Tackling Over-pruning in Variational Autoencoders

Serena Yeung [1] **Anitha Kannan** [2] **Yann Dauphin** [2] **Li Fei-Fei** [1]

Abstract

Variational autoencoders (VAE) are directed generative models that learn factorial latent variables. As noted by Burda et al. (2015), these models exhibit the problem of factor over-pruning where a significant number of stochastic factors fail to learn anything and become inactive. This can limit their modeling power and their ability to learn diverse and meaningful latent representations. In this paper, we evaluate several methods to address this problem and propose a more effective model-based approach called the *epitomic variational autoencoder* (eVAE). The so-called epitomes of this model are groups of mutually exclusive latent factors that compete to explain the data. This approach helps prevent inactive units since each group is pressured to explain the data. We compare the approaches with qualitative and quantitative results on MNIST and TFD datasets. Our results show that eVAE makes efficient use of model capacity and generalizes better than VAE.

1. Introduction

Unsupervised learning holds the promise of learning the inherent structure in data so as to enable many future tasks including generation, prediction and visualization. Generative modeling is an approach to unsupervised learning wherein an explicit stochastic generative model of data is defined; independent draws from this model are to produce samples from the underlying data distribution, while the learned latent structure is useful for prediction, classification and visualization tasks.

Variational autoencoder (VAE) (Kingma & Welling, 2014) is an example of one such generative model. VAE pairs a top-down generative model with a bottom-up recognition network for amortized probabilistic inference, and jointly trains them to maximize a variational lower bound on the data likelihood. A number of recent works use VAE as a modeling framework, including iterative conditional generation of images (Gregor et al., 2015) and conditional future frame prediction (Xue et al., 2016).

The generative model of VAE has a set of independent stochastic latent variables that govern data generation; these variables aim to capture various factors of variation. However, a number of studies (Bowman et al., 2015; Kaae Sonderby et al., 2016; Kingma et al., 2016) have noted that straightforward implementations that optimize the variational bound on the probability of observations converge to a solution in which only a small subset of the stochastic latent units are active. While it may seem advantageous that the model can automatically regularize itself, the optimization leads to learning a suboptimal generative model by limiting its capacity to use only a small number of stochastic units. We call this well-known issue with training VAE as 'over-pruning'. Existing methods propose training schemes to tackle the over-pruning problem that arises due to pre-maturely deactivating units (Bowman et al., 2015; Kaae Sonderby et al., 2016; Kingma et al., 2016). For instance, (Kingma et al., 2016) enforces minimum KL contribution from subsets of latent units while (Bowman et al., 2015) use KL cost annealing. However, these schemes are hand-tuned and takes away the principled regularization scheme that is built into VAE.

We address the over-pruning problem using a model-based approach. We present an extension of VAE called epitomic variational autoencoder (Epitomic VAE, or eVAE, for short) that automatically learns to utilize its model capacity more effectively, leading to better generalization. The motivation for eVAE stems from the following observation: Consider the task of learning a D-dimensional representation for the examples in a given dataset. A single example in the dataset can be sufficiently embedded in a smaller K-dimensional ($K \ll D$) subspace of D. However, different data points may need different subspaces, hence the need for D. Sparse coding methods also exploit a similar hypothesis. Epitomic VAE exploits sparsity using an additional categorical latent variable in the encoder-decoder architecture of the VAE. Each value of the variable acti-

[1]Stanford University, Stanford, CA, USA. Work done during an internship at Facebook AI Research. [2]Facebook AI Research, Menlo Park, CA, USA. Correspondence to: Serena Yeung <serena@cs.stanford.edu>.

Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017. Copyright 2017 by the author(s).

vates only a contiguous subset of latent stochastic variables to generate an observation. This enables learning multiple shared subspaces such that each subspace specializes, and also increases the use of model capacity (Fig. 3), enabling better representation. The choice of the name *Epitomic VAE* comes from the fact that multiple miniature models with shared parameters are trained simultaneously.

The rest of the paper is organized as follows. We first describe variational autoencoders and mathematically show the model pruning effect in § 2 and § 3. We then present our epitomic VAE model in § 4 that overcomes these shortcomings. Experiments showing qualitative and quantitative results are presented in § 5. We discuss related work in § 6, and conclude in § 7.

2. Variational Autoencoders

The generative model of a VAE consists of first generating a sample from a D-dimensional stochastic variable $\mathbf{z}$ that is distributed according to a standard Gaussian:

$$p(\mathbf{z}) = \prod_{d=1}^{D} \mathcal{N}(z_d; 0, 1) \tag{1}$$

Each component z_i captures some latent source of variability in the data. Given $\mathbf{z}$, the N-dimensional observation $\mathbf{x}$ is generated from a parametric family of distributions such as a Gaussian:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_1(\mathbf{z}), \exp(f_2(\mathbf{z}))) \tag{2}$$

where f_1 and f_2 are non-linear deterministic functions of $\mathbf{z}$ modeled using neural networks, and θ denotes the parameters of the generative model.

The model is trained by optimizing the likelihood $p(X|\theta)$ using a dataset X of T *i.i.d.* samples. Since $p(\mathbf{z}|\mathbf{x})$ is intractable, VAE approximates the exact posterior using a variational approximation that is amortized across the training set, using a neural network (recognition network) with parameters ϕ. The resulting variational bound is

$$\begin{aligned} \log p_\theta(X) &= \sum_{t=1}^{T} \log \int_{\mathbf{z}} p_\theta(\mathbf{x}^{(t)}, \mathbf{z}) \\ &\geq \sum_{t=1}^{T} E_{q_\phi(\mathbf{z}|\mathbf{x}^{(t)})} \log p(\mathbf{x}^{(t)}|\mathbf{z}) - KL\Big[q_\phi(\mathbf{z}|\mathbf{x}^{(t)}) \parallel p(\mathbf{z})\Big] \end{aligned} \tag{3}$$

The model is trained using backpropagation to minimize:

$$\begin{aligned} \mathcal{C}_{vae} = &-\sum_{t=1}^{T} E_{q_\phi(\mathbf{z}|\mathbf{x}^{(t)})} \log p(\mathbf{x}^{(t)}|\mathbf{z}) \\ &+ \sum_{t=1}^{T}\sum_{d=1}^{D} KL\Big(q_\phi(z_d|\mathbf{x}^{(t)}) \parallel p(z_d)\Big) \end{aligned} \tag{4}$$

