Variational Variance: Simple, Reliable, Calibrated Heteroscedastic Noise Variance Parameterization

Andrew Stirn
Columbia University
andrew.stirn@cs.columbia.edu

David A. Knowles
Columbia University & New York Genome Center
daknowles@cs.columbia.edu

Abstract

Brittle optimization has been observed to adversely impact model likelihoods for regression and VAEs when simultaneously fitting neural network mappings from a (random) variable onto the mean and variance of a dependent Gaussian variable. Previous works have bolstered optimization and improved likelihoods, but fail other basic posterior predictive checks (PPCs). Under the PPC framework, we propose critiques to test predictive mean and variance calibration and the predictive distribution's ability to generate sensible data. We find that our attractively simple solution, to treat heteroscedastic variance variationally, sufficiently regularizes variance to pass these PPCs. We consider a diverse gamut of existing and novel priors and find our methods preserve or outperform existing model likelihoods while significantly improving parameter calibration and sample quality for regression and VAEs.

1 Introduction

The machine learning community ubiquitously employs neural networks to map conditioning (random) variables onto the parameter space of other model variables. This technique leverages the expressive power of deep learning while preserving probabilistic interpretability. For example, we often map covariates onto the simplex with neural networks to parameterize a categorical distribution over observed labels in classification. Parameterizing the mean and variance of a normal distribution with neural networks is also prevalent (Nix and Weigend, 1994; Kingma and Welling, 2013; Rezende et al., 2014) but problematic. In particular, if our conditional mean network predicts nearly perfectly (i.e. $\mu(x_i) \approx y_i \; \forall \; i \in [N]$), then maximizing the log likelihood will push the variance network $\sigma^2(x_i)$ towards a pathological 0. This tendency coupled with the fact that σ^{-2} (precision) appears as a multiplicative factor in the gradient of the normal log likelihood w.r.t. μ, underlies why jointly optimizing mean and variance networks can be unstable. As the mean estimates $\mu(x_i)$ improve, the log likelihood encourages $\sigma^{-2}(x_i) \to \infty$ such that minuscule errors by the mean network can produce inappropriately large parameter updates. The variance network $\sigma^2(x_i)$ effectively controls the learning rate of the mean network $\mu(x_i)$–increasing it as the mean network improves–in direct opposition to the stochastic gradient descent convergence criteria of Robbins and Monro (1951). While good optima can be found when these criteria are not met, instability has been observed to reduce model likelihoods when optimizing mean and variance networks in regression (Detlefsen et al., 2019) as well as when using mean and variance networks for Gaussian decoder likelihoods in VAEs (Takahashi et al., 2018). Also, Dai and Wipf (2019) identify the theoretical preference of an optimal decoder for zero variance.

While substantial progress has been made in producing accurate means of posited model densities, reliable variance estimation has been more elusive despite its critical importance for numerous applications in today's machine learning arena. For regression, accurate variance estimates enable Bayesian *active learning* (Cohn et al., 1996) and *reinforcement learning* (Ghavamzadeh et al., 2016) regimes where new data is requested or exploration carried out based on predictive variance. Realistic sample generation requires well-calibrated variance and is critical to *adversarial learning* and *data imputation.* We carefully inspected the code of many state-of-the-art VAE methods and found that the generated 'samples' were rarely sampled from the predictive (decoder) distribution. Instead, these methods ancestrally resample latent variables from the variational posterior and report the expected value of the decoder density. This 'sampling' procedure is actually a Monte-Carlo estimate–often using just a single sample from $q(z|x)$–of the posterior predictive mean, $\mathbb{E}[x^*|x] \triangleq \int x^* p(x^*|z) q(z|x)\ dz$. While preserving uncertainty on the latent space by sampling $q(z|x)$, reporting this expectation over the observed data space obfuscates any uncertainty in the predictive density. Furthermore, approximating the posterior predictive distribution, $p(x^*|x) \triangleq \int p(x^*|z) q(z|x)\ dz$, with too few z samples can lead to inaccurate predictive variance estimates. VAE papers claiming improvements to sample quality (van den Oord et al., 2017; Razavi et al., 2019) and imputation (Nazabal et al., 2018; Mattei and Frellsen, 2018b) do *not* sample the predictive distribution despite sometimes fitting a global (homoscedastic) scalar variance to improve mean calibration (Dai and Wipf, 2019).

Perhaps the two most commonly reported performance metrics for regression and VAEs are the log likelihood and the root mean square error (RMSE) of the model's expected value. In VAE papers, it is not always clear if the reported log likelihood is the *expected log likelihood*, $\mathbb{E}_{q(z|x)}[\log p(x^*|z)]$, from the variational objective or the *log posterior predictive probability*, $\log p(x^*|x) = \log \mathbb{E}_{q(z|x)}[p(x^*|z)]$, where x^* is a replicated x. The former is a lower bound of the latter via Jensen's inequality. For consistency, we always use the (posterior) predictive distributions, $p(y|x)$ for regression and $p(x^*|x)$ for VAEs, since they are equally well-defined for the frequentist and Bayesian methods we consider. We too evaluate log likelihoods and mean RMSEs, but also look beyond to variance. To do so, we adopt the framework of posterior predictive checks (PPCs) (Gelman et al., 2013). PPCs posit a well-fit model should, with high probability, produce new data that looks similar to the observed data since any discrepancy could be the result of model misfit or chance. A common PPC is to evaluate the posterior predictive likelihood on a replicated set of the training data or better yet on a held-out validation or test set. Alternatively, one can sample values from the predictive distribution and look for systemic discrepancies with the original data that may indicate model failure. Here, we conduct PPCs from several perspectives. We assess mean and variance calibrations by measuring bias and RMSE between the predictive mean and the data and between the predictive variance and the empirical variance w.r.t. the predictive mean. We also measure bias and RMSE between the original data and samples taken from the predictive distribution; this tests the model's ability to generate sensible data thereby critiquing cooperation of the mean and variance. We provide PPC specifics for regression and VAEs respectively in sections 3 and 4.

In this article, we advocate a Bayesian treatment of the predictive distribution's variance (or rather precision for computational convenience). Treating variance variationally induces a Kullback–Leibler (KL) divergence, which, for an appropriate prior, will produce gradients that prohibit variance from approaching the aforementioned zero pathology (somewhat analogous to logarithmic barriers to enforce constraints in convex optimization) and alleviates the theoretical concern that maximum likelihood estimation (MLE) of continuous VAEs is ill-posed for unbounded likelihood functions (Mattei and Frellsen, 2018a). Detlefsen et al. (2019) and Takahashi et al. (2018) have addressed optimization instabilities to improve regression and VAE model likelihoods, respectively. Here, we look beyond likelihood to holistically improve model performance through the lens of PPCs. In section 2, we review relevant variational inference concepts. In section 3, we formalize our proposed methods for regression and experimentally compare our methods to a variety of baselines. In section 4, we do the same but for VAEs. We emphasize that our proposals broadly apply to both regression and continuous VAEs and notably outperform other methods specific to each context.

2 Amortized Variational Inference

Variational inference (VI) (Blei et al., 2017) posits a family of tractable distributions $q(\Theta; \nu)$ to approximate the true posterior $p(\Theta|\,\mathcal{D})$ over latent variables Θ conditioned on observed data $\mathcal{D}$. We assume i.i.d. data such that $p(\mathcal{D}\,|\Theta) = \prod_{d \in \mathcal{D}} p(d|\Theta)$. Often, and as in the case of *amortized inference* (Kingma and Welling, 2013), we use a neural network with shared learnable parameters ϕ to map data d onto the variational parameters ν (i.e. $f_\phi : d \to \nu$). Amortized VI minimizes the variational posterior's KL divergence from the true posterior, $D_{KL}(q(\Theta|\,\mathcal{D})\ ||\ p(\Theta|\,\mathcal{D}))$, by maximizing the evidence lower bound (ELBO or $\mathcal{L}$ or short),

$$\sum_{d \in \mathcal{D}} \mathop{\mathbb{E}}_{q(\Theta|f_\phi(d))} \big[\log p(d|\Theta)\big] - D_{KL}\big(q(\Theta|f_\phi(d))\ ||\ p(\Theta)\big),$$

since KL divergence is strictly non-negative and the summation of these dual objectives equals a constant. We focus on Gaussian likelihoods with mean μ and precision λ. If we treat λ variationally–we consider it a latent variable (i.e. $\lambda \in \Theta$), specify a prior $p(\lambda)$ to describe its generative process, and employ a variational family $q(\lambda|\cdot)$ to approximate the posterior (or factor thereof)–then the KL divergence above will contain $D_{KL}(q(\lambda|\cdot)\ ||\ p(\lambda))$. This regularizing term can fortify optimization as discussed in section 1 and, with well-informed priors, ideally will find distributions over λ that accurately reflect the local predictive ability of mean network $\mu(\cdot)$.

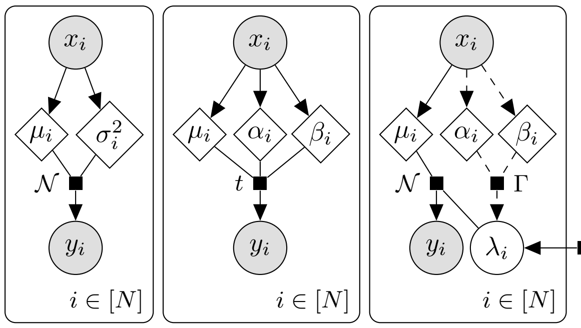

Figure 1: Graphical Models for Regression: Normal, Student's t, and Variational Variance (left to right). Diamonds are deterministic neural network parameter maps. Solid arrows denote the generative process. Dashed arrows define the variational family.

3 Variational Variance for Regression

Homoscedastic regression assumes $y_i = f(x_i) + \epsilon_i$ where $f(x_i)$ is some unknown function of the covariates and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ captures noise on y_i via the unknown global variance parameter σ^2. Notably, the noise ϵ_i has no dependence on the covariates x_i. Maximizing the log likelihood of the data under a normal treatment can be accomplished by first minimizing $\sum_{i=1}^N (y_i - \mu(x_i))^2$ w.r.t. $\mu(\cdot)$ (in our case, a neural network) and thereafter setting the global noise variance $\sigma^2 = N^{-1} \sum_{i=1}^N (y_i - \mu(x_i))^2$. Unfortunately, homoscedasticity is often assumed out of convenience rather than to reflect prior knowledge of the true generative process. One can introduce heteroscedasticity by additionally parameterizing the noise variance with a neural network $\sigma^2(\cdot)$ operating on the covariates. The generative process for this model appears in the leftmost model of fig. 1, has local likelihood $p(y_i|x_i) \triangleq \mathcal{N}(y_i|\mu(x_i), \sigma^2(x_i))$, can be optimized by MLE, and yields a predictive distribution that is simply the factorized normal likelihoods since there are no priors. We refer to this model as the **Normal** model and note $\sigma^2(\cdot)$ applies a softplus as its final activation to ensure positive variances.

