FEW-SHOT LEARNING WITH VARIATIONAL HOMOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot learning of generative models typically falls into two philosophies: conditional models, trained to generate new elements conditioned on observations, and hierarchical Bayesian models, which frame conditioning as inference of latent variables shared by a class. We modify the Variational Autoencoder framework to marry these two approaches, learning a hierarchical Bayesian model by a novel training procedure in which observations are encoded and decoded into new elements from the same class. We call this a Variational Homoencoder (VHE) and apply it to Caltech 101 Silhouettes and the Omniglot dataset. On Omniglot, our hierarchical PixelCNN outperforms existing models on joint likelihood of the data set, and achieves state-of-the-art results on both one-shot generation and one-shot classification tasks. The VHE framework also applies naturally to richer latent structures such as factorial or hierarchical categories. We illustrate this by training models to separate character content from simple variations in drawing style, and to generalise the style of an alphabet to new characters.

1 INTRODUCTION

Data-efficient generalisation is possible only with strong inductive biases. In machine learning such biases can come from hand design, as in the parametrisation of a model, or can be the result of a meta-learning algorithm. Furthermore they may be task-specific, as in discriminative modelling, or may describe the world causally so as to be more naturally reused across many tasks.

Recent work has approached one- and few-shot learning from all of these perspectives. Siamese Networks (Koch, 2015), Matching Networks (Vinyals et al., 2016), Prototypical Networks (Snell et al., 2017), MANNs (Santoro et al., 2016) and ARCs (Shyam et al., 2017) are all discriminatively trained for few-shot classification. Such models can achieve state-of-the-art performance at the task they were trained for, but provide no principled means for transferring knowledge to other tasks.

Other work such as Rezende et al. (2016) has focused on conditional generative models, which take one or a few observations from a class as input, and output a distribution over new elements \( p(x|D) \). Models following this philosophy may be used as classifiers despite not being explicitly trained for this purpose, by comparing conditional likelihoods. They may also be used to generate full sets incrementally as \( p(X) = \prod_i p(x_i|x_1, \ldots, x_{i-1}) \), discussed in Generative Matching Networks (Bartunov & Vetrov, 2016). However, such models do not expose any latent representation of shared structure within a class, and the distributions they induce are a more natural fit to sequences than to sets as they typically lack exchangeability.

Finally are hierarchical approaches which model shared structure through shared latent variables, as \( p(X) = \int p(c) \prod_i p(x_i|c) dc \). For example, Lake et al. (2015) develop a compositional generative model of handwritten characters which, by explicitly modelling shared structure from pen strokes to alphabets, achieves human-level performance at a variety of tasks. Salakhutdinov et al. (2013) follow a similar Bayesian philosophy for a model with less explicit knowledge built in: a Hierarchical Dirichlet Process prior on top of a two layer Boltzmann Machine. Most recently, the Neural Statistician (Edwards & Storkey, 2016) uses amortised inference to support learning in a more expressive generative model.

In this work we propose the Variational Homoencoder (VHE), aiming to combine several advantages of the models described above:
Figure 1: Single step of gradient training in various models. A VAE treats all datapoints as independent, so only a single random element need be encoded and decoded per gradient step. In order to share a latent variable \( c \) across a set of elements \( X \), a Neural Statistician feeds the set through both encoder and decoder networks each step. Instead, we bound the likelihood \( p(X) \) using only random subsamples \( D \) and \( x \) for encoding and decoding. Optionally, \( p(x|c) \) may be defined through a local latent variable \( z \).

1. Like conditional generative approaches, we train on a few-shot generation objective which matches how our model may be used at test time. However, by introducing an encoding cost, we simultaneously optimise a likelihood lower bound for a hierarchical generative model, in which structure shared across elements is made explicit by shared latent variables.

2. Previous work (Edwards & Storkey, 2016) has learned hierarchical Bayesian models by applying Variational Autoencoders to sets, such as classes of images. However, their approach requires feeding full sets through the model per gradient step (see Figure 1), rendering it intractable to train on large classes. In practice, they avoid computational limits by sampling smaller sub-classes as training data. We instead optimise a likelihood bound for the complete dataset, while still constructing this bound by subsampling. This approach not only can improve generalisation, but also departs from previous work by extending to models with richer latent structure, for which the joint likelihood cannot be factorised.

3. As with a Variational Autoencoder, the VHE objective includes both an encoding- and reconstruction-cost. However, sharing latent variables across an entire class reduces the encoding cost per element is significantly. This facilitates use of powerful autoregressive decoders, which otherwise often suffer from ignoring latent variables (Chen et al., 2016). We demonstrate the significance of this by applying a VHE to the Omniglot dataset. Using a PixelCNN decoder (Oord et al., 2016), our generative model is arguably the first with a general purpose architecture to both attain near human-level one-shot classification performance and produce high quality samples in one-shot generation.