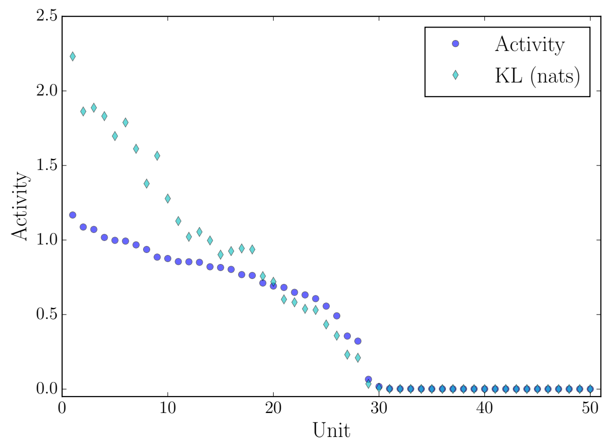


Figure 1: Unit activity and KL term for a 50-unit VAE (sorted by activity). Activity shows correlation with KL term. Highly active units have a high KL with the prior; this mismatch creates a discrepancy between reconstruction and generation performance.

$\mathcal{C}_{vae}$ trade-offs between explaining the data (first term) and ensuring that the posterior distribution is close to the prior $p(\mathbf{z})$ (second term).

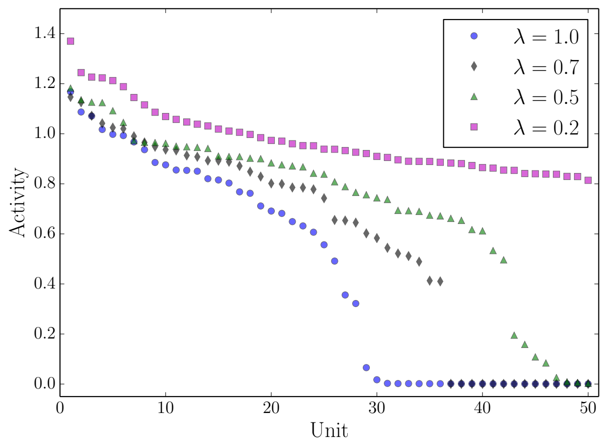
3. Over-pruning

We can better understand over-pruning in the VAE by considering different ways to minimize the sum of two terms in $\mathcal{C}_{vae}$. The first term encourages proper reconstruction while the second term captures the divergence between the posterior $q(z_d)$ and its Gaussian distributed prior $p(z_d)$, independently for each component. The easiest way to minimize the sum is to have a large number of components collapse to the prior $p(z_d)$ to compensate for a few highly non-Gaussian components that help reconstruction. This is achieved by turning off the corresponding component[1]. This behavior is noticeable in the early iterations of training when the model for $\log p(\mathbf{x}|\mathbf{z})$ is quite impoverished, and improvement to the loss can be easily obtained by optimizing this KL term. However, once the units have become inactive, it is almost impossible to resurrect them.

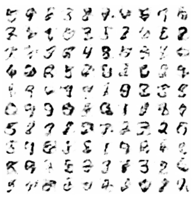
A quantity that is useful in understanding this effect is the activity level of a unit. Following (Burda et al., 2015), we define a unit to be used, or "active", if $A_u = \text{Cov}_x(\mathbb{E}_{u\sim q(u|\mathbf{x})}[u]) > 0.02$. In Figure 1 we plot the activity level and KL-divergence in $\mathcal{C}_{vae}$ for each component of a 50-unit VAE trained on MNIST. This result illustrates that all inactive units have collapsed to the prior, whereas active units are relatively far from the prior.

One could argue that over-pruning is a feature since the

[1]log variance is modeled using the neural network, so turning it off to 0 corresponds to a variance of 1.



(b) $\lambda = 1$


(c) $\lambda = 0.5$

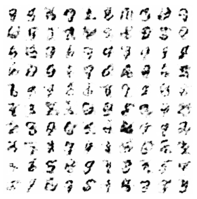

(d) $\lambda = 0.2$

Figure 2: Sorted activity of latent units and corresponding generations on MNIST, for a 50-d VAE with a hidden layer of 500 units. Shown for varying values of the KL weight λ. When $\lambda = 1$, only 30 units are active. As λ is decreased, more units are active; however generation does not improve since the model uses the capacity to model increasingly well only regions of the posterior manifold near training samples (see reconstructions in Fig. 9).

model seems to discard unnecessary capacity. However, we observe over-pruning even when the model underfits the data. Moreover, over-pruning creates a discrepancy between training and generation time since it allows some components $q(z_d)$ to be highly different from the prior. Instead, it is desirable for each individual component to be close to the prior, since generation occurs by sampling from the prior.

3.1. Weighting the KL term

One approach to reducing over-pruning is to introduce a trade-off between the two terms using a parameter λ:

$$-E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] + \lambda \sum_{i=1}^{D} KL\Big(q_\phi(z_i|\mathbf{x}) \parallel p(z_i)\Big)$$

λ controls the importance of keeping the information encoded in $\mathbf{z}$ close to the prior. $\lambda = 0$ corresponds to a vanilla autoencoder, and $\lambda = 1$ to the correct VAE objective. Fig. 2 shows the effect of λ on unit activity and generation. While tuning down λ increases the number of active units, samples generated from the model are still poor. This is because at small values of λ, the model becomes closer to a vanilla autoencoder and hence spends its capacity in ensuring that reconstruction of the training set is optimized (sharper reconstructions as a function of λ are shown in Appendix § 9.1), at the cost of generation capability.

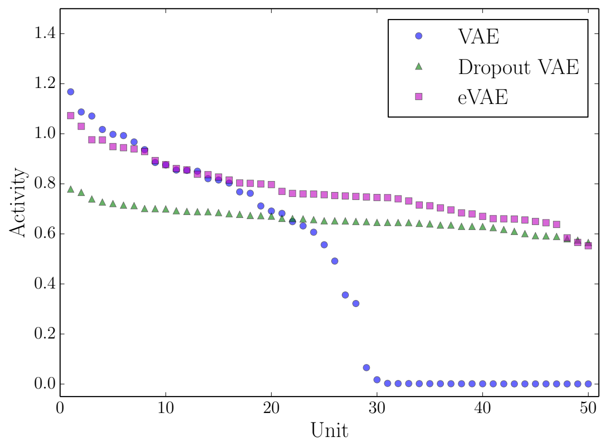


Figure 3: Adding dropout to a VAE (here, dropout rate 0.5 is shown) can prevent the model from pruning units, shown for MNIST. However, in contrast to eVAE, it uses the additional units to encode redundancy, not additional information, and therefore does not address the problem. eVAE is able to utilize the full latent capacity with all units active. Compare generation results for dropout VAE with eVAE in Fig. 4 and Fig. 6, respectively.