3.1 Existing Methods We Use as Baselines

In addition to optimization instabilities, Detlefsen et al. (2019) observed that simultaneously learning neural mean and variance parameterizations can also underestimate the true variance, especially in areas of covariate space with scarce data. They motivate and propose four tricks to ameliorate these issues. First, they argue a batch containing x_i, but lacking other nearby data, while sufficient for updating the mean, is insufficient for updating the variance (unless the mean is already known). Accordingly, they propose a 'locality sampler' that ensures any batch sample (x_i, y_i) is accompanied by its K nearest neighbors (w.r.t. x_i), which are found in pre-training. Unfortunately, nearest neighbor distance can produce meaningless relationships for high dimensional data such as natural images. Second, they optimize the mean and variance networks in isolation (analogous to coordinate ascent). The first half of training fits only the mean network (using a fixed variance) to ensure that, during the latter half of training where coordinate ascent alternates every few batches, variance estimation is feasible since the mean network is presumably now reasonable. Third, they utilize a Gamma-Normal parameterized Student's t, $\mathrm{T}(y_i|x_i) \equiv \int_0^\infty \mathcal{N}(y_i|\mu(x_i), \lambda_i) \operatorname{Gamma}(\lambda_i|\alpha(x_i), \beta(x_i)) d\lambda_i$, as the likelihood, which again results in a predictive distribution that factors into local likelihoods (fig. 1, middle). This parameterization highlights that the Student's t distribution is a scaled mixture of Gaussians with unknown precision λ_i. From one perspective, the Student's t regression model is an MLE problem unto itself. Alternatively, one could consider it an Empirical Bayes MAP (maximum a posteriori) estimation with local likelihood $\mathcal{N}(y_i|\mu(x_i), \lambda_i)$ and local heteroscedastic precision prior $\operatorname{Gamma}(\lambda_i|\alpha(x_i), \beta(x_i))$, where the shared $\alpha(\cdot)$ and $\beta(\cdot)$ neural prior parameterizations are fit during inference. Lastly, they extrapolate variance as a learnable convex combination between the estimated heteroscedastic variance (inverted samples from the parameterized Gamma) and some pre-defined, larger, non-trainable variance. They perform ablation and find that their methods are complementary with the locality sampler and Student's t distribution individually providing the most benefit. We use **Detlefsen** to refer to their top method, which employs all four of their proposals and generally outperforms their chosen baselines: Gaussian processes (Williams and Rasmussen, 2006; Snelson and Ghahramani, 2006; Damianou and Lawrence, 2013), unmodified neural-network parameterizations of mean and variance (Nix and Weigend, 1994; Bishop, 1994; Kingma and Welling, 2013; Rezende et al., 2014), Bayesian neural networks (MacKay, 1992; Hernández-Lobato and Adams, 2015), and Monte-Carlo Drop Out (Gal and Ghahramani, 2016). We independently implemented just their Student's t proposal and refer to it as **Student**. This proposal in isolation is an important baseline since our variational methods also produce a Student's t posterior predictive $p(y|x)$.

3.2 Proposed Regression Methods

In contrast to Detlefsen et al. (2019), we propose a single, simple modification: treat precision variationally (fig. 1, right). The Student's t variance can be undefined and arbitrarily close to ∞, which makes it famously robust against outliers, but, as we will see, unfortunately hamstrings its ability generate sensible data under our PPC framework. Depending on its parameters, a Gamma prior over precision can saturate in a single- (effectively lower bounding variance) or double-sided (upper and lower bounding variance) manner. Thus, we can avoid optimization instabilities while also regularizing variance to pass our PPCs.

Employing amortized VI may appear as a superfluous inference procedure since the exact posterior, $p(\lambda|x, y)$, is available (see supplement), however, it factors into $\prod_{i=1}^{N} p(\lambda_i|x_i, y_i)$ such that local precision depends on both x_i and y_i. This undesirable dual dependence could mean $p(\lambda|y_i, x) \neq p(\lambda|y_j, x)$ when $y_i \neq y_j$. Thus, the exact posterior falls outside the scope of heteroscedasticity, where any realization of x should surjectively map onto variance or its distribution's parameter space. Our variational family, $q(\lambda_i|x_i) \triangleq \text{Gamma}(\lambda_i|\alpha(x_i), \beta(x_i))$ satisfies this requirement. Amortized VI preserves the modeling capacity of the Student's t regression (Detlefsen et al., 2019) as it requires the same number of neural parameterizations and too yields a Student's t posterior predictive. To summarize, we give up posterior exactness for heteroscedasticity and the ability to probabilistically regularize precision's variational distribution.

We employ black-box VI (Ranganath et al., 2014) in conjunction with reparameterization gradients (Salimans et al., 2013; Kingma and Welling, 2013; Rezende et al., 2014; Figurnov et al., 2018) to maximize our variational objective,

$$\begin{aligned}\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \mathop{\mathbb{E}}_{q(\lambda|\alpha(x),\beta(x))} \Big[\log \mathcal{N}(y|\mu(x), \lambda)\Big] \\ - D_{KL}\big(q(\lambda|\alpha(x), \beta(x)) \;||\; p(\lambda)\big), \qquad (1)\end{aligned}$$

w.r.t. the networks $\mu(\cdot)$, $\alpha(\cdot)$, and $\beta(\cdot)$. The first expectation of eq. (1) conveniently evaluates analytically

$$\frac{1}{2}\Big(\psi(\alpha(x)) - \log\beta(x) - \log(2\pi) - \frac{\alpha(x)}{\beta(x)}(y - \mu(x))^2\Big)$$

($\psi(\cdot)$ is the Digamma function) for univariate y and, with a diagonal covariance assumption, for multivariate y. Networks $\alpha(\cdot)$ and $\beta(\cdot)$ employ softplus activations to ensure they give positive parameter values.

3.2.1 Precision Priors

Typically, one defines a generative process before specifying inference methods, yet here, we did the reverse since we seek a probabilistically principled way to regularize a distribution over precision that depends solely on the covariates: $q(\lambda|\alpha(x), \beta(x))$, our variational posterior. As such, we still must define the priors whose regularization effects we wish to evaluate. We consider both homoscedastic priors of form $p(\lambda)$ as well as heteroscedastic priors of form $p(\lambda|x)$, but note we use $p(\lambda)$ as in eq. (1) to refer to both homo- and heteroscedastic priors. Since heteroscedasticity is always available to our variational posterior, $q(\lambda|\alpha(x), \beta(x))$, we really only care about which prior(s) offer optimal PPC performance, but note a heteroscedastic prior creates a generative process with explicit heteroscedasticity, whereas, with a homoscedastic prior, we only recover heteroscedasticity as a result of our inference choices. Performance aside, one may find a philosophical preference to have heteroscedasticity exist congruently in both the generative process and in inference.

We begin by introducing our considered homoscedastic priors. First, we use a standard **Gamma** prior $p(\lambda) \triangleq \text{Gamma}(\lambda; a, b)$, where a and b are scalar parameters specified a priori. Second, we use what we call a **Variational Posterior (VAP)** prior. This prior independently sets $p(\lambda_i) \triangleq q(\lambda_i|\alpha(x_i), \beta(x_i))$ for each data point such that the KL divergence penalty in eq. (1) vanishes. This 'prior' serves as an ablation test to confirm the beneficial regularization of the KL divergence. The variational objective (eq. (1)) with a VAP prior becomes a lower bound of the log predictive likelihood of the Student's t regression via Jensen's inequality. We additionally consider the Empirical Bayes **VAMP** prior (Tomczak and Welling, 2017), which is the prior that maximizes the ELBO: the aggregate posterior $p^*(\lambda) = N^{-1}\sum_{i=j}^{N} q(\lambda|\alpha(x_j), \beta(x_j))$, taken over the N training points. We note that this summation marginalizes out heteroscedasticity. For computational efficiency, Tomczak and Welling (2017) propose using $K < N$ randomly selected (without replacement) training points (pseudo-inputs) instead of all N. They denote the j'th pseudo-input as u_j. Additionally, they introduce the concept of treating pseudo-inputs $\{u_1, \ldots, u_K\}$ as trainable parameters which back propagation can modify, which we denote with **VAMP***.

For heteroscedastic priors, we first consider our novel modification to the VAMP prior, **xVAMP**

$$p(\lambda|x) \triangleq \sum_{j=1}^{K} \pi_j(x) q\big(\lambda|\alpha(u_j), \beta(u_j)\big).$$

Heteroscedasticity is preserved using $\pi(x)$, a neural network that maps x onto the simplex, to determine the mixture proportions. This augmentation decomposes the KL divergence from eq. (1) into

$$\mathop{\mathbb{E}}_{q(\lambda|x)}[\log q(\lambda|x)] - \mathop{\mathbb{E}}_{q(\lambda|x)}\left[\log \sum_{j=1}^{K} \pi_j(x) q(\lambda|u_j)\right], \quad (2)$$

where we evaluate the first term analytically as the Gamma distribution's negative entropy and Monte-

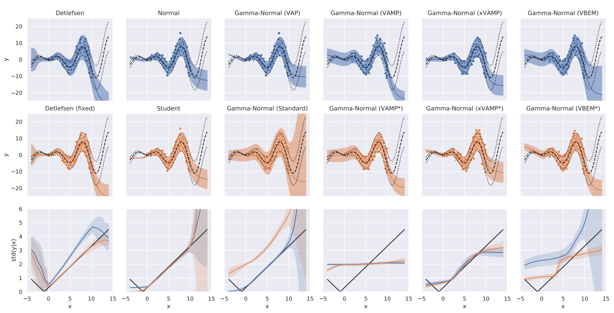

Figure 2: Toy Regression Results. Top two rows: dots are training data, black dashed/dotted lines and colored lines/areas are the true and predictive $\mathbb{E}[x|y] \pm 2 \cdot \sqrt{\text{var}(y|x)}$, respectively. Third row: the true (black) and average predictive (colors correspond to methods above) $\sqrt{\text{var}(y|x)}$ for 20 trials (area is one deviation).

Carlo estimate the second using the log-sum-exp trick. We derive eq. (2) in our supplement. We too consider trainable pseudo-inputs for our xVAMP prior, which we denote as **xVAMP***. Our second heteroscedastic prior is mixture of Gamma distributions

$$p(\lambda|x) \triangleq \sum_{j=1}^{K} \pi_j(x) \,\text{Gamma}(\lambda|a_j, b_j),$$

where again the mixture proportions, $\pi(x)$, depend on the covariates via a trainable mapping onto the simplex. We denote this prior as **VBEM**, which stands for *Variational Bayes Expectation Maximization*, since optimizing the prior parameters during VI is analogous to performing M steps. The resulting KL divergence is identical to eq. (2) except that we replace $q(\lambda|\alpha(u_j), \beta(u_j))$ with $p(\lambda|a_j, b_j)$. The non-trainable set of scalar parameters $\{a_j, b_j\}_{j=1}^K$ is the Cartesian square of a set of scalars ranging from 0.05 to 4.0 (see supplement for details). Again, we consider a version with trainable parameters, which we note as **VBEM***. Here, however, we randomly initialize trainable parameters $\{\hat{a}_1, \hat{b}_1, \ldots, \hat{a}_K, \hat{b}_K\}$ using a Uniform($[-3, 3]$). To ensure valid VBEM* parameters, we apply the softplus to these parameters (e.g. $a_j = \text{softplus}(\hat{a}_j)$). Because precision is local to each data, there is always a 1:1 ratio of likelihoods to KL divergences. Thus, the Bayesian truism that growing the data set will eventually overwhelm the prior does not apply here.