2 Background

2.1 Variational Autoencoders

When dealing with latent variable models of the form \( p(x) = \int_z p(z)p(x|z)dz \), the integration is necessary for both learning and inference but is often intractable to compute in closed form. Variational Autoencoders (VAEs, Kingma & Welling, 2013) provide a method for learning such models by combining the variational inference objective with neural-network based approximate posterior inference. Specifically, a VAE comprises a generative network \( p(z)p(x|z) \) parametrised by \( \theta \), alongside a separate inference network \( q(z|x) \) parameterised by \( \phi \). These are trained jointly to maximise a single objective:

\[
\mathcal{L}_X(\theta, \phi) = \sum_{x \in X} \left[ \log p_\theta(x) - D_{KL} \left( q_\phi(z|x) \parallel p_\theta(z|x) \right) \right] \tag{1}
\]

\[
= \sum_{x \in X} \left[ \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - D_{KL} \left( q_\phi(z|x) \parallel p_\theta(z) \right) \right] \tag{2}
\]
As can be seen from Equation 1, this objective $L_X$ is a lower bound on the total log likelihood of the dataset $\sum_{x \in X} \log p(x)$, while $q(z|x)$ is trained to approximate the true posterior $p(z|x)$ as accurately as possible. If it could match this distribution exactly then the bound would be tight so that the VAE objective equals the true log likelihood of the data. In practice, the resulting model is typically a compromise between two goals: pulling $p$ towards a distribution that assigns high likelihood to the data, but also towards one which allows accurate inference by $q$. Equation 2 provides a formulation for the same objective which can be optimised stochastically, using Monte-Carlo integration to approximate the expectation.

### 2.2 Variational Autoencoders Over Sets

The Neural Statistician (Edwards & Storkey, 2016) is a Variational Autoencoder in which each item to be encoded is itself a set, such as the set $X^{(i)}$ of all images with a particular class label $i$:

$$X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\} \tag{3}$$

The generative model for complete classes, $p(X)$, is described by introduction of a corresponding latent variable $c$. Given $c$, individual $x \in X$ are conditionally independent:

$$p(X) = \int_c p(c) \prod_{x \in X} p(x|c) dc \tag{4}$$

This functional form is justified by de Finetti’s theorem under the assumption that elements within in each set $X$ are exchangeable. The likelihood is again intractable to compute, but it can be bounded below via:

$$\log p(X) \geq L_X = \mathbb{E}_{q(c|X)} \left[ \sum_{x \in X} \log p(x|c) \right] - D_{KL}(q(c|X) \parallel p(c)) \tag{5}$$

### 3 Variational Homoencoders

Unfortunately, calculating the variational lower bound for each class $X$ requires evaluating both $q(c|X)$ and $p(X|c)$, meaning that the entire class must be passed through both networks for each gradient update. This can easily become intractable for classes with hundreds of examples. Indeed, previous work (Edwards & Storkey, 2016) ensures that datasets used for training are always of small size by instead maximising a likelihood lower-bound of randomly sampled subsets.

In this work we replace the variational lower-bound in Equation 5 with a new objective, constructed via sub-sampled datasets of reduced size. We use a constrained variational distribution $q(c|D)$, $D \subseteq X$ for posterior inference and an unbiased stochastic approximation $\log p(x|c)$, $x \in X$ for the likelihood. In the following section we show that the resulting objective can be interpreted as a lower-bound on the log-likelihood of the data.

This bound will typically be loose due to stochasticity in sampling $D$, and we view this as a regularization strategy: we aim to learn latent representations that are quickly inferable from a small number of instances, and the VHE objective is tailored for this purpose.

#### 3.1 The Resampling Trick

We would like to learn a generative model for sets $X$ of the form

$$p(X) = \int_c p(c) \prod_{x \in X} p(x|c) dc \tag{6}$$

We will refer our full dataset as a union of disjoint classes $X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_n$, and use $X(x)$ to refer to the class $X_i \ni x$. Using the standard consequent of Jensen’s inequality, we can lower bound the log-likelihood of each dataset $X$ using an arbitrary distribution $q$. In particular, we give $q$ as a fixed function of arbitrary data.

$$\log p(X) \geq \mathbb{E}_{q(c|D)} \log p(X|c) - D_{KL}(q(c|D) \parallel p(c)), \ \forall D \subset X \tag{7}$$
Splitting up individual likelihoods, we may rewrite

\[
\log p(X) \geq \mathbb{E}_{q(c|D)} \left[ \sum_{x \in X} \log p(x|c) \right] - D_{KL} [q(c|D) \| p(c)], \quad \forall D \subset X
\]

(8)

\[
= \sum_{x \in X} \left[ \mathbb{E}_{q(c|D)} \log p(x|c) - \frac{1}{|X|} D_{KL} [q(c|D) \| p(c)] \right], \quad \forall D \subset X
\]

(9)

\[
def \sum_{x \in X} L(x; D, |X|), \quad \forall D \subset X
\]

(10)

