Faithful Heteroscedastic Regression with Neural Networks

Andrew Stirn
Columbia University (CU)

Hans-Hermann Wessels
New York Genome Center (NYGC)

Megan Schertzer
CU & NYGC

Laura Pereira
NYGC

Neville E. Sanjana
New York University & NYGC

David A. Knowles
CU & NYGC

Abstract

Heteroscedastic regression models a Gaussian variable's mean and variance as a function of covariates. Parametric methods that employ neural networks for these parameter maps can capture complex relationships in the data. Yet, optimizing network parameters via log likelihood gradients can yield suboptimal mean and uncalibrated variance estimates. Current solutions side-step this optimization problem with surrogate objectives or Bayesian treatments. Instead, we make two simple modifications to optimization. Notably, their combination produces a heteroscedastic model with mean estimates that are provably as accurate as those from its homoscedastic counterpart (i.e. fitting the mean under squared error loss). For a wide variety of network and task complexities, we find that mean estimates from existing heteroscedastic solutions can be significantly less accurate than those from an equivalently expressive mean-only model. Our approach provably retains the accuracy of an equally flexible mean-only model while also offering best-in-class variance calibration. Lastly, we show how to leverage our method to recover the underlying heteroscedastic noise variance.

1 INTRODUCTION

Model uncertainty can be categorized as epistemic or aleatoric (Der Kiureghian and Ditlevsen, 2009). Epistemic uncertainty is that which gathering more data or refining models can reduce. For example, uncertainty in model parameters due to lacking sufficient data is epistemic. Aleatoric uncertainty is that which more data or model refinement cannot reduce. Measurement noise is a source of aleatoric uncertainty–growing a noisy dataset will eventually saturate model performance. Ideally, a model's predictive variance accurately captures both epistemic and aleatoric uncertainty; doing so enables active learning (Gal et al., 2017), reinforcement learning (Osband et al., 2016; Chua et al., 2018; Yu et al., 2020), and identification of uncertain predictions. For example, a biologist that uses a model to design high efficacy CRISPR guides might only consider those with low predictive uncertainty.

In regression, noise variance comes in two flavors. Homoscedastic noise variance is constant across all data and can be represented with a fixed global parameter. A model that only learns a mapping from covariates onto the response variable's mean via minimizing the sum of squared errors is implicitly homoscedastic. Heteroscedastic noise variance changes w.r.t. to covariates. Regression models can capture heteroscedasticity in the data by learning a function from covariates to variance in addition to the function from covariates to the mean.

Mapping covariates onto the parameter space of a heteroscedastic Gaussian with neural networks and minimizing the negative log likelihood (NLL) (Nix and Weigend, 1994) is the de-facto method underlying many modern approaches to predictive uncertainty estimation in regression. Kendall and Gal (2017) use Monte Carlo dropout (Gal and Ghahramani, 2016) to accumulate parameter estimates across sampled dropouts such that their empirical distribution better captures predictive uncertainty. Lakshminarayanan et al. (2017) do the same, but accumulate parameter estimates from an ensemble of models instead. Despite its prevalence, minimizing the NLL of a heteroscedastic Gaussian has many pitfalls. Takahashi et al. (2018) identify optimization instabilities that occur when variance is driven towards zero for data whose mean estimates have near-zero error. Skafte et al. (2019) find that Monte Carlo and ensemble approaches can underestimate the true variance. Seitzer et al. (2022) show heteroscedastic variance

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

estimation can degrade predictive mean performance.

Summary of Contributions. Consider a heteroscedastic neural network parameterizing a response variable's mean and covariance. Eliminating all computations in the computational graph that are not ancestors of the mean output yields a mean-only model (a subnetwork of the original heteroscedastic model). We propose using this mean-only model as the baseline for assessing the heteroscedastic model's mean estimates since both models have identical expressive power for mean estimation. We label any heteroscedastic model optimized separately from its mean-only baseline as 'faithful' if both models have equally accurate mean estimates. 'Faithfulness' requires that adding a covariance output head to a mean-only network and retraining it should not worsen its mean estimates. We formalize this notion of 'faithfulness' in definition 2, section 2.

We propose two modifications to a heteroscedastic Gaussian model's NLL minimization that both save computations and are very easy to implement. Notably, their combination guarantees a 'faithful' heteroscedastic model with mean estimates that are equivalent to its mean-only baseline's. We experimentally confirm this claim and show our method achieves best-in-class variance calibration.

Current heteroscedastic methods combine aleatoric and epistemic uncertainty (Kendall and Gal, 2017; Lakshminarayanan et al., 2017) in the predictive variance. Our method offers reliable mean and variance estimates regardless of the true noise variance, which, for certain datasets, we can leverage to accurately decompose predictive variance into its aleatoric and epistemic components. Accurately isolating aleatoric uncertainty enables scientists to study heteroscedasticity in observed systems.

2 METHODS

Preliminaries. Homoscedastic regression models a response variable $Y|X = x \sim \mathcal{N}(\mu(x), \Sigma)$. The mean function $\mu(\cdot)$ operates on covariates x, while the (co)variance parameter has no dependence on x. Heteroscedastic regression models $Y|X = x \sim \mathcal{N}(\mu(x), \Sigma(x))$, with (co)variance now being a function of covariates $\Sigma(x)$. Homoscedasticity is often a convenient rather than accurate assumption. Heteroscedastic models can capture homoscedasticity if $\Sigma(\cdot)$ learns a constant output that matches the true global (co)variance. For brevity, we will use variance and covariance interchangeably henceforth.

Nix and Weigend (1994) propose neural networks as the functional class for the mean and variance function(s); this provides remarkable flexibility to capture complex relationships in the data. To fit the mean and variance network(s) under their proposal, one minimizes the NLL, $\mathcal{L} \equiv$

$$\frac{1}{2}\sum_{(x,y)\in\mathcal{D}} \log|2\pi\Sigma(x)| + \big(y-\mu(x)\big)^T\Sigma(x)^{-1}\big(y-\mu(x)\big).$$

When $\mu(\cdot)$ and $\Sigma(\cdot)$ are from separate network classes

$$\mu(\cdot) \in \mathcal{F}_\mu : (\mathcal{X}, \Theta_\mu) \to \Re^{\dim(\mathcal{Y})} \tag{1}$$
$$\Sigma(\cdot) \in \mathcal{F}_\Sigma : (\mathcal{X}, \Theta_\Sigma) \to \mathbb{S}_{++}^{\dim(\mathcal{Y})\times\dim(\mathcal{Y})},$$

one can separately optimize network parameters θ_μ and θ_Σ by computing gradients $\nabla_{\theta_\mu}\mathcal{L}$ and $\nabla_{\theta_\Sigma}\mathcal{L}$ and using stochastic gradient descent or, like we do, an adaptive gradient algorithm such as Adam (Kingma and Ba, 2014). A more general treatment assumes a single network class

$$(\mu,\Sigma)(\cdot) \in \mathcal{G} : (\mathcal{X}, \Theta) \to (\Re^{\dim(\mathcal{Y})}, \mathbb{S}_{++}^{\dim(\mathcal{Y})\times\dim(\mathcal{Y})}). \tag{2}$$

Whenever $\Theta_\mu \cap \Theta_\Sigma = \emptyset$, $\mu(\cdot)$ and $\Sigma(\cdot)$ are separate networks. Conversely, $\Theta_\mu \cap \Theta_\Sigma \neq \emptyset$, implies $\mu(\cdot)$ and $\Sigma(\cdot)$ share ancestral computations in $\mathcal{G}$'s computational graph that involve shared parameter space $\Theta_\mu \cap \Theta_\Sigma$.

The Problem. Without loss of generality, consider a univariate response ($\dim(Y) = 1$) where $\sigma^2(\cdot)$ is a neural network parameterizing a scalar variance. Then, the gradients of the NLL w.r.t. the parameterizing function outputs are

$$\nabla_{\mu(x)}\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \frac{\mu(x)-y}{\sigma^2(x)} \tag{3}$$

$$\nabla_{\sigma^2(x)}\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \frac{\sigma^2(x)-(y-\mu(x))^2}{2(\sigma^2(x))^2}. \tag{4}$$

Skafte et al. (2019) recognize the $\frac{1}{\sigma^2(x)}$ scaling in both gradients quickens learning for low-variance points and thus biases performance towards regions of data with low noise. Seitzer et al. (2022) further recognize that learning is biased not just against data with higher true variance but also against points whose mean predictions are poor. Here, a model may use high variance (regardless of the true variance) to explain poor mean estimates instead of improving them; this creates a 'rich-get-richer' dynamic, where points with lower predictive variance continuously provide the largest learning signal. Please see Seitzer et al. (2022) for an elegant exposition of this phenomena.

Assessing Mean Estimation. To fairly assess mean estimation accuracy of a heteroscedastic model from $\mathcal{G}$, we propose constructing a mean-only baseline from $\mathcal{F}_\mu$ satisfying definition 1.

Definition 1. *For a heteroscedastic model from network class $\mathcal{G}$ with parameter space Θ (eq. (2)), let $\mathcal{F}_\mu$ with parameter space Θ_μ be the computational subgraph of $\mathcal{G}$ that removes all computational nodes that are not ancestors of the mean output. Clearly then, $\mathcal{F}_\mu \subset \mathcal{G}$ and $\Theta_\mu \subset \Theta$.*

By construction, $\mathcal{G}$ and $\mathcal{F}_\mu$ are equivalently powerful at mean estimation. The decision to use a mean-only model

from $\mathcal{F}_\mu$ as a baseline is well motivated for a couple of reasons. First, prevailing evidence suggests the additional fitting of variance is to blame for poor predictive performance when simultaneously fitting the mean and variance functions of a heteroscedastic model via gradient-based NLL minimization (Takahashi et al., 2018; Skafte et al., 2019; Seitzer et al., 2022). Second, mean-only models that minimize the sum of squared errors (SSE) are equivalent to homoscedastic models with isotropic unit covariance ($\Sigma = I$) and thus cannot get stuck in the cycle of using high local variance to explain poor mean estimates since they equally scale gradients by the inverted global covariance $\Sigma^{-1} = I$.

Mean square error (MSE) or root MSE (RMSE) are common assessments of mean estimation accuracy. We propose a desideratum for a heteroscedastic model is that its MSE is no worse than its mean-only baseline's MSE. If a heteroscedastic model meets this criteria, we say it is 'faithful.' Definition 2 formalizes this notion of faithfulness.

Definition 2. *Let $(\mu, \Sigma)(\cdot) \in \mathcal{G}$ (per eq. (2)) with initial parameters $\theta \in \Theta$ map covariates onto mean estimate $\mu(x;\theta) \approx \mathbb{E}[Y|X = x]$ and covariance estimate $\Sigma(x;\theta) \approx Cov[Y|X = x]$. Let $\mu_0(\cdot) \in \mathcal{F}_\mu$ (satisfying definition 1) with initial parameters $\theta_\mu \subset \theta$ map covariates onto mean estimate $\mu_0(x;\theta_\mu) \approx \mathbb{E}[Y|X = x]$. If $\mu_0(\cdot)$ and $(\mu, \Sigma)(\cdot)$'s parameters are separately optimized with algorithm $\mathcal{A}$ and $\mathbb{E}[|Y - \mu_0(x)|_2 - |Y - \mu(x)|_2] \geq 0$, then $\mathcal{G}$ is **faithful** to $\mathcal{F}_\mu$ under $\mathcal{A}$. Otherwise, $\mathcal{G}$ is **unfaithful** to $\mathcal{F}_\mu$ under $\mathcal{A}$. The expectation is w.r.t. the data distribution $p(X, Y)$ and all sources of randomness in $\mathcal{A}$.*

Using separate mean and variance networks, minimizing the SSE of the mean network first, freezing its parameters, and thereafter optimizing the NLL of the variance network achieves faithfulness under definition 2. This approach has several key deficiencies. It requires monitoring the mean network for convergence in order to transition to fitting the variance network. Sequential optimization is both inefficient and can lead to poor results if mean network optimization halts prematurely. Also, mandating separate networks for the mean and variance prohibits sharing of learned representations. For example, if data points are natural images and the mean network uses convolutional layers, it is likely that the first convolution layer will learn a set of edge detectors that might be redundant to those learned by a convolutional layer in a separate variance network. Relearning representations is computationally wasteful and only gets worse with deeper networks, which are often those with the best performance. Our proposal provably satisfies definition 2 while also addressing these deficiencies.

Our Solution. Consider the highly general network for parameterizing a Gaussian heteroscedastic model in fig. 1, which we partition into three subnetworks:

- Shared representation learner f_{trunk} has parameters θ_z.
- Mean parameter head f_μ has parameters θ_μ.
- Covariance parameter head f_Σ has parameters θ_Σ.