3.2. Dropout VAE

Another approach is to add dropout to the latent variable $\mathbf{z}$ of the VAE (Dropout VAE). While this increases the number of active units (Fig. 3), it generalizes poorly as it uses the dropout layers to merely replicate representation. This results in blurriness in both generation and reconstruction, and illustrates that simply utilizing additional units is not sufficient for proper utilization of these units to model additional factors of variation, as seen in Fig. 4.

4. eVAE: A model-based approach

We propose epitomic variational autoencoders (eVAE) to overcome the over-pruning problem of VAEs. We base this on the observation that while we may need a D-dimensional representation to accurately represent every example in a dataset, each individual example can be represented with a smaller K-dimensional subspace. As an example, consider MNIST with its variability in terms of digits, strokes and thickness of ink, to name a few. While the overall D is large, it is likely that only a few K dimensions of D are needed to capture the variability in strokes of some digits (see Fig. 5). Epitomic VAE can be viewed as a variational autoencoder with latent stochastic dimension D that is composed of a number of smaller variational autoen-

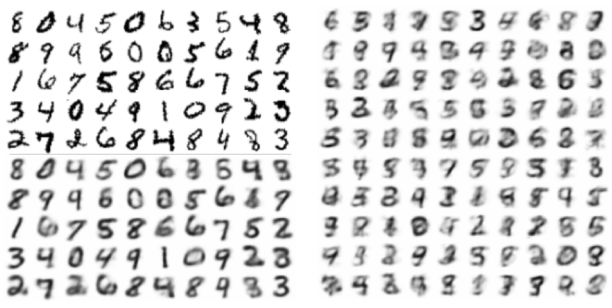

(a) Reconstruction (b) Generation

Figure 4: Dropout VAE on MNIST: Both generation and reconstruction are blurry. This is because the additional active units are used to encode redundant information.

coders called *epitomes*, such that each epitome partially shares its encoder-decoder architecture with other epitomes in the composition. In this paper, we assume simple structured sparsity for each epitome: in particular, only K *contiguous* dimensions of D are active[2].

The generative process can be described as follows: A D-dimensional stochastic variable $\mathbf{z}$ is drawn from a standard multivariate Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$. In tandem, an epitome is implicitly chosen through an epitome selector variable y, which has a uniform prior over possible epitomes. The N-dimensional observation $\mathbf{x}$ is then drawn from a Gaussian distribution:

$$p_\theta(\mathbf{x}|y, \mathbf{z}) = \mathcal{N}(\mathbf{x}; f_1(\mathbf{m}_y \odot \mathbf{z}), \exp(f_2(\mathbf{m}_y \odot \mathbf{z}))) \quad (5)$$

$\mathbf{m}_y$ enforces the epitome constraint: it is also a D-dimensional vector that is zero everywhere except K contiguous dimensions that correspond to the epitome dictated by y. $\odot$ is element-wise multiplication between the two operands. Thus, $\mathbf{m}_y$ masks the dimensions of $\mathbf{z}$ other than those dictated by the choice of y. Fig. 5 illustrates this for an 8-d $\mathbf{z}$ with epitome size $K = 2$, such that there are four possible epitomes (the model also allows for overlapping epitomes, but this is not shown for illustration purposes). Epitome structure is defined using size K and stride s, where $s = 1$ corresponds to full overlap in D dimensions[3]. Our model generalizes the VAE and collapses to a VAE when $D = K = s$.

[2]The model also allows for incorporating other forms of structured sparsity.

[3]The strided epitome structure allows for learning $O(D)$ specialized subspaces, that when sampled during generation can each produce good samples. In contrast, if only a simple sparsity prior is introduced over arbitrary subsets (e.g. with Bernoulli latent units to specify if a unit is active for a particular example), it can lead to poor generation results, which we confirmed empirically but do not report. The reason for this is as follows: due to an exponential number of potential combinations of latent units, sampling a subset from the prior during generation cannot be straightforwardly guaranteed to be a good configuration for a subconcept in the data, and often leads to uninterpretable samples.

$f_1(\diamond)$ and $f_2(\diamond)$ define non-linear deterministic transformations of $\diamond$ modeled using neural networks. Note that the model does not snip off the K dimensions corresponding to an epitome, but instead ignores the $D - K$ dimensions that are not part of the chosen epitome. While the same deterministic functions f_1 and f_2 are used for any choice of epitome, the functions can still specialize due to the sparsity of their inputs. Neighboring epitomes will have more overlap than non-overlapping ones, which manifests itself in the representation space; an intrinsic ordering in the variability is learned.

4.1. Overcoming over-pruning

Following (Kingma & Welling, 2014), we use a recognition network $q(\mathbf{z}, y|\mathbf{x})$ for approximate posterior inference, with the functional form

$$\begin{aligned} q(\mathbf{z}, y|\mathbf{x}) &= q(y|\mathbf{x})q(\mathbf{z}|y, \mathbf{x}) \\ &= q(y|\mathbf{x})\mathcal{N}(\mathbf{z}; \mathbf{m}_y \odot \mu, \exp(\mathbf{m_y} \odot \phi)) \end{aligned} \quad (6)$$