Finally, we can replace the universal quantification with an expectation under any distribution of D (e.g. uniform sampling from X without replacement):

\[
\log p(X) \geq \mathbb{E}_{D \subset X} \sum_{x \in X} L(x; D, |X|) = \sum_{x \in X} \mathbb{E}_{D \subset X} L(x; D, |X|)
\]

(11)

\[
\log p(X) \geq \sum_{x \in X} \mathbb{E}_{D \subset X(x)} L(x; D, |X(x)|)
\]

(12)

This formulation suggests a simple modification to the VAE training procedure, train both the encoder and decoder together. At each step we select an element \(x\), use new resampled elements \(D \subset X(x)\) to construct our approximate posterior \(q(c|D)\), and rescale the encoding cost appropriately. In contrast to a Variational Autoencoder which uses an element’s encoding to reconstruct itself, a Variational Homoencoder uses encoded elements to construct a new element from the same class.

### 3.2 Application to structured datasets

The above derivation applies to a dataset partitioned into disjoint classes \(X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_n\), each with a corresponding latent variable \(c_i\). However, an analogous trick can be applied to datasets with richer categorical structure. In general, if we wish to learn a model for \(X\) in which each latent variable \(c_i\) affects some arbitrary subset \(X_i\) of the data (where the \(X_i\) may overlap), we may still construct a lower bound of the form:

\[
\log p(X) \geq \mathbb{E}_{Q(c|D)} \left[ \sum_{x \in X} \log p(x|c) \right] - D_{KL} [Q(c|D) \| P(c)]
\]

(13)

In particular, we may subsample a set \(D_i \subset X_i\) to infer each latent variable \(c_i\), so that \(Q(c) = \prod_i q_i(c_i|D_i)\). This leads to an analogous training procedure (Equation 14). Figure 2 illustrates how this framework may be applied to data with hierarchical or factorial category structure.

\[
\log p(X) \geq \sum_{x \in X} \mathbb{E}_{D_i \subset X_i \text{ for each } x \in X_i} \left[ \mathbb{E}_{q_i(c_i|D_i) \text{ for each } x \in X_i} \log p(x|c) - \sum_{i: x \in X_i} \frac{1}{|X_i|} D_{KL} \left( q_i(c_i|D_i) \| p(c_i) \right) \right]
\]

(14)

![Diagram](image.png)

Figure 2: Application of VHE framework to hierarchical (left) and factorial (right) models. Given an element \(x\) such that \(x \in X_1\) and \(x \in X_2\), an approximate posterior is constructed for the corresponding shared latent variables \(c_1, c_2\) using subsampled sets \(D_1 \subset X_1, D_2 \subset X_2\).
3.3 Discussion

Powerful Decoder Models

As evident in Equation 9, the VHE objective also provides a formal motivation for KL rescaling in the variational objective (a commonly-used technique to increase utilisation of latent variables) by sharing these variables across many elements.

For powerful decoder models, in which inference can be particularly challenging, a common failure mode of VAEs is to avoid the penalty for inaccurate inference by learning a decoder \( p(x|z) \) with no dependence on the latent space, allowing \( q \) to match the (now fixed) posterior exactly. This has been discussed in Chen et al. (2016), who suggest crippling the decoder to alleviate this problem.

This failure mode is very plausible in a VAE for sets with a sufficiently powerful generative network, for example, if the inference network \( q \) is not able to reduce its approximation error \( D_{KL}[q(c|D) \parallel p(c|D)] \) to below the total correlation of \( D \). This could easily happen whenever \( |D| \) is too small, or the inference network \( q \) is too weak. Variational Homoencoders suggest a potential remedy to this, encouraging use of the latent space by reusing the same latent variables across a large set \( X \). This allows a VHE to learn useful representations even with \( |D| = 1 \), while at the same time utilising a powerful decoder model such as a PixelCNN to achieve highly accurate density estimation.

Constrained Posterior Approximation

In a VAE, use of a recognition network encourages learning of generative models whose structure permits accurate amortised inference. In a VHE, this recognition network takes only small subsets of a class as input, which additionally encourages that the true posterior \( p(c|X) \) can be well approximated from only a few examples of \( X \). For a subset \( D \subset X \), \( q(c|D) \) is implicitly trained to minimise the KL divergence from this posterior in expectation, over all possible classes \( X \) consistent with \( D \).