3.3 Toy Data

Detlefsen et al. (2019) simulate heteroscedastic data. Similarly, we define a toy process $y \triangleq x \cdot \sin(x) + \epsilon$ where $\epsilon|x \sim \mathcal{N}(0, [0.3 \cdot (1 + x)]^2)$. We sample training covariates uniformly from $[0, 10]$ and test over equally spaced points in $[-4, 14]$. Detlefsen et al. (2019)'s code's normal log likelihood's log variance term was missing a $\frac{1}{2}$ (this bug only affected this particular experiment). We ran their code with and without our correction to assess its impact. We also mimicked their implementation specifics (see supplement). We find all methods adeptly estimate the true mean on the training interval $[0, 10]$ (top two rows of subplots, fig. 2). Fixing Detlefsen et al. (2019)'s bug significantly improves their ability to learn the true variance on $[0, 10]$. Our VAP prior is equally accurate on this interval. We set the parameters for the standard Gamma prior using the MLE parameters of a Gamma distribution fit over the the training interval's true precision values. The standard Gamma and VBEM priors similarly overestimate the true variance on parts of $[0, 10]$, but are the only methods to consistently overestimate the variance outside the training interval $[0, 10]$ (a desiderata of Detlefsen et al. (2019)). The VAMP and VAMP* priors are poor at capturing heteroscedastic variance, which we attribute to the aforementioned marginalization of heteroscedasticity. Namely, $D_{KL}(q(\lambda|x)||K^{-1}\sum_{j=1}^{K} q(\lambda|u_j))$ is minimized when the variational distributions are approximately uniform. Indeed, they predict homoscedasticity with a nearly constant standard deviation that is approximately equal to the expected value of the true standard deviation over the training interval, 1.8. The xVAMP and xVAMP* priors generally capture the true variance albeit with some inaccuracy near the closure of $[0, 10]$. VBEM* behaves similarly but with added inaccuracy across the entire training interval.

3.4 UCI Data

We consider many of the same UCI datasets as Detlefsen et al. (2019) and process them similarly: we independently whiten all features and targets to enforce zero mean and unit variance, while reporting performance metrics for the original target scalings. We collect metrics from randomly held-out validation sets that each constitute 10% of the data across 20 trials. We match the remaining implementation specifics to Detlefsen et al. (2019) (see supplement for details).

We report model likelihood, $N^{-1}\sum_{i=1}^{N} \log p(y_i|x_i)$, normalized by N, the number of validation data. Recall $p(y_i|x_i)$, the (posterior) predictive likelihood, is a Student's t for all methods except the Normal model. The remaining PPC metrics require residuals for the predictive mean $\mathbb{E}[y_i|x_i] - y_i$, variance $\text{var}[y_i|x_i] - \left(\mathbb{E}[y_i|x_i] - y_i\right)^2$, and samples $\left(y^* \sim p(y_i|x_i)\right) - y_i$. We note the expectations and variances are w.r.t. $p(y_i|x_i)$. The predictive variance is $\sigma^2(x)$ for the Normal model and $\forall \alpha(x) > 1 : \frac{\beta(x)}{\alpha(x)-1}$ (i.e. the expectation of an Inverse-Gamma), which is always available since we

Table 1: UCI Log Predictive Probability (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Sun et al. (2019)	N/A	*-2.30±0.04*	–	*-3.10±0.02*	*-0.68±0.02*	*7.13±0.02*
Detlefsen	N/A	-2.98±0.09	8.77±0.24	-3.66±0.08	-4.90±0.27	9.67±0.19
Normal	N/A	-2.42±0.23	13.20±1.35	-3.06±0.17	-0.48±0.69	14.15±0.17
Student	N/A	-2.37±0.19	**17.19±0.21**	-3.10±0.17	0.22±0.31	13.60±0.39
Gamma-Normal	VAP	-2.36±0.17	15.52±0.24	-3.12±0.17	0.17±0.44	13.36±0.41
	Standard	-2.48±0.29	11.28±0.02	-3.20±0.16	-1.05±0.18	12.33±0.16
	VAMP	-2.39±0.17	14.37±0.17	-3.09±0.16	-0.18±0.21	14.16±0.78
	VAMP*	-2.39±0.16	14.38±0.12	-3.09±0.16	-0.16±0.20	13.96±0.88
	xVAMP	**-2.33±0.17**	15.38±0.24	-3.01±0.14	0.05±0.28	13.50±0.59
	xVAMP*	-2.33±0.17	15.41±0.18	-3.01±0.13	0.11±0.39	13.34±0.47
	VBEM	-2.46±0.11	4.57±1.00	-3.11±0.07	-4.52±0.26	9.02±0.61
	VBEM*	-2.36±0.14	14.64±0.16	**-2.99±0.13**	**0.49±0.28**	**14.42±0.15**

Algorithm	Prior	power plant (9568, 4, 1)	superconductivity (21263, 81, 1)	wine-red (1599, 11, 1)	wine-white (4898, 11, 1)	yacht (308, 6, 1)
Sun et al. (2019)	N/A	*-2.83±0.01*	–	–	–	*-1.03±0.03*
Detlefsen	N/A	-3.26±9.1e-03	-5.21±0.02	-1.04±0.06	-1.12±0.04	-3.15±0.10
Normal	N/A	-2.82±0.05	-3.51±0.10	-0.92±0.05	-1.05±0.04	-1.55±0.65
Student	N/A	**-2.78±0.03**	-3.41±0.05	**-0.80±0.10**	-1.05±0.04	-1.73±0.59
Gamma-Normal	VAP	-2.81±0.04	-3.45±0.06	-0.87±0.06	-1.04±0.04	-1.79±0.50
	Standard	-2.88±0.03	-3.45±0.04	-0.98±0.07	-1.13±0.05	-1.73±0.38
	VAMP	-2.83±0.03	-3.94±0.02	-0.94±0.05	-1.05±0.04	-2.83±0.70
	VAMP*	-2.83±0.03	-3.94±0.03	-0.94±0.05	-1.05±0.04	-2.77±0.77
	xVAMP	-2.81±0.04	-3.40±0.04	-0.90±0.05	-1.03±0.04	-1.68±0.38
	xVAMP*	-2.81±0.04	**-3.39±0.05**	-0.89±0.06	-1.03±0.04	-1.71±0.47
	VBEM	-2.89±0.05	-3.77±0.09	-0.91±0.05	-1.03±0.03	-2.64±0.23
	VBEM*	-2.81±0.03	-3.41±0.04	-0.89±0.06	**-1.03±0.04**	**-1.11±0.57**

offset $\alpha(x)$'s softplus output by 1. For each residual, we compute bias (i.e. the mean) and root mean square error (RMSE) over the N validation points.

Table 1 contains UCI log predictive probabilities with top performers in bold. We include recent competitive results (Sun et al., 2019), which we italicize to emphasize that they are reported–we did not reproduce them. Often the reported models utilized larger neural networks than we did. If multiple architectures were reported, we selected the one closest in size to ours. We report tables corresponding to the remaining six PPC metrics in our supplement, though we include a summary in table 2, which tallies the number of the datasets for which a method was the top performer and, in parentheses, was statistically indistinguishable from the winner according to a two-sided Kolmogorov–Smirnov test with a $p \leq 0.05$ significance threshold. Because we could not obtain all PPC metrics for reported results, we exclude them from the summary table, but append them to the corresponding PPC table in our supplement if they were available.

Examining table 2, we find that our baselines generally under perform in each category compared to our methods. Student's and VAP's under performance confirms the benefit of regularizing variance. Our VBEM* prior offers the best balance of performance, winning log likelihood and posting a competitive number of statistical ties for the remaining PPCs. Since particular applications may place higher emphasis on certain PPC categories, we encourage readers to utilize table 2 as a guide when picking a method most appropriate for their application.

4 Variational Variance for VAEs

The variational autoencoder (VAE) (Kingma and Welling, 2013) is a deep latent variable model (DLVM) that provides computationally efficient VI for a generative process from a low-dimensional latent local variable z_i to high-dimensional data x_i. We place a $p(z) \triangleq \mathcal{N}(0, I)$ prior on the latent variables and perform inference by defining $q(z|x) \triangleq \mathcal{N}(z|\mu_z(x), \sigma_z^2(x))$, where $\mu_z(x)$ and $\sigma_z^2(x)$ are bifurcated outputs of the same neural network. A softplus is applied to the variances to ensure positivity. The VAE's ELBO is

$$\sum_{x \in \mathcal{D}} \mathop{\mathbb{E}}_{q(z|x)} \Big[\log \mathcal{N}(x|\mu_x(z), \sigma_x^2(z)) \Big] - D_{KL}\big(q(z|x) \;||\; p(z)\big). \tag{3}$$

Parameter maps $\mu_x(z)$ and $\sigma_x^2(z)$ can be either bifurcated outputs of the same neural network (**VAE**) or separate neural networks (**VAE-Split**). We evaluate both architectures with and without batch normalization (**+ BN**). Detlefsen et al. (2019) apply their regression proposals to VAEs. We refer the interested reader to their manuscript for details. Takahashi et al. (2018) propose using a Student's t likelihood to bolster optimization and improve model likelihood. Their

Table 2: UCI Regression Summary: reported as wins (statistical ties)

