
HyperVAE: Variational Hyper-Encoding Network

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 θ are drawn from a distribution $p(\theta)$ which is modeled by a hyper-level VAE. Given a target distribution, we predict the posterior distribution of the latent code, then use a matrix-network decoder to generate a posterior distribution $q(\theta)$. HyperVAE can encode the parameters θ 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 evaluate HyperVAE in density estimation tasks, outlier detection and discovery of novel design classes.

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 [3, 4, 6, 8, 5] 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.

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, 11]. In 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) [10, 14] 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) [15], 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.

2 Preliminaries

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 approach that jointly learns the generative model $p(x|z; \theta)$ and a variational inference model $q(z|x; \theta)$ ¹. Its ELBO objective,

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

lower-bounds the marginal log-likelihood $\log p(x; \theta)$. In practice, Monte Carlo estimator of the ELBO's gradient is used to update the parameters θ . The form of q and p in Eq. 1 makes an encoder and a decoder, hence the name auto-encoder [10]. MLE of a single θ maximizing the above objective is susceptible to overfitting, especially when $\theta \in \Theta$ is high dimensional.

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 :

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

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^2 .

HyperVAE We propose a framework for this problem called *HyperVAE* as depicted in Fig. 1 (b). 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 :

$$p(\theta, D) = \int p(\theta, D|u)p(u)du = \int p(D|\theta)p(\theta|u)p(u)du \quad (3)$$

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

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

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.

¹We use $\theta = (\theta_p, \theta_q)$ to denote the set of parameters for p and q .

²We assume a Dirac delta distribution for γ , i.e. a point estimate, in this study.

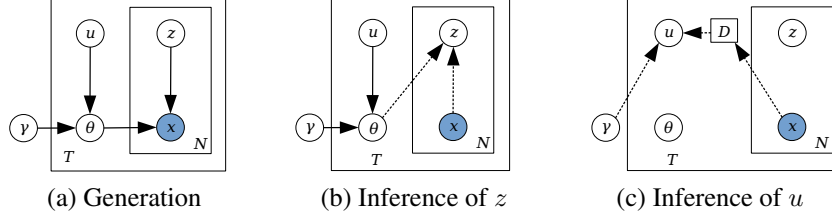


Figure 1: HyperVAE graphical model. $D = \{x_n\}_{n=1}^N$.

Inference of u (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 $\mu(\cdot)$ and $\sigma^2(\cdot)$. We approximate $q(u|D, \theta) \approx q(u|D)$. The next problem is that since D_t is a set, $q(u|D_t)$ is a function of set. Here we use a simple method to summarize D_t into a vector, $d_t = s(D_t)$, 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 Equivalent to MDL

Let assume we are working with discretized distributions each with its own precision. Under the crude 2-part code MDL, the expected code length for transmitting a dataset D and the model parameters θ for Eq. 2 is $L(D) = L(D|\theta) + L(\theta)$, where $L(D|\theta) = -\log P(D; \theta)$ is the code length of 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:

$$L(D) = L(D|\theta) + L(\theta|u) + L(u) \quad (4)$$

$$= L(D|\theta(u)) + L(u) \quad (5)$$

where we choose a Dirac delta distribution for θ , $p(\theta|u) = \delta(\theta(u))$. Using bits-back coding, this code length should be shorter by exploiting the fact that the choice of u can carry additional information up to the entropy of the variational posterior distribution $q(u|D)$ [17]. The expected code length for u over $q(u|D)$ is then:

$$L(u) = \mathbb{E}[-\log P(u) + \log Q(u|D)] = \mathbb{E}\left[\log \frac{q(u|D)}{p(u)}\right] = \text{KL}(q(u|D)||p(u)) \quad (6)$$

The final expected code length of D in HyperVAE is then:

$$L(D) = \mathbb{E}_{q(u|D)} [L(D|\theta(u))] + \text{KL}(q(u|D)||p(u)) \quad (7)$$

3.4 Training objective for HyperVAE

We train the HyperVAE parameters by minimizing the description length in Eq. 7. The description length of a dataset $D = \{x_i\}_{i=1}^{|D|}$ is the summation of the description length of every data point in it, $L(D|\theta(u)) = \sum_{i=1}^{|D|} L(x_i|\theta(u))$. Application of the bits-back coding argument to each $L(x_i|\theta)$ gives:

$$L(x_i|\theta) = \mathbb{E}_{q_\theta(z|x_i)} [L(x_i|z, \theta)] + \text{KL}(q_\theta(z|x_i)||p(z)) \quad (8)$$

where we ignored the dependence of θ on u to avoid clutter. Substitute Eq. 8 into Eq. 7 we obtain the training objective for HyperVAE as follows:

$$L(D) = \mathbb{E}_{q(u|D)} \left[\sum_{i=1}^{|D|} \mathbb{E}_{q_{\theta(u)}(z|x_i)} [L(x_i|z, \theta(u)) + \text{KL}(q_{\theta(u)}(z|x_i)||p(z))] \right] + \text{KL}(q(u|D)||p(u))$$

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.

