QuIL: Quantifying Quantization-induced Information Loss in Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

Floating-point bit-string representations of parameters are ubiquitous building blocks of neural networks, and their quantization into lower precision formats is increasingly used to reduce their computational requirements. The degradation that quantization induces in the model is, however, not well studied, and its detection relies on task-specific heuristics. We propose QuIL: An information-theoretic approach for measuring quantization-induced information loss. By modelling the probability distribution of the underlying floating-point bit-strings of the parameters with a probabilistic circuit, we can measure information loss as the change in entropy across quantization levels. Experiments on neural-network regression, vision transformers, and GPT models show that QuIL, computed from the parameter posterior alone, identifies the bit-width at which quantization begins to degrade accuracy, calibration, and predictive density, offering a principled basis for selecting parameter precision.

1 Introduction

Model parameters are never manipulated as idealized infinite precision real numbers on a computer, which becomes exceedingly apparent in the low-bit-width representations popular in modern machine learning (Ma et al., 2024). Instead, parameters are manipulated as finite-length *bit-strings* (ordered sequences of 0s and 1s) that are interpreted under a chosen number system, most commonly floating-point arithmetic (IEEE, 2019; Knuth, 1997; Sterbenz, 1974). As finite objects, they are approximations of the parameter. From the perspective of *probabilistic numerics*, such numerical approximation introduces uncertainty that should be explicitly modelled and quantified (Hennig et al., 2015; Cockayne et al., 2019; Hennig et al., 2022). We apply this perspective to quantization and expose rather than hide the inherent uncertainty over its idealized infinite precision value when quantizing model parameters in neural networks.

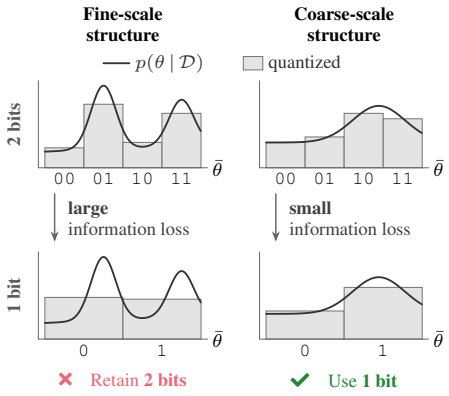

Figure 1: The same precision reduction can remove different amounts of posterior information. **QuIL** aims to capture this.

Many existing works characterize the effects of model parameter quantization through task-specific loss metrics (*e.g.*, accuracy, curvature). While such metrics establish a practical consequence of quantization, they only indirectly measure what is lost from the model. Taking a probabilistic perspective, we instead consider the parameter posterior distribution, which encodes information about the data distribution, and how its *entropy* (a measure of information, Shannon, 1948) is affected by quantization. We ask the questions: *(i)* how can we measure quantization-induced information loss from the posterior, and *(ii)* is this quantity sufficient on its own for predicting the downstream task performance degradation. We investigate by defining the parameter posterior directly over their bit-string representations, and define a novel measure computed from its change in entropy. The intuition of our method is shown in Fig. 1: the same reduction in bit-width can remove different amounts

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of posterior structure, reflecting different amounts of loss of information on the data distribution, depending on how probability is distributed across the quantization bins.

We introduce **QuIL**[1], a metric for *quantization-induced information loss*. On a bounded numerical domain, we obtain lower-precision representations of a fitted posterior by merging cells, preserving their combined probability mass, and making the density uniform within each merged cell. We then define **QuIL** as the difference in entropy between the fine posterior and its coarse representation, and argue that it is a measure of the information lost due to quantization. This gives a non-negative measure of the posterior structure removed by the specified coarsening, which we hypothesize to be a good indicator of quantization-induced model degradation.

To compute this quantity for realistic models, we extend BitVI (Sladek et al., 2025) to floating-point bit-string representations. Its probabilistic circuit representation provides a tractable hierarchy from which all coarsenings can be derived without refitting. Once the posterior is fitted, **QuIL** can be evaluated without the need of further downstream performance measurements. Its computation is exact for that fitted distribution, with its accuracy relative to the true posterior depending on the posterior approximation accuracy. We evaluate **QuIL**'s relationship to model degradation empirically, and show quantitative and qualitative results indicating **QuIL** can predict quantization-induced model degradation at several quantization levels.

