Variational Hyper-Encoding Networks

Phuoc Nguyen[1], Truyen Tran[1], Sunil Gupta[1], Santu Rana[1], Hieu-Chi Dam[2], and Svetha Venkatesh[1]

[1] A2I2, Deakin University, Geelong, Australia
{phuoc.nguyen, truyen.tran, sunil.gupta, santu.rana, svetha.venkatesh}@deakin.edu.au
[2] Japan Advanced Institute of Science and Technology, Japan
dam@jaist.ac.jp

Abstract. We propose a framework called HyperVAE for encoding distributions of distributions. When a target distribution is modeled by a VAE, its neural network parameters are sampled from a distribution in the model space modeled by a hyper-level VAE. We propose a variational inference framework to implicitly encode the parameter distributions into a low dimensional Gaussian distribution. Given a target distribution, we predict the posterior distribution of the latent code, then use a matrix-network decoder to generate a posterior distribution for the parameters. HyperVAE can encode the target parameters in full in contrast to common hyper-networks practices, which generate only the scale and bias vectors to modify the target-network parameters. Thus HyperVAE preserves information about the model for each task in the latent space. We derive the training objective for HyperVAE using the minimum description length (MDL) principle to reduce the complexity of HyperVAE. We evaluate HyperVAE in density estimation tasks, outlier detection and discovery of novel design classes, demonstrating its efficacy.

Keywords: Deep generative models, meta-learning, hyper networks

1 Introduction

Humans can extract meta knowledge across tasks such that when presented with an unseen task they can use this meta knowledge, adapt it to the new context and quickly solve the new task. Recent advance in meta-learning [5, 8] shows that it is possible to learn a single model such that when presented with a new task, it can quickly adapt to the new distribution and accurately classify unseen test points. Since meta-learning algorithms are designed for few-shot or one-shot learning where labeled data exists, it faces challenges *when there is none*[1] to assist backpropagation when testing.

Hyper-networks [9] can generate the weights for a target network given a set of embedding vectors of those weights. Due to its generative advantage, it can be used to generate a distribution of parameters for a target network [9, 12]. In

[1] This is not the same as zero-shot learning where label description is available.

practice, due to the high dimensional parameter space, it only generates scaling factors and biases for the target network. This poses a problem that the weight embedding vectors only encode partial information about the target task, and thus are not guaranteed to perform well on unseen tasks.

On the other hand, variational autoencoders (VAEs) [11, 19] is a class of deep generative models that can model complex distributions. A major attractive feature of VAEs is that we can draw from simple, low-dimensional distributions (such as isotropic Gaussians), and the model will generate high-dimensional data instantly without going through expensive procedures like those in the classic MCMC. This suggests VAEs can be highly useful for high dimensional design exploration [7]. In this work, we lift this idea to one more abstraction level, that is, using a *hyper VAE to generate VAE models*. While the VAEs work at the individual design level, the *hyper VAE* works at the class level. This permits far more flexibility in exploration, because not only we can explore designs within a class, we can explore multiple classes. The main insight here is that the model parameters can also be treated as a design in a model design space. Hence, we can generate the model parameters using another VAE given some latent low-dimensional variable.

We propose HyperVAE, a novel class of VAEs, as a powerful deep generative model to learn to generate the parameters of VAE networks for modeling the distribution of different tasks. The versatility of the HyperVAE to produce VAE models allows it to be applied for a variety of problems where model flexibility is required, including density estimation, outlier detection, and novelty seeking. For the latter, since HyperVAE enforces a smooth transition in the model family, interpolating in this space will enable us to extrapolate to models of new tasks which are *close* to trained tasks. Thus as global search techniques can guide the generation of latent spaces of VAEs, search enables HyperVAE to produce novel classes of discovery. We use Bayesian Optimization (BO) [20], to search in the low dimensional encoding space of VAE. Once a low dimensional design is suggested, we can decode it to the corresponding high dimensional design.

We demonstrate the ability of HyperVAE on three tasks: density estimation, robust outlier detection and discovery of unseen design classes. Our main contributions and results are: (i) Development of a hyper-encoding framework, guided through MDL; (ii) Construction of a versatile HyperVAE model that can tackle density estimation tasks and outlier detection; and (iii) Demonstration of novel designs produced from our model coupled with BO.