For a data-generating distribution \( p_d \) we may equivalently describe the VHE objective (Eq. 12) as

\[
\mathbb{E}_{p_d(D)} \mathbb{E}_{p_d(X|D)} \left[ \log p(x) - \frac{1}{|X|} D_{KL}[q(c|D) \parallel p(c|X)] \right]
\]

Note that the variational gap on the right side of this equation is itself bounded by:

\[
\mathbb{E}_{p_d(X|D)} D_{KL}[q(c|D) \parallel p(c|X)] \geq D_{KL}[q(c|D) \parallel \mathbb{E}_{p_d(X|D)} p(c|X)] \geq 0
\]

The left inequality is tightest when \( p(c|X) \) matches \( p(c|D) \) well across all \( X \) consistent with \( D \), and exact only when these are equal. We view this aspect of the VHE loss as regulariser for constrained posterior approximation, encouraging models for which the posterior \( p(c|X) \) can be well determined by sampled subsets \( D \subset X \). This reflects how we expect the model to be used at test time, and in practice we have found this ‘loose’ bound to perform well in our experiments. It is also possible for this bound to be tightened by introducing an auxiliary inference network (see supplement). However, in our experiments so far, we have not yet found this to confer any additional benefit.
4 EXPERIMENTAL RESULTS

4.1 CLASS-LEVEL MODELS

Sharing latent variables across large datasets allows us to use more powerful autoregressive decoders, producing sharper samples and achieving higher test likelihood. We demonstrate this by training a multilayer PixelCNN on the Omniglot dataset. Our model is inspired by the recently proposed PixelVAE (Gulrajani et al., 2016) with addition of a Spatial Transformer (Jaderberg et al., 2015) in the generative model. Each character class is associated with latent variable $c$, drawn from a learned PixelCNN prior. For each character $x$, a sampled affine transform $t_x$ is applied to this template and the result is used to condition a Gated PixelCNN (Oord et al., 2016). We use a CNN with BatchNorm (Ioffe & Szegedy, 2015) for $q(t|x)$, and for $q(c|D)$ we use a Spatial Transformer followed by a mean and a single convolution, outputting a diagonal Gaussian posterior.

We train Variational Homoencoders on a random sample of 1200 Omniglot classes, as in Edwards & Storkey (2016); Santoro et al. (2016); Vinyals et al. (2016). Images are scaled to 28x28 pixels, dynamically binarised, and augmented by 8 transformations to produce new classes. We additionally use 20 small random affine transformations to create new instances within each class. All models were built in Torch 7 (Collobert et al., 2011) and optimised using Adam (Kingma & Welling, 2013). For comparison, we also trained several alternative models on the same dataset. In particular, we compare the performance of this model to one with a deconvolutional architecture (following Edwards & Storkey, 2016) trained for 300 epochs. For both architectures, we also test several other models using different objectives with the sample computational cost of training: a Neural Statistician trained on sampled subsets $D \subset X$ with $|D| = 5$, and two intermediate objectives:

Resample only: $\mathbb{E}_{D \subset X} \left[ \mathbb{E}_{q(c|D)} \log p(x|c) - \frac{1}{|D|} D_{KL} [q(c|D) \parallel p(c)] \right]$ (17)

Rescale only: $\mathbb{E}_{D \subset X} \left[ \mathbb{E}_{q(c|D)} \log p(x|c) - \frac{1}{|X|} D_{KL} [q(c|D) \parallel p(c)] \right]$ (18)

To aid optimisation, we use 50 epochs to anneal the encoding cost and perform 5 independent training runs from which we select the model with the lowest training error. This was necessary to ensure a fair comparison with the Neural Statistician objective, which otherwise converges to a local optimum with $q(c|D) = p(c)$. We additionally implemented ‘sample dropout’ (Edwards & Storkey, 2016) but found that this did not affect results.

Figure 3: Autoregressive Variational Homoencoder for Omniglot characters.

Figure 4: One-shot same-class samples generated by our model. Cues were drawn at random from test data.
Table 1: Comparison of deep learning techniques for Omniglot classification

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (20-way)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
<td></td>
</tr>
<tr>
<td><strong>Generative models, log (p(X))</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generative Matching Networks*</td>
<td>77.0%</td>
<td>91.0%</td>
<td></td>
</tr>
<tr>
<td>Neural Statistician</td>
<td>93.2%</td>
<td>98.1%</td>
<td></td>
</tr>
<tr>
<td>Variational Homoencoder</td>
<td>95.2%</td>
<td><strong>98.8%</strong></td>
<td></td>
</tr>
<tr>
<td>**Discriminative models, log (q(y</td>
<td>x, X, Y))**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Siamese Networks</td>
<td>88.1%</td>
<td>97.0%</td>
<td></td>
</tr>
<tr>
<td>Matching Networks</td>
<td>93.8%</td>
<td>98.7%</td>
<td></td>
</tr>
<tr>
<td>Prototypical Networks</td>
<td>96.0%</td>
<td>98.9%</td>
<td></td>
</tr>
<tr>
<td>Attentive Recurrent Comparators</td>
<td>97.5%</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

*Uses train/test split from Lake et al. (2015)

Table 1 shows the 20-way classification accuracy of VHEs trained on 1-shot and 5-shot objectives. Both networks are state-of-the-art amongst deep generative models, and competitive when compared to models discriminatively trained for classification. As shown in Figure 4, the same models can be used for generation of new instances in a class, producing samples which are sharp and identifiable.

