Relaxed-Responsibility Hierarchical Discrete VAEs

Matthew Willetts[*] UCL **Xenia Miscouridou** Imperial College **Stephen Roberts** University of Oxford **Chris Holmes** University of Oxford

Abstract

Successfully training Variational Autoencoders (VAEs) with a hierarchy of discrete latent variables remains an area of active research. Vector-Quantised VAEs are a powerful approach to discrete VAEs, but naive hierarchical extensions can be unstable when training. 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 that gives better performance and more stable training. This enables a novel approach to hierarchical discrete variational autoencoders with numerous layers of latent variables (here up to 32) that we train end-to-end. Within hierarchical probabilistic deep generative models with discrete latent variables trained end-to-end, we achieve state-of-the-art bits-per-dim results for various standard datasets.

1 Introduction

Probabilistic deep generative models, such as Variational Autoencoders (VAEs), have had significant and continuing success in learning continuous representations of data (Kingma & Welling, 2014; Rezende et al., 2014; Kingma et al., 2016; Vahdat & Kautz, 2020; Child, 2021). The learning of discrete representations has also flourished (Grathwohl et al., 2018; Oord et al., 2017; Razavi et al., 2019; Fortuin et al., 2019; Pervez et al., 2020) and remains an active area of research. However, training rich hierarchical models with discrete latent variables for high-dimensional data remains a problem in the field (Liévin et al., 2019; Williams et al., 2020; Pervez et al., 2020).

Here we propose an effective, scalable method for learning hierarchical discrete representations of image data within a unified probabilistic framework. This work builds on Vector-Quantised Variational Autoencoders (VQ–VAEs) (Oord et al., 2017) and their relaxation (Sønderby et al., 2017).

VQ–VAEs reach surprisingly poor raw bits-per-dim (bpd), a scaled form of the ELBO, on both the train and test sets. Thus to achieve good performance they require post-hoc training of density estimators on learnt embeddings. This motivates us to develop a novel variety of hierarchical discrete VAEs. These models, which we call *Relaxed-Responsibility Vector-Quantised VAEs* or RRVQ–VAEs, achieve state of the art bits-per-dim for this class of models.

2 Background: Vector Quantised Variational Autoencoders

The Vector-Quantised Variational Autoencoder (VQ–VAE) (Oord et al., 2017) is a density estimator for high dimensional data. Instead of having continuous latent variables, as in the vanilla VAE, the latents $\mathbf{z}$ are a set of M discrete variables $\mathbf{z} = \{z^1, \ldots, z^M\}$ each of dimensionality K. The joint $p_\theta(\mathbf{x}, \mathbf{z})$ factorises as for a vanilla VAE, with $p(\mathbf{z}) = \prod_{m=1}^{M} \mathrm{Cat}\left(z^m | \frac{1}{K}\right)$.

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 $\mathbf{E}^k \in \mathbb{R}^{d_e}$, d_e

[*]Correspondence to `mwilletts@turing.ac.uk`.

being the dimensionality of the embedding space. 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 there is a one-hot (i.e. deterministic) posterior

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

rVQ–VAEs Instead of the deterministic posterior found in a vanilla VQ–VAE, a Gumbel-Softmax distribution (Maddison et al., 2017; Jang et al., 2017) can be used to specify a posterior distribution from which we can take differentiable samples (Sønderby et al., 2017). Thus 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 (Sønderby et al., 2017),

$$q(\mathbf{z}|\mathbf{x}) = \prod_{m=1}^{M} \text{Cat}\left(\mathbf{z}_m|\pi_\phi^m\left(\mathbf{x}\right)\right), \text{ where } \pi_\phi^{m,k}(\mathbf{x}) \propto \exp\left(-\frac{1}{2}\left|\mathbf{e}_\phi^m(\mathbf{x}) - \mathbf{E}^k\right|_2^2\right). \tag{2}$$

These *Relaxed*–VQ–VAEs (henceforth rVQ–VAEs) have been shown to make better use of their latent variables than the deterministic base model, obtaining higher values of the evidence lower bound (ELBO) both at train and test time (Sønderby et al., 2017).

3 Relaxed-Responsibility Hierarchical Discrete VAEs

3.1 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^{th} latent variable in the ℓ^{th} layer. We wish to have an autoregressive structure *between* layers. Inspired by the ResNet VAEs (Kingma et al., 2016), we choose our generative model's factorisation to be

$$p_\theta\left(\mathbf{x}, \vec{\mathbf{z}}\right) = p_\theta\left(\mathbf{x}|\vec{\mathbf{z}}\right) p_\theta\left(\vec{\mathbf{z}}\right) = p_\theta(\mathbf{x}|\vec{\mathbf{z}}) p(\mathbf{z}_L) \prod_{\ell=1}^{L-1} p\left(\mathbf{z}_\ell|\mathbf{z}_{>\ell}\right) \tag{3}$$

where $p(\mathbf{z}_\ell|\mathbf{z}_{>\ell}) = \text{Cat}\left(\mathbf{z}_\ell|\pi_{\theta,\ell} = f_{\theta,\ell}(\mathbf{z}_{>\ell})\right)$ and $p_\theta(\mathbf{z}_L) = \text{Cat}\left(\mathbf{z}_L|\pi_{\theta,L}\right)$. Similarly, q factorises as

$$q_\phi(\vec{\mathbf{z}}|\mathbf{x}) = q_\phi(\mathbf{z}_L|\mathbf{x}) \prod_{\ell=1}^{L-1} q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell}, \mathbf{x}). \tag{4}$$