2 Variational Autoencoder (VAE)

Let x denote an $\mathcal{X}$-value random variable associated with a $\mathcal{Z}$-value random variable z through a joint distribution $p(x, z)$. We consider a parametric family $\mathcal{P}$ of generative models $p(x, z; \theta)$ factorized as a conditional $p(x|z; \theta)$ and a simple prior $p(z)$, usually chosen as $\mathcal{N}(0, I)$. Maximum likelihood estimate (MLE) of $\theta \in \Theta$, where Θ is the parameter space, over the marginal $\log p(x; \theta) = \log \int p(x, z; \theta) dz$ is intractable, thus requiring alternatives such as expectation-

maximization and variational inference. VAE is an amortized variational inference that jointly learns the generative model $p(x|z;\theta)$ and the variational inference model $q(z|x;\theta)$[2]. Its ELBO objective,

$$\mathcal{L}(x,p,q;\theta) = \mathbb{E}_{q(z|x;\theta)} \log p(x|z;\theta) - D_{KL}\left(q(z|x;\theta)\|p(z)\right) \tag{1}$$

lower-bounds the marginal log-likelihood $\log p(x;\theta)$. In practice, Monte Carlo estimate of the ELBO's gradient is used to update θ. The form of q and p in Eq. 1 makes an encoder and a decoder, hence the name auto-encoder [11].

3 Variational Hyper-Encoding Networks

We assume a setting where there is a sequence of datasets (or tasks) and model parameters $\{(D_t, \theta_t)\}_t$ a sender wish to transmit to a receiver using a minimal combined code length.

3.1 Hyper-auto-encoding problem

Given a set of T distributions $\{D_t\}_{t=1}^T$ called tasks, each containing samples $x \sim p_{D_t}(x)$, our problem is first fitting each parametric model $p(x;\theta_t)$, parameterized by $\theta_t \in \Theta$, to each D_t:

$$\hat{\theta}_t = \underset{\theta\in\Theta}{\operatorname{argmax}}\, p(D_t;\theta) \tag{2}$$

then fitting a parametric model $p(\theta;\gamma)$, parameterized by $\gamma \in \Gamma$ to the set $\{\hat{\theta}_t\}_{t=1}^T$. However, there are major drawbacks to this approach. First, the number of tasks may be insufficient to fit a large enough number of θ_t for fitting $p(\theta;\gamma)$. Second, although we may resample D_t and refit θ_t to create more samples, it is computationally expensive. A more practical approach is to jointly learn the distribution of θ and D.

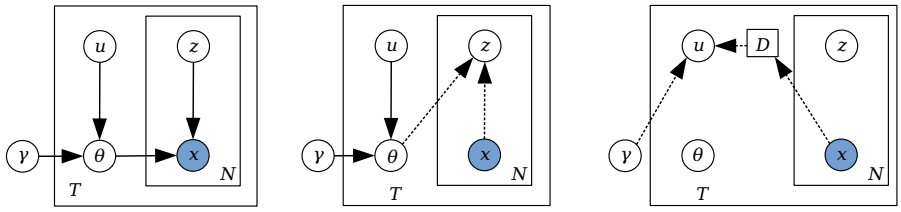
3.2 Hyper-encoding problem

Our problem is to learn the joint distribution $p(\theta, D;\gamma)$ for some parameters γ[3].

HyperVAE We propose a framework for this problem called *HyperVAE* as depicted in Fig. 1. The main insight here is that the *VAE model parameters* $\theta \in \Theta$ can also be treated as a normal input in the parameter space Θ. Hence, we can generate the model parameters θ using *another VAE at the hyper level* whose generative process is $p_\gamma(\theta|u)$ for some low-dimensional latent variable $u \sim p(u) \equiv \mathcal{N}(0, I)$, the prior distribution defined over the latent manifold $\mathcal{U}$ of Θ. The joint distribution $p(\theta, D;\gamma)$ can be expressed as the marginal over the latent representation u:

$$p(\theta, D) = \int p(\theta, D|u)p(u)\mathrm{d}u = \int p(D|\theta)p(\theta|u)p(u)\mathrm{d}u \tag{3}$$

[2] We use $\theta = (\theta_p, \theta_q)$ to denote the set of parameters for p and q.
[3] We assume a Dirac delta distribution for γ, i.e. a point estimate, in this study.



(a) Generation (b) Inference of z (c) Inference of u

Fig. 1. HyperVAE networks, $D = \{x_n\}_{n=1}^N$.

Generation of a random data point x is as follows, c.f. , Fig. 1 (a):

$$\begin{aligned} u_t &\sim N(0, I) \\ \theta_t &\sim p_\gamma(\theta \mid u_t) \\ z &\sim N(0, I) \\ x &\sim p_{\theta_t}(x \mid z) \end{aligned}$$

Inference of z given x and θ, Fig. 1 (b), is approximated by a Gaussian distribution, $q(z|x,\theta) = \mathcal{N}(z|\mu_\theta(x), \sigma_\theta^2(x))$, where μ_θ and σ_θ^2 are neural networks generating the mean and variance parameter vectors.