Table 3 compares the results of models trained using alternative objectives, both on a deconvolutional architecture and on the hierarchical PixelCNN architecture described. In the former case, all models yield similar classification performance. However, when using a more expressive PixelCNN decoder, the Neural Statistician objective learns to place very little information in the posterior \(q(c|D)\), which can be seen both in the samples (Figure 5) and classification performance. Our careful training suggests that this is not an optimisation difficulty but is core to the objective, as discussed in Chen et al. (2016). In this case, VHE improves generalisation by encouraging use of the latent space. While KL rescaling is a common technique used to achieve this, we find that when used alone it leads to overfitting that is equally harmful to classification performance.

Because our goal is to model shared structure across images, we evaluate the joint log likelihood of entire character classes. From this perspective, a single element VAE will perform poorly as it treats all datapoints as independent, optimising a sum over log likelihoods for each element. By sharing latent variables across all elements of the same class, a VHE can improve upon this considerably.

Most previous work which evaluates likelihood uses the train/test split of Burda et al. (2015). However, our most appropriate comparison is with Generative Matching Networks (Bartunov & Vetrov, 2016) as they also model dependencies within a class. We thus train a VHE under the same conditions as them, using the harder test split from Lake et al. (2015) with no data augmentation. We evaluate the joint log likelihood of full character classes from the test set, normalised by the number of elements, using importance weighting with \(k=500\) samples from \(q(c|X)\). Table 4.1 compares these log likelihoods, with VHE achieving state-of-the-art. To make clear the contribution of shared latent variables, we also compare with a VAE model with the same architecture.

Figure 5: 5-shot samples generated by each model (more in supplement). With a PixelCNN, architecture both NS and Resample underutilise the latent space and so produce unfaithful samples.
Table 3: Joint log-likelihood of full Omniglot classes

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Class NLL / image</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Independent models</strong></td>
<td></td>
</tr>
<tr>
<td>DRAW [7]</td>
<td>96.5 nats</td>
</tr>
<tr>
<td>Conv DRAW [8]</td>
<td>91.0 nats</td>
</tr>
<tr>
<td>VLAE [3]</td>
<td>89.83 nats</td>
</tr>
<tr>
<td>VAE (same architecture as VHE)</td>
<td>71.44 nats</td>
</tr>
<tr>
<td><strong>Conditional models</strong></td>
<td></td>
</tr>
<tr>
<td>Generative Matching Networks [1]</td>
<td>62.42 nats</td>
</tr>
<tr>
<td><strong>Shared-latent models</strong></td>
<td></td>
</tr>
<tr>
<td>Variational Homoencoder</td>
<td>61.65 nats</td>
</tr>
</tbody>
</table>

4.2 Modelling richer category structure

To demonstrate how the VHE framework may apply to models with richer category structure, we build both a hierarchical and a factorial VHE (see Figure 2) using simple modifications to the above architectures. For the hierarchical VHE, we extend the deconvolutional model with an extra latent layer $a$ using the same encoder and decoder architecture as $c$. This is used to encode alphabet level structure for the Omniglot dataset, learning a generative model for alphabets of the form

$$ p(\mathcal{A}) = \int p(a) \prod_{X_i \in \mathcal{A}} \int p(c_i | a) \prod_{x_{ij} \in X_i} p(x_{ij} | c_i, a) dc_i da $$

Again, we train this model using a single objective, using separately resampled subsets $D^a$ and $D^c$ to infer each latent variable (see supplement). We then test our model at both one-shot character generation and 5-shot alphabet generation, using samples from previously unseen alphabets. As shown in Figure 6, our single trained model is able to learn structure at both layers of abstraction.

For the factorial VHE, we extend the PixelCNN model with a 20-dimensional Gaussian latent variable $s$, which is appended to $c$ when conditioning the PixelCNN decoder. $c$ is used to represent the character in an image and $s$ to represent the style. We extend the Omniglot dataset by assigning each image to one of 30 randomly generated styles (independent of its character class) and recolouring it accordingly. Once trained, we can use the model to re-render characters in a new style, either sampled from the prior $p(s)$ or inferred from a separate character using the encoder $q(s | D^s)$. Figure 8 shows samples from two separate models for manipulating background colour and pen stroke.

Figure 6: Conditional samples from a VHE trained on object silhouettes (see supplement).

Figure 7: Conditional samples from both character (top) and alphabet (bottom) levels of the same hierarchical model.

Figure 8: Characters rendered with new pen stroke (top) or background (bottom). Style may be sampled from the prior or inferred from another image.

We thank the authors of Bartunov & Vetrov (2016) for providing us with this comparison.
5 CONCLUSION

We introduced the Variational Homoencoder: a deep generative model trained by a novel training procedure which resembles few-shot generation. This framework allows latent variables to be shared across large numbers of elements in the training set, encouraging them to be well utilised even with highly expressive decoder networks. We demonstrate this by training a hierarchical PixelCNN model of the Omniglot dataset, and show that our novel training objective is responsible for the state-of-the-art results it achieves. This model is arguably the first which uses a general purpose architecture to both attain near human-level one-shot classification performance and produce high quality samples in one-shot generation.