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

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{\vec{\mathbf{z}}\sim q} \log p_\theta(\mathbf{x}|\vec{\mathbf{z}}) - \text{KL}\left(q_\phi(\mathbf{z}_L|\mathbf{x})||p(\mathbf{z}_L\right) - \sum_{\ell=1}^{L-1} \mathbb{E}_{\mathbf{z}_{>\ell}\sim q} \text{KL}(q_\phi(\mathbf{z}_\ell|\mathbf{z}_{>\ell}, \mathbf{x})||p_\theta(\mathbf{z}_\ell|\mathbf{z}_{>\ell}). \tag{5}$$

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

3.2 Relaxed-Responsibility Vector-Quantisation

Our first main contribution is a new method of parameterising the generative model and the approximate posterior for models containing vector-quantised discrete latents. This 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, hierarchical models of this form had low performance and were often unstable during training, and that the changes we propose are synergistic – working better together than either alone.

3.2.1 Proposal for q

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 (Bishop, 2006, §10.2). Eq (2) is equivalent to saying that the posterior at each position in $\mathbf{z}$ is proportional 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 mixture component. The responsibilities then used for defining $\pi_{\phi,\ell}(\mathbf{e}_{\phi,\ell})$ are

$$\pi_{\phi,\ell}^{m,k}(\mathbf{e}_{\phi,\ell}) \propto \frac{\exp\left(-\frac{1}{2\mathbf{E}_{\ell,\Sigma}^k}\left|\mathbf{e}_{\phi,\ell}^m - \mathbf{E}_{\mu,\ell}^k\right|_2^2\right)}{\sqrt{(2\pi)^{d_e}\,\mathbf{E}_{\ell,\Sigma}^k}}, \tag{6}$$

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 ℓ^{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 (4).

By learning $\mathbf{E}_{\Sigma,\ell}$, codebook embeddings with large diagonal covariance will have their means used preferentially when the output 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' (Jacobs et al., 1991) when $\mathbf{e}_{\phi,\ell}$ is close.

3.2.2 Proposal for p

What about for the generative model? 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 VQ–VAEs that directly parameterised these conditional probabilities.

Training instability in VAEs come from large KL values that then lead to numerical overflow and large gradients.

What is a reasonable, flexible form for the generative model that will provide stable training while preserving or even improving performance? We find that rVQ parameterisation of discrete variables leads to less-peaked, higher-entropy distributions, than a naive implementation using a softmax of raw logits—the method we found to be unstable. See Appendix D for theoretical and experimental study of this.

Thus we choose to parameterise the conditional distributions in $p_\theta(\vec{\mathbf{z}})$ along the same lines as for q, namely rVQ-parameterisation via an embedding space, rather than directly using raw logits, and sharing the same codebooks as for q. So, in the generative model each conditional distribution in $p_\theta(\vec{\mathbf{z}})$, rather than receiving the parameters needed to define it directly, instead has its probabilities parameterised via the responsibilities given by embeddings $\mathbf{e}_{\theta,\ell} \in \mathbb{R}^{M\times d_e}$ output by a deep net:

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

3.3 Overall Model

By combining Relaxed-Responsibility VQ with a hierarchical discrete VAE structure, we obtain our proposed model, a Relaxed-Responsibility Vector Quantised VAE (RRVQ–VAE). 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 (3-4): in the generative model we have an autoregressive structure over layers, and similarly in the posterior each layer of latents is conditioned both on $\mathbf{x}$ and on those above it in the hierarchy. See Fig A.1 for a graphical representation of this model. 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.

4 Experiments

We train our models on CIFAR-10, SVHN and CelebA. We train very deep models with $L = 32$ layers. For each of CIFAR-10 and SVHN the models 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.

The number of latent variables per layer decreases as we ascend the hierarchy, as represented in Fig A.1. We decrease the number of latents by a factor of 4 every 8 layers, forming 4 blocks each of decreasing numbers of latents. Each layer of latent variables has its own pair of codebooks for means and diagonal covariances. For further model description and implementation details, see Appendix C.

4.1 Numerical Results

Table 1: Bits Per Dim Results: Comparison of our model, RRVQ–VAE, to various baselines in bits-per-dim (bpd) for train & test sets – lower better. We benchmark against rVQ–VAEs, VIMCO-trained discrete VAEs (Oord et al., 2017) and FouST-trained models with binary latents and $L = 1$ or $L = 4$ layers (Pervez et al., 2020). For additional context we also give values for hierarchical Spatial-VAEs, the conditionally-Gaussian latent variable version of our models.