Inference of u is shown in Fig. 1 (c). We also assume a Gaussian posterior distribution $q(u|D,\theta) = \mathcal{N}(u|\mu(d_t), \sigma^2(d_t))$ parameterized by neural networks $\mu(.)$ and $\sigma^2(.)$. Since θ can be trained on D thus depending on D, we can approximate this dependency implicitly using the inference network itself, thus $q(u|D,\theta) \approx q(u|D)$. The next problem is that since D_t is a set, $q(u|D_t)$ is a function of set, which is an interesting problem on its own. Here we use a simple method to summarize D_t into a vector,

$$d_t = s(D_t) \tag{4}$$

and this turns $q(u|D_t)$ into $q(u|d_t)$. For example, $s(\cdot)$ can be a mean function, a random draw from the set, or a description of the set. In this study, we simply choose a random draw x from the set D_t.

3.3 Minimum Description Length

It is well-known that variational inference is equivalent to the Minimum Description Length (MDL) principle [10]. In this section, we use MDL to compute the total code length of the model and data misfits. From the total code length, we show that a shorter code length and a simpler model can be achieved by redesigning the distribution of the model space. We used a Dirac delta distribution centered at $\mu(u)$ for θ given each latent code u, $p(\theta|u) = \delta_{\mu(u)}(\theta)$ parameterized by a neural network $\mu(u)$ for each latent code u. This results in an implicit distribution for θ represented by the compound distribution $p(\theta) = \int \delta_{\mu(u)}(\theta)p(u)\mathrm{d}u$.

Under the crude 2-part code MDL, the expected code length for transmitting a dataset D and the model parameters θ in the encoding problem in Eq. 2 is $L(D) = L(D|\theta) + L(\theta)$, where $L(D|\theta) = -\log p(D;\theta)$[4] is the code length of the data given the model θ, and $L(\theta) = -\log p(\theta)$ is the code length of the model itself. Under the HyperVAE the code length of D is:

$$L(D) = L(D|\theta) + L(\theta|u) + L(u) \tag{5}$$

If we choose a Dirac delta distribution for θ, $p(\theta|u) = \delta_{\mu(u)}(\theta)$ then θ is deterministic from u and we can eliminate the code length $L(\theta|u)$, thus making the total code length shorter:

$$L(D) = L(D|\theta(u)) + L(u) \tag{6}$$

Additionally, bits-back coding can recover the additional information in the entropy of the variational posterior distribution $q(u|D)$, thus this information should be subtracted from the total code length [10, 22]. The total expected code length is then:

$$\begin{aligned} L(D) &= \mathbb{E}\left[L\left(D|\theta(u)\right) - \log p(u) + \log q(u|D)\right] \\ &= \mathbb{E}\left[L\left(D|\theta(u)\right)\right] + D_{KL}\left(q(u|D)\|p(u)\right) \end{aligned} \tag{7}$$

where the expectation is taken over the posterior distribution $q(u|D)$. The description length of a dataset $D = \{x_i\}_{i=1}^{|D|}$ is the summation of the description length of every data point:

$$\begin{aligned} L\left(D|\theta(u)\right) &= \sum_{i=1}^{|D|} L\left(x_i|\theta(u)\right) \\ &= \sum_{i=1}^{|D|} \left(\mathbb{E}_{q_\theta(z|x_i)} L\left(x_i|z,\theta\right) + D_{KL}\left(q_\theta(z|x_i)\|p(z)\right)\right) \end{aligned} \tag{8}$$

where we ignored the dependence of θ on u to avoid clutter. We train the HyperVAE parameters by minimizing the description length in Eq. 7. In our experiment, we scale down this objective by multiplying it by $1/|D|$ to have a similar scale as a normal VAE's objective. The training objective for HyperVAE is then:

$$L(D) = \frac{1}{|D|}\left[L\left(D|\theta(u)\right) + D_{KL}\left(q(u|D)\|p(u)\right)\right] \tag{9}$$

Mini-batches as tasks In practice, the number of tasks is too small to adequately train the hyper-parameters γ. Here we simulate tasks using data mini-batches in the typical stochastic gradient learning. That is, each mini-batch is treated as a task. To qualify as a task, each mini-batch needs to come from the same class. For example, for handwritten digits, the class is the digit label.

[4] We abused the notation and use p to denote both a density and a probability mass function. Bits-back coding is applicable to continuous distributions [10].

3.4 Compact hyper-decoder architecture

Since neural networks weights are matrices that are highly structured and often overparameterized, we found that a more efficient method is to use a matrix generation network [3] for generating the weights. More concretely, a matrix hyper-layer receives an input matrix H and computes a weight matrix W as $W = \sigma(UHV + B)$, where U, V, B are parameters. As an example, if H is a 1D matrix of size 400×1 and a target weight W of size 400×400, a matrix-layer will require 176 thousand parameters, a 3 order of magnitude reduction from 64.16 million parameters of the standard fully-connected hyper-layer. This compactness allows for complex decoder architecture for generating the target network, unlike hypernetworks methods which rely on a linear layer of an embedding vector for each target-network layer.