Furthermore, we show that the VHE framework may be naturally applied to models with richer latent structure. We highlight this by training a hierarchical model which generalises the style of an alphabet to produce new characters, and a factorial model which separates the content and drawing style of coloured character images.

REFERENCES


6 Supplementary Material

6.1 Simple Distributions

To understand the behaviour of the VHE objective, we apply it to few-shot learning of simple 1D distributions. We create five toy datasets as follows, each split into 100 classes of 100 examples:

**Gaussian:** Each class is Normal with fixed $\sigma^2$, and with $\mu$ drawn from a Gaussian hyperprior.

**Mixture of Gaussians:** Each class is an even mixture of two Gaussians, with fixed $\sigma^2$, fixed separation, and location drawn from a Gaussian hyperprior.

**von Mises:** Each class is von Mises with fixed $\kappa$, and with $\mu$ drawn from a Uniform hyperprior.

**Gamma:** Each class is Gamma with fixed $\beta$, and with $\alpha$ drawn from a Uniform hyperprior.

**Discrete:** Each class is a subset of integers 1 to 8, either small (1-4), large (5-8), odd or even.

We train models on each of these datasets using a simple architecture with a single latent variable $c \sim \text{Normal}(0, 1)$. We use a generator $p(x|c)$ containing one fully connected layer which outputs parameters of the true distribution type (Gaussian, Mixture of two Gaussians, etc), and for the inference network $q(c|D)$ we constrain our architecture to a simple linear map and a mean operation (using learned embeddings for the Discrete dataset) with a Gaussian posterior. All models are built in Torch 7 [Collobert et al., 2011] and optimised using Adam [Kingma & Welling, 2013] for 200 epochs. To aid optimisation, we also use an additional 50 epochs to anneal the encoding cost, and perform 3 independent training runs from which we select the model with the lowest training error.

We then test on two few-shot generalisation tasks: conditional generation of new examples, evaluated as test log-likelihood under the approximate posterior $q(c|D)$, and classification. Finally, we use importance weighting to evaluate the average joint log-likelihood of an entire class under the generative model $p$.

Results of these experiments are shown in Figure 9, and elucidate several properties of our objective:

- A Variational Homoencoder matches or outperforms other tested models at both few-shot generation (by conditional log likelihood) and few-shot classification, when the input size $|D|$ is sufficiently small.

- As is common in a VAE framework, we find cases for which NS encodes no information in the posterior $q(c|D)$. Our careful training suggests that this is not an optimisation difficulty but is core to the objective, as discussed in Chen et al. [2016]. In these cases, VHE improves generalisation by encouraging use of the latent space. While KL rescaling is a standard technique to achieve this, we find that when used alone it leads to overfitting and poor conditional NLL.

- Simultaneously, the VHE objective learns a hierarchical generative model $p$ with good marginal likelihood. For sufficiently small $|D|$ we find that this significantly improves upon the likelihood achieved by other objectives, including versus a single-element VAE (corresponding to NS with $|D| = 1$). However, when $|D|$ is sufficiently large, the benefit of resampling is lost and the NS objective may achieve equal or better class likelihood. At $|D| = |X| = 100$, all objectives become equivalent.
Figure 9: Comparison of models with shared architecture, trained by various objectives on synthetic datasets. $|D|$ refers to the size of the encoder input during training. Note that some lines extend above the limit of the y-axis. Top row: $|D|$-shot generation loss $-\mathbb{E}_{c \sim q(c|D)} \log p(x'|c)$; Second row: $|D|$-shot binary classification error, by minimising conditional NLL; Third row: Joint NLL of all elements in a class normalised by the number of elements, calculated by importance weighting on 200 samples from $q(c|X)$; Bottom row: Mean encoded information $\text{KL}[q(c|D) \parallel p(c)]$.

6.2 Tightened Variational Bound

The likelihood lower bound in the VHE objective may also be tightened by introduction of an auxiliary network $r(D|c, X)$, trained to infer which subset $D \subset X$ was sampled during the inference procedure. This meta-inference approach was introduced in Salimans et al. (2015) to develop stochastic variational posteriors using MCMC inference, and has recently been applied to approximate inference evaluation (Cusumano-Towner & Mansinghka, 2017). Applied to Equation 12, this yields a modified bound for the VHE objective

$$\log p(X) \geq \sum_{x \in X} \mathbb{E}_{q(D|X(x))} \left[ \log p(x|c) - \frac{1}{|X(x)|} \log \frac{p(c) r(D|c, X(x))}{q(D|X(x)) q(D)} \right]$$

where $q(D|X)$ describes the stochastic procedure for sampling $D \subset X$. We have experimented with a simple functional form $r(D|c, X) = \prod_i r(d_i|c, X) \propto \prod_i [f_{\psi}(c) \cdot \xi_{d_i}]$, learning parameters $\psi$ and embeddings $\{\xi_d : d \in X\}$. However, we have so far found that this conferred no additional benefit over the loose bound used in our experiments.