MODEL	TEST BPD	TRAIN BPD
CIFAR-10		
VIMCO	5.14	-
rVQ–VAE	4.77	4.87
FouST, $L = 4$	4.16	-
FouST, $L = 1$	4.02	-
RRVQ–VAE, $L = 32$	**3.94**	**3.81**
SPATIAL–VAE, $L = 32$	3.55	3.49
SVHN		
rVQ–VAE	3.73	4.17
RRVQ–VAE, $L = 32$	**2.30**	**2.52**
SPATIAL–VAE, $L = 32$	1.94	2.07
CELEBA		
rVQ–VAE	5.31	5.31
RRVQ–VAE, $L = 32$	**2.97**	**2.97**
SPATIAL–VAE, $L = 32$	2.54	2.58

We show in Table 1 numerical results from our $L = 32$ models, benchmarked against rVQ–VAEs and various baselines. We measure the bits-per-dim (bpd) for the training and test set (using non-relaxed categorical distributions). These results show clearly the benefit our approach brings to discrete VAEs, from the improved values reached of test and train bits-per-dim. Our models help close the gap between discrete latent variable models and those with continuous latent variables.

5 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 within the approximate posterior using for inference and in 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. RRVQ–VAEs are the highest-performance unified probabilistic deep generative models with hierarchies of discrete latent variables to be trained end-to-end on the datasets studied.

Bibliography

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. When is “Nearest Neighbor” Meaningful? In *Proceedings of the 7th International Conference on Database Theory*, volume 1540, pp. 217–235, 1999. ISBN 3540654526. doi:10.1007/3-540-49257-7_15.

Bishop, C. M. *Pattern Recognition and Machine Learning*. New York, 2006. ISBN 9780387310732. doi:10.1117/1.2819119.

Child, R. Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. In *ICLR*, 2021.

Fortuin, V., Hüser, M., Locatello, F., Strathmann, H., and Rätsch, G. SOM-VAE: Interpretable discrete representation learning on time series. In *ICLR*, 2019.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

Gregor, K., Besse, F., Rezende, D. J., Danihelka, I., and Wierstra, D. Towards conceptual compression. In *NeurIPS*, 2016.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991. ISSN 0899-7667. doi:10.1162/neco.1991.3.1.79.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding Variational Bayes. In *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved Variational Inference with Inverse Autoregressive Flow. In *NeurIPS*, 2016.

Liévin, V., Dittadi, A., Maaløe, L., and Winther, O. Towards Hierarchical Discrete Variational Autoencoders. In *Advances in Approximate Bayesian Inference*, 2019.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 2017.

Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. Neural Discrete Representation Learning. *NeurIPS*, 2017.

Pervez, A., Cohen, T., and Gavves, E. Low Bias Low Variance Gradient Estimates for Hierarchical Boolean Stochastic Networks. In *ICML*, 2020.

Razavi, A., Oord, A. v. d., and Vinyals, O. Generating Diverse High-Fidelity Images with VQ-VAE-2. *NeurIPS*, 2019.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*, 2014.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. PixelCNN++: Improving the PixelCnn with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Sønderby, C. K., Poole, B., and Mnih, A. Continuous Relaxation Training of Discrete Latent Variable Image Models. In *NeurIPS Bayesian Deep Learning Workshop*, 2017.

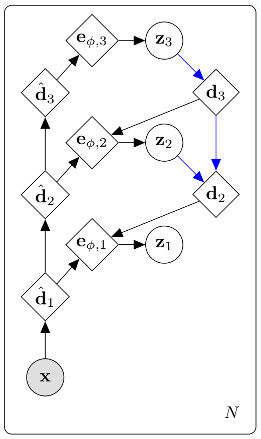
Vahdat, A. and Kautz, J. NVAE: A Deep Hierarchical Variational Autoencoder. In *NeurIPS*, 2020.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional Image Generation with PixelCNN Decoders. In *NeurIPS*, 2016.

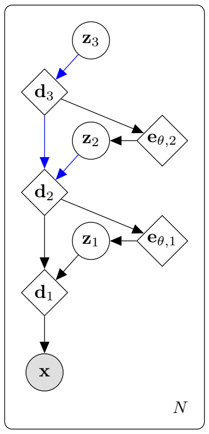
Williams, W., Ringer, S., Ash, T., Hughes, J., MacLeod, D., and Dougherty, J. Hierarchical Quantized Autoencoders. *NeurIPS*, 2020.

Appendix for Relaxed-Responsibility Hierarchical Discrete VAEs

A Graphical Representation of RRVQ-VAE



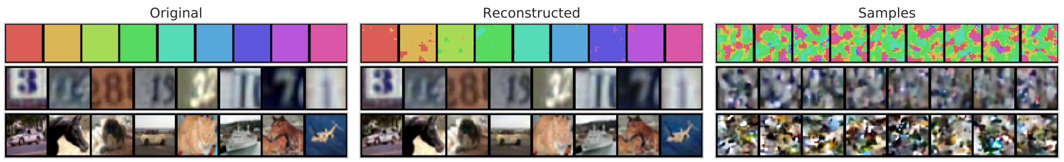
(a) Variational Posterior



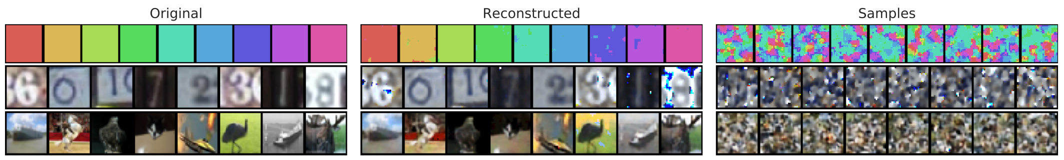
(b) Gen. Model

Figure A.1: RRVQ–VAE with $L = 3$, (a) variational posterior and (b) generative model, as defined in Eq (5). 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. We decrease the multiplicity by a factor of 4 at each layer.