Algorithm	Prior	LL	Mean Bias	Mean RMSE	Var Bias	Var RMSE	Sample Bias	Sample RMSE
Detlefsen	N/A	0 (0)	2 (3)	1 (1)	0 (0)	0 (1)	0 (4)	0 (0)
Normal	N/A	0 (3)	1 (7)	2 (7)	1 (6)	2 (7)	2 (9)	**3** (6)
Student	N/A	3 (**7**)	0 (7)	0 (6)	0 (5)	0 (3)	0 (9)	2 (5)
Gamma-Normal	VAP	0 (4)	0 (9)	0 (6)	0 (7)	0 (4)	**3** (9)	0 (5)
	Standard	0 (0)	0 (7)	0 (6)	**2** (6)	0 (5)	0 (7)	0 (5)
	VAMP	0 (3)	**3** (**10**)	0 (8)	1 (7)	**3** (**9**)	1 (**10**)	1 (**9**)
	VAMP*	0 (3)	0 (**10**)	1 (8)	**2** (7)	1 (**9**)	0 (**10**)	2 (**9**)
	xVAMP	1 (4)	2 (9)	0 (7)	1 (6)	1 (7)	0 (9)	2 (6)
	xVAMP*	1 (4)	1 (9)	0 (7)	1 (**8**)	0 (8)	1 (9)	0 (6)
	VBEM	0 (1)	0 (**10**)	**5** (**10**)	0 (0)	2 (6)	1 (9)	0 (0)
	VBEM*	**5** (**7**)	1 (9)	1 (7)	**2** (5)	1 (7)	2 (9)	0 (5)

method, **VAE-Student**, results in an ELBO

$$\sum_{x\in\mathcal{D}} \mathop{\mathbb{E}}_{q(z|x)}\Big[\log \mathrm{T}(x|\mu_x(z),\lambda_x(z),\nu_x(z))\Big] - D_{KL}\big(q(z|x)\;||\;p(z)\big) \tag{4}$$

with three separate neural networks, $\mu_x(z)$, $\lambda_x(z)$ and $\nu_x(z)$ for mean, precision, and degrees-of-freedom, respectively. Since the Student's t variance is undefined for $\nu_x(z) \in (0,1]$, infinite for $\nu_x(z) \in (1,2]$, and arbitrarily close to ∞ for $\nu_x(z) \approx 2$, we restrict $\nu_x(z) > 3$ using a shifted softplus. We found that allowing the posterior predictive to attain these high variances worsens its PPC performance beyond what we report. Takahashi et al. (2018) additionally propose their **MAP-VAE** where precision is absorbed into the likelihood: $p(\lambda|z) \triangleq \text{Gamma}(\lambda|a,b)$ for predefined constants a and b. The MAP-VAE's ELBO is identical to eq. (3) except for the additional log likelihood term and replacing $\sigma_x^2(z)$ with a network that outputs precision. Our method, **V3AE** (variational variance VAE) treats precision variationally and uses $q(\lambda|z) \triangleq \text{Gamma}(\lambda|\alpha(z),\beta(z))$ as its variational family. We consider the same priors discussed in section 3.2.1 for λ, except that we now condition on latent codes z_i. Our resulting ELBO,

$$\sum_{x\in\mathcal{D}} \mathop{\mathbb{E}}_{q(z|x)}\Bigg[\mathop{\mathbb{E}}_{q(\lambda|z)}\Big[\log\mathcal{N}(x|\mu_x(z),\lambda)\Big] - D_{KL}\big(q(\lambda|\alpha(z),\beta(z))\;||\;p(\lambda)\big)\Bigg] - D_{KL}\big(q(z|x)\;||\;p(z)\big), \tag{5}$$

introduces a KL divergence that regularizes the predictive variance. See supplement for additional details.

A posterior predictive distribution is always the expected likelihood w.r.t. the (variational) posterior. The VAE (+ BN), VAE-Split (+ BN), and MAP-VAE's predictive distribution is the expected normal likelihood w.r.t. $q(z|x)$. Because the decoder employs neural networks that operate on z, this integral is not analytically available. We therefore estimate it with 20 Monte-Carlo samples. The resulting approximation becomes a uniform mixture of Gaussians with one component for each z sample. Similarly, the VAE-Student's approximate posterior predictive is a uniform mixture of Student's t distributions. Our V3AE methods have two variational distributions $q(z|x)$ and $q(\lambda|z)$. Integrating the V3AE's normal likelihood w.r.t. $q(\lambda|z)$ is analytically tractable and gives back a Student's t. Thereafter, we approximate $q(z|x)$ integration again by taking a uniform mixture over z samples of Student t distributions.

Having explicated the posterior predictive approximations for each VAE method, we can now define our PPCs. We report normalized model likelihood, $N^{-1}\sum_{i=1}^N \log p(x_i^*|x_i)$, for which x_i^* denotes a replicated x_i (i.e. $x_i^* \equiv x_i$). Our other PPCs require residuals for the predictive mean: $\mathbb{E}[x_i^*|x_i] - x_i$, variance: $\text{var}[x_i^*|x_i] - (\mathbb{E}[x_i^*|x_i] - x_i)^2$, and samples: $\big(x^* \sim p(x_i^*|x_i)\big) - x_i$. We rely on TensorFlow's mixture distribution support for generating the log likelihoods, means, variances, and samples associate with our posterior predictive approximations. Here, we focus on mean/sample RMSE and variance bias.

We report VAE PPC metrics for the Fashion MNIST dataset in table 3. Top performers are in bold as well as any method that is statistically indistinguishable using the same test from section 3.4. In fig. 3, we curate a subset of the VAE methods to qualitatively visualize our PPCs. We include tabular results for MNIST in our supplement, but note they trend similarly. Therein, one can also find similar figures but for all methods with additional data samples.

Comparing the Fixed-Variance VAEs confirms Dai and Wipf (2019)'s claim that variance impacts mean quality. While fixing variance to 0.001 produces the crispest means and samples, we sacrifice tremendous likelihood, heteroscedasticity, and our ability to analyze model uncertainty. The Student VAE dominates

Table 3: VAE PPCs for Fashion MNIST (mean±std.)

Method	LL	Mean RMSE	Var Bias	Sample RMSE
Fixed-Var. VAE (1.0)	-730.05±0.11	0.15±1.8e-03	0.98±5.7e-04	1.01±3.7e-04
Fixed-Var. VAE (0.001)	-1452.44±3.65	**9.4e-02±4.6e-05**	-7.8e-03±8.6e-06	**9.9e-02±4.7e-05**
VAE	2154.31±42.11	0.25±1.4e-03	3.3e-02±1.5e-03	0.39±3.1e-03
VAE + BN	1639.39±15.33	0.20±1.8e-03	2.1e-02±3.0e-03	0.31±5.6e-03
VAE-Split	2099.28±39.97	0.27±2.9e-03	4.7e-02±1.6e-03	0.45±4.8e-03
VAE-Split + BN	1948.30±25.87	0.26±6.2e-03	3.1e-02±3.6e-03	0.41±1.1e-02
Detlefsen	-1624.12±8.3e-03	0.16±8.2e-04	9.97±3.3e-04	3.17±1.2e-03
MAP-VAE	1003.51±32.75	0.11±4.1e-03	-9.1e-03±6.2e-04	0.13±4.8e-03
Student-VAE	**3134.52±18.60**	0.29±3.3e-03	7.4e-02±1.6e-02	0.49±2.2e-02
V3AE-VAP	2146.46±67.83	0.28±3.5e-03	**9.9e-04±3.0e-03**	0.40±8.2e-03
V3AE-Gamma	1201.95±25.25	0.11±2.8e-03	-8.0e-03±4.1e-04	0.12±3.4e-03
V3AE-VAMP	1632.22±12.89	0.17±1.3e-03	1.5e-03±2.7e-04	0.25±2.4e-03
V3AE-VAMP*	1630.10±17.87	0.18±2.6e-03	1.3e-03±2.5e-04	0.25±3.4e-03
V3AE-xVAMP	1601.60±21.49	0.18±2.5e-03	1.3e-03±4.1e-04	0.25±3.8e-03
V3AE-xVAMP*	1619.97±25.95	0.18±3.3e-03	1.5e-03±4.6e-04	0.25±5.0e-03
V3AE-VBEM	306.46±1.04	0.10±6.8e-04	6.4e-02±9.7e-05	0.29±3.3e-04
V3AE-VBEM*	1153.11±4.20	0.10±5.5e-04	**4.4e-04±3.9e-05**	0.15±8.0e-04

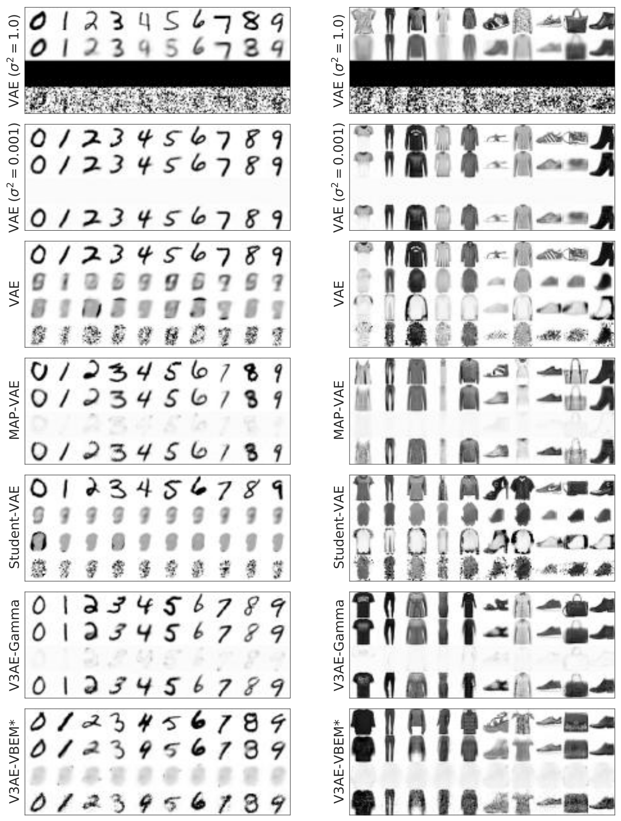

Figure 3: VAE PPC Visualization: The rows within a subplot from top to bottom are randomly selected test data followed by the posterior predictive mean and variance and a sample from it. Pixel values are clamped to $[0, 1]$, when PPC values exit this interval.

likelihood, but fails terribly at our other PPCs rendering predictive means and samples unrecognizable. This paradox highlights the need for our additional PPCs. The standard VAEs with heteroscedastic variances attain likelihoods similar to many of our methods, but perform notably worse than those same methods on the other PPCs. Detlefsen's VAE has the worst variance calibration and sample quality because their variance extrapolation learns to fully weight the larger constant variance of 10, which is excessive for data in $[0, 1]$. The MAP-VAE and our V3AE-Gamma make up much of likelihood lost when fixing variance to 0.001. These two methods also produce crisp samples and well-calibrated predictive means and variances. We note, however, our V3AE-Gamma attains superior likelihoods. Interestingly, these two methods learn predictive variances that indicate the models are most uncertain at edge localization.

5 Conclusion

In this manuscript, we motivate and advocate a probabilistically principled method for regularizing neural network variance parameterizations that broadly applies to regression and continuous VAEs. Our experiments highlight that model likelihood is not necessarily the complete story. Baseline methods with high likelihoods can have poor predictive mean and variance calibrations and also exhibit systematic sampling discrepancies. As we discuss, these undesirable characteristics unfortunately are indicative of model failure. Conversely, our proposed methods boast competitive model likelihoods and improve performance under this holistic set of model critiques. Additionally, our methods preserve heteroscedasticity and thereby enable model uncertainty analyses. Thus, our methods may be of interest to those working on reinforcement learning, active learning, and data imputation tasks.

References

Bishop, C. M. (1994). Mixture density networks.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Dai, B. and Wipf, D. (2019). Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.

Detlefsen, N., Jørgensen, M., and Hauberg, S. (2019). Reliable training and estimation of variance networks. In *Advances in Neural Information Processing Systems*, pages 6323–6333.