6.3 Variational Bound for Hierarchical Models

The resampling trick may be applied iteratively, to construct likelihood bounds over hierarchically organised data. Expanding on Equation 12 suppose that we have collection of datasets

$$X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_N$$

For example, each $X_i$ might be a different alphabet whose latent description $a$ generates many character classes $X_i$, and for each of these a corresponding latent $c_i$ is used to generate many images $x_{ij}$. From this perspective, we would like to learn a generative model for alphabets $\mathcal{X}$ of the form

$$p(\mathcal{X}) = \int p(a) \prod_{X_i \subset \mathcal{X}} \int p(c|a) \prod_{x \in X_i} p(x|c, a) dc da$$

12
Reapplying the same trick as before yields a bound taken over all elements $x$:

$$
\log p(X) \geq \sum_{x \in X} \mathbb{E}_{D^c \subset X(x)} \left[ \frac{1}{|X(x)|} \sum_{c \in D^c} \log p(x|c) - \frac{1}{|X(x)|} D_{KL}(q_c(c|D^c, a) \parallel p(c|a)) \right] - \frac{1}{|X(x)|} D_{KL}(q_c(c|D^c, a) \parallel p(c|a)) \right] 
$$

This suggests an analogous hierarchical resampling trick: Summing over every element $x$, we can bound the log likelihood of the full hierarchy by resampling subsets $D^c, D^a$, etc. at each level to construct an approximate posterior. All networks are trained together by this single objective, sampling $x, D^a$ and $D^c$ for each gradient step. Note that this procedure need only require passing sampled elements, rather than full classes, into the upper-level encoder $q_a$.

6.4 Omniglot Architecture

6.5 PixelCNN Omniglot Architecture

6.5.1 Methodology

Our architecture uses a 8x28x28 latent variable $c$, with a full architecture detailed below. For our classification experiments, we trained 5 models on each of the objectives (VHE, Rescale only, Resample only, NS). Occasionally we found instability in optimisation, causing sudden large increases in the training objective. When this happened, we halted and restarted training. All models were trained for 100 epochs on 1000 characters from the training set (the remaining 200 have been used as validation data for model selection). Finally, for each objective we selected the parameters achieving the best training error.

Note that we did not optimise or select models based on classification performance, other than through our development of our model’s architecture. However, we find that classification performance is well correlated the generative training objective, as can be seen in the full table of results.

We perform classification by calculating the expected conditional likelihood under the variational posterior: $E_{q(c|x)} p(x|c)$. This is approximated using 20 samples for the outer expectation, and importance sampling with $k = 10$ for the inner integral $p(x|c) = E_{q(t|x)} \frac{p(t)}{q(t|x)} p(x|c, t)$

To evaluate and compare log likelihood, we trained 5 more models with the same architecture, this time on the canonical 30-20 alphabet split of Lake et al. We did not augment our training data. Again, we split the background set into training data (25 alphabets) and validation data (5) but do not use the validation set in training or evaluation for our final results. We estimate the total class log likelihood by importance weighting, using $k=20$ importance samples of the class latent $c$ and $k=10$ importance samples of the transformation latent $t$ for each instance.
### 6.5.2 Conditional Samples on Omniglot

<table>
<thead>
<tr>
<th>Baseline Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Neural Statistician [10]</strong></td>
</tr>
<tr>
<td><strong>Resample Only</strong></td>
</tr>
<tr>
<td><strong>Rescale Only</strong></td>
</tr>
<tr>
<td><strong>Variational Homeencoder</strong></td>
</tr>
<tr>
<td><strong>PixelCNN Architecture</strong></td>
</tr>
<tr>
<td><strong>Neural Statistician</strong></td>
</tr>
<tr>
<td><strong>Resample Only</strong></td>
</tr>
<tr>
<td><strong>Rescale Only</strong></td>
</tr>
<tr>
<td><strong>Variational Homeencoder</strong></td>
</tr>
</tbody>
</table>
6.5.3 Model Specification

\([d]\) denotes a dimension \(d\) tensor. \(\{t\}\) denotes a set with elements of type \(t\). Posteriors \(q\) are Gaussian.

\(P(C)\)

A PixelCNN with autoregressive weights along only the spatial (not depth) dimensions of \(c\). We use 2 layers of masked \(64\times3\times3\) convolutions, followed by a ReLU and two \(8\times1\times1\) convolutions corresponding to the mean and log variance of a Gaussian posterior for the following pixel.