3.5 Applications

We can use the HyperVAE framework for density estimation, outlier detection and novelty discovery. In the following, we use *HyperVAE* to denote the whole VAE-of-VAEs framework, *hyper VAE* for the hyper level VAE, and *main VAE* for the VAE of each target task.

Density estimation After training, HyperVAE can be used to estimate the density of a new dataset/task. Let D_t is the new task data. We first infer the posterior distribution $q(u|D_t) \approx q(u|d_t) = \mathcal{N}(u|\mu(d_t), \sigma^2(d_t))$, where d_t is a summary of D_t, Eq. 4, which we choose as random in this study. Next we select the mean of this posterior distribution and decode it into θ using $p_\gamma(\theta|u)$. We use this θ to create the main VAE for D_t then use importance sampling to estimate the density of $x \in D_t$ as follows:

$$p(x) = \mathbb{E}_{q(z|x,\theta)} \frac{p(x|z)p(z)}{q(z|x,\theta)} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(x|z_i)p(z_i)}{q(z_i|x,\theta)}$$

where N is a chosen number of importance samples, $\{z_i\}_{i=1}^N$ are samples from the proposal distribution $q(z|x,\theta)$ to reduce the variance of the density estimate, and $p(z_i)/q(z_i|x,\theta)$ is the multiplicative adjustment to compensate for sampling from $q(z|x,\theta)$ instead of $p(z)$.

Outlier detection Similar to the density estimation application above, we first encode a test vector x_t into a latent distribution $q(u|x_t)$ then decode its mean vector into θ_t to create a VAE model. We then use the description length of x_t, c.f. Eq. 8, under this VAE as the outlier score. Our assumption is that outliers are unseen to the trained model, thus incompressible under this model and should have longer description lengths.

Novelty discovery HyperVAE provides an extra dimension for exploring the model space Θ in addition to exploring the design space $\mathcal{X}$. Once trained, the network can guide exploration of new VAE models for new tasks with certain similarity to the trained tasks.

Given no prior information, we can freely draw models $\theta(u)$ from $u \sim p(u)$ and designs $x \sim p_\theta(x|z)$ with $z \sim p_{\theta(u)}(z)$ and search for the desired x^* satisfying some property $F(x^*)$. An intuitive approach is to employ a global search technique such as Bayesian Optimization (BO) in both the model latent space of u and in the data latent space of z. However searching for both $u \in \mathcal{U}$ and $z \in \mathcal{Z}$ is expensive due to the combined number of dimensions can be very high. Furthermore, reducing the latent dimension would affect the capacity of VAE. To overcome this major challenge, we use BO for optimizing the z space and replace the search in u space by an iterative search heuristic. The workflow starts with an initial exemplar x_0^* which can be completely uninformative (e.g., an empty image for digits or a random design), or properly guided (e.g., from the best choice thus far in the database, or from what is found by VAE+BO itself). The search process for the optimal design at step $t = 1, 2, ..., T$ is as follows:

$$\begin{aligned} &u_t \sim q(u \mid d_{t-1}); \qquad \theta_t = g_\gamma(u_t); \\ &z^* \leftarrow \mathrm{BO}(g_{\theta_t}(z)); \qquad x_t^* \leftarrow g(z^*). \end{aligned} \tag{10}$$

where $d_{t-1} \leftarrow x_{t-1}^*$.

The optimization step in the z space maximizes a function $\max_x F(x) = \max_z F \circ g_{\theta_t}(z)$ for a fixed generator θ_t. Let z_t^* and thus $x_t^* = g_{\theta_{t-1}}(z_{t-1}^*)$ be the solution found at step t. The generator parameter in the subsequent step is set as $\theta_t \leftarrow \theta\,(\mu(x_t^*))$ where μ is the posterior mean. Thus the HyperVAE step transforms the objective function with respect to z by shifting θ.

4 Experiments

We evaluate HyperVAE on three tasks: density estimation, robust outlier detection, and novel discovery.

4.1 Data sets

We use four datasets: MNIST handwritten digits, Omniglot handwritten characters, Fashion MNIST, and Aluminium Alloys datasets. The MNIST contains 60,000 training and 10,000 test examples of 10 classes ranging from 0 to 9. The Omniglot contains 24,345 training and 8,070 test examples. The Fashion MNIST dataset contains the same number of training and test examples as well as the number of classes. In these three datasets, the images are statically binarized to have pixel values in $\{0, 1\}$.

The Alloys dataset (https://tinyurl.com/tmah538), previously studied in [16], consists of 15,000 aluminium alloys. Aluminium alloy is a combination of about 85% aluminium and other elements.

Phase diagram contains important characteristics of alloys, representing variations between the states of compounds at different temperatures and pressures. They also contain thermodynamic properties of the phases. In this experiment, a phase diagram is coded as a 2D matrix, in which each cell is the prevalence of a phase at a particular temperature.