Figurnov, M., Mohamed, S., and Mnih, A. (2018). Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pages 441–452.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2016). Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

Mattei, P.-A. and Frellsen, J. (2018a). Leveraging the exact likelihood of deep latent variable models. In *Advances in Neural Information Processing Systems*, pages 3855–3866.

Mattei, P.-A. and Frellsen, J. (2018b). Miwae: Deep generative modelling and imputation of incomplete data. *arXiv preprint arXiv:1812.02633*.

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2018). Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822.

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Salimans, T., Knowles, D. A., et al. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*.

Takahashi, H., Iwata, T., Yamanaka, Y., Yamada, M., and Yagi, S. (2018). Student-t variational autoencoder for robust density estimation. In *IJCAI*, pages 2696–2702.

Tomczak, J. M. and Welling, M. (2017). Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

van den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

6 Introduction

Hereafter, we include supplementary material for our manuscript. Please refer to our arxiv version for the latest results. Our code is available at `https://github.com/astirn/VariationalVariance`. We organize this supplement using the same section names as the main article. Any reference to the supplement from the main article will appear in the corresponding section. Figure and table numbers continue from the main article.

7 Amortized Variational Inference

This section is intentionally blank since its supplemental section is not necessary.

8 Variational Variance for Regression

8.1 Existing Methods We Use as Baselines

This section is intentionally blank since its supplemental section is not necessary.

8.2 Proposed Regression Methods

The following derivation proves that precision's true posterior for regression locally factorizes into a distribution that depends both on the covariates x_i and responses y_i. As we discuss in the main article, this dual dependence implies the true posterior falls outside the scope of heteroscedasticity due to the additional dependence on y_i.

$$\begin{aligned}
p(\lambda|y,x) &= \frac{p(y,\lambda|x)}{p(y|x)} = \frac{p(y,\lambda|x)}{\int p(y,\lambda|x)d\lambda} = \frac{p(y|x,\lambda)p(\lambda)}{\int p(y,\lambda|x)d\lambda} \\
&= \frac{\prod_{i=1}^n \mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)}{\int \prod_{i=1}^n \mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)d\lambda_i} \\
&= \frac{\prod_{i=1}^n \mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)}{\prod_{i=1}^n \int \mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)d\lambda_i} \\
&= \prod_{i=1}^n p(\lambda_i|y_i,x_i)
\end{aligned}$$

Above, we use $p(\lambda_i|y_i,x_i) \triangleq \frac{\mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)}{\int \mathcal{N}(y_i|\mu(x_i),\lambda_i)p(\lambda_i)d\lambda_i}$ to symbolically capture the local factorization.

8.2.1 Precision Priors

Here, we derive the xVAMP ELBO and decompose its KL divergence. The xVAMP generative process is

$$\begin{aligned}
u_1,\ldots,u_K &\sim \text{UniformWithoutReplacement}(\{x_1,\ldots,x_N\}) \\
\lambda_i|x_i &\sim p(\lambda_i|x_i,u_1,\ldots,u_K) \triangleq \sum_{j=1}^K \pi_j(x_i)\cdot q(\lambda_i|u_j) \\
y_i|x_i,\lambda_i &\sim p(y_i|x_i,\lambda_i) \triangleq \mathcal{N}(y_i|\mu(x_i),\lambda_i).
\end{aligned}$$

Please note that we treat $\{u_j\}_{j=1}^K$ as prior parameters (not random variables).

This generative process leads to the local (per-point) ELBO

$$\begin{aligned}
\log p(y_i|x_i) &= \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log p(y_i|x_i,\lambda_i) - \log\frac{q(\lambda_i|x_i)}{p(\lambda_i|x_i)} + \log\frac{q(\lambda_i|x_i)}{p(\lambda_i|x_i,y_i)}\right]\\
&= \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log p(y_i|x_i,\lambda_i)\right] - D_{KL}\big(q(\lambda_i|x_i)||p(\lambda_i|x_i)\big) + D_{KL}\big(q(\lambda_i|x_i)||p(\lambda_i|x_i,y_i)\big)\\
&\geq \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log p(y_i|x_i,\lambda_i)\right] - D_{KL}\big(q(\lambda_i|x_i)||p(\lambda_i|x_i)\big)\\
&= \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log p(y_i|x_i,\lambda_i) - \log q(\lambda_i|x_i) + \log\sum_{j=1}^{K}\pi_j(x_i)q(\lambda_i|u_j)\right].
\end{aligned}$$

From the ELBO, we determine

$$\begin{aligned}
D_{KL}\big(q(\lambda_i|x_i)||p(\lambda_i|x_i)\big) &= \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log q(\lambda_i|x_i) - \log\sum_{j=1}^{K}\pi_j(x_i)q(\lambda_i|u_j)\right]\\
&= -\mathbb{H}\big[q(\lambda_i|x_i)\big] - \mathop{\mathbb{E}}_{q(\lambda_i|x_i)}\left[\log\sum_{j=1}^{K}\pi_j(x_i)q(\lambda_i|u_j)\right].
\end{aligned} \tag{6}$$

For VBEM's prior parameters we use the Cartesian square of a set of scalars ranging from 0.05 to 4.0. That set of integers is

$$\{0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}.$$

8.3 Toy Data

Here, we provide the exact implementation details used during the toy regression experiments. For all methods, we employ neural networks with a single hidden layer of 50 sigmoid neurons to match Detlefsen et al. (2019). For our VAMP(*), xVAMP(*), and VBEM* priors, we set $K = 20$. For VAMP(*) and xVAMP(*), we sample

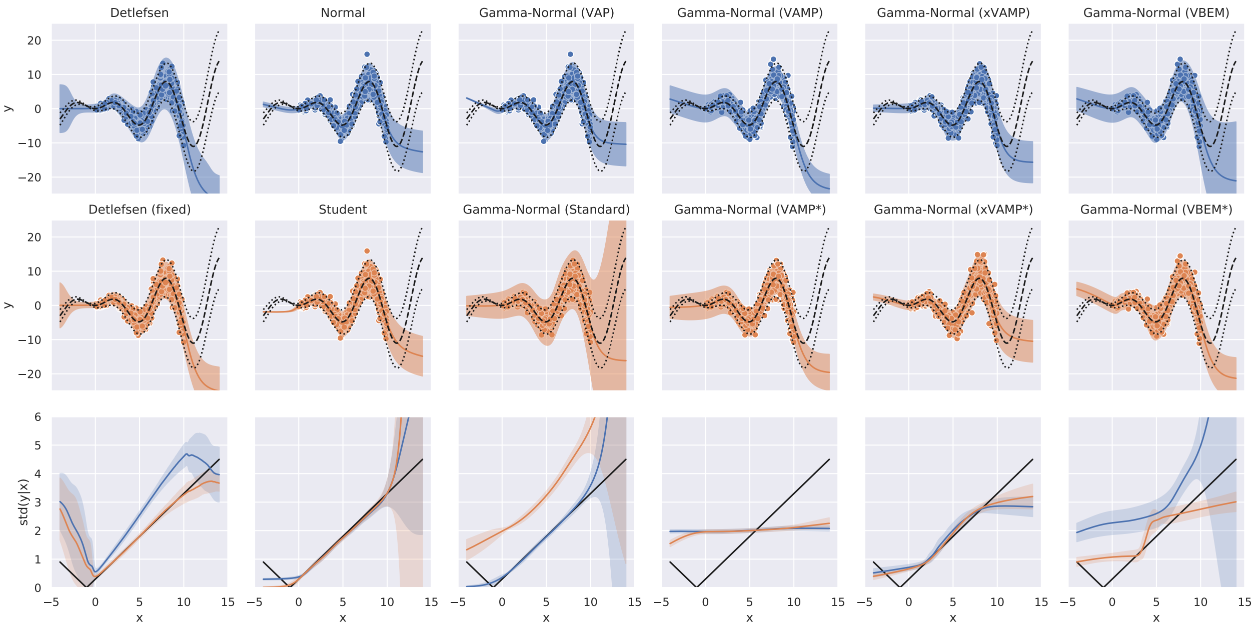

Figure 4: Toy Regression Results. Top two rows: dots are training data, black dashed/dotted lines and colored lines/areas are the true and predictive $\mathbb{E}[x|y] \pm 2 \cdot \sqrt{\text{var}(y|x)}$, respectively. Third row: the true (black) and average predictive (colors correspond to methods above) $\sqrt{\text{var}(y|x)}$ for 20 trials (area is one deviation).

pseudo-inputs $u_i \overset{iid}{\sim} \text{Uniform}([-4, 14])$. Like Detlefsen et al. (2019), we use ADAM (Kingma and Ba, 2014) for optimization. While Detlefsen et al. (2019) employ separate optimizers for the mean and variance networks that respectively use 1e-2 and 1e-3 as learning rates, we employ a single ADAM instance with a learning rate of 5e-3. We run all algorithms for 6e3 epochs without batching (i.e. the single batch contains all 500 training points). We ran the toy experiments on a NVIDA RTX2070.

We additionally include an enlarged copy (fig. 4) of the main article's toy data figure for enhanced clarity.

8.4 UCI Data

Here, we provide the experimental specifics for the UCI regression experiments. Detlefsen et al. (2019) again employ a neural network with a single hidden layer, but now with 50 ReLU neurons. We use the same network architecture except with ELU neurons, which we found to be more robust during our Monte-Carlo estimation of the right-most term of eq. (6)'s RHS. The baseline code (Detlefsen et al., 2019) allows training to run for some number of batch iterations, whereas our code uses the notion of an epoch, which encompasses the number of batches required to see each example in the training set exactly once. To keep things equal, we allow each algorithm to run for a dataset-specific number of batch iterations with a batch size of 256, which we convert to epochs ($\lceil \frac{\text{iterations}}{\text{batch size}} \rceil$) for our methods. All UCI datasets use 2e4 batch iterations except for those with larger ($N > 9000$) sample sizes (e.g. carbon, naval, power plant, and superconductivity), which use 1e5 batch iterations. Here, Detlefsen et al. (2019) use 1e-2 and 1e-4 as learning rates for the mean and variance networks, respectively. We use 1e-3 as the learning rate for our single ADAM instance. We use $a = 1$ and $b = 0.001$ as the standard Gamma prior's parameters. For VAMP(*) and xVAMP(*) we sample $K = 100$ pseudo-inputs uniformly from the training set without replacement. We also use $K = 100$ for our VBEM* prior. We employ early stopping on the validation set's log (posterior) predictive likelihood with a patience of 50 epochs. We implemented an equivalent early stopping mechanism in the baseline code (Detlefsen et al., 2019), in which we also introduced support for multivariate response variables. We ran the UCI experiments on a NVIDA RTX2070 and were able to parallelize up to five trials (i.e. five concurrent training sessions for any of the tested models).

We include the remaining PPC metrics in tables 4 to 9, where we bold just the top performer. We italicize any cited (i.e. reported) results, which we never bold since we did not validate those methods under our experimental configurations (e.g. some reported results use larger neural networks). Statistical ties for all PPC metrics are tallied in table 2 of our original manuscript.