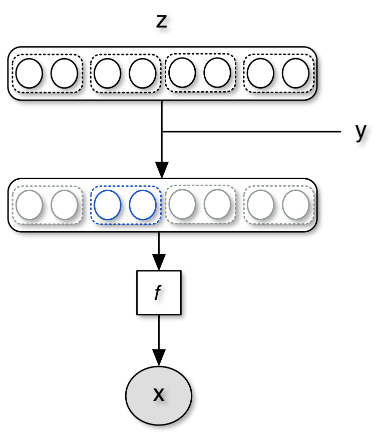
where $\mu = \mathbf{h_1}(\mathbf{x})$ and $\phi = \mathbf{h_2}(\mathbf{x})$ are neural networks that map $\mathbf{x}$ to D-dimensional space. We use a similar masking operation as the generative model. Unlike the generative model (eq. 5), the masking operation defined by y operates directly on outputs of the recognition network that characterizes the parameters of $q(\mathbf{z}|y, \mathbf{x})$. Similar to VAE, the lower bound on the log probability of a dataset can be derived, leading to the cost function (negative bound):

$$\begin{aligned} \mathcal{C}_{evae} = &- \sum_{t=1}^{T} E_{q(\mathbf{z},y|\mathbf{x}^{(t)})}[\log p(\mathbf{x}^{(t)}|y, \mathbf{z})] \\ &+ \sum_{t=1}^{T} KL\Big[q_\phi(y|\mathbf{x}^{(t)}) \parallel p_\theta(y)\Big] \\ &+ \sum_{t=1}^{T}\sum_{y} q_\phi(y|\mathbf{x}^{(t)}) KL\Big[q_\phi(\mathbf{z}|y, \mathbf{x}^{(t)}) \parallel p_\theta(\mathbf{z})\Big] \end{aligned} \quad (7)$$

eVAE departs from VAE in how the contribution from the KL term is constrained. Consider the third term expanded:

$$\begin{aligned} &\sum_{t=1}^{T}\sum_{y} q_\phi(y|\mathbf{x}^{(t)}) KL\Big[q_\phi(\mathbf{z}|y, \mathbf{x}^{(t)}) \parallel p_\theta(\mathbf{z})\Big] \\ &= \sum_{t=1}^{T}\sum_{y} q_\phi(y|\mathbf{x}^{(t)}) \sum_{d=1}^{D} \mathbf{1}[m_{d,y} = 1] KL\Big[q(z_d|\mathbf{x}^{(t)}) \parallel p(z_d)\Big], \end{aligned} \quad (8)$$

where $\mathbf{1}[\star]$ is an indicator variable that evaluates to 1 if only if its operand $\star$ is true. Unlike in VAE where this KL



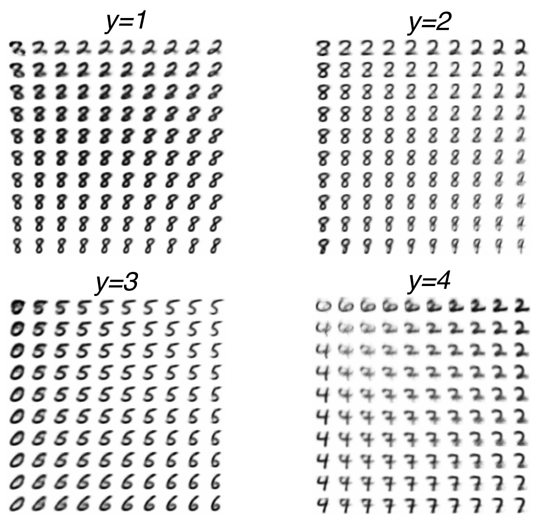


Figure 5: Left: Illustration of an epitomic VAE with dimension D=8, epitome size K=2 and stride S=2. In this depiction, the second epitome is active. Right: Learned manifolds on MNIST for 4 different epitomes in a 20-d eVAE with size $K = 2$ and stride $s = 1$. We observe that each epitome specializes on a coherent subset of examples; this enables increasing the diversity of the samples generated while maintaining quality of the samples when the latent dimension is large.

term decomposes into independent KL for each z_i, contiguous dimensions of $\mathbf{z}$ are constrained by the choice of y. In addition, the number of KL terms that will contribute to $\mathcal{C}_{evae}$ for an input $\mathbf{x}^{(t)}$ is exactly K by model design, with the other $D - K$ dimensions set to provide contribution of zero. Thus, only a fraction of examples in the training set contributes a possible non-zero value to z_d's KL term in $\mathcal{C}_{evae}$. This gives eVAE the ability to use more total units without having to prematurely prune the model to optimize the bound. In contrast, for $\mathcal{C}_{vae}$ to have a small contribution from the KL term of a particular z_d, it has to infer that unit to have zero mean and unit variance for many examples in the training set. In practice, this results in VAE completely inactivating units.

Fig. 3 compares the activity levels of eVAE with VAE and dropout VAE. Even though Dropout VAE has similar activity profile to eVAE, its generative model is impoverished as it focuses on replicating representation as opposed to modeling variability. This can be evidenced by comparing the generation results from the two models, in Fig. 4 and Fig. 6.

4.2. Training

The generative model and the recognition network are trained simultaneously, by minimizing $\mathcal{C}_{evae}$ in Eq. 7.

For the stochastic continuous variable $\mathbf{z}$, we use the reparameterization trick as in VAE, which reparameterizes the recognition distribution in terms of auxiliary variables with fixed distributions. This allows efficient sampling from the posterior distribution as it becomes a deterministic function of inputs and auxiliary variables.

For the discrete variable y, we cannot use the reparameterization trick. We therefore approximate $q(y|\mathbf{x})$ by a point estimate y^* so that $q(y|\mathbf{x}) = \delta(y - y^*)$, where δ evaluates to 1 only if $y = y^*$ and the best $y^* = \arg\min \mathcal{C}_{evae}$. We also explored modeling $q(y|\mathbf{x}) = Mult(h(\mathbf{x}))$ as a discrete distribution with h being a neural network. In this case, the backward pass requires either using REINFORCE or passing through gradients for the categorical sampler. In our experiments, we found that these approaches did not work well, especially when the number of possible values of y becomes large. We leave this as future work to explore.

The recognition network first computes μ and ϕ. It is then combined with the optimal y^* for each example, to arrive at the final posterior. The model is trained using a simple algorithm outlined in Alg. 1. Backpropagation with minibatch updates is used, with each minibatch constructed to be balanced with respect to epitome assignment.