These three subnetworks comprise a heteroscedastic network from $\mathcal{G}$ (eq. (2), outer gray box fig. 1) such that $(\theta_z, \theta_\mu, \theta_\Sigma) \in \Theta$. Subnetworks f_{trunk} and f_μ comprise a mean-only network from $\mathcal{F}_\mu$ (eq. (1), inner gray ellipse fig. 1) such that $(\theta_z, \theta_\mu) \in \Theta_\mu$. Thus, $\mathcal{F}_\mu$ in fig. 1 is the mean-only baseline satisfying definition 1 to which we will compare a heteroscedastic model from $\mathcal{G}$. If f_{trunk} is the identity function, then the mean and covariance networks are separate. Alternatively, f_{trunk} could be a multi-layer network with vector output z. One possible $f_\Sigma(z; w_\Sigma, b_\Sigma) = \text{diag}(\text{softplus}(z^T w_\Sigma + b_\Sigma))$ adds a single output layer.

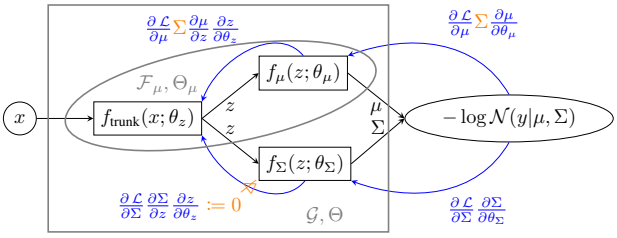


Figure 1: Heteroscedastic Network Partitions

Conventional optimization of fig. 1 forward passes covariates x (black arrows) to compute the NLL and then backward passes its gradient w.r.t. network parameters (blue arrows and equations) to render parameter updates (Nix and Weigend, 1994). We propose two changes (shown in orange) to gradient computation and its backward pass:

Proposal 1: Scale $\nabla_{\mu(x)} \mathcal{L}$ by Σ, its inverse Jacobian (i.e. a Newton step instead of a gradient step).

Proposal 2: Stop $\nabla_{\Sigma(x)} \mathcal{L}$ from contributing to updates for any shared parameters θ_z in f_{trunk}.

Together, these modifications ensure the mean subnetwork $\mu(x) = (f_\mu \circ f_{\text{trunk}})(x)$ will undergo optimization identically as if it were removed from $\mathcal{G}$ and trained on its own. Hence, our solution guarantees $\mathcal{G}$ will be faithful to $\mathcal{F}_\mu$ according to definition 2. Our first modification replaces the mean's gradient (eq. (3)) with its second-order Newton step

$$\sum_{(x,y)\in\mathcal{D}} \Sigma(x)\Sigma^{-1}(x)(\mu(x) - y) = \sum_{(x,y)\in\mathcal{D}} \mu(x) - y,$$

which is recognizable as the gradient of a homoscedastic model with isotropic unit covariance (i.e. from minimizing SSE). When $\dim(Y) > 1$, the $\Sigma(x)\Sigma^{-1}(x)$ cancellation saves a matrix inversion and matrix-vector product. Our second modification

$$\frac{\partial \mathcal{L}}{\partial \theta_z} = \left(\frac{\partial \mathcal{L}}{\partial \mu} \frac{\partial \mu}{\partial z} + \cancel{\frac{\partial \mathcal{L}}{\partial \Sigma} \frac{\partial \Sigma}{\partial z}}^{:=0} \right) \frac{\partial z}{\partial \theta_z}$$

eliminates the covariance's influence on any shared parameters θ_z. Thus, by shielding θ_z from $\frac{\partial \mathcal{L}}{\partial \Sigma}\frac{\partial \Sigma}{\partial z}\frac{\partial z}{\partial \theta_z}$ (the influence from $\Sigma(x)$) and ensuring equivalent parameter updates Δ_{θ_μ} and Δ_{θ_z} between a heteroscedastic model and its homoscedastic mean-only baseline, we have provably satisfied definition 2. By equally weighting all error terms, our proposal cannot enter the deleterious 'rich-get-richer' cycle where poor mean estimates are explained by high variance and subsequently ignored. Minimizing $\mathcal{L}_{\text{ours}} \equiv$

$$\sum_{(x,y)\in\mathcal{D}} \frac{\left|y-\mu(x)\right|_2^2}{2} - \log \mathcal{N}\left(y; \lfloor\mu(x)\rfloor, \Sigma(\lfloor f_{\text{trunk}}(x)\rfloor)\right) \quad (5)$$

is an equivalent implementation of our two proposals. We use $\lfloor\cdot\rfloor$ to denote the stop gradient operation, which converts its operand to a constant from the optimizer's perspective. As before, $\mu(x) = (f_\mu \circ f_{\text{trunk}})(x)$. Equation (5) is easy to implement in popular deep learning frameworks and is compatible with any gradient-based optimizer.

Theorem 1. *Under the same assumptions of definition 2, gradient-based optimization of $\mathcal{L}_{ours}$ (eq. (5)) guarantees $\mathcal{G}$'s faithfulness to $\mathcal{F}_\mu$.*

Proof. Differentiating $\mathcal{L}_{\text{ours}}$, one finds that

$$\frac{\partial \mathcal{L}_{\text{ours}}}{\partial \theta_\mu} = \sum_{(x,y)\in\mathcal{D}} \Big(f_\mu(z) - y\Big)\frac{\partial f_\mu(z)}{\partial \theta_\mu}$$
$$\frac{\partial \mathcal{L}_{\text{ours}}}{\partial \theta_z} = \sum_{(x,y)\in\mathcal{D}} \Big(f_\mu(z) - y\Big)\frac{\partial f_\mu(z)}{\partial z}\frac{\partial z}{\partial \theta_z},$$

which are equivalent to those arising from a SSE $\sum_{(x,y)\in\mathcal{D}} \frac{1}{2}\left|y-\mu(x)\right|_2^2$ loss. Thus, if both the heteroscedastic and homoscedastic optimizations experience the same randomness (e.g. by using a random number seed), then parameter updates Δ_{θ_μ} and Δ_{θ_z} will be identical at every optimization step. Because equivalence holds for every step for a single random number seed, it holds for all steps for any random number seed. □

Related Solutions. This article considers parametric methods that map a data point's covariates directly onto the response variable's parameter space. In contrast, Gaussian process (GP) regression models are non-parametric insofar that they require evaluating a kernel on all training (or inducing) points to compute the predictive mean and variance for a query point (Rasmussen and Williams, 2006). Most GPs including those that use neural networks to project covariates to a latent space before kernel evaluation (Wilson et al., 2016) assume homoscedastic isotropic noise $\sigma^2 I$ and cannot capture heteroscedastic noise variance. While a homoscedastic GP's predictive variance does change w.r.t. covariates, this change is in epistemic uncertainty only (e.g. x_*'s distance to training/inducing points). Technically, heteroscedasticity refers to noise variance only. Goldberg et al. (1997) introduce a GP with heteroscedastic noise assumptions. That said, GPs are a fundamentally different model class than the parametric methods we consider. We only compare performances within model classes to avoid confounding model selection and optimization methods.

Nix and Weigend (1994) first proposed using neural networks as Gaussian parameter maps. As such, we include their proposal in all of our experiments. Their technique, despite its optimization pitfalls (Seitzer et al., 2022), underlies Monte Carlo dropout (Kendall and Gal, 2017) and Deep Ensembles (Lakshminarayanan et al., 2017), which average outputs from Nix and Weigend (1994) over dropout samples and random initializations, respectively. Both of these methods produce a uniform mixture of Normals as the predictive distribution. In section 3, this underlying model experimentally underperforms, and we do not trust averaging to fix it. Instead, our supplement adapts our proposals to these two additional model classes and finds significant performance gains.

Several approaches adopt a Bayesian perspective to alleviate optimization problems, which results in a (Student's) t predictive distribution. Takahashi et al. (2018) parameterize a t-distribution using mean, precision, and degrees-of-freedom network(s). Integration over Gamma-distributed precisions seemingly improves optimization stability for points whose mean errors are small since integration almost surely includes non-zero variances. Skafte et al. (2019) use the same t-distribution parameterization with several other modifications. Stirn and Knowles (2020) place a prior over precision and perform variational inference. For well-chosen priors, the resulting Kullback–Leibler (KL) divergence in the variational objective can regularize variance away from very small (those that cause optimization instability (Takahashi et al., 2018)) and very large (those that produce poor estimates via the 'rich-get-richer' cycle (Seitzer et al., 2022)) variances. Our supplement adapts our proposals to the t-distribution model and again finds substantial performance gains.

Within the class of parametric methods with a Normal predictive, Seitzer et al. (2022) propose their β-NLL loss

$$\sum_{(x,y)\in\mathcal{D}} \lfloor\sigma^{2\beta}(x)\rfloor\left(\frac{1}{2}\log\sigma^2(x) + \frac{(y-\mu(x))^2}{2\sigma^2(x)}\right).$$

Again, $\lfloor\cdot\rfloor$ is the stop gradient operation. Setting $\beta = 0$, is equivalent to conventional NLL minimization. For $\beta = 1$, the gradients of β-NLL and SSE w.r.t. mean estimates are identical, but gradients w.r.t. variance estimates will still influence shared parameters. Changing β from 0 to 1 changes the curvature of the gradient w.r.t. variance function outputs (see supplement). Seitzer et al. (2022) recommend $\beta = 0.5$ for the best balance between RMSE and log likelihood performance, but never check variance calibration.

While $\beta = 1$ seemingly prevents poor mean estimates explained by high variance from being ignored, it is important to test what effect, if any, $\beta \neq 0$ has on variance calibration. We examine mean estimation accuracy and variance calibration for $\beta \in \{0.5, 1\}$ in our experiments.

3 EXPERIMENTS

Our experiments examine mean accuracies, variance calibrations, and log likelihoods from our method and our chosen baselines. **Unit Variance Homoscedastic** is the mean-only baseline (satisfying definition 1) and uses a nominal homoscedastic isotropic unit covariance for variance calibration and log likelihood measurements. **Conventional Heteroscedastic** denotes standard NLL minimization of a heteroscedastic model (Nix and Weigend, 1994). We label results for **Beta NLL** (Seitzer et al., 2022) with the β setting in parentheses. **Proposal 1** and **Proposal 2** apply our two proposals from section 2 in isolation to confirm both are necessary. Our method, **Faithful Heteroscedastic**, utilizes proposals 1 and 2 for guaranteed faithfulness.

3.1 Convergence of the Predictive Mean and Variance

Figure 2 plots the convergence behaviour for all methods with a Normal predictive distribution. Our supplement has plots for Deep Ensemble, MC Dropout, and t-distribution based methods with similar findings. We generate data similarly to Skafte et al. (2019); Stirn and Knowles (2020); Seitzer et al. (2022). We sample $x \sim \text{Uniform}(2.5, 7.5)$ and then set $y \triangleq x \cdot \sin(x) + \epsilon$ with $\epsilon | x \sim \mathcal{N}(0, 0.1 + |0.5x|)$. We add $(X = 0.5, Y = 0.5 \cdot \sin(0.5))$ and $(X = 9.5, Y = 9.5 \cdot \sin(9.5))$ as isolated points for a total of 500 samples (shown in blue, isolated points appear larger for visual convenience). We highlight regions of covariate space containing data with gray backgrounds. We plot the predictive mean and variance every 2000 training epochs in the top and bottom rows of fig. 2, respectively. The dashed lines show the true mean and variance.

This experiment employs a neural network with a single hidden layer for f_{trunk}. The mean and variance heads are each a single layer. The mean-only model, Unit Variance Homoscedastic, converges on both isolated points while estimating the true mean over the data rich interval $[2.5, 7.5]$. This behavior not only mimics a GP (to which this network architecture has theoretical connections (Rasmussen and Williams, 2006)) but also confirms the mean subnetwork of the heteroscedastic model has the flexibility to do the same. However, the Conventional Heteroscedastic model, after quickly fitting its mean to the majority of points, has a drastic error at $X = 9.5$. As Seitzer et al. (2022) suspect, the model quickly increases variance to explain this large error, which effectively eliminates this point's learning signal and ultimately prohibits convergence thereto. Our Proposal 2 does the same. This problem does not fully disappear for Beta NLL (0.5). The Beta NLL (1.0), Proposal 1, and Faithful Heteroscedastic models all produce accurate mean estimates with convergence at both isolated points and accurate variance estimates.

3.2 UCI Regression

The following experiments utilize publicly available regression datasets from the UCI repository[1]. We scale response variables to have zero mean and unit variance. We now use a neural network with two hidden layers as the shared representation learner f_{trunk}. The mean and variance heads are single layers operating on f_{trunk}'s output. To best approximate the expectation in definition 2, we first divide each UCI dataset into ten folds. We then aggregate the predictions of each held-out fold from the ten models resulting from cross-validation such that we have a held-out prediction for every data point (Höfling and Tibshirani, 2008). Every model uses the same fold assignments and, for every fold, shares the same initial parameters. Our supplement contains additional details.