B Sampling and Reconstructing in VQ–VAEs



(a) rVQ–VAE



(b) Spatial–VAE

Figure B.2: 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) rVQ–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 apply to images generally, 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 the image (Oord et al., 2017). 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. This indicates severe aggregate posterior–prior mismatch. Samples from this prior fail to capture the structure needed, i.e. the dependencies between the M latents that are 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 (van den Oord et al., 2016; Salimans et al., 2017) over the latent representations to then sample from. This is followed in the two- and three-layer extension of VQ–VAEs as well (Razavi et al., 2019).

Conversely, in VAEs with continuous latent variables the reconstructions are generally found to be somewhat blurry, while samples tend to have more coherent structure. In a standard VAE with $p(\mathbf{z}) = \prod_{i=1}^{M} \mathcal{N}\left(z^m|0, 1\right)$ 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 related 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, and the decoder mean are output by MLPs. Those 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: $p_\theta(\mathbf{z}) = \prod_{m=1}^{M} \mathcal{N}\left(\mathbf{z}^m|\mathbf{0}, \mathbb{I}\right)$ 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 rVQ–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 B.2 for the resulting reconstructions and samples for the three datasets for both models. We also provide examples of toy MLP-parameterised VQ-VAEs providing coherent samples in Appendix H.

Embedding an image into the latent space for reconstruction is relatively easy. For the discrete model, with high probability the encoder outputs embeddings $\mathbf{e}_\phi(\mathbf{x})$ that are close to the appropriate codebook embedding, and this is the case for each of the M spatially-arranged latents. Similarly, at each latent position, the Spatial–VAE encoder learns to place posterior probability over the appropriate latent space region. 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.

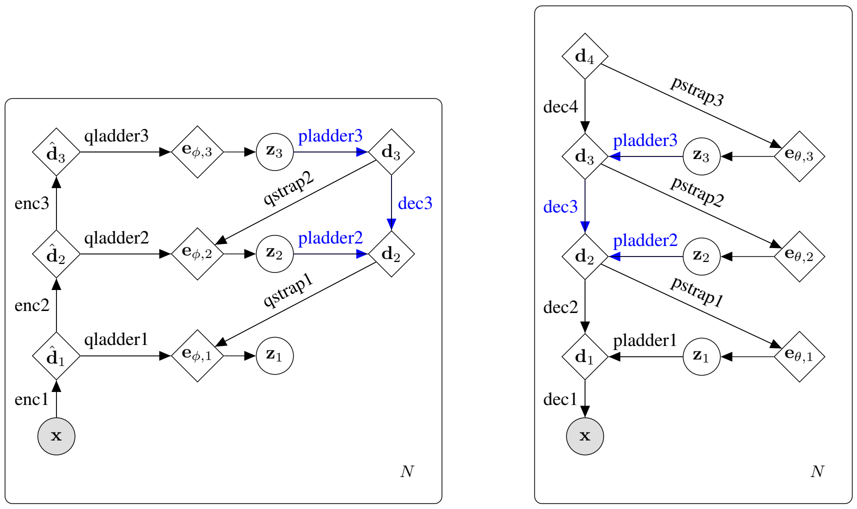
The poor quality of naive VQ–VAE draws is not intrinsically from having discrete latent variables, but from having discrete latent variables *that are arranged spatially and are parameterised in both the posterior and generative models using convolutional neural networks*. However, it is the choice to 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 δ functions, so it resembles an empirical data distribution. Thus training a high-performance density estimator is reasonable and provides realistic draws (Oord et al., 2017; Razavi et al., 2019). In this manner of operation, the encoder-decoder networks are tools for non-linear dimensionality-reduction, so that the density estimator can be trained in a lower-dimensional space, the learnt latent space, rather than directly on the raw data. While that is a proven and performative approach, our goal is to combine the benefits of VQ–VAEs (high quality reconstructions, the desirable property of learning discrete representations, ease of training) with having a unified modelling approach, with models trained end-to-end.

We develop ways to make discrete VAEs more expressive and flexible by adding hierarchical structure. This removes the need of a two-stage training process, and gives us the benefits of hierarchical representations such as having different layers learning different aspects of the data. Further, if autoregressive models are used to produce samples, we are required to perform as many forward passes through the model as there are latent variables. In the hierarchical case, as in Razavi et al. (2019), this remains the case.

In this paper we are training very deep hierarchies of latent variables, up to 32 layers. Therefore, if we had autoregressive models for sampling in a hierarchical model of this form, the additional calls that would be needed to produce a single sample would be very demanding. For our deepest models trained on 32×32 images it would be ≈ 2000 internal, sequential forward passes. For 64×64 images it would be $\approx 10,000$. Instead, with our approach, we are able to generate samples using a single forward pass.