Table 4: UCI Mean Bias (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

		boston	carbon	concrete	energy	naval
Algorithm	Prior	(506, 13, 1)	(10721, 5, 3)	(1030, 8, 1)	(768, 8, 2)	(11934, 16, 2)
Detlefsen	N/A	-0.67±1.81	-9.8e-04±3.5e-03	-0.03±3.30	0.15±0.87	-2.8e-04±6.3e-04
Normal	N/A	-0.28±0.56	5.5e-05±2.1e-04	0.32±0.66	0.06±0.10	-1.3e-04±1.2e-04
Student	N/A	-0.50±0.51	-4.3e-05±1.4e-04	-0.04±0.72	-0.06±0.13	-1.0e-04±1.5e-04
Gamma-Normal	VAP	-0.31±0.54	1.5e-04±2.2e-04	0.06±0.73	-0.03±0.12	-4.1e-05±8.1e-05
	Standard	-0.39±0.67	5.0e-05±1.9e-04	0.09±0.72	0.04±0.10	-1.4e-04±8.5e-05
	VAMP	**-0.18±0.52**	**9.5e-06±2.3e-04**	-0.10±0.58	**0.02±0.11**	-4.6e-06±5.5e-05
	VAMP*	-0.18±0.52	7.9e-05±2.0e-04	-0.10±0.58	0.02±0.11	-8.9e-06±6.0e-05
	xVAMP	-0.25±0.54	4.6e-05±1.8e-04	**9.2e-03±0.63**	0.03±0.11	**-1.8e-06±7.1e-05**
	xVAMP*	-0.24±0.54	1.3e-04±2.0e-04	0.03±0.66	0.02±0.11	3.8e-06±6.2e-05
	VBEM	-0.18±0.54	7.3e-05±3.8e-04	-0.01±0.55	0.03±0.11	1.5e-05±2.8e-05
	VBEM*	-0.19±0.58	-2.2e-05±2.1e-04	-0.02±0.63	0.03±0.11	1.9e-05±5.8e-05

		power plant	superconductivity	wine-red	wine-white	yacht
Algorithm	Prior	(9568, 4, 1)	(21263, 81, 1)	(1599, 11, 1)	(4898, 11, 1)	(308, 6, 1)
Detlefsen	N/A	0.15±0.79	1.22±3.11	0.02±0.12	**2.8e-03±0.08**	**-0.06±1.19**
Normal	N/A	-9.9e-03±0.07	0.90±0.62	**-5.1e-03±0.06**	-3.3e-03±0.03	-0.96±0.82
Student	N/A	0.06±0.08	0.29±0.54	-0.07±0.06	-0.02±0.03	-7.47±2.71
Gamma-Normal	VAP	0.02±0.05	-0.08±0.33	-0.01±0.06	-7.0e-03±0.03	-7.22±2.74
	Standard	-9.6e-03±0.11	0.74±0.33	-0.02±0.06	-0.01±0.03	-5.42±1.86
	VAMP	3.2e-03±0.06	0.09±0.38	-0.01±0.06	-0.01±0.03	-0.62±1.53
	VAMP*	3.2e-03±0.06	0.06±0.39	-0.01±0.06	-0.01±0.03	-0.60±1.53
	xVAMP	6.5e-03±0.07	-0.03±0.41	-9.6e-03±0.06	-7.9e-03±0.03	-5.44±2.46
	xVAMP*	2.0e-03±0.06	**0.01±0.49**	-8.8e-03±0.06	-8.7e-03±0.03	-5.59±2.61
	VBEM	1.2e-03±0.10	0.25±0.45	-9.6e-03±0.06	-5.8e-03±0.03	-0.17±0.52
	VBEM*	**3.0e-05±0.07**	0.09±0.46	-9.2e-03±0.06	-7.6e-03±0.02	-2.00±1.30

Table 5: UCI Mean RMSE (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Sun et al. (2019)	N/A	*2.38±0.10*	–	*4.94±0.18*	*0.41±0.02*	*1.2e-04±0.00*
Detlefsen	N/A	4.48±1.06	0.02±4.6e-03	8.13±1.65	2.05±0.49	4.2e-03±6.3e-04
Normal	N/A	3.36±1.29	**7.5e-03±3.3e-03**	6.05±0.66	1.30±0.14	3.5e-03±3.1e-04
Student	N/A	3.62±1.42	7.6e-03±3.3e-03	6.71±0.81	1.42±0.17	3.4e-03±5.0e-04
Gamma-Normal	VAP	3.44±1.21	7.7e-03±3.3e-03	6.61±0.84	1.38±0.15	3.2e-03±5.3e-04
	Standard	3.82±1.72	7.6e-03±3.3e-03	6.63±0.70	1.31±0.14	3.2e-03±5.1e-04
	VAMP	3.15±1.06	7.8e-03±3.2e-03	5.47±1.00	1.36±0.13	1.2e-03±1.0e-03
	VAMP*	3.15±1.05	7.8e-03±3.2e-03	5.47±1.00	1.36±0.13	1.6e-03±1.3e-03
	xVAMP	3.25±1.16	7.6e-03±3.3e-03	5.61±0.67	1.36±0.14	3.2e-03±5.2e-04
	xVAMP*	3.28±1.17	7.6e-03±3.3e-03	5.72±0.59	1.36±0.14	3.2e-03±4.9e-04
	VBEM	**3.14±1.07**	8.7e-03±3.3e-03	**5.26±0.58**	1.36±0.14	**5.6e-04±1.6e-04**
	VBEM*	3.18±1.12	7.6e-03±3.3e-03	5.59±0.70	**1.30±0.13**	2.4e-03±2.8e-04

Algorithm	Prior	power plant (9568, 4, 1)	superconductivity (21263, 81, 1)	wine-red (1599, 11, 1)	wine-white (4898, 11, 1)	yacht (308, 6, 1)
Sun et al. (2019)	N/A	*4.10±0.05*	–	–	–	*0.61±0.07*
Detlefsen	N/A	4.33±0.27	17.72±1.29	0.71±0.06	0.76±0.04	**2.42±1.06**
Normal	N/A	**4.12±0.20**	14.53±0.44	0.62±0.03	0.70±0.04	3.42±2.30
Student	N/A	4.12±0.19	14.85±0.42	0.63±0.03	0.71±0.03	15.03±3.30
Gamma-Normal	VAP	4.14±0.21	14.83±0.48	0.62±0.03	0.70±0.03	14.70±3.31
	Standard	4.18±0.18	14.44±0.43	0.63±0.03	0.72±0.03	12.17±2.38
	VAMP	4.16±0.20	12.81±0.33	0.62±0.03	0.70±0.04	5.42±3.54
	VAMP*	4.16±0.20	**12.80±0.35**	0.62±0.03	0.70±0.04	5.30±3.65
	xVAMP	4.14±0.20	14.13±0.39	0.62±0.03	0.70±0.04	12.30±3.09
	xVAMP*	4.13±0.21	14.25±0.42	0.62±0.03	0.70±0.03	12.51±3.20
	VBEM	4.16±0.19	13.13±0.37	**0.62±0.03**	**0.69±0.03**	3.51±1.46
	VBEM*	4.12±0.19	14.08±0.42	0.62±0.03	0.69±0.03	5.33±2.58

Table 6: UCI Variance Bias (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Detlefsen	N/A	1.0e+02±79.11	9.8e-05±1.6e-04	2.2e+02±91.85	18.60±8.88	nan±nan
Normal	N/A	31.63±1.5e+02	3.5e+23±1.6e+24	-2.01±8.67	-0.16±0.24	3.1e-07±2.0e-06
Student	N/A	18.08±63.79	0.12±0.23	-2.20±9.28	24.00±85.42	4.9e-06±2.2e-05
Gamma-Normal	VAP	3.3e+02±1.3e+03	0.25±1.11	-2.13±7.85	0.04±0.31	3.1e-07±6.8e-07
	Standard	3.18±20.09	1.5e-04±5.8e-05	0.76±11.35	0.31±0.40	3.7e-06±2.9e-06
	VAMP	-2.96±7.96	-6.6e-06±6.0e-05	-6.15±5.18	-0.15±0.39	1.7e-07±2.9e-07
	VAMP*	-3.00±7.84	**-6.0e-06±6.0e-05**	-6.16±5.18	-0.13±0.40	**1.3e-07±3.3e-07**
	xVAMP	0.65±16.20	2.7e-05±9.7e-05	-4.82±4.63	**6.0e-03±0.36**	3.1e-07±7.4e-07
	xVAMP*	0.51±20.17	5.0e-04±2.2e-03	-4.66±5.07	-8.5e-03±0.36	2.7e-07±6.3e-07
	VBEM	6.74±8.48	0.01±4.5e-03	25.86±8.94	22.06±5.58	3.6e-05±1.4e-05
	VBEM*	**-0.11±8.62**	-7.2e-06±6.1e-05	**-0.58±5.05**	0.02±0.28	3.9e-07±4.9e-07

Algorithm	Prior	power plant (9568, 4, 1)	superconductivity (21263, 81, 1)	wine-red (1599, 11, 1)	wine-white (4898, 11, 1)	yacht (308, 6, 1)
Detlefsen	N/A	69.25±2.40	5.5e+04±6.2e+03	2.16±1.57	0.83±0.36	96.62±54.08
Normal	N/A	**0.05±1.53**	2.3e+13±1.0e+14	-3.8e-03±0.04	-0.02±0.06	20.68±54.95
Student	N/A	-0.27±1.47	1.6e+05±3.3e+05	12.52±30.71	-5.6e-03±0.05	1.7e+03±2.3e+03
Gamma-Normal	VAP	0.52±1.29	9.0e+05±2.6e+06	0.03±0.05	0.13±0.64	1.3e+03±1.5e+03
	Standard	2.34±1.43	1.1e+02±81.21	0.04±0.11	**-2.3e-03±0.05**	**-7.28±40.88**
	VAMP	0.89±1.04	**-9.83±7.97**	0.05±0.06	-8.8e-03±0.04	38.05±83.39
	VAMP*	0.89±1.04	-9.89±7.98	0.05±0.06	-8.9e-03±0.04	38.07±83.29
	xVAMP	0.46±1.25	14.40±42.90	3.5e-03±0.05	-0.03±0.03	4.8e+02±1.7e+03
	xVAMP*	0.44±1.24	1.3e+02±4.7e+02	**2.1e-03±0.05**	-0.03±0.03	1.7e+02±1.5e+02
	VBEM	16.53±9.32	91.44±25.39	0.08±0.04	0.07±0.04	20.70±25.23
	VBEM*	1.86±1.44	9.87±16.22	0.05±0.06	0.01±0.04	26.48±26.88