We report mean estimation accuracy with RMSE. For variance, we use expected calibration error (ECE) (Kuleshov et al., 2018). For bin edges $0 = p_0 < p_1 < \ldots < p_{m-1} \leq p_m = 1$, ECE computes bin probabilities

$$\hat{p}_j = \frac{|\{(x, y) \in \mathcal{D} : p_{j-1} < F(y|x) \leq p_j\}|}{|\,\mathcal{D}\,|},$$

where $F(y|x)$ is the Normal CDF. For a well-calibrated model, $\hat{p}_j \approx p_j - p_{j-1}$. Accordingly, ECE is defined as $\sum_{j=1}^{m} (\hat{p}_j - (p_j - p_{j-1}))^2$. We report RMSE, ECE, and log likelihood (LL) in table 1 for all methods with Normal predictive distributions. Our supplement has tables for Deep Ensemble, Monte Carlo Dropout, and t-distribution based methods with similar findings. Any model whose predictive squared errors are greater than those of the mean-only baseline (Unit Variance Homoscedastic) according to a one-sided paired t-test with a $p < 0.05$ threshold is considered empirically unfaithful and has its RMSE, ECE, and LL struck out. We use a paired t-test since the squared errors from two models are not independent–each pair results from the same held out data point. For each dataset, we bold the smallest RMSE and, using the same significance test, identify any statistical ties, which too are bold. We also bold the smallest ECE, but ignore struck-out values from any unfaithful models. We use a G-test to identify statistically significant differences between the histograms generated from a model's $\hat{p}_j$ values. Any faithful model, whose p-value ≥ 0.05 when compared to the faithful model with the best ECE is considered tied for best. Among faithful models, we bold the largest LL and any statistical ties according to a one-sided Kolmogorov-Smirnov test with a 0.05 threshold. Tallying the number of wins/ties for each

[1] Our code downloads and processes these datasets.

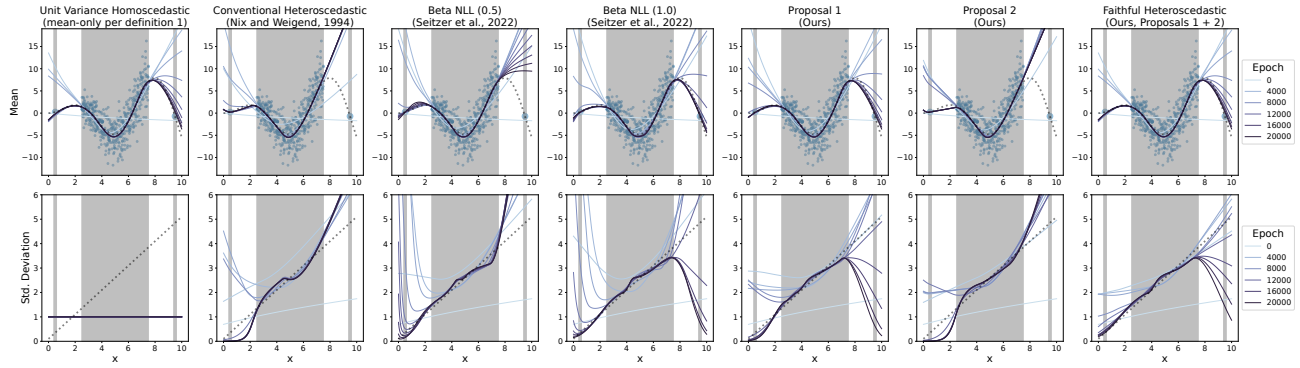


Figure 2: Convergence Behavior of Methods with Normal Predictive Distributions

Table 1: UCI Regression Performance of Methods with Normal Predictive Distributions

	Unit Variance Homoscedastic (mean-only per definition 1)			Conventional Heteroscedastic (Nix and Weigend, 1994)			Beta NLL (0.5) (Seitzer et al., 2022)			Beta NLL (1.0) (Seitzer et al., 2022)			Proposal 1 (Ours)			Proposal 2 (Ours)			Faithful Heteroscedastic (Ours, Proposals 1 + 2)		
Dataset	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL
boston	0.304	0.168	-0.965	~~0.374~~	~~0.00977~~	~~-2.04~~	**0.341**	**0.0224**	**-12**	**0.335**	**0.0269**	**-4.5**	~~0.355~~	~~0.00973~~	~~-0.886~~	~~0.355~~	~~0.0121~~	~~-2.27~~	**0.304**	0.0303	**-17.4**
carbon	0.0489	0.401	-2.76	~~0.0493~~	~~0.00247~~	~~3.89~~	**0.0488**	0.00152	**-30.8**	~~0.05~~	~~0.00727~~	~~-9.76e+04~~	~~0.0818~~	~~5.48e-05~~	~~4.78~~	~~0.0491~~	~~0.00225~~	~~5.38~~	**0.0489**	**0.00124**	**-9.49**
concrete	0.265	0.127	-0.954	~~0.315~~	~~0.00601~~	~~-2.24~~	**0.269**	0.0348	-59.8	**0.263**	0.0417	-22.2	~~0.293~~	~~0.0103~~	~~-2~~	~~0.294~~	~~0.00941~~	~~-1.2~~	**0.265**	**0.0282**	**-2.81**
energy	0.16	0.314	-1.85	~~0.193~~	~~0.00236~~	~~3.39~~	~~0.182~~	~~0.00139~~	~~3.9~~	**0.168**	0.00242	**3.29**	~~0.195~~	~~0.00161~~	~~2.79~~	~~0.193~~	~~0.00966~~	~~2.6~~	**0.16**	**0.00127**	**3.35**
naval	0.0247	0.401	-1.84	~~0.533~~	~~0.00432~~	~~3.23~~	~~0.159~~	~~0.0133~~	~~1.85~~	~~0.0262~~	~~0.00116~~	~~6.6~~	~~0.207~~	~~0.000175~~	~~2.64~~	~~0.581~~	~~0.00212~~	~~2.8~~	**0.0247**	**0.00197**	**6.63**
power plant	0.22	0.0105	-0.943	~~0.227~~	~~0.000175~~	~~0.0106~~	~~0.224~~	~~0.000199~~	~~0.0456~~	~~0.223~~	~~0.00034~~	~~-53.2~~	~~0.235~~	~~0.000155~~	~~0.0385~~	~~0.223~~	~~0.000127~~	~~0.0722~~	**0.22**	**0.000183**	**0.0937**
protein	0.00257	0.489	-0.919	~~0.029~~	~~0.00291~~	~~3.73~~	~~0.00397~~	~~0.00599~~	~~4.57~~	~~0.00308~~	~~0.00342~~	~~4.62~~	~~0.0372~~	~~8.67e-05~~	~~2.1~~	~~0.0675~~	~~0.00957~~	~~2.67~~	**0.00257**	**0.00591**	**4.68**
superconductivity	0.326	0.0104	-0.972	~~0.388~~	~~0.00109~~	~~-10~~	**0.327**	0.002	-23.7	**0.327**	0.00184	-7.57	~~0.37~~	~~0.000297~~	~~-0.181~~	~~0.358~~	~~0.00174~~	~~-1.51~~	**0.326**	**0.00103**	**-0.291**
wine-red	0.769	0.00601	-1.21	**0.769**	0.00203	-3.96	**0.769**	0.00181	-3.77	**0.767**	0.0019	**-4.69**	0.773	0.00186	-1.7	**0.769**	**0.00159**	**-1.19**	**0.769**	0.00242	**-1.22**
wine-white	0.777	0.0015	-1.22	~~0.786~~	~~0.000868~~	~~-22.6~~	**0.784**	0.001	-1.92	~~0.784~~	~~0.000767~~	~~-7.37~~	~~0.787~~	~~0.000935~~	~~-2.51~~	**0.779**	0.000613	**-1.27**	**0.777**	**0.000514**	**-1.2**
yacht	0.0226	0.882	-0.919	~~0.657~~	~~0.0331~~	~~-0.769~~	**0.0226**	**0.00442**	**2.29**	**0.0226**	0.00955	1.86	~~0.135~~	~~0.0103~~	~~0.738~~	~~0.454~~	~~0.0876~~	~~-0.0822~~	**0.0226**	0.0174	**1.33**
Total wins or ties	–	–	–	*1*	*0*	*0*	*7*	*2*	*3*	*6*	*1*	*3*	*0*	*0*	*0*	*2*	*1*	*2*	*11*	*8*	*11*

metric, we find our Faithful Heteroscedastic model is never empirically unfaithful and offers the best combination of mean estimation accuracy, variance calibration, and data log likelihood. Proposals 1 and 2 underperform in isolation, confirming both are necessary to achieve faithfulness.

3.3 Decomposing Variance Estimates

Kendall and Gal (2017); Lakshminarayanan et al. (2017) want predictive variance to include epistemic and aleatoric uncertainty. Conversely, we seek to accurately decompose predictive variance into its epistemic and aleatoric components. Consider a variational autoencoder (VAE) (Kingma and Welling, 2013) that stochastically encodes an image into a low-dimensional space and employs a Gaussian decoder. A heteroscedastic Gaussian decoder will use per-pixel predictive variance to explain reconstruction errors. Collecting more data is unlikely to reduce these variances. Yet, we could replace the VAE with a deterministic autoencoder to trivially achieve infinite likelihood: use the identity function to autoencode the mean and set variance to zero. Thus, we argue a VAE's variance estimates for a set of unique images are entirely epistemic (i.e. only reducible with a model change) and capture compression loss.

If we define f_{trunk} as a stochastic, convolutional encoder $q(z;x)$, then fig. 1 becomes a VAE. A VAE minimizes

$$\mathop{\mathbb{E}}_{q(z;x)}[-\log \mathcal{N}(x; f_\mu(z), f_\Sigma(z))] + D_{KL}\big(q(z;x)||p(z)\big).$$

The KL divergence is simply a regularization loss for f_{trunk}.

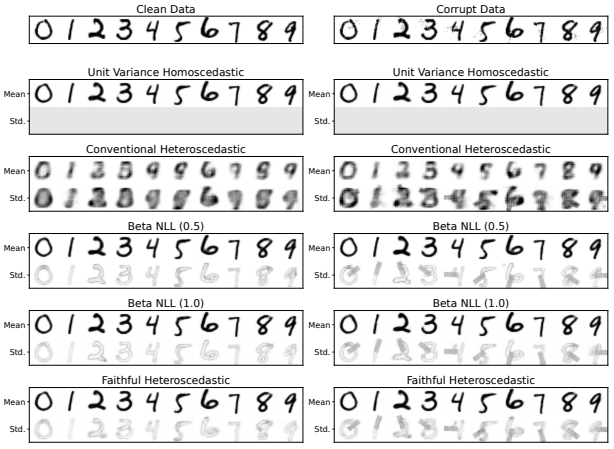


Figure 3: VAE Predictive Moments for MNIST

Using a single sample from $q(z;x)$, we use our chosen baselines' proposed objective for the NLL. We note, however, our loss is equivalent to optimization changes such that we are not changing the VAE model but rather its optimization. Parameter heads f_μ and f_Σ are each transposes of the encoder. Our supplement contains additional details.

The left column of fig. 3 shows an example set of images and each model's corresponding mean and standard deviation estimates. The Conventional Heteroscedastic model uses high variance to explain poor mean fits. All other models produce sensible looking means. Their estimated vari-

Table 2: VAE Performance of Methods with Normal Predictive Distributions

		Unit Variance Homoscedastic (mean-only per definition 1)			Conventional Heteroscedastic (Nix and Weigend, 1994)			Beta NLL (0.5) (Seitzer et al., 2022)			Beta NLL (1.0) (Seitzer et al., 2022)			Proposal 1 (Ours)			Proposal 2 (Ours)			Faithful Heteroscedastic (Ours, Proposals 1 + 2)		
Dataset		RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL
fashion-mnist	clean	0.923	0.00232	-1.05e+03	~~1.61~~	~~0.00013~~	~~-711~~	~~0.962~~	~~0.000161~~	~~78.6~~	~~0.945~~	~~0.000374~~	~~-452~~	~~0.935~~	~~2.26e-05~~	~~-144~~	~~1.52~~	~~0.000322~~	~~-477~~	**0.923**	**7.12e-05**	**-217**
	corrupt	1.17	0.00251	-1.26e+03	~~1.77~~	~~0.000178~~	~~-482~~	~~1.2~~	~~0.000296~~	~~-81.9~~	~~1.19~~	~~0.000576~~	~~-976~~	~~1.19~~	~~2.4e-05~~	~~-320~~	~~1.64~~	~~0.00042~~	~~-370~~	**1.17**	**6.33e-05**	**-392**
mnist	clean	0.759	0.00923	-946	~~2.27~~	~~0.00072~~	~~117~~	~~0.85~~	~~0.000182~~	~~-3.56e+03~~	~~0.78~~	~~0.00106~~	~~-741~~	~~0.792~~	~~1.89e-05~~	~~546~~	~~2.14~~	~~0.00112~~	~~-565~~	**0.759**	**3.16e-05**	**119**
	corrupt	1.04	0.00839	-1.15e+03	~~2.33~~	~~0.000555~~	~~-361~~	~~1.07~~	~~0.000243~~	~~-1.53e+04~~	~~1.07~~	~~0.00668~~	~~-1.43e+05~~	~~1.06~~	~~2.51e-05~~	~~220~~	~~1.8~~	~~0.00236~~	~~-413~~	**1.04**	**5.78e-05**	**-358**
Total wins or ties		–	–	–	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*4*	*4*	*4*

ances suggest compression loss most significantly affects edge localization, which we argue is epistemic.