4.2 Model settings

We use a similar architecture for the encoder and decoder of all VAE in all datasets. The encoder has 2 convolution layers with 32 and 64 filters of size 3×3, stride 2, followed by one dense layer with 100 hidden units, then two parallel dense layers to output the mean and log variance of $q(z|x;\theta)$. The decoder architecture exactly reverses that of the encoder to map from z to x, with transposed convolution layers in place of convolution layers, and outputs the Bernoulli mean of $p(x|z;\theta)$. For the alloys dataset, the convolution layers are replaced by matrix layers with size 200×200, as in [4]. We also use a similar architecture for HyperVAE in all datasets. The encoder uses the same architecture as the VAE's encoder. The decoder use a dense layer with 100 hidden units, followed by L parallel matrix layers generating the weights, biases, and filters of the main VAE network, resembling the parameter θ. The input to the matrix layer is reshaped into size 20×20. All layers except the last layer use RELU activation. The z-dimension and u-dimension is 10 for all datasets. We used Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate $\eta = 0.0003$, minibatches of size 100, and ran for 10000 iterations or when the models converge.

4.3 Model behavior

We study whether the HyperVAE learns a meaningful latent representation and data distribution for the MNIST and Omniglot datasets. We use negative log-likelihood (NLL) and $D_{KL}(q(z|x)\|p(z))$ as measures. NLL is calculated using importance sampling with 1024 samples.

Table 1. Negative log-likelihood (-LL), and $D_{KL}(q(z|x)\|p(z))$ (KL). Smaller values are better.

		VAE	MetaVAE	HyperVAE
MNIST	-LL	99.4	93.0	**88.2**
	KL	18.8	15.5	18.5
Omniglot	-LL	111.4	128.1	**105.5**
	KL	17.1	**13.2**	18.1
Fashion MNIST	-LL	237.7	232.7	**231.8**
	KL	14.5	**13.7**	13.9

Table 1 compares the performance of VAE, MetaVAE, and HyperVAE. As shown, HyperVAE has better NLL on the three datasets. MetaVAE has smallest

KL, which is due to it has a separate and fixed generator for each task. HyperVAE has slightly smaller KL on the MNIST and Fashion MNIST dataset than VAE. Note that better log-likelihoods can be achieved by increasing the number of latent dimensions, e.g. $\dim(z) = 50$, instead of $\dim(z) = 10$ in this experiment.

Table 2. Number of parameters (rounded to thousands).

	VAE	MetaVAE	HyperVAE
Inference	445	445	445
Generative	445	$445 \times$ #task	445
Total	890	$445 + 445 \times$ #task	890

Table 2 compares the number of parameters between networks. Note that while MetaVAE shares the same inference network for all tasks, it needs a separate generative network for each task. For HyperVAE, the trainable parameters are from the hyper level VAE, whereas the main VAE network of each task obtains its parameters by sampling from the HyperVAE network. Therefore, for the comparison we only count the number of trainable parameters, which is what eventually saved to disk. The real parameters for the target networks will be generated on-the-fly given a target task. Thus, it will take extra generation time for each task, c.f. Table. 3.

Table 3. Time measured in milliseconds for a batch of 100 inputs.

	Generation	Inference	Total time
VAE	0.12	0.12	0.24
MetaVAE	0.12	0.12	0.24
HyperVAE	0.12 (x)	0.12 (z)	1.11
	0.75 (θ)	0.12 (u)	

Overfitting VAE is trained on the combined dataset therefore it is less affected by overfitting due to high variance in the data. Whereas MetaVAE is more susceptible to overfitting when the number of examples in the target task is small, which is the case for Omniglot dataset, c.f. Table 1. While the training of MetaVAE's encoder is amortized across all datasets, the training of its decoder is task-specific. As a result, when a (new) task has a small number of examples, the low variance data causes overfitting to this task's decoder. Therefore MetaVAE is not suitable for transfer learning to new/unseen tasks. HyperVAE can avoid overfitting by taking a Bayesian approach.

HyperVAE complexity The algorithmic complexity of HyperVAE is about double that of VAE, since it is a VAE of VAE, plus the extra generation time of the parameters. Specifically, it runs the VAE at the hyper level to sample a weight

parameter θ, then it runs the VAE to reconstruct a set of inputs given this parameter θ. Due to the difference in matrix sizes of different layers in the target network, we generate each weight matrix and bias vector at a time, resulting in $O(D)$ time[5] with D being the depth of the target network. Therefore, the time complexity of the hyper generation network is $O(L_{\text{hyper}}+L_{\text{VAE}}+D)$, where L_{hyper} and L_{VAE} are the number of layers of the hyper and the primary generation networks respectively, and we assumed the average hidden size of the layers is a constant. However, more efficient methods is also possible. For example, inspired from [9], we can reshape matrices into batches of blocks of the same size, then stacking along the batch dimension in to a large 3D tensor. Then, we can use a matrix network to generate this tensor in $O(1)$ time[6] whence the time complex will be $O(L_{\text{hyper}} + L_{\text{VAE}})$. We leave this implementation for future investigation. Table 3 shows the wall-clock time comparison between methods on a Tesla P100 GPU.