5. Experiments

We present experimental results on two datasets, MNIST (LeCun et al., 1998) and Toronto Faces Database (TFD) (Susskind et al., 2010). We use standard splits for both MNIST and TFD. In our experiments, the encoder and decoder are fully-connected networks, and we show results for different depths and number of units of per layer. ReLU nonlinearities are used, and models are trained using the Adam update rule (Kingma & Ba, 2014) for 200 epochs (MNIST) and 250 epochs (TFD), with base

Algorithm 1 Learning Epitomic VAE

```
1: $\theta$, $\phi$ ←Initialize parameters
2: for until convergence of parameters ($\theta$, $\phi$) do
3:     Assign each $\mathbf{x}$ to its best $y^* = \arg\min \mathcal{C}_{evae}$
4:     Randomize and partition data into minibatches, with each minibatch having proportionate number of examples $\forall\, y$
5:     for k $\in$ numbatches do
6:         Update model parameters using $k^{th}$ minibatch consisting of $\mathbf{x}, y$ pairs
7:     end for
8: end for
```

learning rate 0.001. We emphasize that in all experiments, we optimize the correct lower bound for the corresponding models.

5.1. Qualitative results: Reconstruction vs. Generation

We first qualitatively illustrate the ability of eVAE to overcome over-pruning and utilize latent capacity to model greater variability in data. Fig. 6 compares generation results for VAE and eVAE for different dimensions D of latent variable $\mathbf{z}$. With $D = 2$, VAE generates realistic digits but suffers from lack of diversity. When D is increased to 5, the generation exhibits some greater variability but also begins to degrade in quality. As D is further increased to 10 and 20, the degradation continues. Contrast this with eVAE performance on generation: as the dimension D of $\mathbf{z}$ is increased while maintaining epitomes of size $K = 2$, eVAE is able to model greater variability in the data. Highlighted digits in the 20-d eVAE show multiple styles such as crossed versus un-crossed 7, and pointed, round, thick, and thin 4s. For both models, reconstruction improves with increasing D as provided in Appendix Fig. 10.

5.2. Choice of epitome size

We next investigate how the choice of epitome size, K, affects generation performance. We measure sample quality using the Parzen window estimator (Rifai et al., 2012). Fig. 7 shows the Parzen log-density for different choices of epitome size on MNIST, with encoder and decoder consisting of a single deterministic layer of 500 units. Epitomes are non-overlapping, and the results are grouped by total dimension D of the latent variable $\mathbf{z}$. For comparison, we also show the log-density for VAE models with the same dimension D, and for mixture VAE (mVAE), an ablative version of eVAE where parameters are not shared. mVAE can also be seen as a mixture of independent VAEs trained in the same manner as eVAE. The number of deterministic units in each mVAE component is computed so that the total number of parameters is comparable to eVAE.

As we increase D, the performance of VAE drops significantly, due to over-pruning. In fact, the number of active units for VAE are 8, 22 and 24, corresponding to D values of 8, 24 and 48, respectively. In contrast, eVAE performance increases as we increase D, with an epitome size K that is significantly smaller than D. This confirms the advantage of using eVAE to ensure good generation performance. Table 1 provides more comparisons. eVAE also performs comparably or better than mVAE at all epitome sizes. An explanation is that due to the parameter sharing in eVAE, each epitome benefits from general features learned across the training set.

5.3. Increasing complexity of encoder and decoder

Here we investigate the impact of encoder and decoder architectures with respect to over-pruning and generation performance. We vary model complexity through number of layers L of deterministic hidden units, and number of hidden units H in each deterministic layer. Table 1 shows the Parzen log-densities of VAE, mVAE and eVAE models trained on TFD with different latent dimension D (See Appendix §9.3 for MNIST). All epitomes are non-overlapping and of size $K = 5$. We observe that for VAE, increasing the number of hidden units H (e.g. from 500 to 1000) for a fixed network depth L has a negligible effect on the number of active units and performance. On the other hand, as the depth of the encoder and decoder L is increased, the number of active units in VAE decreases though performance is still able to improve. This illustrates that increase in the complexity of the interactions through multiple layers counteract the perils of the over-pruning. However, this comes with the cost of substantial increase in the number of model parameters to be learned.

In contrast, for any given model configuration, eVAE is able to avoid the over-pruning effect in the number of active units and outperform VAE. Table 1 also shows results for mVAE, the ablative version of eVAE where parameters are not shared. The number of deterministic units per layer in each mVAE component is computed to have total number of parameters comparable to eVAE. These results are in line with the intuition that parameter sharing is helpful in more challenging settings when each epitome can also benefit from general features learned across the training set.

5.4. Log-likelihood evaluation

Table 2 shows importance weighted estimates as the mean of $\mathcal{L}_{5000}$ for VAE and eVAE on MNIST, with different dimensions D of latent variable z. All models have 2 deterministic hidden layers of 200 units, and are trained in 8 stages with a learning rate of $0.001 \cdot 10^{\frac{-1}{7}}$ for 3^i epochs, for each stage $i = 0...7$, following Burda et al. (2015). The VAE model has 20 active units at all D, so is not able