C Details of Model Architecture



(a) Variational Posterior (b) Gen. Model

Figure C.3: RRVQ–VAE with $L = 3$ as an example. (a) The variational posterior and (b) generative model, as defined in Eq (5). Blue arrows indicate shared networks. For simplicity the codebooks are not represented. Each labelled arrow corresponds to a network, described below.

Our network implementation has the basic structure of a ResNet VAE (Kingma et al., 2016), including in neural network hyperparameters (other than our choice to narrow the number of latents as we ascent the hierarchy, the necessary changes for RRVQ as opposed to Gaussian latents, and that we train with a slightly slower initial learning rate). Now we describe the structure of each variety of network inside our model. enc1/dec1 are convolutions/transposed convolutions that down/upscale their inputs using a stride of 2. All the other subnetworks of the enc/dec deterministic backbones are each implemented as a single resnet block – each dec_ using a transposed convolution internally. When the mappings between two layers of latent variables requires a resizing, the identity path of the network performs a differentiable rescaling operation.

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 implemented as a single convolutional layer. The networks qstrap_ and pstrap_ too are each implemented as a single convolutional layer, and carry out upscaling using a stride of 2. They output in the embedding space. The embeddings used to define each layer's posterior distribution are the sum of the outputs of that layer's qstrap_ and qladder_ networks. The embeddings used for the generative model's internal conditional probabilities are simply the outputs of each pstrap_.

For the $L = 32$ model runs, the backbones have 256 channels, and the $\mathbf{e}$ representations are 32 dimensional. The layer's codebooks hold 256 embeddings each. The likelihood function is the same discretised logistic likelihood as in Kingma et al. (2016). As in Kingma et al. (2016), we use weight normalisation, ELU activations and free-bits regularisation.

The top-most latent variable in the generative model can be set to be uniform over embeddings, or can be parameterised by a similar procedure as for the rest via a $\mathbf{d}_L$ that is a learnable parameter (rather than itself the output of a network). See Fig C.3 for a representation of this – here for $L = 3$ we have $\mathbf{d}_4$ in the generative model parameterising $p_\theta(\mathbf{z}_3)$.

When training with just $L = 5$ layers, as opposed to 32, it is as if we remove the corresponding intermediate latent variables along with their ladder_, strap_ networks, so now the enc_ and dec_ networks are composed of 4 resnet blocks between latents. We also promote the remaining ladder_, strap_ networks to themselves be composed of 4 resnet blocks.

We train using AdaMax with batch size 64 and 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 (SVHN + CIFAR-10)/5 (CelebA) epochs, down to a minimum of 5×10^{-5}. The initial learning rate is 8×10^{-4}. We train with for up to 500 (SVHN + CIFAR-10)/160 (CelebA) epochs or until convergence. We used Azure VMs with NVIDIA M60 GPUs to train our models – using a single M60 to train a model takes ≈ 10 days for SVHN and CIFAR10. For the CelebA multi-GPU training is necessary.

D Worst-Case Entropy of rVQ and Softmax-parameterised Discrete Distributions

In Theorems 1 and 2 we consider the worst-case arrangement of codebook means/logits for rVQ and Softmax respectively, such that a single large-magnitude value of the underlying network outputs has maximum impact driving the resulting discrete distribution to be close to one-hot.

Theorem 1. *(Minimum entropy from rVQ) Consider the worst-case arrangement of rVQ codebooks vectors, i.e. resulting in the minimum entropy categorical distribution: all but one of the codebook embeddings are an equal and greater distance away from the input embedding. For large-magnitude input embeddings of distance d from the solitary, closest codebook embedding along the line of separation and the remaining $K-1$ codebook embeddings at a distance $d+\delta$ along the same line of separation, the entropy of the resulting categorical distribution, Eq (2) is, to first order*

$$\mathcal{H}_{\mathrm{rVQ}} = (K-1)(1+g)\exp(-g) + O\left(\exp(-g)^2\right) \tag{D.1}$$

where $g = \left(\frac{\delta^2}{2} + \delta d\right)$. Proof: See Appendix D.1.

Theorem 2. *(Minimum entropy from Softmax) Consider the worst-case arrangement of logits, i.e. resulting in the minimum entropy categorical distribution: all but one of the logits take the same value c, with one logit taking the larger value $c+\ell$, $\ell > 0$. For large-magnitude difference in logits ℓ, the entropy of the resulting categorical distribution is, to first order,*

$$\mathcal{H}_{\mathrm{softmax}} = (K-1)(1+\ell)\exp(-\ell) + O\left(\exp(-\ell)^2\right) \tag{D.2}$$

Proof: See Appendix D.2.

We computationally verify the tightness of these first order approximations in Appendix D.3 and find them to be very accurate, quickly become correct to one part in 10^{-6}.

Corollary 2.1. *Viewing $\ell + c = d$ as the large-magnitude output of a neural network, for large d rVQ-parameterised categorical distributions have higher entropy than softmax-parameterised ones if $\delta < 1$.*

This tells us that for large neural network outputs rVQ-parameterised distributions have higher entropy than those parameterised via logits, even for the most unlucky arrangement of codebook embeddings, as long as the largest distance between codebooks is < 1.