We now add simulated heteroscedastic noise to each image, where the standard deviation is a rectangular patch whose magnitude is 25% the dynamic range of the images. The patch is rotated according to an image's class label (fig. 4, top) to simulate heteroscedastic aleatoric uncertainty. The right column of fig. 3 shows an example set of corrupt images and each model's moment estimates when trained on the corrupt data. Except for the Conventional Heteroscedastic model, the variance estimates contain the same epistemic uncertainty from the clean data as well as the added aleatoric noise variance. To isolate heteroscedastic aleatoric uncertainty for an image, we subtract the variance output from a model trained on clean data from that of a model trained on corrupt data. The well-known Bias-Variance equality supports this decomposition:

$$\underbrace{\mathrm{Var}[y|x]}_{\text{noise variance}} = \underbrace{\mathbb{E}[(y-\mu(x))^2]}_{\text{aleatoric and epistemic}} - \underbrace{(\mathbb{E}[y|x]-\mu(x))^2}_{\text{only epistemic mean uncertainty}}$$
$$\approx \underbrace{\Sigma_{\text{noisy}}(x)}_{\text{model trained w/ noisy } y|x} - \underbrace{\Sigma_{\text{clean}}(x)}_{\text{model trained w/ } \mathbb{E}[y|x]} . \tag{6}$$

We plot the average difference in predictive variance $(\Sigma_{\text{noisy}}(x) - \Sigma_{\text{clean}}(x))$ over all images within a class in fig. 4. The ability to recover the true noise variance suggests that, for datasets where we have clean and noisy measurements, we can use well-calibrated heteroscedastic models to decompose the predicted variance into its aleatoric and epistemic components. Our supplement includes full size images for all datasets.

Table 2 reports performance and identifies wins/ties as described in section 3.2. Again, our Faithful Heteroscedastic model offers the best combined performance. All other models were empirically unfaithful.

3.4 Noise Variance in CRISPR-Cas13 efficacy

Corrupting a clean set of images with heteroscedastic noise in section 3.3 may seem contrived, but a similar situation frequently arises in biology. Biologists will often replicate experiments to account for technical variation. Averaging replicates conditioned on covariates $y_1|x, \ldots y_r|x$ creates a de-noised dataset with approximate $\mathbb{E}[y|x]$ values.

CRISPR-Cas13 is a system for knocking down gene expression. Unlike Cas9, which modifies DNA, Cas13 targets and degrades specific RNA transcripts. Cas13 systems use a guide RNA (gRNA) to recruit the Cas13 protein to destroy complementary RNA transcripts in its host. However, Cas13 has known sequence preferences making some gRNAs more effective than others. We consider three Cas13 datasets: one measures guide efficacy via flow cytometry in HEK293 cells (Wessels et al., 2020), and the other two use survival screens to measure efficacy in HEK293 and A375 cells, respectively.[2] For each dataset, we have multiple efficacy scores for each target sequence as a result of the replicated experiments. We predict a scalar value that measures knockdown efficacy (a more negative value indicates higher efficacy) from one-hot encoded target sequence. Shared representation learner f_{trunk} uses convolutional layers for motif discovery. The parameter heads are fully-connected networks. See supplement for details.

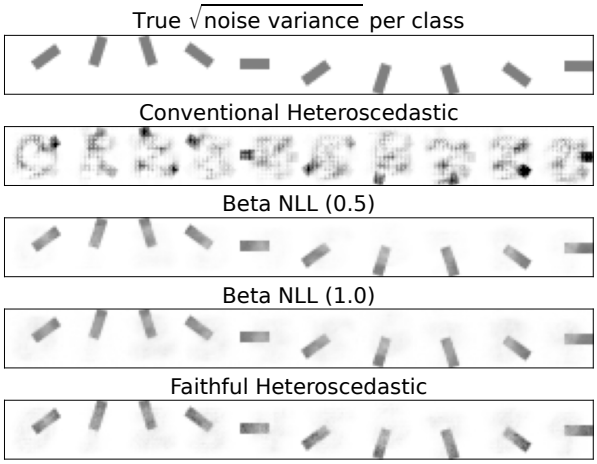


Figure 4: VAE Recovered Noise Variance for MNIST

SHAP values quantify each feature's impact to a model's output w.r.t. the average output taken over a set of background samples (Lundberg and Lee, 2017). Using the same cross-validation strategy from section 3.2, we collect the predictive moments and their SHAP values for every data point; we do this for all models twice: once when trained on raw replicates and again when trained on their average. Table 3 reports performance, identifying wins/ties as described in section 3.2. While our Faithful Heteroscedastic model does not always have the best RMSE, it offers the best ECE and LL performance.

[2]These datasets can be found in our public code repository.

Table 3: Modeling CRISPR Cas13 Efficacy

Dataset	Observations	Unit Variance Homoscedastic (mean-only per definition 1) RMSE	ECE	LL	Conventional Heteroscedastic (Nix and Weigend, 1994) RMSE	ECE	LL	Beta NLL (0.5) (Seitzer et al., 2022) RMSE	ECE	LL	Beta NLL (1.0) (Seitzer et al., 2022) RMSE	ECE	LL	Proposal 1 (Ours) RMSE	ECE	LL	Proposal 2 (Ours) RMSE	ECE	LL	Faithful Heteroscedastic (Ours, Proposals 1 + 2) RMSE	ECE	LL
flow cytometry (HEK293)	means	0.279	0.0209	-0.958	~~0.285~~	~~0.0925~~	~~-11.4~~	**0.272**	**0.0214**	-1.27	**0.274**	0.0258	-1.25	~~0.297~~	~~0.0088~~	~~-0.915~~	~~0.289~~	~~0.0599~~	~~-5.21~~	0.279	0.0219	**-1.04**
	replicates	0.294	0.00673	-0.962	0.295	0.0117	-1.2	**0.286**	0.00697	-0.658	0.294	0.00695	-0.65	~~0.304~~	~~0.00925~~	~~-1.12~~	~~0.301~~	~~0.00635~~	~~-0.697~~	0.294	**0.00459**	**-0.511**
survival screen (A375)	means	0.31	0.00596	-0.967	~~0.313~~	~~0.000577~~	~~-0.251~~	**0.309**	0.00131	-0.337	**0.309**	0.00066	-0.262	~~0.314~~	~~0.000948~~	~~-0.328~~	~~0.312~~	~~0.000578~~	~~-0.246~~	**0.31**	**0.000599**	**-0.248**
	replicates	0.36	0.00158	-0.984	~~0.361~~	~~0.000324~~	~~-0.422~~	**0.36**	0.000311	-0.422	**0.36**	0.000243	-0.406	~~0.363~~	~~0.00033~~	~~-0.437~~	~~0.361~~	~~0.000183~~	~~-0.391~~	**0.36**	**0.000206**	**-0.394**
survival screen (HEK293)	means	0.304	0.0178	-0.965	~~0.309~~	~~0.0111~~	~~-1.92~~	**0.298**	0.00362	-0.261	0.301	0.00271	-0.211	~~0.308~~	~~0.0033~~	~~-0.282~~	~~0.318~~	~~0.00433~~	~~-0.439~~	0.304	**0.00235**	**-0.194**
	replicates	0.346	0.00281	-0.979	~~0.349~~	~~0.00128~~	~~-0.45~~	**0.345**	0.000663	-0.337	~~0.348~~	~~0.000551~~	~~-0.332~~	~~0.352~~	~~0.000859~~	~~-0.403~~	~~0.351~~	~~0.000869~~	~~-0.381~~	**0.346**	**0.000453**	**-0.316**
Total wins or ties		–	–	–	*0*	*0*	*0*	*6*	*1*	*0*	*3*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*3*	*5*	*6*

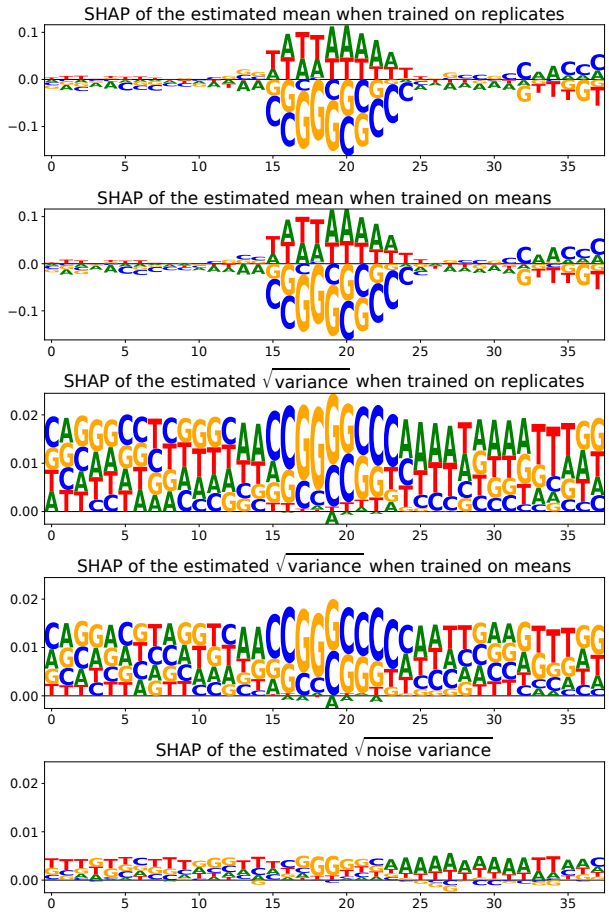


Figure 5: SHAP Analysis of Flow Cytometry (HEK293) Data Using our Proposed Faithful Heteroscedastic Model

We reduce each feature's SHAP values over all held-out data via averaging. The first two rows of fig. 5 plot the reduced SHAP values for our model's mean estimates when trained on raw and averaged replicates, respectively. As expected, our model's mean estimates are unaffected by reduced noise variance and recover known Cas13 sequence preferences in HEK293 cells as measured by flow cytometry (Wessels et al., 2020). The next two rows of fig. 5, plot the reduced SHAP values for our model's standard deviation estimates when trained on raw and averaged replicates, respectively. The standard deviation's magnitude appropriately decreases for the model trained on the averaged replicates. The last row of fig. 5, subtracts the fourth row from the third (as in eq. (6)) and shows that heteroscedasticity in CRISPR-Cas13 has a similar sequence dependence as the mean, suggesting more effective guides have higher variance–a common heteroscedastic outcome. To the best of our knowledge, we are the first to model sequence-dependent heteroscedasticity in CRISPR-Cas13. Our supplement has SHAP plots for the other datasets and models.

4 DISCUSSION

If the parameterizing neural network for a heteroscedastic Gaussian model has all components removed that are not necessary for generating mean estimates, the resulting subnetwork will have the same expressive power for mean estimation as the full model. Therefore, we posit any heteroscedastic model's mean estimation accuracy should be as good as this mean-only baseline; we say a model and its optimization are 'faithful' upon achieving this. Yet, conventional minimization of a heteroscedastic model's NLL can worsen its mean estimation accuracy (Seitzer et al., 2022). Our optimization proposals provably ensure 'faithfulness' and are equivalent to a modified loss function, making our method exceptionally easy to implement in popular deep learning frameworks. One limitation of our approach, a byproduct of theorem 1, is that our method can never have more accurate mean estimates than its mean-only baseline. There are a few examples of the Beta NLL model (Seitzer et al., 2022) outperforming its mean-only baseline's mean estimation accuracy.

Our experiments include datasets ranging from simple scalar regression to auto-encoding natural images and predicting CRISPR-Cas13 efficacy from RNA sequence. The tested network architectures are equally diverse. Regardless of task or network complexity, our method overwhelmingly offers the best combination of mean estimation accuracy, variance calibration, and model log likelihood. With our reliable mean and variance estimates, we demonstrate how to isolate aleatoric uncertainty in the predictive variance from epistemic uncertainty. Our technique accurately recovers heteroscedasticity in simulated experiments. Additionally, our model recovers known CRISPR-Cas13 sequence preferences and identifies low-levels of sequence-dependent (heteroscedastic) noise variance. When model predictions inform high-impact decision-making (e.g. those that affect health or society), accurate mean and uncertainty estimates are critical. Thus,

our method represents a step towards improving model trustworthiness and reliability.

Acknowledgements

A.S. devised the ideas in this article and implemented all computational experiments. H.W. collected and prepared both HEK293 datasets. M.S. & L.P. collected and prepared the A735 dataset. N.S. supervises H.W. D.A.K supervises A.S. & M.S.

References

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. Advances in neural information processing systems, 31.

Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? Structural safety, 31(2):105–112.

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. PMLR.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In International Conference on Machine Learning, pages 1183–1192. PMLR.

Goldberg, P., Williams, C., and Bishop, C. (1997). Regression with input-dependent noise: A gaussian process treatment. Advances in neural information processing systems, 10.

Höfling, H. and Tibshirani, R. (2008). A study of prevalidation. The Annals of Applied Statistics, 2(2):643–664.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? Advances in neural information processing systems, 30.