Table 7: UCI Variance RMSE (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Detlefsen	N/A	2.5e+02±3.1e+02	2.8e-03±1.8e-03	2.9e+02±1.1e+02	21.58±10.55	nan±nan
Normal	N/A	2.8e+02±1.1e+03	2.0e+25±8.8e+25	84.48±46.55	**2.70±0.50**	3.6e-05±1.5e-05
Student	N/A	1.1e+02±1.7e+02	1.19±2.12	86.73±25.52	1.0e+02±3.6e+02	1.4e-04±4.8e-04
Gamma-Normal	VAP	2.4e+03±9.5e+03	4.74±20.81	80.57±26.95	3.64±1.10	2.3e-05±8.7e-06
	Standard	76.10±83.67	2.8e-03±1.8e-03	92.78±28.09	3.05±1.00	3.0e-05±1.1e-05
	VAMP	31.89±37.72	**2.7e-03±1.9e-03**	60.50±29.84	4.19±0.79	**5.5e-06±7.0e-06**
	VAMP*	**31.82±37.33**	2.7e-03±1.9e-03	60.49±29.84	4.19±0.79	7.8e-06±9.1e-06
	xVAMP	63.40±84.86	3.0e-03±1.7e-03	59.74±20.73	3.36±0.88	2.3e-05±8.7e-06
	xVAMP*	64.27±1.2e+02	0.02±0.05	62.39±19.00	3.19±0.78	2.2e-05±6.3e-06
	VBEM	38.91±35.36	0.01±4.3e-03	59.36±13.87	26.09±6.92	4.2e-05±1.6e-05
	VBEM*	39.54±42.93	2.8e-03±1.9e-03	**58.92±19.43**	3.72±0.47	1.5e-05±3.8e-06
Algorithm	Prior	power plant (9568, 4, 1)	superconductivity (21263, 81, 1)	wine-red (1599, 11, 1)	wine-white (4898, 11, 1)	yacht (308, 6, 1)
Detlefsen	N/A	85.20±10.25	2.2e+05±1.7e+04	7.27±9.73	2.91±3.65	1.1e+02±70.70
Normal	N/A	**43.93±18.42**	1.1e+15±4.8e+15	0.63±0.09	1.05±0.82	1.2e+02±2.7e+02
Student	N/A	44.88±18.75	5.8e+06±1.3e+07	1.2e+02±3.4e+02	0.96±0.35	5.7e+03±8.2e+03
Gamma-Normal	VAP	44.40±18.82	4.0e+07±1.2e+08	0.65±0.10	3.97±13.91	4.8e+03±5.7e+03
	Standard	45.18±18.49	1.5e+03±1.7e+03	0.77±0.38	0.94±0.20	1.1e+02±66.59
	VAMP	44.78±18.79	**4.4e+02±55.55**	0.63±0.09	0.87±0.25	99.44±1.2e+02
	VAMP*	44.78±18.79	4.4e+02±57.11	0.63±0.09	0.87±0.25	98.63±1.3e+02
	xVAMP	44.44±18.84	1.1e+03±1.8e+03	**0.63±0.11**	0.85±0.23	1.4e+03±4.8e+03
	xVAMP*	44.36±18.82	6.4e+03±2.1e+04	0.63±0.11	0.85±0.23	5.6e+02±4.9e+02
	VBEM	49.60±16.88	4.5e+02±59.88	0.63±0.11	**0.82±0.11**	**39.19±37.73**
	VBEM*	44.68±18.61	6.3e+02±4.2e+02	0.64±0.10	0.86±0.28	93.02±86.57

Table 8: UCI Sample Bias (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Detlefsen	N/A	-0.92±2.52	-9.4e-04±3.6e-03	0.49±3.93	0.17±1.03	-2.6e-04±6.1e-04
Normal	N/A	**-0.19±0.99**	4.6e+09±2.0e+10	0.39±1.05	0.06±0.15	-1.2e-04±1.4e-04
Student	N/A	-0.58±0.60	-4.3e-05±1.5e-04	-0.14±1.09	-0.07±0.15	-8.3e-05±2.0e-04
Gamma-Normal	VAP	-0.34±0.54	1.5e-04±2.1e-04	**-0.02±1.18**	-0.04±0.14	**-7.0e-06±1.1e-04**
	Standard	-0.39±0.59	7.2e-05±2.9e-04	0.04±1.02	0.04±0.13	-1.3e-04±1.2e-04
	VAMP	-0.23±0.47	**-1.8e-06±2.9e-04**	-0.17±0.83	0.01±0.12	7.4e-06±5.5e-05
	VAMP*	-0.24±0.47	6.7e-05±2.7e-04	-0.17±0.83	0.02±0.12	8.2e-06±5.9e-05
	xVAMP	-0.23±0.69	-1.8e-05±2.4e-04	-0.08±0.88	6.0e-03±0.16	-2.2e-05±1.2e-04
	xVAMP*	-0.23±0.68	7.9e-05±2.6e-04	-0.09±0.97	**2.6e-03±0.15**	-1.8e-05±1.2e-04
	VBEM	-0.19±0.85	-4.2e-04±2.0e-03	-0.15±1.07	-0.09±0.49	-2.3e-05±1.4e-04
	VBEM*	-0.24±0.70	3.3e-06±3.0e-04	0.02±0.79	0.02±0.10	1.8e-05±6.8e-05
Algorithm	Prior	power plant (9568, 4, 1)	superconductivity (21263, 81, 1)	wine-red (1599, 11, 1)	wine-white (4898, 11, 1)	yacht (308, 6, 1)
Detlefsen	N/A	0.12±0.77	1.15±4.61	0.07±0.18	-4.9e-03±0.08	-0.19±2.16
Normal	N/A	0.02±0.16	3.6e+04±1.6e+05	**1.7e-03±0.08**	3.8e-03±0.05	-0.70±1.15
Student	N/A	0.04±0.13	0.29±0.61	-0.07±0.08	-0.03±0.04	-6.26±4.34
Gamma-Normal	VAP	**-7.6e-03±0.15**	-0.06±0.41	-0.02±0.08	-0.02±0.04	-6.16±4.37
	Standard	-0.04±0.16	0.70±0.51	-0.03±0.08	-0.02±0.04	-4.92±2.49
	VAMP	-0.02±0.13	0.16±0.46	-0.02±0.08	-0.02±0.04	-0.62±1.94
	VAMP*	-0.02±0.13	0.14±0.45	-0.02±0.08	-0.02±0.04	-0.58±1.94
	xVAMP	-0.02±0.18	-0.16±0.62	-0.02±0.08	-0.02±0.04	-5.13±2.98
	xVAMP*	-0.02±0.15	-0.13±0.62	-0.02±0.08	-0.02±0.04	-5.36±3.12
	VBEM	-0.04±0.21	0.10±0.67	-0.02±0.08	-0.01±0.04	**-0.19±0.89**
	VBEM*	0.04±0.15	**0.04±0.57**	-7.3e-03±0.06	**-2.3e-03±0.04**	-1.74±1.59

Table 9: UCI Sample RMSE (mean±std.) Tuples below dataset are $(N_{\text{observations}}, \dim(x), \dim(y))$.

Algorithm	Prior	boston (506, 13, 1)	carbon (10721, 5, 3)	concrete (1030, 8, 1)	energy (768, 8, 2)	naval (11934, 16, 2)
Detlefsen	N/A	12.02±3.89	0.03±3.6e-03	17.93±2.55	5.07±0.98	6.2e-03±5.7e-04
Normal	N/A	4.92±3.57	2.6e+11±1.2e+12	8.23±1.08	**1.85±0.21**	5.0e-03±5.4e-04
Student	N/A	4.64±1.10	**8.1e-03±3.1e-03**	9.18±1.36	2.07±0.37	5.0e-03±1.7e-03
Gamma-Normal	VAP	4.69±0.86	0.01±2.2e-03	9.42±1.82	2.02±0.35	4.5e-03±7.2e-04
	Standard	4.92±2.18	0.02±1.8e-03	8.67±1.20	1.88±0.38	4.6e-03±9.1e-04
	VAMP	4.27±0.87	0.01±1.9e-03	7.27±1.05	1.93±0.19	**1.8e-03±1.4e-03**
	VAMP*	4.26±0.86	0.01±2.0e-03	**7.27±1.05**	1.93±0.20	2.2e-03±1.8e-03
	xVAMP	**4.23±1.15**	0.01±2.0e-03	7.84±1.05	1.88±0.29	4.5e-03±7.5e-04
	xVAMP*	4.23±1.14	0.01±2.5e-03	8.00±1.01	1.87±0.30	4.5e-03±7.1e-04
	VBEM	5.03±0.92	0.11±0.03	9.21±0.98	5.13±0.83	5.9e-03±1.5e-03
	VBEM*	4.41±1.07	0.01±1.9e-03	7.90±1.10	1.85±0.30	3.5e-03±3.7e-04
Algorithm	**Prior**	**power plant (9568, 4, 1)**	**superconductivity (21263, 81, 1)**	**wine-red (1599, 11, 1)**	**wine-white (4898, 11, 1)**	**yacht (308, 6, 1)**
Detlefsen	N/A	10.36±0.28	2.4e+02±15.98	1.43±0.34	1.27±0.10	10.71±2.19
Normal	N/A	5.85±0.20	1.7e+06±7.4e+06	**0.86±0.07**	0.98±0.03	**4.73±3.68**
Student	N/A	**5.79±0.28**	21.25±1.46	0.88±0.06	0.99±0.04	20.24±7.84
Gamma-Normal	VAP	5.90±0.26	21.05±0.81	0.89±0.06	0.99±0.04	20.00±7.47
	Standard	6.01±0.51	23.75±13.27	0.90±0.13	0.98±0.06	14.20±4.40
	VAMP	5.97±0.28	17.86±0.41	0.90±0.07	0.99±0.06	8.67±6.43
	VAMP*	5.97±0.28	**17.85±0.42**	0.90±0.07	0.99±0.06	8.50±6.57
	xVAMP	5.92±0.20	19.98±0.49	0.89±0.05	**0.97±0.06**	15.57±5.76
	xVAMP*	5.91±0.20	20.19±0.77	0.89±0.05	0.97±0.06	15.81±5.05
	VBEM	7.17±0.65	20.92±0.70	0.93±0.06	1.02±0.05	6.66±2.53
	VBEM*	6.00±0.20	19.78±0.48	0.89±0.07	0.97±0.03	6.84±4.58

9 Variational Variance for VAEs

For the VAE experiments, we use ADAM with a 5e-5 learning rate. All Monte-Carlo (MC) approximations use 20 samples. We found additional samples did not improve log posterior predictive probability approximations. Since our VAMP(*), xVAMP(*), and VBEM* priors require twice as many MC samples ($q(\lambda|x)$ in addition to $q(z|x)$), their memory footprint is higher, requiring a batch size of 125 on a NVIDA RTX2070. The remaining models use a batch size of 256. Because the lower batch size has twice as many batch updates per epoch, those models train for half (500) the number of epochs used by the other models (1000). We employ early stopping on the validation set's log posterior predictive probability with a patience of 25 for the 500 epoch models and 50 for the 1000 epoch models. We use an encoder architecture with hidden layers of sizes 512, 256, and 128, each of which applies an ELU activation. The decoder architecture is the transpose of the encoder. The dimensions of the latent variable, $\dim(z)$, are 10 for MNIST and 25 for Fashion MNIST.