D.1 Proof of Theorem 1

Proof. Our distribution of interest is a rVQ distribution, ie Eq (2), where we have the worst possible arrangement of our K-member codebooks – the arrangement that leads to the minimum possible entropy, and we also assume the worst possible positions of the embedding vector $\mathbf{e}$. The arrangement that leads to this is having all but one of the codebook vectors at one point and a single codebook separated a distance δ from them, with the embedding vector $\mathbf{e}$ lying along the line defined by those two positions a distance d from the outlier codebook vector and $d+\delta$ from the remaining $K-1$

codebook vectors. We note that this arrangement is closely related to that considered in Beyer et al. (1999), § 3.5.2.

This gives us a distribution $p(\mathbf{z}|\boldsymbol{\pi})$, where

$$\pi^i = \begin{cases} \frac{1}{Z}\exp\left(-\frac{1}{2}d^2\right) & \text{if } i = 1 \\ \frac{1}{Z}\exp\left(-\frac{1}{2}(d+\delta)^2\right) & \text{otherwise.} \end{cases} \tag{D.3}$$

and

$$Z = \exp\left(-\frac{1}{2}d^2\right) + (K-1)\exp\left(-\frac{1}{2}(d+\delta)^2\right). \tag{D.4}$$

The entropy of this discrete distribution is thus:

$$\begin{aligned}
\mathcal{H}_{\text{rVQ}} &= -\sum_{i=1}^{K} \pi^i \log \pi^i && \text{(D.5)} \\
&= -\frac{\exp\left(-\frac{1}{2}d^2\right)}{Z}\log\left(\frac{\exp\left(-\frac{1}{2}d^2\right)}{Z}\right) - (K-1)\frac{\exp\left(-\frac{1}{2}(d+\delta)^2\right)}{Z}\log\left(\frac{\exp\left(-\frac{1}{2}(d+\delta)^2\right)}{Z}\right) && \text{(D.6)} \\
&= -\frac{\exp\left(-\frac{1}{2}d^2\right)}{Z}\left(-\frac{1}{2}d^2 - \log Z + (K-1)\exp\left(-\frac{1}{2}\delta^2 - \delta d\right)\left(-\frac{1}{2}(d+\delta)^2 - \log Z\right)\right). && \text{(D.7)}
\end{aligned}$$

Now let us consider the value of this in the limit of large d, $d \gg \delta$. First, let us expand $\frac{\exp\left(-\frac{1}{2}d^2\right)}{Z}$ using the first order expansion $(1+x)^{-1} \approx 1 - x$ for $|x| \ll 1$.

$$\begin{aligned}
\frac{\exp\left(-\frac{1}{2}d^2\right)}{Z} &= \frac{\exp\left(-\frac{1}{2}d^2\right)}{\exp\left(-\frac{1}{2}d^2\right) + (K-1)\exp\left(-\frac{1}{2}(d+\delta)^2\right)} && \text{(D.8)} \\
&= \frac{1}{1 + (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)} && \text{(D.9)} \\
&= 1 - (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right) + O\left(\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)^2\right). && \text{(D.10)}
\end{aligned}$$

Second, let us expand $\log Z$ using the first order expansion $\log(1+x) \approx x$ for $|x| \ll 1$.

$$\begin{aligned}
\log Z &= \log\left(\exp\left(-\frac{1}{2}d^2\right) + (K-1)\exp\left(-\frac{1}{2}(d+\delta)^2\right)\right) && \text{(D.11)} \\
&= \log\left(\exp\left(-\frac{1}{2}d^2\right)\left(1 + (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\right)\right) && \text{(D.12)} \\
&= -\frac{1}{2}d^2 + \log\left(1 + (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\right) && \text{(D.13)} \\
&= -\frac{1}{2}d^2 + (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right) + O\left(\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)^2\right). && \text{(D.14)}
\end{aligned}$$

Taking Eqs (D.10,D.14) and subbing back into Eq (D.7), we get

$$\begin{aligned}
\mathcal{H}_{\text{rVQ}} &= \left[1 - (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\right]\left[(K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\left(1 + \frac{1}{2}(\delta + d)^2 - \frac{1}{2}d^2\right)\right. \\
&\qquad \left. + O\left(\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)^2\right)\right] && \text{(D.15)} \\
&= (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\left(1 + \frac{1}{2}(\delta^2 + 2\delta d)\right) + O\left(\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)^2\right). && \text{(D.16)}
\end{aligned}$$

Giving us, to first order in $\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)$,

$$\mathcal{H}_{\text{rVQ}} \approx (K-1)\exp\left(-\frac{1}{2}(\delta^2 + 2\delta d)\right)\left(1 + \frac{1}{2}\left(\delta^2 + 2\delta d\right)\right) \tag{D.17}$$

as required. □

D.2 Proof of Theorem 2

Proof. Our distribution of interest is a discrete distribution defined as a softmax of K raw logits, where we have the worst possible arrangement of the logit outputs – the arrangement that leads to the minimum possible entropy. The arrangement that leads to this is having all but one of the logits take one value c and a single logit taking the value $c + \ell$, $\ell > 0$.