Contributions Our contributions are as follows:

- We formulate parameter quantization as posterior coarsening and establish a non-negative information-loss measure through exact posterior entropy computation.
- We extend BitVI to floating-point representations, enabling tractable evaluation of **QuIL** across precisions and aggregation across parameter groups under the mean-field model.
- We show on feedforward MLPs, vision transformers, and GPT models that using the parameter posterior alone, **QuIL** can identify the bit-width at which quantization becomes harmful to downstream model performance.

1.1 Related work

Quantization is a practical tool for making neural networks cheaper to store and run. Early and widely used post-training methods have studied how to choose clipping ranges, correct weight and activation scales, and reduce rounding error without retraining the full model (Banner et al., 2019; Nagel et al., 2019; 2020). Recent methods address activation outliers, smooth difficult channels, or reconstruct weights layer by layer or block by block, enabling accurate low-bit inference for transformers and language models (Dettmers et al., 2022; Yao et al., 2022; Frantar et al., 2022; Xiao et al., 2023; Lin et al., 2024). **QuIL** complements these deployment methods by characterizing posterior information loss under a specified precision reduction.

Sensitivity-based methods relate quantization to the geometry of a loss. The HAWQ family (Dong et al., 2019; 2020; Yao et al., 2021) uses Hessian information to guide mixed-precision allocation, while OBC (Frantar & Alistarh, 2022) and GPTQ (Frantar et al., 2022) use curvature in layerwise reconstruction. These criteria measure local perturbation or reconstruction costs. **QuIL** instead measures the change in a posterior under a specified coarsening, providing a complementary distributional criterion for studying precision reductions.

Bayesian approaches already connect posterior uncertainty to compression. Bayesian compression uses weight uncertainty to select fixed-point precision (Louizos et al., 2017), and Bayesian Bits learns quantization and pruning gates in a variational framework (Van Baalen et al., 2020). BitVI ties approximate inference to bit-string representations and provides tractable distributions over parameter values (Sladek et al., 2025). Building on this representation, our contribution is to quantify the information removed by coarsening a fitted posterior. This connects bit-string inference to the probabilistic numerics question of what is lost through numerical approximation.

[1]The name comes from a *quill*: the fineness of its stroke limits the detail it can convey.

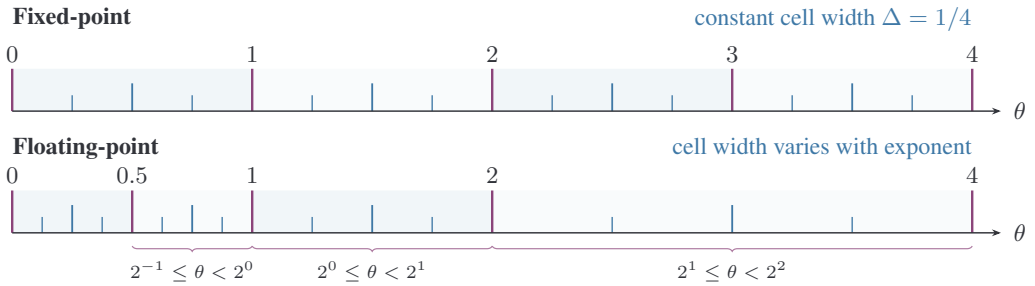

Figure 2: **Equal bit budgets, different numerical resolution.** Partitioning of the domain of θ, each with the same number of cells shown. Tall ticks delimit coarse intervals; shorter ticks show two successive binary refinements. Fixed-point cells have constant width. Floating-point uses 2 exponent and 2 mantissa bits, with cell widths doubling across successive normal exponent ranges.