Table 4. Outlier detection on MNIST. AUC: Area Under ROC Curve, FPR: False Positive Rate, FNR: False Negative Rate, KL: KL divergence, -EL: mean negative loglikelihood and KL.

		VAE	MetaVAE		HyperVAE
			KL	-EL	
MNIST	AUC	93.0	54.7	52.2	**95.3**
	FPR	16.3	47.5	49.4	**15.6**
	FNR	15.5	45.0	50.5	**8.0**
Omniglot	AUC	98.3	87.3	97.5	**98.7**
	FPR	5.5	18.8	7.2	**4.9**
	FNR	6.4	20.9	9.0	**5.9**
Fashion MNIST	AUC	74.6	58.2	56.8	**76.8**
	FPR	**33.5**	44.1	45.8	33.6
	FNR	32.0	43.5	44.5	**28.7**

4.4 Robust outlier detection

Next, we study HyperVAE model for outlier detection tasks. We use three datasets: MNIST, Omniglot, and Fashion MNIST to create three outlier detection experiments. For each experiment, we select one dataset as the normal class and 20% random samples from another dataset as outliers. All methods are trained on only normal data. VAE and HyperVAE use the negative loglikelihood and KL for calculating the outlier score, which is equivalent to the negative ELBO. MetaVAE does not have a generative network for new data. Therefore we use two scoring methods: (1) KL divergence only, and (2) mean

[5] We assumed a matrix multiplication takes $O(1)$ time in GPU.
[6] Batched matrix multiplication can be paralleled in GPU.

negative log-likelihood and KL, using all trained generative networks. For training MetaVAE and HyperVAE, we define the task as before, i.e. the data in each task have a similar class label.

Table 4 compares the performance of all methods on the three datasets. Overall, HyperVAE has better AUCs compared to VAE and MetaVAE. The MetaVAE has the lowest AUC. This could be due to the use of a discrete set of generative networks for each task, making it unable to handle new, unlabeled data.

While the false positive rates of VAE and HyperVAE models are similar, the false negative rates for HyperVAE are lower than that of VAE. This is because HyperVAE was trained across tasks, thus it has a better support between tasks.

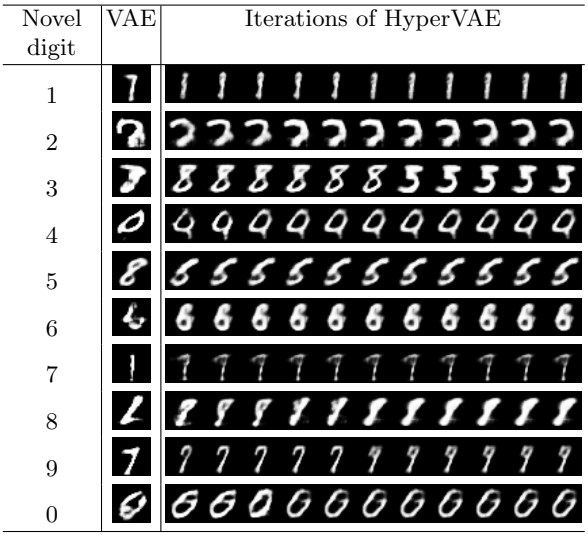
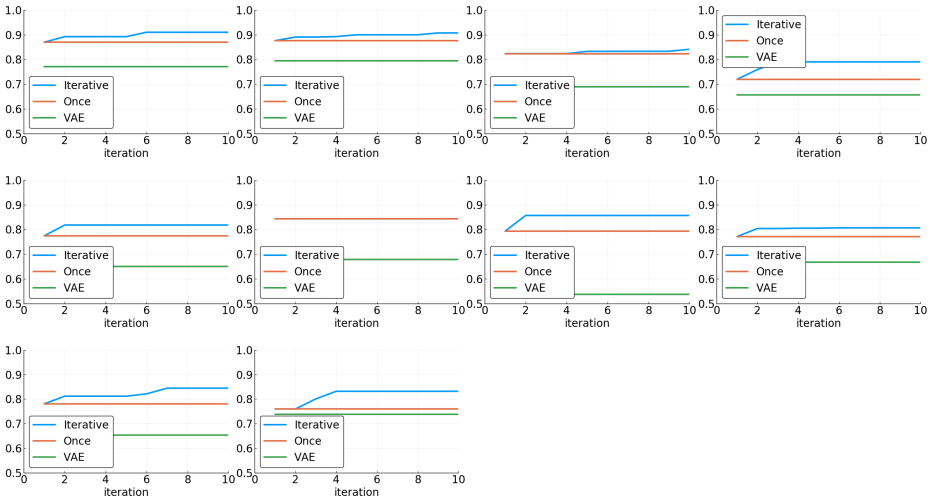


Fig. 2. Best digits found at iterative steps in searching for a new class of digits, corresponding to the performance curves in Fig. 3.