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.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In International conference on machine learning, pages 2796–2804. PMLR.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems, 30.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

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.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. Advances in neural information processing systems, 29.

Rasmussen, C. E. and Williams, C. K. I. (2006). Gaussian processes for machine learning. Adaptive computation and machine learning. MIT Press, Cambridge, Mass. OCLC: ocm61285753.

Seitzer, M., Tavakoli, A., Antic, D., and Martius, G. (2022). On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. In International Conference on Learning Representations.

Skafte, N., Jørgensen, M., and Hauberg, S. (2019). Reliable training and estimation of variance networks. Advances in Neural Information Processing Systems, 32.

Stirn, A. and Knowles, D. A. (2020). Variational variance: Simple and reliable predictive variance parameterization. arXiv preprint arXiv:2006.04910.

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.

Wessels, H.-H., Méndez-Mancilla, A., Guo, X., Legut, M., Daniloski, Z., and Sanjana, N. E. (2020). Massively parallel Cas13 screens reveal principles for guide RNA design. Nature Biotechnology, 38(6):722–727.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In Artificial intelligence and statistics, pages 370–378. PMLR.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. Advances in Neural Information Processing Systems, 33:14129–14142.

A FURTHER DISCUSSION OF RELATED WORK

Skafte et al. (2019) propose four modifications to improve variance estimates. While their combination yields a Student's t predictive distribution, one of their proposals on its own admits a Normal model. Utilizing separate mean and variance networks, they use the first half of training to fit just the mean via minimizing sum of squared errors (SSE). Thereafter, they alternate between minimizing the negative log likelihood (NLL) w.r.t. mean network parameters and minimizing the NLL w.r.t. variance network parameters. The exact number of batches between alternations is a configurable hyperparameter. They argue that fitting the variance should wait until mean estimates are reasonable. This proposal is similar to the faithful approach discussed in section 2 of our main manuscript–use separate mean and variance networks, minimize the SSE of the mean network, freeze its parameters, and then minimize the NLL of the variance network–and suffers the same deficiencies. Also, without freezing mean network parameters before fitting the separate variance network, their approach is not provably faithful. That said, if the mean network is sufficiently converged, then Skafte et al. (2019) can avoid the deleterious 'rich-get-richer' cycle where poor mean estimates are explained by high variance and subsequently ignored (Seitzer et al., 2022). However, their implementation has the added complication of running two adaptive gradient optimizers–one for the mean network and one for the variance network. The gradient w.r.t. one network's parameters could be drastically different after an interleaved adjustment to the other network's parameters. It is unclear what effect, if any, this might have on Adam's (Kingma and Ba, 2014) performance. Stirn and Knowles (2020) find that Skafte et al. (2019)'s combination of proposals can severely overestimate predictive variance in high-dimensional settings.

As discussed in our main manuscript, Seitzer et al. (2022) propose their β-NLL loss

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \lfloor \sigma^{2\beta}(x) \rfloor \left(\frac{1}{2} \log \sigma^2(x) + \frac{(y-\mu(x))^2}{2\sigma^2(x)} \right).$$

When $\beta = 0$, their objective is the same as the NLL. For $0 < \beta \leq 1$, the gradients become

$$\nabla_{\mu(x)} \mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \frac{\mu(x) - y}{\sigma^{(2-2\beta)}(x)}, \quad \nabla_{\sigma^2(x)} \mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \frac{\sigma^2(x) - (y-\mu(x))^2}{2\sigma^{(4-2\beta)}(x)}.$$

The gradient w.r.t. the mean estimate is linear in $\mu(x)$ for all $\beta \in [0, 1]$. The gradient w.r.t. the variance estimate is a difference of powers of $\sigma^2(x)$. Changing these powers naturally affects the curvature of the gradient. Supplement fig. 6 plots $\nabla_{\sigma^2(x)} \mathcal{L}$ when $(y - \mu(x))^2 = 1$ for different values of β.

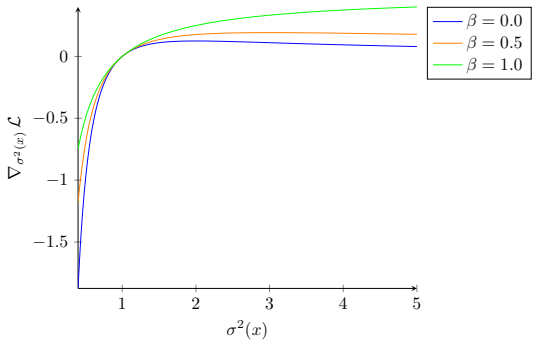


Figure 6: Gradient of β-NLL w.r.t. Variance for Unit Mean Error at Different β Settings

While the optima at $\sigma^2(x) = 1$ does not move, the magnitude of the gradient about the optima does. To the left of the optima, the magnitude of the gradient decreases as β increases. The opposite effect occurs to the right of the optima–magnitude of the gradient increases as β increases. Also, at $\beta = 1$, $\nabla_{\sigma^2(x)} \mathcal{L}$ becomes fully concave. Given these changes, it is important to test their predictive variance calibration.

Our first proposal scales the gradient w.r.t. the mean estimate by its inverse Jacobian and is identical to using $\beta = 1$ for $\nabla_{\mu(x)} \mathcal{L}$. Our proposals do not alter the NLL's gradient w.r.t. the variance estimate and thus implicitly set $\beta = 0$ for $\nabla_{\sigma^2(x)} \mathcal{L}$. Experimentally, we consistently offer better variance calibration than Seitzer et al. (2022).

B ADAPTING TO OTHER MODEL CLASSES

As stated in the 'Related Solutions' portion of our main manuscript (section 2), we only compare performances within model classes to avoid confounding model selection and optimization methods. We generally define a model class by the resulting predictive distribution. Our main article only considers Normal predictive distributions. The following subsections adapt our proposals to different types of predictive distributions and compare to relevant methods. Upcoming supplement appendix C contains the architectural details for the models used in section 3 of our main manuscript. Unless otherwise noted the following additional model classes use the same architectures.

B.1 Adaptation to Deep Ensemble Methods

Lakshminarayanan et al. (2017) use an ensemble of heteroscedastic Gaussian models with neural network parameter maps to improve predictive uncertainty estimation. Their predictive distribution is a uniform mixture of M models

$$p(Y|X=x) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{N}(Y|\mu(x,\theta_m),\Sigma(x,\theta_m)),$$

where the component models are separately initialized and trained via conventional NLL minimization. To construct a 'faithful' ensemble, we simply train the M components with our method to guarantee each component is 'faithful.' The Unit Variance Homoscedastic is a uniform mixture of M mean-only models:

$$p(Y|X=x) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{N}(Y|\mu(x,\theta_m), I_{\dim(Y)}).$$

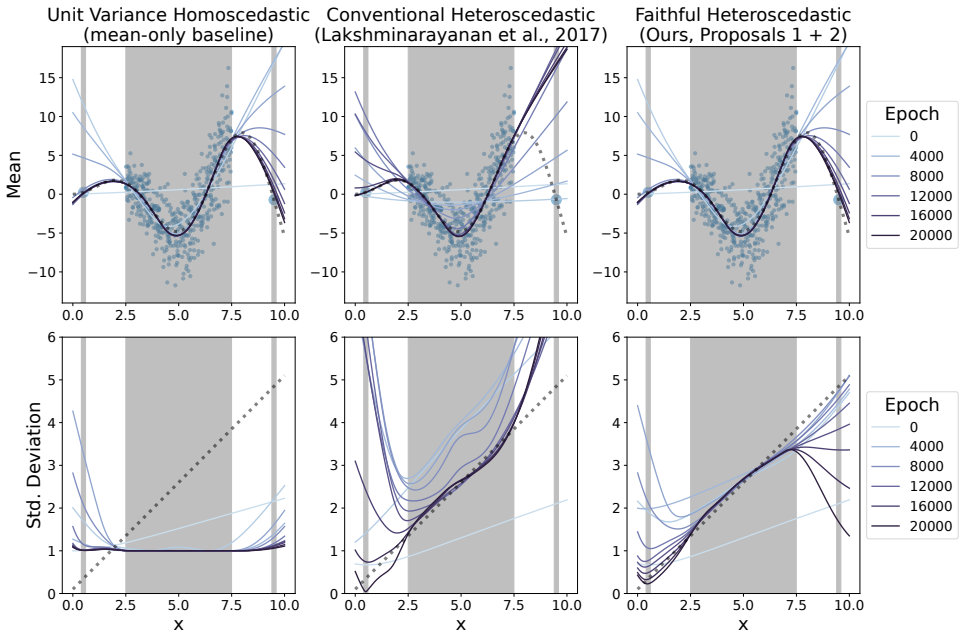


Figure 7: Convergence Behavior of Deep Ensemble Methods

Figure 7 examines the convergence behavior of Deep Ensemble methods. Given the convergence flaws of the underlying component model (our main report; Seitzer et al. (2022)), it is unsurprising that an ensemble of these flawed models does not perform well. Indeed, the Conventional Heteroscedastic Deep Ensemble (Lakshminarayanan et al., 2017) uses high variance to explain mean errors at $X = 9.5$ despite having the flexibility to converge thereto as demonstrated by the Unit Variance Homoscedastic Deep Ensemble. While the Unit Variance Homoscedastic Deep Ensemble uses a fixed variance for each of its components, the predictive variance of the ensemble changes w.r.t. x. The increase in variance outside the data-rich interval $[2.5, 7.5]$ arises from epistemic uncertainty in the M mean estimates. In particular, if any of the $\mu(x, \theta_m)$

components have slightly different outputs, then the variance of the predictive mixture increases. Table 4 reports UCI regression performances for the Deep Ensemble methods. Our faithful variant overwhelmingly outperforms the ensemble of conventionally optimized models.

Table 4: UCI Regression Performance of Deep Ensemble Methods

	Unit Variance Homoscedastic (mean-only baseline)			Conventional Heteroscedastic (Lakshminarayanan et al., 2017)			Faithful Heteroscedastic (ours)		
Dataset	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL
boston	0.32	0.172	-0.975	~~0.374~~	~~0.00793~~	~~-0.446~~	**0.32**	**0.0053**	**-5.86**
carbon	0.0487	0.48	-2.76	**0.0488**	0.0169	**9.35**	**0.0487**	**0.0111**	4.99
concrete	0.25	0.154	-0.955	~~0.292~~	~~0.00165~~	~~-0.725~~	**0.25**	**0.00374**	**-0.267**
energy	0.159	0.327	-1.85	~~0.192~~	~~0.0331~~	~~4.38~~	**0.159**	**0.0238**	**4.24**
naval	0.0215	0.857	-1.84	~~0.313~~	~~0.215~~	~~5.32~~	**0.0215**	**0.188**	**6.68**
power plant	0.215	0.0275	-0.943	~~0.223~~	~~0.000265~~	~~0.158~~	**0.215**	**0.000225**	**0.155**
protein	0.00193	0.4	-0.919	~~0.0484~~	~~0.228~~	~~3.92~~	**0.00193**	**0.28**	**4.61**
superconductivity	0.293	0.0162	-0.97	~~0.375~~	~~0.000344~~	~~0.344~~	**0.293**	**0.00104**	**0.185**
wine-red	0.762	0.00564	-1.21	**0.765**	0.00197	**-1.11**	**0.762**	**0.00181**	-1.14
wine-white	0.748	0.00216	-1.21	~~0.756~~	~~0.00044~~	~~-1.3~~	**0.748**	**0.000356**	**-1.1**
yacht	0.0244	0.383	-0.919	~~0.663~~	~~0.096~~	~~-0.744~~	**0.0244**	**0.0376**	**2.18**
Total wins or ties	–	–	–	*2*	*0*	*2*	*11*	*11*	*9*

B.2 Adaptation to Monte Carlo Dropout Methods

Gal and Ghahramani (2016) propose using dropout during training and prediction to approximate a Gaussian Process. During prediction, they generate M mean estimates from the same network but for different dropout samples; this results in a predictive uniform mixture of a mean-only model evaluated over M dropout samples d_m:

$$p(Y|X=x) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{N}(Y|\mu(x, d_m, \theta), I_{\dim(Y)}).$$

Kendall and Gal (2017) replace the underlying homoscedastic model with a heteroscedastic model, which results in a predictive uniform mixture of a heteroscedastic model evaluated over M dropout samples:

$$p(Y|X=x) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{N}(Y|\mu(x, d_m, \theta), \Sigma(x, d_m, \theta)). \tag{7}$$

Both Gal and Ghahramani (2016) and Kendall and Gal (2017) use conventional NLL minimization with the addition of dropout. Because Gal and Ghahramani (2016) assume homoscedastic noise variance, they avoid the deleterious 'rich-get-richer' cycle of using high variance to explain and ignore poor mean estimates (Seitzer et al., 2022). Kendall and Gal (2017), however, map covariates to heteroscedastic noise variance estimates and thus are susceptible to entering this cycle. To construct a 'faithful' variant, we simply optimize the underlying Gaussian model with our method and dropout. Evaluation of our 'faithful' variant's predictive distribution is identically over dropout samples as in supplement eq. (7).