We include PPC metrics for MNIST in table 10, which we could not fit in the main report. Additionally, we include figs. 5 and 6, which are similar to fig. 3 (main article) but have additional samples for *all* tested methods.

Table 10: VAE PPCs for MNIST (mean±std.)

Method	LL	Mean RMSE	Var Bias	Sample RMSE
Fixed-Var. VAE (1.0)	-732.10±0.11	0.17±1.1e-03	0.98±4.2e-04	1.02±2.5e-04
Fixed-Var. VAE (0.001)	-2902.66±29.23	**0.11±3.3e-04**	-1.2e-02±7.4e-05	**0.12±3.1e-04**
VAE	2593.51±267.72	0.25±2.7e-03	4.3e-02±2.6e-02	0.41±3.5e-02
VAE + BN	2386.70±23.17	0.25±1.8e-03	0.13±2.6e-02	0.50±2.6e-02
VAE-Split	2282.32±65.63	0.25±2.7e-03	7.4e-02±2.6e-02	0.44±3.2e-02
VAE-Split + BN	2482.36±75.34	0.28±4.4e-03	6.2e-02±1.1e-02	0.47±1.5e-02
Detlefsen	-1561.89±1.36	0.18±6.7e-04	9.12±2.1e-02	3.03±3.2e-03
MAP-VAE	1291.42±6.94	0.13±1.9e-03	-1.3e-02±4.1e-04	0.15±2.0e-03
Student-VAE	**4826.82±530.95**	0.27±1.7e-02	0.38±0.45	0.68±0.28
V3AE-VAP	3243.11±445.47	0.24±3.5e-03	**8.1e-04±9.7e-04**	0.34±5.5e-03
V3AE-Gamma	1495.01±2.75	0.13±7.0e-04	-1.2e-02±1.9e-04	0.15±9.2e-04
V3AE-VAMP	2355.12±13.40	0.20±6.7e-04	**6.2e-04±1.1e-03**	0.28±1.7e-03
V3AE-VAMP*	2270.76±41.89	0.20±7.9e-04	**1.2e-03±1.1e-03**	0.29±2.2e-03
V3AE-xVAMP	2323.38±94.35	0.20±2.6e-03	**1.9e-03±6.8e-04**	0.29±3.2e-03
V3AE-xVAMP*	2280.13±48.29	0.20±2.0e-03	**6.5e-04±7.2e-04**	0.29±3.7e-03
V3AE-VBEM	296.95±0.92	0.12±8.1e-04	6.1e-02±2.7e-04	0.30±2.7e-04
V3AE-VBEM*	2107.63±5.44	0.14±1.2e-03	**1.6e-03±1.1e-04**	0.20±1.6e-03

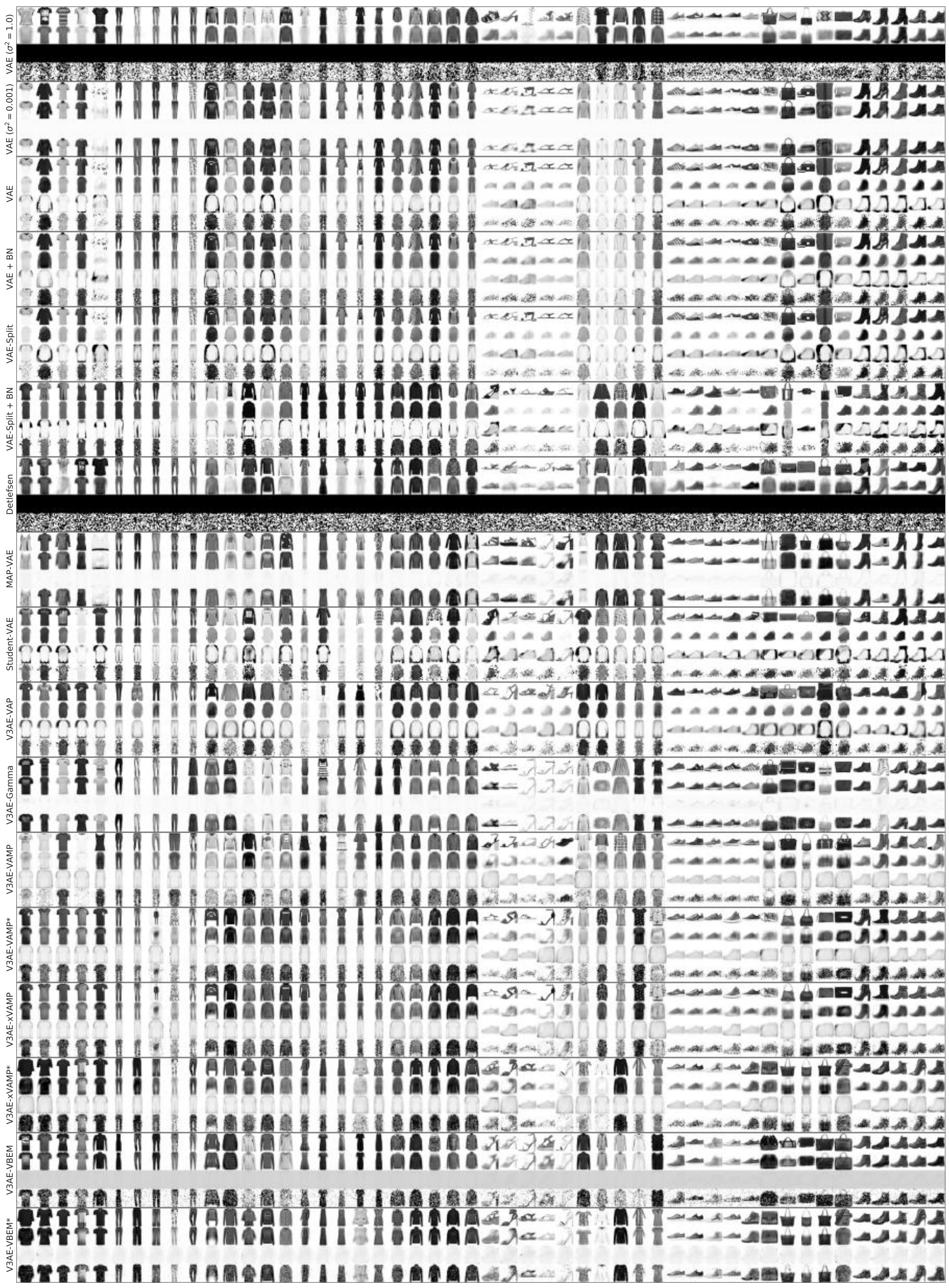

Figure 5: VAE PPC Visualization for Fashion MNIST: The rows within a subplot from top to bottom are randomly selected test data followed by the posterior predictive mean and variance and a sample from it. Pixel values are clamped to [0, 1], when PPC values exit this interval.

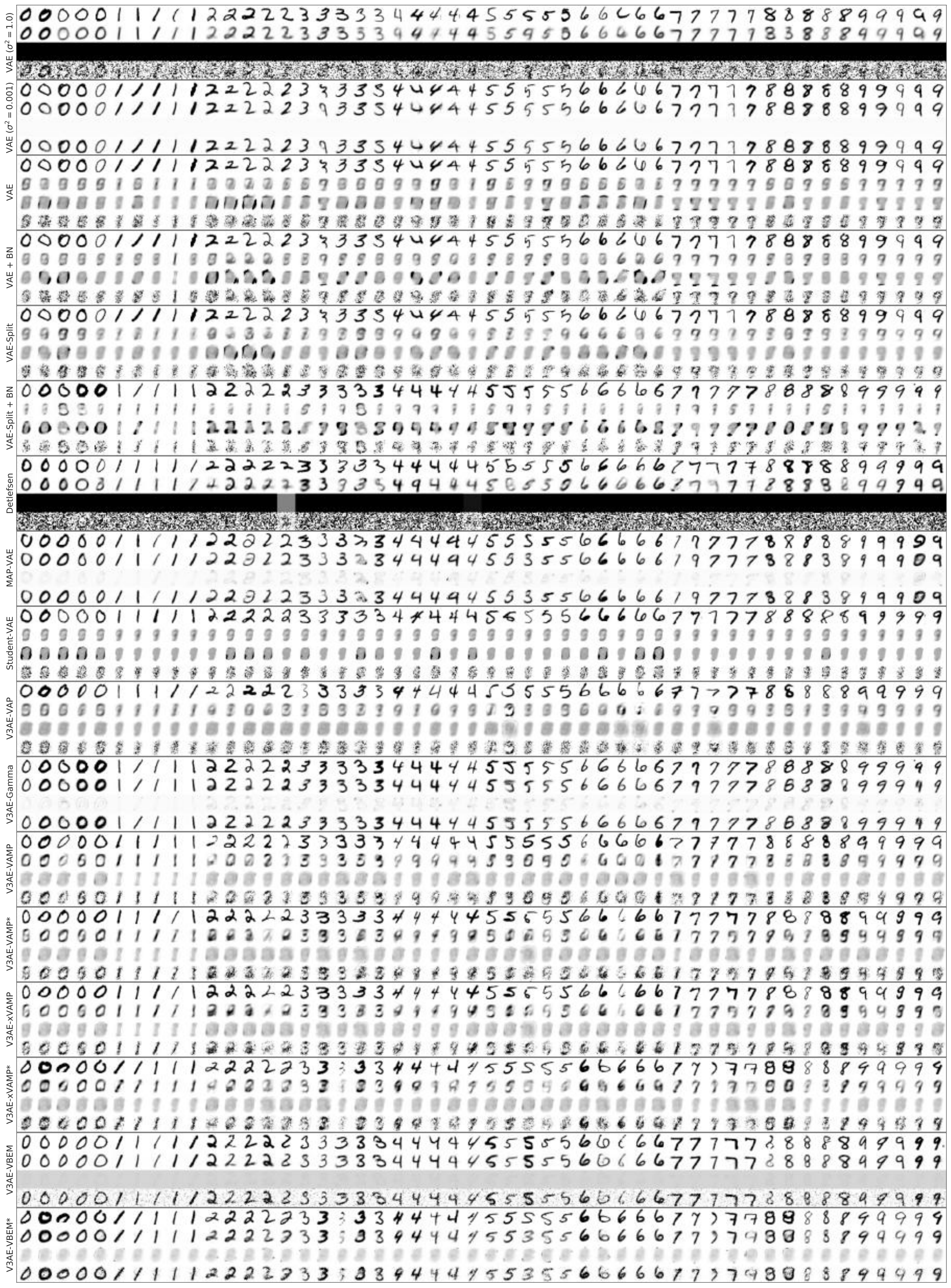

Figure 6: VAE PPC Visualization for MNIST: The rows within a subplot from top to bottom are randomly selected test data followed by the posterior predictive mean and variance and a sample from it. Pixel values are clamped to $[0, 1]$, when PPC values exit this interval.