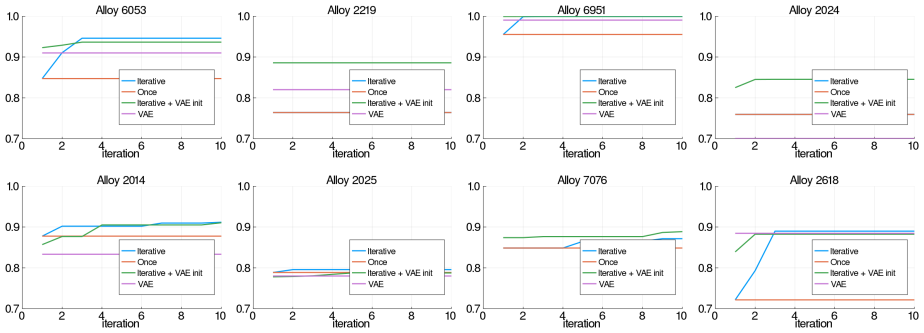
4.5 Novelty discovery

We demonstrate the effectiveness of HyperVAE+BO for finding realistic designs close to an ideal design, which lies outside known design classes. The performance measure is how close we get to the given ideal design, as measured in cosine distance for simplicity.

In each of the following two experiments, the BO objective is to search for a novel unseen design x^*, an unseen digit or alloy, by maximizing a Cosine distance $F(x^*)$. The maximum number of BO iterations is set to 300 and the search space is $[-5, 5]$ for each z and u dimension.



(a) Searching for unseen MNIST digits $\{1, \ldots, 9, 0\}$, from left to right, top to bottom.



(b) Searching for unseen Alloys.

Fig. 3. Searching for: (a) unseen digits, and (b) unseen alloys designs. Cosine distance between target and best found vs iterations. Best viewed in color.

Digit discovery This experiment illustrates the capability of HyperVAE+BO in novel exploration on MNIST. For each experiment, one digit is held out. We used nine digit classes for training and tested the model ability to search for high quality digits of the remaining unseen digit class. BO is applied to search for new digits that are similar to a given new exemplar in the z-space.

In the iterative process, an empty image $d_1 = \mathbf{0}$ is given at the first step, and subsequently updated as $d_t = x^*_{t-1}$. After each step t we set $u_t = \mu(x^*_t)$. The quality curves are presented Fig. 3 (a). Examples of discovery process are listed in Fig. 2. The figures show that VAE has a very limited capability to support exploration outside the known regions, while HyperVAE is much more flexible, even without the iterative process (#Step = 1). With more iterative refinements, the quality of the explored samples improves.

Alloy discovery We now use the framework to search for a new class of alloys. For each experiment, one alloy is held out. Models are trained on the remaining 29 alloys. We work on the phase space as a representation of the material composition space, to take advantage of the closeness of phase space to the target performance. We treat the phase diagrams as matrices whose values are proportions of phases at different temperatures. The goal is to search for a new class of alloys that is similar to the "ideal" alloy that has not been seen in any previous alloy classes. BO is applied to search for new alloys that are similar to a given new ideal alloy in the space of z. In the iterative process, we can initialize the search by an uninformative model $u_1 = \mathbf{0}$ or the one found by VAE+BO (the "Iterative + VAE init"). Subsequently the model is updated by setting $d_t = x^*_{t-1}$. The u variable is set to $u_t = \mu(x^*_t)$ after each step t.

We utilize the matrix structure of the phase diagram and avoid overfitting by using matrix representation for the input [4]. To inversely map the phase diagram back to the element composition, we use the inverse program learned from the phase-composition dataset, as described in [16]. To verify that the found materials are realistic (to account for the possible error made by the inverse program), we run the Thermo-Calc software to generate the phase diagrams. These computed phase diagrams are compared against the discovered phase diagrams. The result from Thermo-Calc confirms that the found alloys are in the target class.

To examine the effect of initialization to HyperVAE+BO performance, we initialized it by either uninformative hypothetical alloy (e.g., with hyper prior of zeros), with the alloy found by VAE+BO, or with a chosen known alloy. The performance curves are shown in Fig. 3 (b). "Once" means running HyperVAE for just one step. "Iterative + VAE init" means initialization of $d_1 = x^*$ by VAE. It shows: (a) For a majority of cases, HyperVAE+BO initialized uninformatively could find a better solution than VAE+BO, and (b) initializing HyperVAE+BO with solution found by VAE+BO boosts the performance further, sometimes a lot more. This suggests that care must be taken for initializing HyperVAE+BO.