This gives us a distribution $p(\mathbf{z}|\boldsymbol{\pi})$, where

$$\pi^i = \begin{cases} \frac{1}{Z}\exp(c+\ell) & \text{if } i = 1 \\ \frac{1}{Z}\exp(c) & \text{otherwise} \end{cases} \tag{D.18}$$

and

$$Z = \exp(c+\ell) + (K-1)\exp(c). \tag{D.19}$$

The entropy of this discrete distribution is thus:

$$\mathcal{H}_{\text{softmax}} = -\sum_{i=1}^{K} \pi^i \log \pi^i \tag{D.20}$$

$$= -\frac{\exp(\ell+c)}{Z}(\ell + c - \log Z) - (K-1)\frac{\exp(c)}{Z}(c - \log Z). \tag{D.21}$$

Now let us consider the value of this in the limit of large ℓ, $\ell \gg c$. First, let us expand $\frac{1}{Z}$ using the first order expansion $(1+x)^{-1} \approx 1 - x$ for $|x| \ll 1$.

$$\frac{1}{Z} = \frac{1}{\exp(c+\ell) + (K-1)\exp(c)} \tag{D.22}$$

$$= \frac{1}{\exp(\ell+c)}\frac{1}{1+(K-1)\exp(-\ell)} \tag{D.23}$$

$$= \exp(-\ell - c)\left(1 - (K-1)\exp(-\ell) + O(\exp(-\ell)^2)\right). \tag{D.24}$$

Second, let us expand $\log Z$ using the first order expansion $\log(1+x) \approx x$ for $|x| \ll 1$.

$$\log Z = \log(\exp(c+\ell) + (K-1)\exp(c)) \tag{D.25}$$

$$= \log(\exp(c+\ell)(1 + (K-1)\exp(-\ell))) \tag{D.26}$$

$$= c + \ell + \log(1 + (K-1)\exp(-\ell)) \tag{D.27}$$

$$= c + \ell + (K-1)\exp(-\ell) + O\left(\exp(-\ell)^2\right). \tag{D.28}$$

Taking Eqs (D.24,D.28) and subbing back into Eq (D.21), keeping terms to first order in $\exp(-\ell)$ we get

$$\mathcal{H}_{\text{softmax}} \approx (K-1)\exp(-\ell)(1+\ell) \tag{D.29}$$

as required.

□

D.3 Experimental Evaluation

In order to empirically verify the bounds above, we compare the exact entropy to these first-order approximations for both methods' worst-case scenarios. We find the approximation to be highly accurate for inputs > 10, with proportional error $\approx 10^{-6}$ for each.

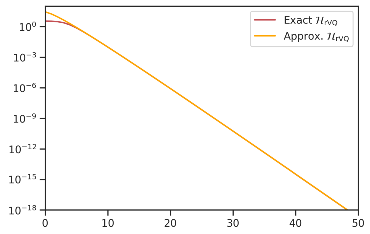


Figure D.4: rVQ worst-case entropy as a function of d, calculated exactly and using Eq (D.17), for $\delta = 1$. Note this is a logarithmic plot.

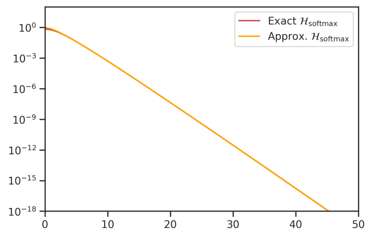


Figure D.5: Softmax worst-case entropy as a function of d, calculated exactly and using Eq (D.29), for $c = 0$. Note this is a logarithmic plot.

As an additional check on the rVQ results, we create random codebooks of embeddings uniformly distributed over the hypersphere with radius 0.5 and calculate $\mathcal{H}$ as a function of d. We do this for 20,000 sampled codebooks per value of d, each of 256 entries, in an embedding space with $d_e = 32$. The entropy we get from simulation shows an entirely different trend from the 'worst-case' calculations. This is reasonable as the worst-possible arrangement is very unlikely to occur.

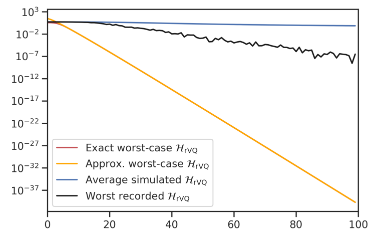


Figure D.6: rVQ entropy as a function of d, calculated for the worst case both exactly and using Eq (D.17), for $\delta = 1$, along with the average entropy from simulated codebooks with codebook embeddings uniform over the radius 0.5 hypersphere and the worst recorded entropy from that simulation procedure at each distance. Note this is a logarithmic plot.

D.4 Proof of Corollary 2.1

Proof. In big-O notation, viewing each as functions of their underlying neural parameterisation,

$$\mathcal{H}_{\mathrm{rVQ}} = O\left(d \exp\left(-\delta d\right)\right) \tag{D.30}$$

and

$$\mathcal{H}_{\mathrm{softmax}} = O\left(\ell \exp\left(-1 \times \ell\right)\right) \tag{D.31}$$

So if $\delta < 1$, the for large network outputs $\mathcal{H}_{\mathrm{rVQ}} > \mathcal{H}_{\mathrm{softmax}}$. $\square$

E Samples and Reconstructions

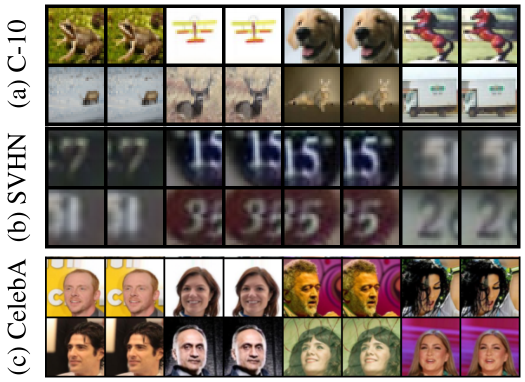


Figure E.7: Reconstructions: we demonstrate the high quality reconstructions of our approach for CIFAR-10, SVHN and CelebA. In each pair, left is the reconstruction, right the original.