Figure 8 examines the convergence behavior of the Monte Carlo Dropout methods. Unlike for the other model classes, we use two hidden layers in f_{trunk} for Monte Carlo Dropout methods during our convergence experiments. Using dropout with just a single layer deeply affected performance of all Monte Carlo Dropout methods. With two layers, adding dropout does not resolve NLL optimization flaws. The Conventional Heteroscedastic Monte Carlo Dropout (Kendall and Gal, 2017) uses high variance to explain the mean error at $X = 9.5$ despite having the flexibility to converge thereto as demonstrated by the Unit Variance Homoscedastic Monte Carlo Dropout (Gal and Ghahramani, 2016). As with the Unit Variance Homoscedastic Deep Ensemble, the Unit Variance Homoscedastic Monte Carlo Dropout model's predictive variance changes w.r.t. x outside the data-rich interval $[2.5, 7.5]$ to capture epistemic uncertainty in the mean. Table 5 reports UCI regression performances for the Monte Carlo Dropout methods. Our faithful variant overwhelmingly outperforms Kendall and Gal (2017).

B.3 Adaptation to the Student's t-distribution

In section 2 of our main manuscript, we discuss existing methods for improving heteroscedastic regression performance that yield a Student's t predictive distribution (Takahashi et al., 2018; Skafte et al., 2019; Stirn and Knowles, 2020). Our

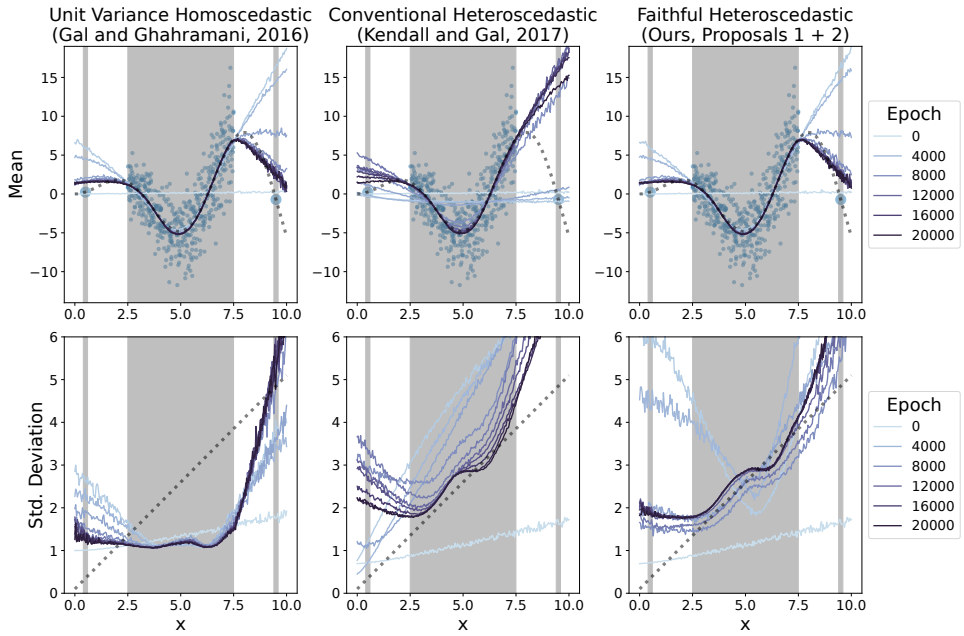


Figure 8: Convergence Behavior of Monte Carlo Dropout Methods

Table 5: UCI Regression Performance of Monte Carlo Dropout Methods

	Unit Variance Homoscedastic (Gal and Ghahramani, 2016)			Conventional Heteroscedastic (Kendall and Gal, 2017)			Faithful Heteroscedastic (ours)		
Dataset	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL
boston	0.332	0.179	-0.988	~~0.419~~	~~0.0176~~	~~-0.225~~	**0.332**	**0.0253**	**-0.308**
carbon	0.134	0.796	-2.81	**0.117**	0.331	**1.56**	0.134	**0.305**	**1.11**
concrete	0.299	0.162	-0.978	~~0.322~~	~~0.0113~~	~~-0.197~~	**0.299**	**0.0125**	**-0.208**
energy	0.215	0.313	-1.88	~~0.227~~	~~0.0934~~	~~1.23~~	**0.215**	**0.141**	**0.606**
naval	0.15	0.327	-1.88	~~0.237~~	~~0.0597~~	~~0.872~~	**0.15**	**0.0671**	**0.893**
power plant	0.25	0.02	-0.955	**0.248**	**0.000682**	**-0.0395**	0.25	0.000747	**-0.0587**
protein	0.0473	0.25	-0.926	~~0.472~~	~~0.146~~	~~0.271~~	**0.0473**	**0.248**	**-0.22**
superconductivity	0.339	0.0218	-0.986	~~0.483~~	~~0.000891~~	~~-0.096~~	**0.339**	**0.0023**	**-0.16**
wine-red	0.773	0.00523	-1.23	**0.778**	0.00177	**-1.15**	**0.773**	**0.00177**	-1.17
wine-white	0.761	0.00185	-1.22	~~0.788~~	~~0.000475~~	~~-1.17~~	**0.761**	**0.000468**	**-1.19**
yacht	0.0996	0.654	-0.935	~~0.662~~	~~0.11~~	~~-0.844~~	**0.0996**	**0.348**	**0.567**
Total wins or ties	–	–	–	*3*	*1*	*3*	*9*	*10*	*10*

proposals are also adaptable to the t-distribution. To do so, we add a new parameter head such that the network partitions are now:

- Shared representation learner f_{trunk} has parameters θ_z.
- Location parameter head f_μ has parameters θ_μ.
- Scale parameter head f_σ has parameters θ_σ.
- Degrees-of-freedom parameter head f_ν has parameters θ_ν, for which we constrain outputs to $(3, \infty)$ via a shifted softplus (we do this for our implementations of Takahashi et al. (2018); Stirn and Knowles (2020) as well).

Similar to our proposal for heteroscedastic Normal models, we can guarantee faithfulness when optimizing a heteroscedastic t-distribution's log likelihood $\mathcal{L}$ by:

1. Setting $\nabla_{\mu(x)} \mathcal{L} \coloneqq \sum_{(x,y)\in\mathcal{D}} \mu(x) - y$ (i.e. the gradient resulting from minimizing the sum of squared errors)
2. Stopping $\nabla_{\sigma(x)} \mathcal{L}$ and $\nabla_{\nu(x)} \mathcal{L}$ from contributing to updates for any shared parameters θ_z

The first proposal is equivalent to optimizing a t-distribution in the limit as the degrees of freedom $\to \infty$. At prediction time, our Unit Variance Homoscedastic variant approximates this limit by outputting a constant 100 for the degrees-of-freedom parameter and setting $\sigma = \sqrt{(100-2)/100}$ to ensure unit variance.

Figure 9 examines the convergence behavior of the t-distributed methods. Our Faithful Heteroscedastic variant is the only heteroscedastic method that can converge on the isolated point at $X = 9.5$. Table 6 reports UCI regression performances for the t-distributed methods. Our Faithful Heteroscedastic variant is never empirically unfaithful and is the top-performing method.

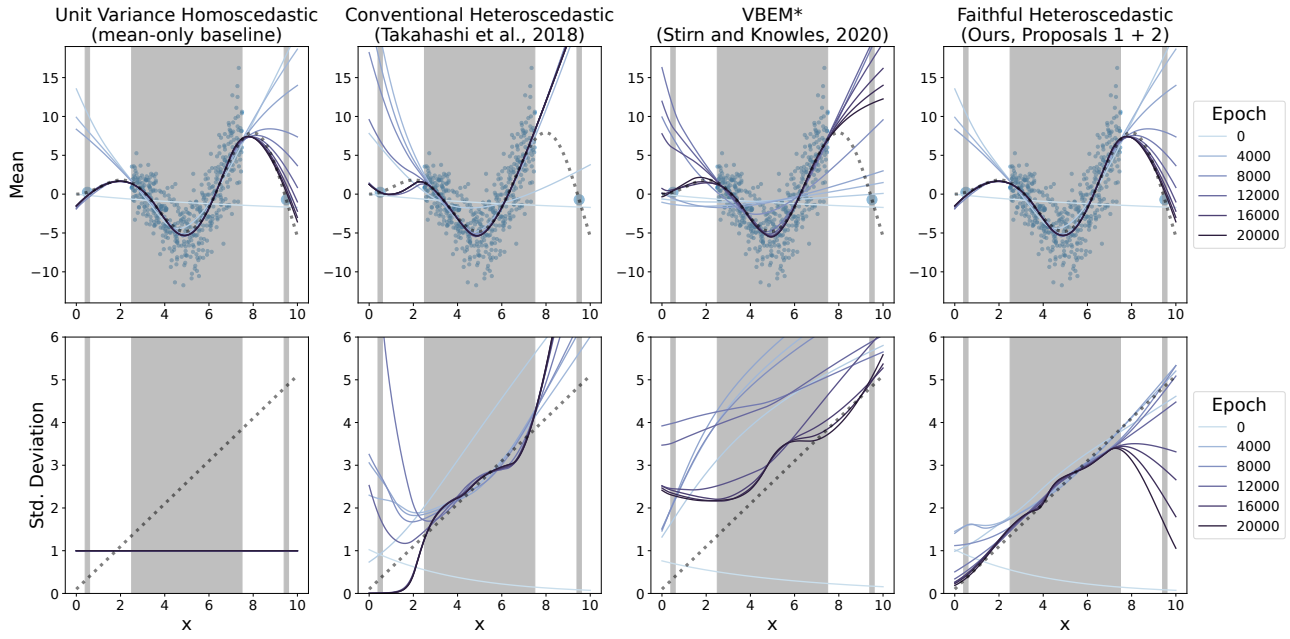


Figure 9: Convergence Behavior of t-distributed Methods

Table 6: UCI Regression Performance of t-distributed Methods

	Unit Variance Homoscedastic (mean-only baseline)			Conventional Heteroscedastic (Takahashi et al., 2018)			VBEM* (Stirn and Knowles, 2020)			Faithful Heteroscedastic (ours)		
Dataset	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL	RMSE	ECE	LL
boston	0.304	0.168	-0.959	~~0.387~~	~~0.017~~	~~-1.05~~	**0.339**	**0.0137**	**-0.311**	**0.304**	0.026	-1.63
carbon	0.0489	0.279	-2.74	**0.0488**	0.000339	**12.6**	**0.0488**	0.0054	**9.01**	0.0489	**0.000134**	**11.8**
concrete	0.265	0.127	-0.947	~~0.322~~	~~0.0151~~	~~-2.53~~	**0.27**	**0.0134**	**-0.22**	**0.265**	0.0306	**-0.93**
energy	0.16	0.268	-1.84	~~0.196~~	~~0.00164~~	~~4.48~~	~~0.185~~	~~0.0228~~	~~2.96~~	**0.16**	**0.00104**	**3.94**
naval	0.0247	0.271	-1.82	~~0.521~~	~~0.000101~~	~~4.69~~	~~0.0482~~	~~0.0034~~	~~6.34~~	**0.0247**	**0.000171**	**7.04**
power plant	0.22	0.0105	-0.936	~~0.227~~	~~0.000179~~	~~0.15~~	~~0.221~~	~~0.000177~~	~~0.122~~	**0.22**	**0.000152**	**0.154**
protein	0.00257	0.49	-0.911	~~0.0287~~	~~0.000721~~	4	~~0.00383~~	~~0.00429~~	~~4.38~~	**0.00257**	**0.00137**	**4.94**
superconductivity	0.326	0.0104	-0.966	~~0.397~~	~~0.00153~~	~~0.16~~	~~0.347~~	~~0.0011~~	~~0.132~~	**0.326**	**0.000993**	**0.02**
wine-red	0.769	0.00601	-1.21	~~0.782~~	~~0.00249~~	~~-1.36~~	**0.771**	0.00567	**-1.16**	**0.769**	**0.00247**	-1.19
wine-white	0.777	0.0015	-1.22	~~0.783~~	~~0.000786~~	~~-1.29~~	**0.777**	0.000711	**-1.16**	**0.777**	**0.00041**	-1.17
yacht	0.0226	0.375	-0.912	~~0.677~~	~~0.0603~~	~~-0.689~~	**0.0193**	0.103	**2.42**	0.0226	**0.0161**	2.29
Total wins or ties	–	–	–	*1*	*0*	*1*	*6*	*2*	*6*	*9*	*9*	*7*

C EXPERIMENT DETAILS

In the following subsections, we provide additional details needed to reproduce our experiments. To most exactly reproduce our results, please see or use our provided code. By default, our code uses the same random number seeds that we used and enables GPU determinism. In all experiments, we assume diagonal covariance and parameterize its standard deviation, that is we replace (co)variance function f_Σ with diagonal standard deviation function f_σ.

C.1 Convergence Experiment Details

Supplement table 7 has the neural network architecture used to generate figure 2 in our main manuscript. Every method fits the same 500 data points, which fully comprise a batch, and submits their objective to Adam (Kingma and Ba, 2014) with a 1e-3 learning rate for 20,000 epochs. Our main manuscript contains the remaining implementation details.