We examine the results of the ten most difficult to find alloy targets, i.e. the alloys whose distance to their nearest alloy are largest, in descending order

of difficulty. Table 5 shows the element composition errors of found alloys. The results show that most alloys are found to be in the target class and all found alloys are close to the boundaries of their targets (at ±20%). Table 5 also shows that the Thermo-Calc phase calculation agrees with the predicted phase, i.e. small errors. The alloy 6951 and 6463 have the smallest errors compared to others.

Table 5. Column A - Element composition errors of found alloys (the composition is predicted by the method in [16]). The found alloys are expected to reside within ±20% relative error to the target alloy to stay within its class. **Column B** - Verification of phase in Thermo-Calc simulator, where the phase error is calculated as the mean error relative to the maximum proportion of each phase. The errors of the best method are reported. Alloys are ranked by their relative distance to the nearest neighbor, in decreasing order.

Alloy	A - Element error (%)	B - Phase error (%)
6053	**11.3**	3.0
2219	26.6	3.0
6951	**20.0**	0.4
2024	31.5	3.5
2014	**18.5**	1.4
2025	**4.4**	2.9
7076	31.8	2.5
2618	23.9	3.6

5 Related Work

Our method can be considered as a lossless compression strategy where the HyperVAE compresses a family of networks that parameterize the parameters of distributions across datasets. The total code length of both the model and data misfits are minimized using HyperVAE, thus help it generalize to unseen data. This is in contrast to the lossy compression strategy [1] where local information of images are freely decoded independent of the compressed information.

The HyperVAE shares some insight with the recent MetaVAE [2], but this is different from ours in the target and modeling, where the latent z is factored into data latent variable z and the task latent variable u. HyperVAE is related to Bayesian VAE, where the model is also a random variable generated from some hyper-prior. There has been some work on priors of VAE [13, 21], but using VAE as a prior for VAE is new.

HyperGAN [18] is a recent attempt to generate the parameters of model for classification. This framework generates all parameters from a single low dimension Gaussian noise vector. Bayesian neural networks (BNN) in [23] also use GAN framework for generating network parameters θ that looks real similar to one drawn from BNN trained with stochastic gradient Langevin dynamics.

However, GAN is not very successful for exploration but more for generating realistic samples.

Continual learning are gaining ground in recent years. Variational continual learning [15], for example, solves catastrophic forgetting problems in supervised learning, but it still needs a set of prototype points for old tasks. [17] tackles this problem in unsupervised tasks and also does task inference as ours, however our settings and approaches are different. Meta-learning frameworks for classification and regression [6, 24, 14] is another direction where the purpose is to learn agnostic models that can quickly adapt to a new task.

6 Conclusion

We proposed a new method called HyperVAE for encoding a family of neural network models into a simple distribution of latent representations. A neural network instance sampled from this family is capable of modeling the end task in which the family is trained on. Furthermore, by explicitly training the variational hyper-encoder network over a complex distribution of tasks, the hyper-network learns the smooth manifold of the family encoded in the posterior distribution of the family. This enables the model to extrapolate to new tasks close to trained tasks, and to transfer common factors of variation across tasks. In the handwritten digit example, the transferable factors may include writing styles, font face and size. It can be thought of as expanding the support of the distribution of trained model, thus is useful for downstream tasks such as searching for a data distribution close to existing ones and reducing the false positive error in outlier detection.

Acknowledgments

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

References

1. Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.
2. Kristy Choi, Mike Wu, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. *arXiv preprint arXiv:1902.01950*, 2019.
3. Kien Do, Truyen Tran, and Svetha Venkatesh. Matrix-centric neural networks. *arXiv preprint arXiv:1703.01454*, 2017.
4. Kien Do, Truyen Tran, and Svetha Venkatesh. Learning deep matrix representations. *arXiv preprint arXiv:1703.01454*, 2018.
5. Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

6. Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *ICLR*, 2018.
7. Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.
8. Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.
9. David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.
10. Geoffrey Hinton and Drew Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer, 1993.
11. Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
12. David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.
13. Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. Variational memory encoder-decoder. In *NeurIPS*, 2018.
14. Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *ICLR'18*, 2018.
15. Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *ICLR*, 2018.
16. Phuoc Nguyen, Truyen Tran, Sunil Gupta, Santu Rana, Matthew Barnett, and Svetha Venkatesh. Incomplete conditional density estimation for fast materials discovery. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 549–557. SIAM, 2019.
17. Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 7645–7655, 2019.
18. Neale Ratzlaff and Li Fuxin. HyperGAN: A Generative Model for Diverse, Performant Neural Networks. *ICML*, 2019.
19. Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.
20. Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.
21. Jakub M Tomczak and Max Welling. VAE with a VampPrior. *arXiv preprint arXiv:1705.07120*, 2017.
22. James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.
23. Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial distillation of bayesian neural network posteriors. In *International Conference on Machine Learning*, pages 5177–5186, 2018.
24. Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.