\(P(T)\)

t: \([16]\) Normal(0, 1)

\(P(X—C,T)\)

c: \([8x28x28]\), t: \([16]\) \(\rightarrow\) x: \([1x28x28]\)

<table>
<thead>
<tr>
<th>Input</th>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t)</td>
<td>Linear</td>
<td>(t_2: [6])</td>
</tr>
<tr>
<td>(c_2)</td>
<td>Spatial Transformer</td>
<td>(y_1: [8x28x28])</td>
</tr>
<tr>
<td>(y_1)</td>
<td>(64\times3\times3) Conv; Relu; BatchNorm</td>
<td>(y_2: [64x28x28])</td>
</tr>
<tr>
<td>(y_2)</td>
<td>(64\times3\times3) Conv; Relu; BatchNorm</td>
<td>(y_3: [64x28x28])</td>
</tr>
<tr>
<td>(y_3)</td>
<td>(64\times3\times3) Conv; Relu; BatchNorm</td>
<td>(y_4: [64x28x28])</td>
</tr>
<tr>
<td>(y_4)</td>
<td>(64\times3\times3) Conv; Relu; BatchNorm</td>
<td>(y_5: [64x28x28])</td>
</tr>
<tr>
<td>(y_5)</td>
<td>(64\times3\times3) Conv; Relu; BatchNorm</td>
<td>(y: [64x28x28])</td>
</tr>
<tr>
<td>(y)</td>
<td>PixelCNN</td>
<td>x: ([1x28x28])</td>
</tr>
</tbody>
</table>

PixelCNN is gated by \(y\), and is autoregressive along only the spatial (not depth) dimensions of \(c\). We use 2 layers of masked \(64\times3\times3\) convolutions, followed by a ReLU, a \(2\times1\times1\) convolution and a softmax, corresponding to a Bernoulli distribution on the following pixel.

\(Q(C—D)\)

\(D: \{[1x28x28]\} \rightarrow c: [8x28x28]\)

<table>
<thead>
<tr>
<th>Input</th>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D)</td>
<td>STNq</td>
<td>(Y: {[1x28x28]})</td>
</tr>
<tr>
<td>(Y)</td>
<td>Mean</td>
<td>(y: [1x28x28])</td>
</tr>
<tr>
<td>(y)</td>
<td>(16x28x28) Conv</td>
<td>(\text{mu: [8x28x28], logvar: [8x28x28]})</td>
</tr>
</tbody>
</table>

\(Q(T—X)\)

\(x: [1x28x28] \rightarrow t: [16]\)

<table>
<thead>
<tr>
<th>Input</th>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>(32\times3\times3) Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>(y_1: [32x15x15])</td>
</tr>
<tr>
<td>(y_1)</td>
<td>(32\times3\times3) Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>(y_2: [32x8x8])</td>
</tr>
<tr>
<td>(y_2)</td>
<td>(32\times3\times3) Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>(y_3: [32x4x4])</td>
</tr>
<tr>
<td>(y_3)</td>
<td>(32\times3\times3) Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>(y_4: [32x2x2])</td>
</tr>
<tr>
<td>(y_4)</td>
<td>(32\times3\times3) Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>(y_5: [32x1x1])</td>
</tr>
<tr>
<td>(y_5)</td>
<td>Linear</td>
<td>(\text{mu: [16], logvar: [16]})</td>
</tr>
</tbody>
</table>
Spatial Transformer STNq

\[ x: [1x28x28] \rightarrow y: [1x28x28] \]

<table>
<thead>
<tr>
<th>Input</th>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>16x3x3 Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>( y_1: [16x15x15] )</td>
</tr>
<tr>
<td>( y_1 )</td>
<td>16x3x3 Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>( y_2: [16x8x8] )</td>
</tr>
<tr>
<td>( y_2 )</td>
<td>16x3x3 Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>( y_3: [16x4x4] )</td>
</tr>
<tr>
<td>( y_3 )</td>
<td>16x3x3 Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>( y_4: [16x2x2] )</td>
</tr>
<tr>
<td>( y_4 )</td>
<td>16x3x3 Conv; 2x2 Max Pooling; ReLU; BatchNorm</td>
<td>( y_5: [16x1x1] )</td>
</tr>
<tr>
<td>( y_5 )</td>
<td>Linear</td>
<td>( y_6: [6] )</td>
</tr>
<tr>
<td>( x, y_6 )</td>
<td>Spatial Transformer Network</td>
<td>( y: [1x28x28] )</td>
</tr>
</tbody>
</table>

6.6 Hierarchical Omniglot Architecture

We extend the same architecture described in Appendix B of [Edwards & Storkey, 2016], with only a simple modification: we introduce a new latent layer containing a 64-dimensional variable \( a \), with a Gaussian prior. We give \( p(c|a) \) the same functional form as \( p(z|c) \), and give \( q(a|D^n) \) the same functional form as \( q(c|D^n) \) using the shared encoder.

Figure 10: 10-shot alphabet generation samples from the hierarchical model.
6.7 Conditional Samples on Silhouettes Dataset

We created a VHE using the sampled deconvolutional architecture as applied to omniglot, and trained it on the Caltech-101 Silhouettes dataset. 10 object classes were held out as test data, which we use to generate both 5-shot and 5-shot conditional samples.