Table 7: Neural Network Architecture: Convergence Experiments

f_{trunk}	f_μ	f_σ
Dense (50 elu units)	Dense (1 linear unit)	Dense (1 softplus unit)

C.2 UCI Experiment Details

Supplement table 8 has the neural network architecture used in section 3.2 of our main manuscript. We use Adam (Kingma and Ba, 2014) with a 1e-3 learning rate for a maximum of 60k epochs. Because the UCI datasets are relatively small, every batch contains the entire training set. We stop training early if the validation root-mean-square error (RMSE) has not improved for 100 epochs. We always restore the model weights from the epoch with the best RMSE. We monitor RMSE instead of NLL in order to give every model the best chance of being faithful.

Table 8: Neural Network Architecture: UCI Experiments

f_{trunk}	f_μ	f_σ
Dense (50 elu units) Dense (50 elu units)	Dense ($\dim(Y)$ linear units)	Dense ($\dim(Y)$ softplus units)

C.3 VAE Experiment Details

Supplement table 9 has the neural network architecture used in section 3.3 of our main manuscript. The encoder, f_{trunk}, parameterizes a $\dim(z) = 16$ latent embedding. We use Adam (Kingma and Ba, 2014) with a 1e-3 learning rate for a maximum of 2k epochs with a 2048 batch size. We monitor RMSE, perform early stopping, and weight restoration as discussed in supplement appendix C.2. We rotate our noise template by $\frac{2\pi}{\text{num. classes.}}(\text{class label})$ radians. The training/validation split assignments are those provided by the TensorFlow Datasets API.

Table 9: Neural Network Architecture: VAE Experiments

f_{trunk}	f_μ	f_σ
Conv ($32 \times 5 \times 5$, stride 2, elu)	Dense (128 elu units)	Dense (128 elu units)
Conv ($32 \times 5 \times 5$, stride 2, elu)	Dense (1568 elu units)	Dense (1568 elu units)
Dense (128 elu units)	ConvT ($32 \times 5 \times 5$, stride 2, elu)	ConvT ($32 \times 5 \times 5$, stride 2, elu)
Dense (16 linear units, 16 softplus units)	ConvT ($1 \times 5 \times 5$, stride 2, linear)	ConvT ($1 \times 5 \times 5$, stride 2, softplus)
Sampling Layer		

C.4 CRISPR-Cas13 Experiment Details

Supplement table 10 has the neural network architecture used in section 3.4 of our main manuscript. We use Adam (Kingma and Ba, 2014) with a 1e-3 learning rate for a maximum of 2k epochs with a 2048 batch size. We monitor RMSE, perform early stopping, and weight restoration as discussed in supplement appendix C.2. For each dataset, we have either two or three replicate measurements for computing the mean (i.e. de-noised) efficacy scores. We use 10k randomly selected training points as the background to compute SHAP values (Lundberg and Lee, 2017) for a validation fold. If we have fewer than 10k training points, we use the entire training set. Their online documentation recommends using training points as the background. For visual aesthetics, we add back a scaled version of the average model output over the background samples to the respective sequence SHAP values. We scale this value by $\frac{1}{\text{sequence length}}$.

Table 10: Neural Network Architecture: CRISPR-Cas13 Experiments

f_{trunk}	f_μ	f_σ
Conv (64×4, stride 1, relu)	Dense (128 elu units)	Dense (128 elu units)
Conv(64×4, stride 1, relu)	Dense (32 elu units)	Dense (32 elu units)
MaxPool1D (pool 2, stride 1)	Dense (1 linear unit)	Dense (1 softplus unit)

D ADDITIONAL RESULTS

The following pages provide results supplementary to those from our main manuscript.

D.1 Additional VAE Results

Supplement figs. 10 and 11 are larger versions of figures 3 and 4 from our main manuscript. Supplement figs. 12 and 13 are the same but for the Fashion MNIST dataset.

D.2 Additional CRISPR-Cas13 Results

Supplement figs. 14 to 16 contain the complete set of figures (those similar to figure 5 of our main manuscript) for all models and all CRISPR-Cas13 datasets. The first two rows of every subplot are the average SHAP values of the estimated mean when trained on replicate and mean efficacy scores, respectively. The next two rows are the same but for the average SHAP values of the estimated standard deviation. The final row subtracts the average SHAP values of the estimated standard deviation when trained on means from those when trained on replicates and represents the SHAP values of the estimated square root of noise variance.