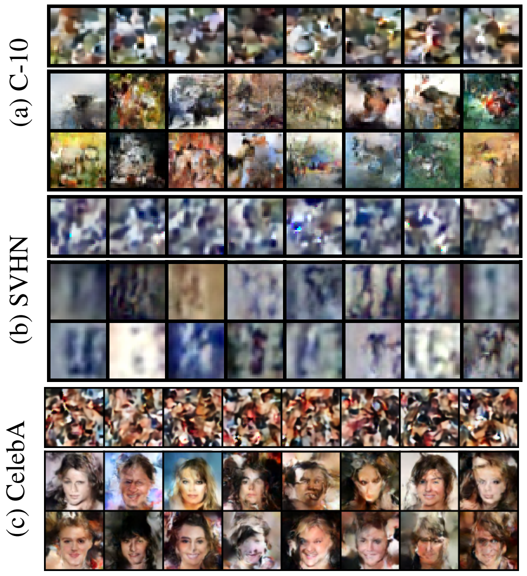


Figure E.8: Sampling: we perform ancestral sampling for single-layer rVQ–VAE baselines (top row) and our $L = 32$ models (middle and bottom), for CIFAR-10, SVHN and CelebA.

F Compression using RRVQ models

For our $L = 5$ models, our latents $\vec{\mathbf{z}}$ are in 5 layers of size $\mathbf{M} = \{16 \times 16, 8 \times 8, 4 \times 4, 2 \times 2, 1 \times 1\}$. $\{\mathbf{E}_{\mu,\ell}, \mathbf{E}_{\Sigma,\ell}\}$. 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.

In Fig F.9 we compress (top) CelebA images using (middle) our $L = 5$ model and (bottom) using JPEG to the same compression ratio (CR) [same experimental protocol as Gregor et al. (2016). We are compressing 64×64 images into 2275 bits, a CR of $\frac{98304}{2275} \approx 43$. Our approach outperforms JPEG, maintaining more visual information. Unlike JPEG, ours does not introduce blocky artefacts.

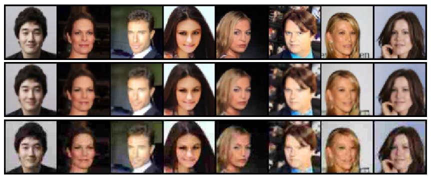


Figure F.9: *Top*: Original image, *Middle*: RRVQ $L = 5$ compression, *Bottom*: JPEG at same compression ratio. Best viewed zoomed in.

G Ablation Study

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

p:		Direct-Cat		Embedding-Cat
CIFAR-10				
$\sigma = 1$	Train:	5.00	Train:	5.06
	Test:	5.05	Test:	5.11
σ learnt	Train:	5.08	Train:	**4.40** *(RRVQ)*
	Test:	5.10	Test:	**4.65** *(RRVQ)*
SVHN				
$\sigma = 1$	Train:	3.44	Train:	3.51
	Test:	3.32	Test:	3.41
σ learnt	Train:	–	Train:	**3.02** *(RRVQ)*
	Test:	–	Test:	**2.96** *(RRVQ)*

How does our approach compare to other possible hierarchical extensions of rVQ–VAEs? For models with $L = 5$ we trained various ablations of our proposal: with or without a learnt codebook of covariances; and with the generative model represented either via a Relaxed-VQ lookup or directly outputting a (log) probability over embeddings. All of these have an ELBO as in Eq (5), but we are varying how we parameterise p and q. In Table G.1 we show the test and train bpd obtained for these hierarchical discrete VAEs.

We can see that $\sigma = 1$ with *direct probabilities in* p (the top-left corner results for each dataset), arguably being the most naive approach, is outperformed by $\approx$ half of a bpd by full RRVQ (bottom-right). Interestingly, if any of the two changes (p via embedding & σ learnt) are made in isolation, the result is either no substantial change in performance (if anything, slight degradation) or rendered training so unstable that it was impossible to obtain a result. Clearly a synergistic property takes place here; these two changes made *together* lead to improved performance of the models.

H MLP rVQ-VAEs

For completeness, in Fig H.10 we train an MLP rVQ-VAE on our colour swatch data, to demonstrate that samples from such a model show consistent colour cast (further, samples show new colours beyond the training set). That is, ancestral samples look like the training data (ie with consistent colour) unlike single-latent-layer convolutional models.

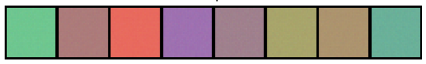


Figure H.10: MLP-rVQ-VAE samples, trained on toy colour-swatch dateset.