2 METHODS

Quantization makes parameter values indistinguishable when they share a lower-precision code. To quantify the information removed, we ask how much structure a parameter posterior assigns to distinctions that the coarser representation cannot express. This requires a posterior tied to numerical cells and a consistent way to coarsen it. Therefore, we first construct a tractable posterior over a hierarchy of bit-string representations, then measure the change induced by merging its cells. All resolutions are derived from one fitted posterior approximation, separating information loss due to coarsening from changes due to refitting or variance in the approximation algorithm.

2.1 PRELIMINARIES: BIT-STRING REPRESENTATION OF MODEL WEIGHTS

Parameters in a neural network are *necessarily* represented by finite bit-strings. In particular, let $\theta_k \in \mathbb{R}$ represent a continuous parameter, such as a weight of a neural network $f_{\boldsymbol{\theta}}$ with $\theta_k \in \boldsymbol{\theta}$. Further, let $\boldsymbol{b} \in \mathbb{B}^B$ be the B-bit *floating-point bit-string* $\boldsymbol{b} = \{b_{B-1}, b_{B-2}, \ldots, b_0\}, b_i \in \{0, 1\}$ that represents θ_k. Then, we obtain θ_k from $\boldsymbol{b}$ by the mapping function $g_{\text{float}} \colon \mathbb{B}^B \to \mathbb{R}$:

$$\theta = f_{\text{float}}(\mathbf{b}) = \underbrace{(-1)^{b_{B-1}}}_{\text{sign}} \cdot \underbrace{\left[2^{\left(\sum_{i=M}^{B-2} b_i 2^{i-M}\right)-\kappa}\right]}_{\text{exponent}} \cdot \underbrace{\left(1 + \sum_{j=0}^{M-1} b_j 2^{j-M}\right)}_{\text{mantissa}}, \tag{1}$$

where $E \in \mathbb{Z}$ and $M \in \mathbb{Z}$ represent the number of exponent and mantissa bits, respectively, and κ is a bias term. This mapping function (excluding special tokens) is the standard IEEE 754 floating-point representation (IEEE, 2019). Note that `bfloat16` (Wang & Kanwar, 2019), specifically designed for deep learning, is a special case of Eq. (1) with a different assignment of bits than IEEE 754.

In quantization, we are interested in finding a minimal precision representation B' of $f_{\boldsymbol{\theta}}$, *i.e.*, a representation of the weights $\theta_k \in \boldsymbol{\theta}$ each with $B' \ll B$ bits, such that adding additional bits to B' yields a minimal gain of information. To obtain an actionable objective, we need to decide what to measure the information gain over. In this work, we take a probabilistic stance and argue that a natural approach is to use an 'intrinsic' measure of information gain derived from the Bayes posterior over the weights. Hence, in the following, we will first introduce how to represent and estimate the Bayes posterior over model parameters represented using the floating-point number system, IEEE 754 or `bfloat` (Sec. 2.2). Then we will introduce **QuIL**, a principled metric of quantization-induced uncertainty which allows for effective and safe model parameter quantization (Sec. 2.3).

2.2 ESTIMATING THE BAYES POSTERIOR OVER BIT-STRINGS

Our probabilistic construction requires a posterior representation that supports fitting cell probabilities, aggregating them across resolutions, and computing the change in entropy exactly. For this, we build on BitVI (Sladek et al., 2025), which provides these operations through a probabilistic circuit (PC, see Choi et al., 2020; Vergari et al., 2021) over bit-strings. For this, we define a tree-structured PC which makes the relationship between resolutions explicit, *i.e.*, a path in the PC indexes a fine cell, and a

prefix identifies the union of its descendants. Further, we extend the fixed-point construction of Sladek et al. (2025) to floating-point, as this number format is more widely used in machine learning models.

Probabilistic circuits express probability distributions as computational graphs with sum, product, and leaf nodes arranged in a directed acyclic graph (DAG) structure. If the structure and leaf nodes fulfill certain properties, such as *decomposability, smoothness, or determinism* (see Choi et al., 2020, for further detail), then the computation of certain probabilistic inferences, such as computing moments, can be guaranteed to be computable exactly in linear time in the model complexity. At the same time, the formalism of PCs also allows to reason about expressivity of the model and has recently been exploited to derive tractable yet highly flexible probabilistic models (Loconte et al., 2024; Yu et al., 2023).