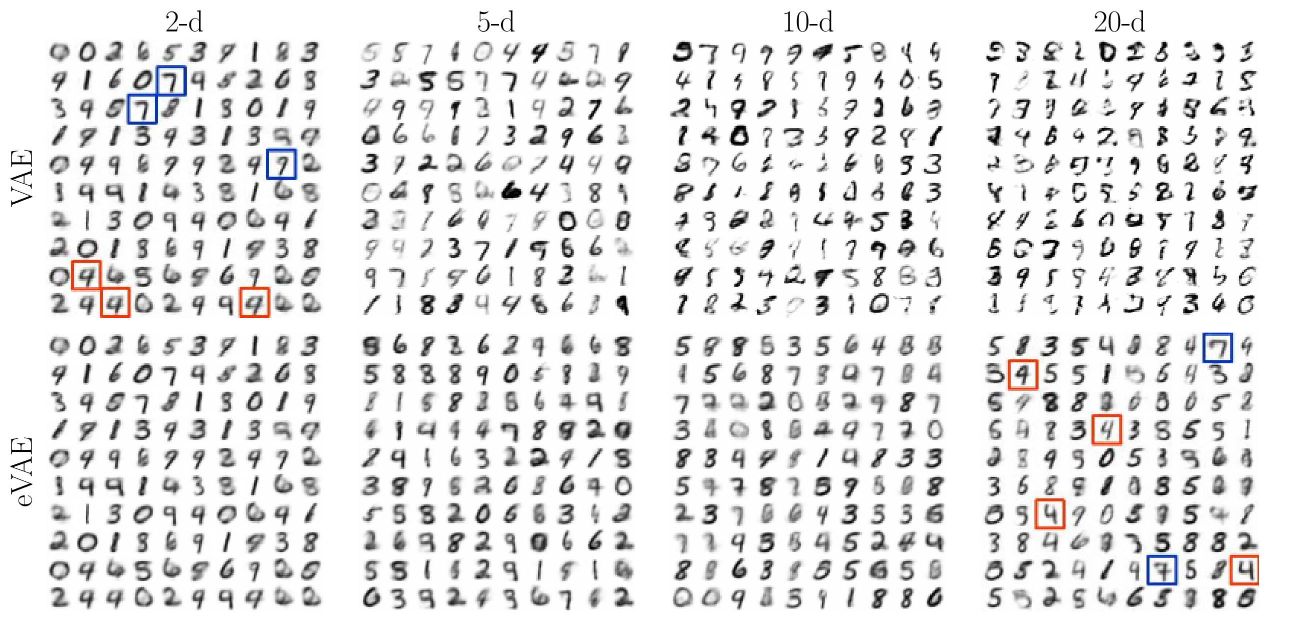


Figure 6: Generations from VAE and eVAE models for different dimensions of latent variable $\mathbf{z}$. In this experiment, we maintain a simple encoder-decoder architecture with a single layer of 500 deterministic units (samples from best architecture are in Fig. 8). Across each row are 2-d, 5-d, 10-d, and 20-d models. VAE generation quality degrades as latent dimension increases, and it is unable to effectively use added capacity to model greater variability. eVAE overcomes the problem by modeling multiple shared subspaces, here 2-d (overlapping) epitomes are maintained as the latent dimension is increased. Learned epitome manifolds from the 20-d model are shown in Fig. 5. Boxed digits highlight the difference in variability that the VAE vs. eVAE model is able to achieve.

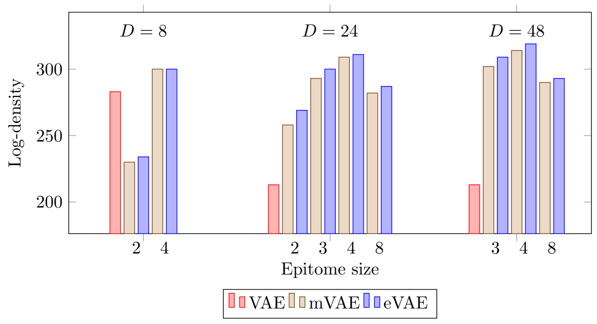


Figure 7: Epitome size vs. Parzen log-density in nats for different values of D on MNIST. For each D, the optimal epitome size is significantly smaller than D.

		H=500		H=1000	
		L=2	L=3	L=2	L=3
D=15	VAE	2173(15)	2180(15)	2149(15)	2116(15)
	mVAE	2276(15)	2314(15)	**2298**(15)	2343(15)
	eVAE	**2298**(15)	**2353**(15)	2278(15)	**2367**(15)
D=25	VAE	2067(25)	2085(25)	2037(25)	2101(25)
	mVAE	2287(25)	2306(25)	**2332**(25)	2351(25)
	eVAE	**2309**(25)	**2371**(25)	2297(25)	**2371**(25)
D=50	VAE	1920(50)	2062(29)	1886(50)	2066(30)
	mVAE	2253(50)	2327(50)	2280(50)	2358(50)
	eVAE	**2314**(50)	**2359**(50)	**2302**(50)	**2365**(50)

Table 1: Parzen log-densities in nats of VAE, mVAE and eVAE for increasing model capacity on TFD. Across each row shows performance as the number of encoder and decoder layers L increases for a fixed number of hidden units H in each layer, and as H increases. Number of active units are indicated in parentheses.

to improve performance with increasing D. On the other hand, eVAE is able to leverage the additional latent capacity to improve on the log-likelihood. Note that these results can be improved through the tighter lower bound of IWAE (Burda et al., 2015), but this is an orthogonal consideration since epitomic training can also improve IWAE.

5.5. Sample-based evaluation

In Table 3 we compare the generative performance of eVAE with other models through their samples. Encoders and decoders have $L = 2$ layers of $H = 1000$ deterministic units. $D = 8$ for MNIST, and $D = 15$ for TFD. VAE, mVAE, and eVAE refer to the best performing models over all architectures from Table 1. For MNIST, the VAE model is $(L, H, D) = (3, 500, 8)$, mVAE is $(3, 1000, 24)$, and eVAE is $(3, 500, 48)$. For TFD, the VAE model is $(3, 500, 15)$, mVAE is $(3, 1000, 50)$, and eVAE is $(3, 500, 25)$. We observe that eVAE significantly improves over VAE and is competitive with several state-of-the-art models, notably Adversarial Autoencoders. Samples from eVAE on MNIST and TFD are shown in Fig. 8.

D	VAE	eVAE
50	86.76	86.13
100	86.91	85.73
200	87.09	**85.53**

Table 2: Importance-weighted log-likelihood estimates as the mean of $\mathcal{L}_{5000}$ for VAE and eVAE on MNIST, with different dimensions D of latent variable z. All models have 2 deterministic hidden layers of 200 units, and eVAE models use epitomes of size $K = 25$. As D increases, VAE is not able to take advantage of additional units to improve performance, while eVAE is.

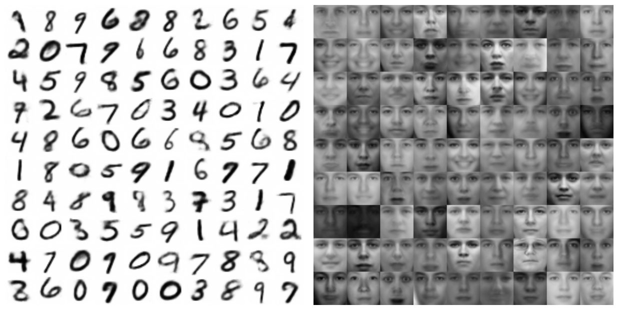

(a) (b)

Figure 8: eVAE samples for: (a) MNIST, and (b) TFD.