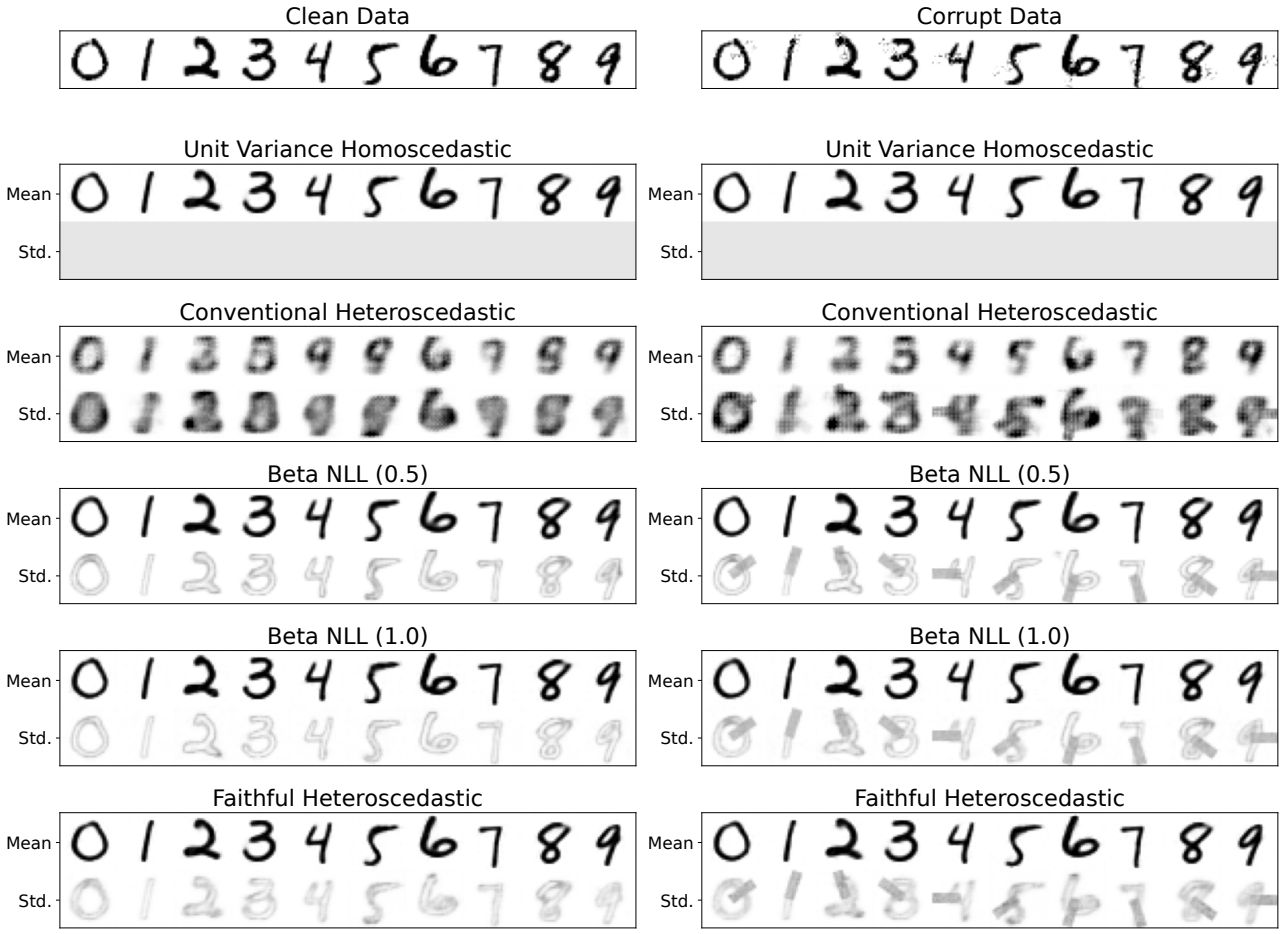


Figure 10: VAE Predictive Moments for MNIST

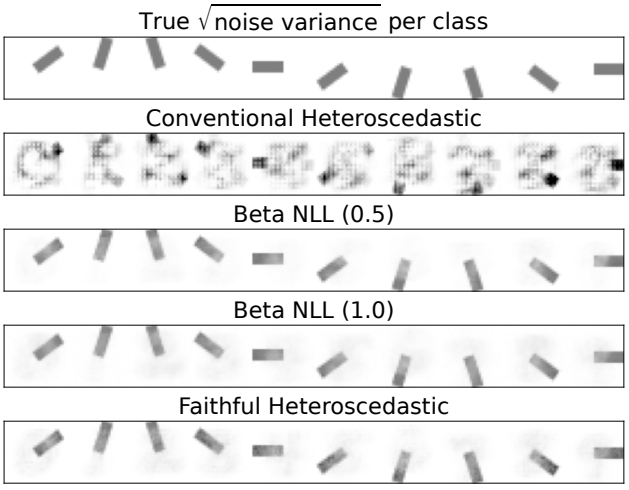


Figure 11: VAE Recovered Noise Variance for MNIST

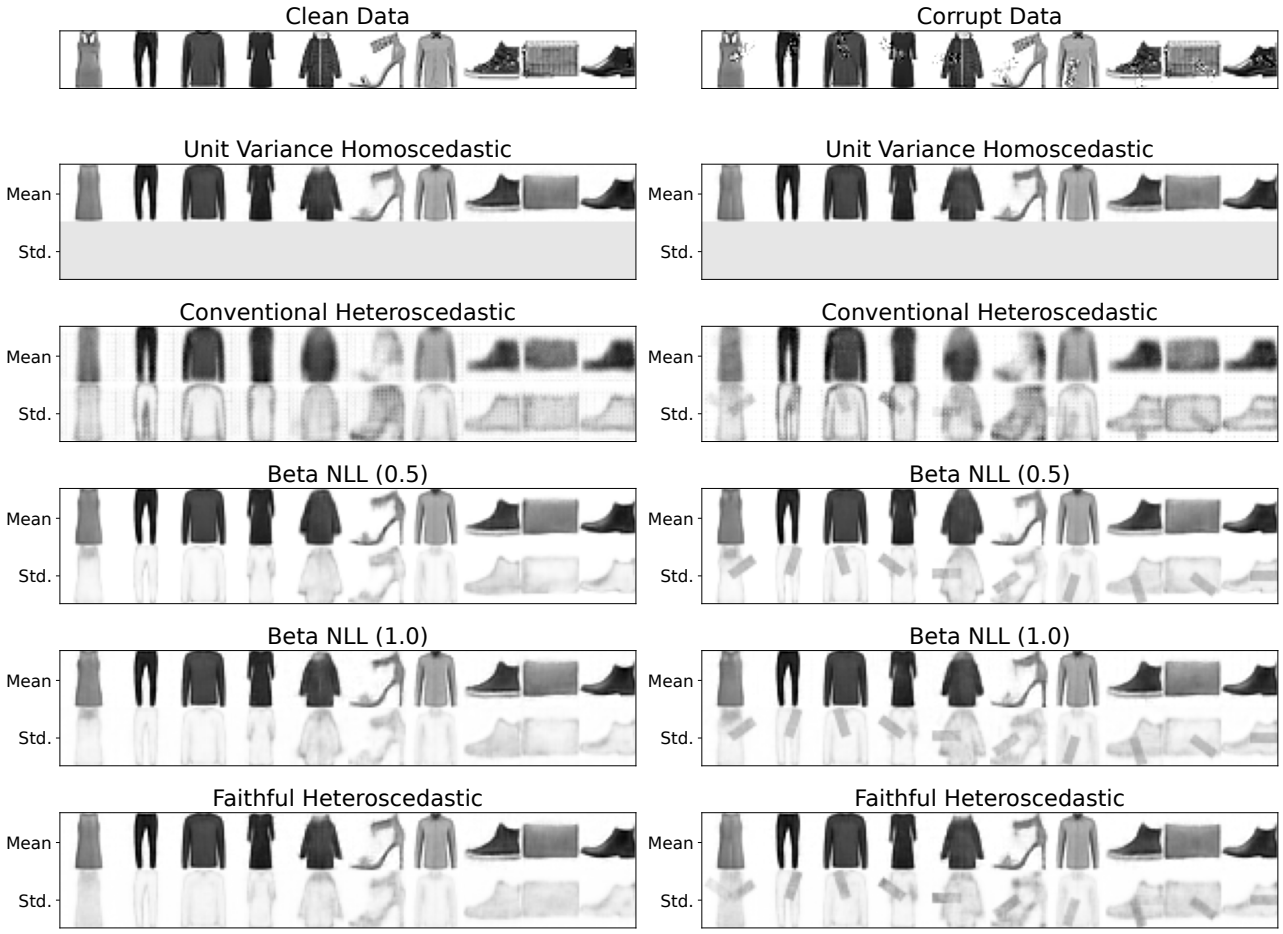


Figure 12: VAE Predictive Moments for Fashion MNIST

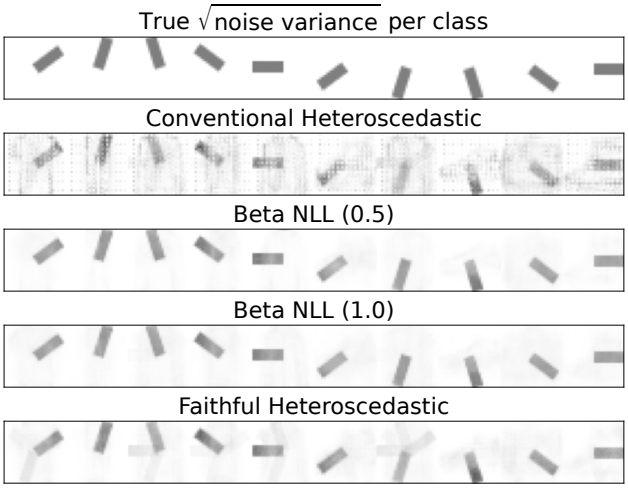
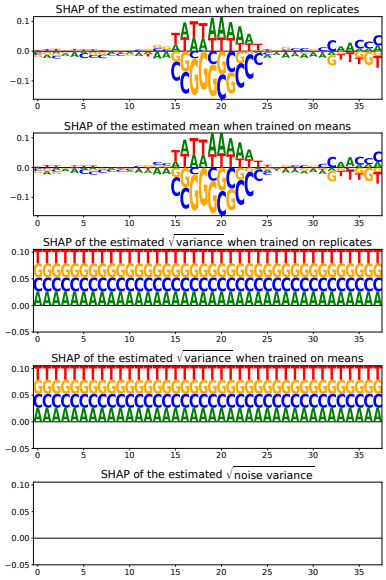
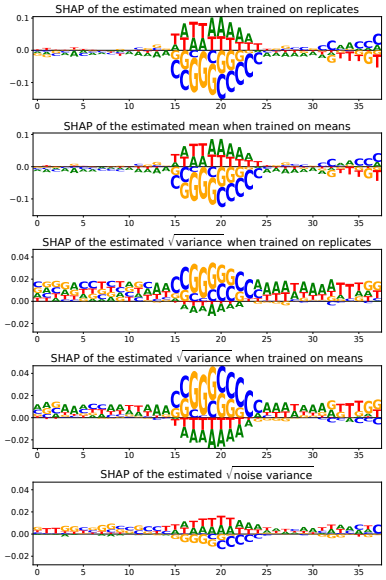


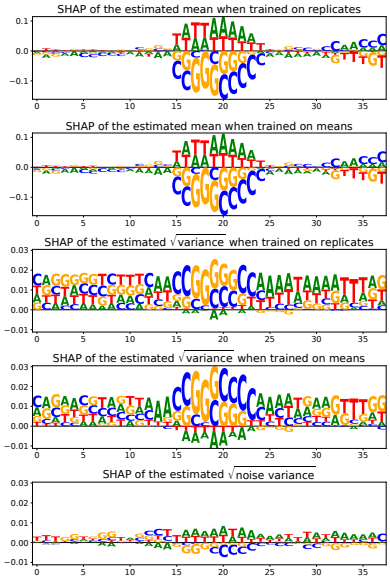
Figure 13: VAE Recovered Noise Variance for Fashion MNIST



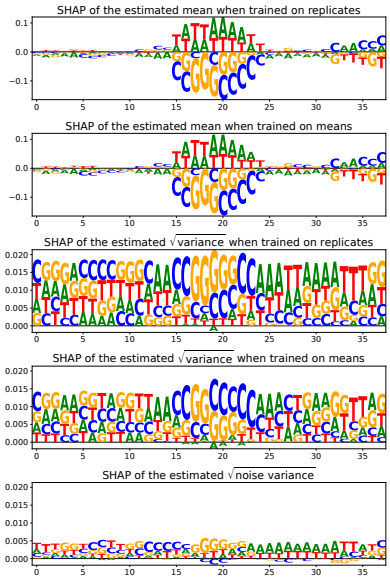
(a) Unit Variance



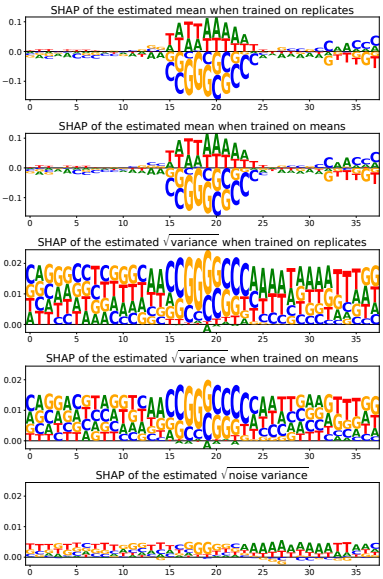
(b) Heteroscedastic



(c) Beta NLL (0.5)

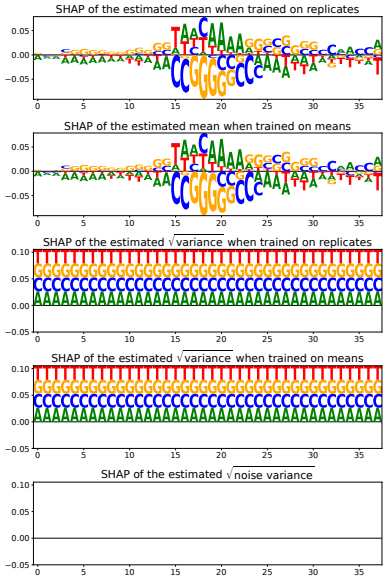


(d) Beta NLL (1.0)

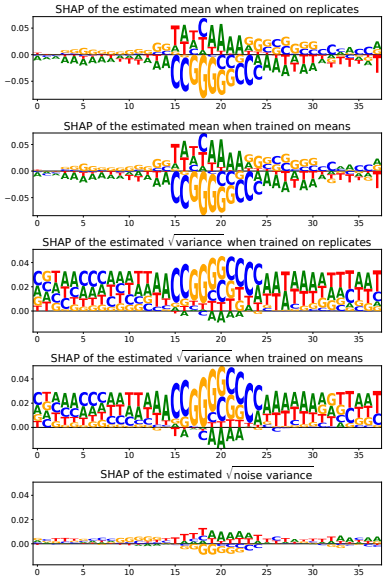


(e) Faithful Heteroscedastic

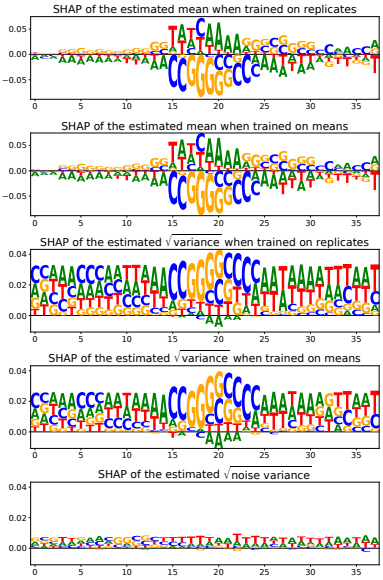
Figure 14: SHAP Values for Flow Cytometry (HEK293) Dataset



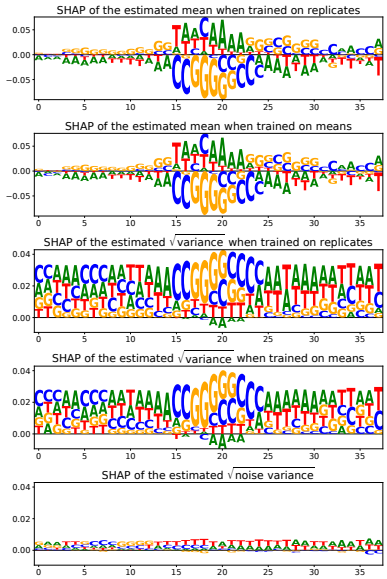
(a) Unit Variance



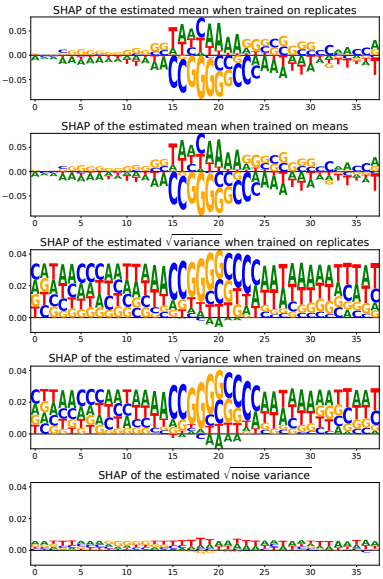
(b) Heteroscedastic



(c) Beta NLL (0.5)

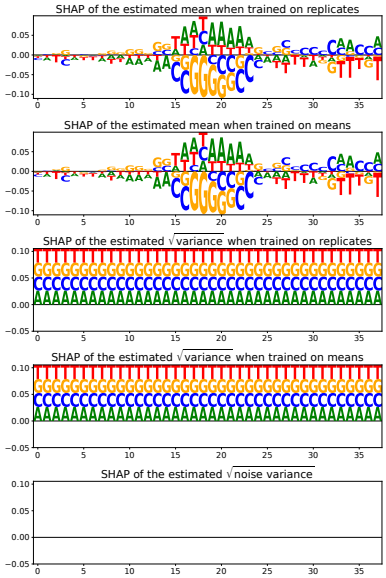


(d) Beta NLL (1.0)

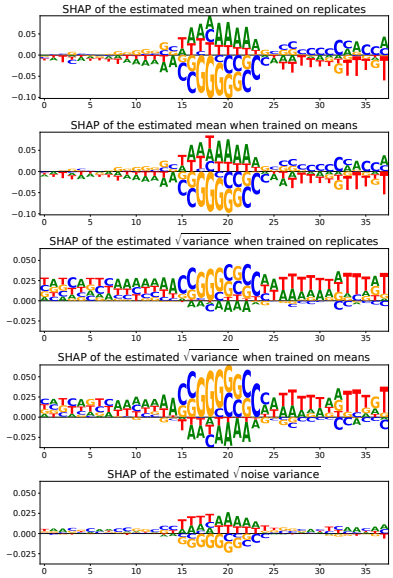


(e) Faithful Heteroscedastic

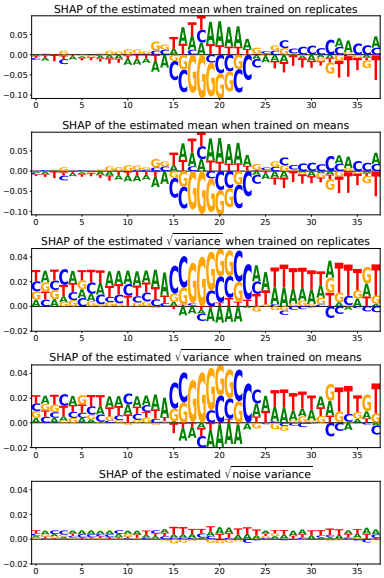
Figure 15: SHAP Values for Survival Screen (A375) Dataset



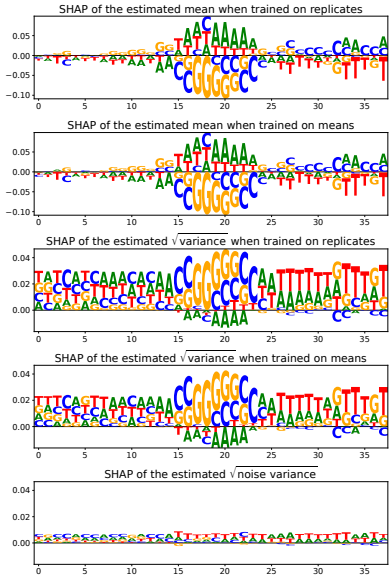
(a) Unit Variance



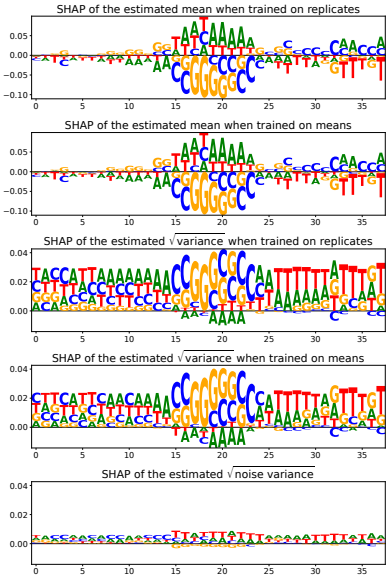
(b) Heteroscedastic



(c) Beta NLL (0.5)



(d) Beta NLL (1.0)



(e) Faithful Heteroscedastic

Figure 16: SHAP Values for Survival Screen (HEK293) Dataset