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.

3.5 Compact hyper-decoder architecture

Since neural networks weights are matrices that are highly structured and often overparameterized, we use an efficient matrix network to decode 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.

3.6 Applications

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 , 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 $x \in D_t$.

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 . The search process for the optimal design at step $t = 1, 2, \dots, T$ is as follows:

$$\begin{aligned} u_t &\sim q(u | d_{t-1}); & \theta_t &= g_\gamma(u_t); \\ z^* &\leftarrow \text{BO}(g_{\theta_t}(z)); & x_t^* &\leftarrow g(z^*). \end{aligned} \quad (9)$$

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.

4 Experiments

We use MNIST handwritten digits, Omniglot handwritten characters, and Fashion MNIST. In these three datasets, the images are statically binarized to have pixel values in $\{0, 1\}$.

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 $p(x|z; \theta)$. 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.

4.1 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 $\text{KL}(q(z|x)||p(z))$ as measures. NLL is calculated using importance sampling with 1024 samples. We compare VAE and HyperVAE in Table 1, and as seen the HyperVAE outperforms the VAE on NLL on MNIST dataset and on NLL and KL on Omniglot dataset.

Table 1: Negative ELBO (-EL), negative log-likelihood (-LL), $\text{KL}(q(z|x)||p(z))$, and $\text{KL}(q(u|x)||p(u))$ (u). Smaller values are better.

	MNIST			Omniglot			Fashion MNIST		
	-EL	-LL	KL	-EL	-LL	KL	-EL	-LL	KL
VAE	118.2	99.4	18.8	128.6	111.4	17.1	252.2	237.7	14.5
MetaVAE	108.6	93.0	15.5	141.1	128.1	13.2	245.2	232.7	13.7
HyperVAE	106.8	88.2	18.5 5.7 (u)	124.7	105.5	18.1 2.7 (u)	245.7	231.8	13.9 0.4 (u)

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.

Table 2: Number of parameters (thousands) of the methods.

	Inference network	Generative network	Total
VAE	445	445	890
MetaVAE	445	$445 \times \text{\#task}$	$445 + 445 \times \text{\#task}$
HyperVAE	445	445	890

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 and using the MDL principle for regularization.

HyperVAE complexity The algorithmic complexity of HyperVAE is just double that of VAE since it is a VAE of VAE. 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 . The weight generation network generates each weight matrix and bias vector at a time at its last layer. Therefore, the time complexity of the hyper generation network is $O((L_{\text{hyper}} + L_{\text{VAE}})H)$, where L_{hyper} and L_{VAE} are the number of layers of the hyper and the primary generation networks respectively, and H is the average hidden size of the layers.

4.2 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 ELBO for calculating the outlier score, since the description length in Eq. 8 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, and (2) mean negative ELBO 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 3 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.

Table 3: Outlier detection on MNIST. AUC: Area Under ROC Curve, FPR: False Positive Rate, FNR: False Negative Rate, KL: KL divergence, -EL: mean negative ELBO.

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

4.3 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.

4.3.1 Digit discovery



















Novel digit	1	2	3	4	5	6	7	8	9	0
VAE										
HyperVAE										

Figure 2: Best digits found at iterative steps in searching for a new class of digits.

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 $u_t = \mu(x_t^*)$. 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.

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 [12, 16], but using VAE as a prior for VAE is new.

HyperGAN [13] 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 [18] 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.

6 Conclusion

We proposed a new method called HyperVAE for encoding a family of neural network models into a simple distribution of latent representations.

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] 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.
- [4] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *ICLR*, 2018.
- [5] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. *arXiv preprint arXiv:1902.08438*, 2019.
- [6] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 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] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [11] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.
- [12] Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. Variational memory encoder-decoder. In *NeurIPS*, 2018.
- [13] Neale Ratzlaff and Li Fuxin. HyperGAN: A Generative Model for Diverse, Performant Neural Networks. *ICML*, 2019.
- [14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.
- [15] 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.
- [16] Jakub M Tomczak and Max Welling. VAE with a VampPrior. *arXiv preprint arXiv:1705.07120*, 2017.
- [17] James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.
- [18] 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.