Method	MNIST(10K)	TFD(10K)
DBN (Hinton et al., 2006)	138 ± 2	1909 ± 66
Deep CAE (Bengio et al., 2013)	121 ± 1	2110 ± 50
Deep GSN (Thibodeau-Laufer et al., 2014)	214 ± 1	1890 ± 29
GAN (Goodfellow et al., 2014)	225 ± 2	2057 ± 26
GMMN + AE (Li et al., 2015)	282 ± 2	2204 ± 20
Adversarial AE (Makhzani et al., 2015)	$\mathbf{340 \pm 2}$	2252 ± 16
VAE	325 ± 2	2180 ± 20
mVAE	$\mathbf{338 \pm 2}$	2358 ± 20
eVAE	337 ± 2	$\mathbf{2371 \pm 20}$

Table 3: Parzen log-densities in nats on MNIST and TFD. VAE, mVAE, and eVAE refer to the best performing models over all architectures from Table 1.

6. Related Work

A number of applications use variational autoencoders as a building block. In (Gregor et al., 2015), a generative model for images is proposed in which the generator of the VAE is an attention-based recurrent model that is conditioned on the canvas drawn so far. (Eslami et al., 2016) proposes a VAE-based recurrent generative model that describes images as formed by sequentially choosing an object to draw and adding it to a canvas that is updated over time. In (Kulkarni et al., 2015), VAEs are used for rendering 3D objects. Conditional variants of VAE are also used for attribute specific image generation (Yan et al., 2015) and future frame synthesis (Xue et al., 2016). All these applications suffer from the problem of model over-pruning and hence have adopted strategies that takes away the clean mathematical formulation of VAE. We have discussed these in § 3. A complementary approach to the problem of model pruning in VAE was proposed in (Burda et al., 2015); the idea is to improve the variational bound by using multiple weighted posterior samples. Epitomic VAE provides improved latent capacity even when only a single sample is drawn from the posterior.

Methods to increase the flexibility of posterior inference are proposed in (Salimans et al., 2015; Rezende & Mohamed, 2016; Kingma et al., 2016). In (Rezende & Mohamed, 2016), posterior approximation is constructed by transforming a simple initial density into a complex one with a sequence of invertible transformations. (Kingma et al., 2016) augments the flexibility of the posterior through autoregression over projections of stochastic latent variables. However, the problem of over-pruning still persists: for instance, (Kingma et al., 2016) enforces a minimum information constraint to ensure all units are used.

Related is research in unsupervised sparse overcomplete representations, especially with group sparsity constraints *c.f.* (Gregor et al., 2011; Jenatton et al., 2011). In the epitomic VAE, we have similar motivations that enable learning better generative models of data.

7. Conclusion

This paper introduces Epitomic VAE, an extension of variational autoencoders, to address the problem of model over-pruning, which has limited the generation capability of VAEs in high-dimensional spaces. Based on the intuition that subconcepts can be modeled with fewer dimensions than the full latent space, epitomic VAE models the latent space as multiple shared subspaces that have learned specializations. We show how this model addresses the model over-pruning problem in a principled manner, and present qualitative and quantitative analysis of how eVAE enables increased utilization of the model capacity to model greater data variability. We believe that modeling the latent space as multiple structured subspaces is a promising direction of work, and allows for increased effective capacity that has potential to be combined with methods for increasing the flexibility of posterior inference.

8. Acknowledgments

We thank Marc'Aurelio Ranzato, Joost van Amersfoort and Ross Girshick for helpful discussions. We also borrowed the term 'epitome' from an earlier work of (Jojic et al., 2003).

References

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *ICML (1)*, pp. 552–560, 2013.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R, and S, Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Burda, Yuri, Grosse, Roger B., and Salakhutdinov, Ruslan. Importance weighted autoencoders. *ICLR*, 2015.

Eslami, S. M. Ali, Heess, Nicolas, Weber, Theophane, Tassa, Yuval, Kavukcuoglu, Koray, and Hinton, Geoffrey E. Attend, infer, repeat: Fast scene understanding with generative models. *CoRR*, abs/1603.08575, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gregor, Karol, Szlam, Arthur, and LeCun, Yann. Structured sparse coding via lateral inhibition. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2011.

Gregor, Karol, Danihelka, Ivo, Graves, Alex, Rezende, Danilo Jimenez, and Wierstra, Daan. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.046239*, 2015.

Hinton, Geoffrey E, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12, 2011.

Jojic, Nebojsa, Frey, Brendan J., and Kannan, Anitha. Epitomic analysis of appearance and shape. In *Proceedings of International Conference on Computer Vision*, 2003.

Kaae Sonderby, C., Raiko, T., Maale, L., Kaae Snderby, S., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Kingma, D.P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Kulkarni, T.D., Whitney, W., Kohli, P., and Tenenbaum, J.B. Deep convolutional inverse graphics network. *NIPS*, 2015.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

Li, Yujia, Swersky, Kevin, and Zemel, Rich. Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727, 2015.

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, and Goodfellow, Ian J. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015. URL http://arxiv.org/abs/1511.05644.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2016.

Rifai, Salah, Bengio, Yoshua, Dauphin, Yann, and Vincent, Pascal. A generative process for sampling contractive auto-encoders. *arXiv preprint arXiv:1206.6434*, 2012.

Salimans, T., Kingma, D.P., and Welling, M. Markov chain monte carlo and variational inference: Bridging the gap. *ICML*, 2015.

Susskind, Josh M, Anderson, Adam K, and Hinton, Geoffrey E. The toronto face database. *Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep*, 3, 2010.

Thibodeau-Laufer, Eric, Alain, Guillaume, and Yosinski, Jason. Deep generative stochastic networks trainable by backprop. 2014.

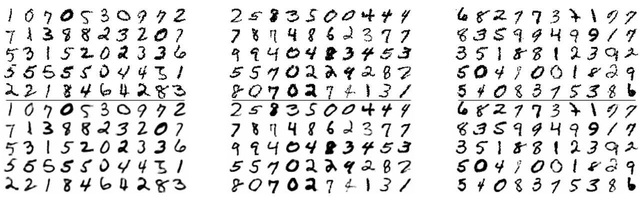
Xue, Tianfan, Wu, Jiajun, Bouman, Katherine L., and Freeman, William T. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *arXiv preprint arXiv:1607.02586*, 2016.

Yan, Xinchen, Yang, Jimei, Sohn, Kihyuk, and Lee, Honglak. Attribute2image: Conditional image generation from visual attributes. *CoRR*, abs/1512.00570, 2015.

