1$ | SVHN | 0.44 | **0.38** |
| Direct-Cat in $p$, $\sigma$ learnt | SVHN | – | – |
| Embed-Cat in $p$, $\sigma = 1$ | SVHN | **0.42** | 0.41 |
| RRVQ–VAE | SVHN | 0.45 | 0.44 |