9. Appendix

9.1. Effect of KL weight λ on reconstruction

We visualize VAE reconstructions as the KL term weight λ is tuned down to keep latent units active. The top half of each figure are the original digits, and the bottom half are the corresponding reconstructions. While reconstruction performance is good, generation is poor (Fig. 2). This illustrates that VAE learns to model well only regions of the posterior manifold near training samples, instead of generalizing to model well the full posterior manifold.



(a) $\lambda = 1.0$ (b) $\lambda = 0.5$ (c) $\lambda = 0.2$

Figure 9: Reconstructions for a 50-d VAE with KL weight $\lambda = 1$, 0.5, and 0.2. The top half of each figure are the original digits, and the bottom half are the corresponding reconstructions.

9.2. Effect of increasing latent dimension on reconstruction

In § 5.1 in the main paper, Fig. 6 shows the effect of increasing latent dimension on generation for VAE and eVAE models. Here we show the effect of the same factor on reconstruction quality for the models (Fig. 10). The top half of each figure are the original digits, and the bottom half are the corresponding reconstructions. As the dimension of the latent variable $\mathbf{z}$ increases from 2-d to 20-d, VAE reconstruction becomes very sharp (the best model), but generation degrades (Fig. 6). On the other hand, eVAE is able to achieve both good reconstruction and generation.

9.3. MNIST encoder and decoder complexity

Table 4 shows the effect of increasing complexity in the encoder and decoder complexity on MNIST. We vary model complexity through number of layers L of deterministic hidden units, and number of hidden units H in each deterministic layer. Parzen log-densities are provided for VAE, mVAE and eVAE models trained on MNIST with different latent dimension D. The effect of model complexity on active units and performance aligns with that for TFD in the main paper.

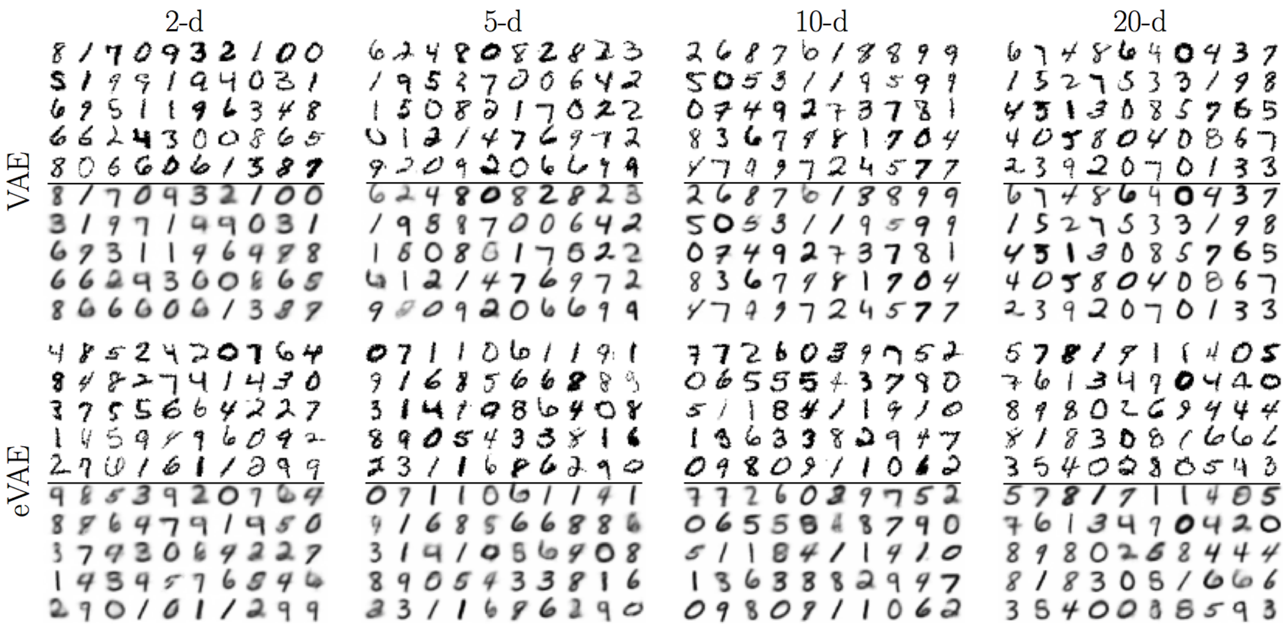


Figure 10: Reconstructions from VAE and eVAE models for different dimensions of latent variable $\mathbf{z}$. Across each row are 2-d, 5-d, 10-d, and 20-d models. The top half of each figure are the original digits, and the bottom half are the corresponding reconstructions. The eVAE models multiple shared subspaces by maintaining 2-d (overlapping) epitomes as the latent dimension is increased. In contrast to VAE, eVAE achieves both good reconstruction and generation.

		$H = 500$			$H = 1000$		
		$L = 1$	$L = 2$	$L = 3$	$L = 1$	$L = 2$	$L = 3$
$D = 8$	VAE	283(8)	292(8)	325(8)	283(8)	290(8)	322(6)
	mVAE	**300**(8)	328(8)	**337**(8)	309(8)	**333**(8)	**335**(8)
	eVAE	**300**(8)	**330**(8)	**337**(8)	**312**(8)	331(8)	334(8)
$D = 24$	VAE	213(22)	273(11)	305(8)	219(24)	270(12)	311(7)
	mVAE	309(24)	330(24)	**336**(24)	313(24)	**333**(24)	**338**(24)
	eVAE	**311**(24)	**331**(24)	**336**(24)	**317**(24)	**332**(24)	**336**(24)
$D = 48$	VAE	213(24)	267(13)	308(8)	224(24)	273(12)	309(8)
	mVAE	314(48)	**334**(48)	336(48)	315(48)	333(48)	**337**(48)
	eVAE	**319**(48)	**334**(48)	**337**(48)	**321**(48)	**334**(48)	**332**(48)

Table 4: Parzen log-densities in nats of VAE, mVAE and eVAE for increasing model parameters, trained on MNIST with different dimensions D of latent variable $\mathbf{z}$. For mVAE and eVAE models, the maximum over epitomes of size $K = 3$ and $K = 4$ is used. All epitomes are non-overlapping. Across each row shows performance as the number of encoder and decoder layers L increases for a fixed number of hidden units H in each layer, and as H increases. Number of active units are indicated in parentheses.