For a scalar parameter θ, let $\boldsymbol{b} = (b_{B-1}, \ldots, b_0)$ contain one sign bit, E exponent bits, and M mantissa bits, with $B = 1 + E + M$. Each code indexes a bounded interval $I_{\boldsymbol{b}}$ of width $\Delta_{\boldsymbol{b}}$. Mantissa bits refine intervals within an exponent range, so cell widths vary with magnitude. We include sub-normals and interpret all-ones exponents as finite values; the mapping is specified in Sec. A.2. The domain and cell boundaries are fixed before fitting.

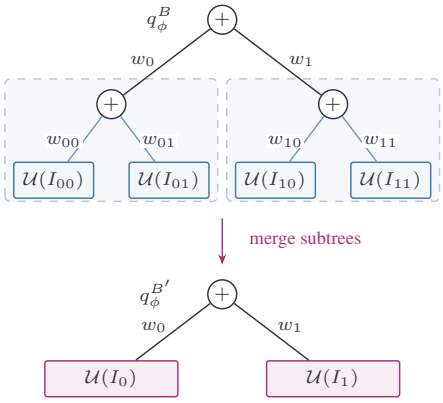

Figure 3: **One posterior, multiple resolutions.** A two-level circuit fragment. Weighted sums allocate posterior mass to uniform leaves. Coarsening replaces each shaded subtree by $\mathcal{U}(I_\varepsilon)$, where $I_\varepsilon = I_{\varepsilon 0} \cup I_{\varepsilon 1}$, preserving its incoming mass w_ε.

The PC mirrors this partition hierarchy (Fig. 3). Each sum node stores the conditional probabilities of the next bit given its prefix, with outgoing weights summing to one. Multiplying these weights along a path gives the cell probability $P_\phi(\boldsymbol{b})$. Finally, its leaf follows a uniform distribution on $I_{\boldsymbol{b}}$, giving $q_\phi^B(\theta) = P_\phi(\boldsymbol{b})/\Delta_{\boldsymbol{b}}$ with $\boldsymbol{b} = g^{-1}(\theta)$. Thus, the circuit represents a continuous density whose resolved structure is encoded by cell masses. Its disjoint leaf supports make it a deterministic probabilistic circuit (Choi et al., 2020), enabling exact differential entropy calculation w.r.t. the Lebesgue measure, *i.e.*,

$$h(q_\phi^B) = -\int_\theta \frac{P_\phi(g^{-1}(\theta))}{\Delta_{g^{-1}(\theta)}} \log\left(\frac{P_\phi(g^{-1}(\theta))}{\Delta_{g^{-1}(\theta)}}\right) \mathrm{d}\theta \tag{2}$$

$$= -\sum_{\boldsymbol{b}} P_\phi(\boldsymbol{b}) \log P_\phi(\boldsymbol{b}) + \sum_{\boldsymbol{b}} P_\phi(\boldsymbol{b}) \log \Delta_{\boldsymbol{b}}. \tag{3}$$

This expression can be evaluated through a single upwards pass of the PC and its computation is linear in the complexity of the PC. The derivation is given in Sec. A.4.1. Note that in this work, we use the natural logarithms and report information in nats rather than bits.

Given data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and D network parameters, we approximate $p(\boldsymbol{\theta} \mid \mathcal{D})$ with the mean-field family $q_\phi^B(\boldsymbol{\theta}) = \prod_{k=1}^D q_{\phi,k}^B(\theta_k)$. To find the optimal circuit weights ϕ, we maximize the evidence lower bound (ELBO):

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi^B}\left[\sum_{i=1}^n \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right] + \sum_{k=1}^D h(q_{\phi,k}^B). \tag{4}$$

We estimate the expectation using K samples generated via inverse-CDF sampling, *i.e.*, $\theta_k^{(s)} = F^{-1}_{q^B_{\phi,k}}(u_k^{(s)})$ with $u_k^{(s)} \sim \mathcal{U}(0,1)$, and use pathwise gradients (Sladek et al., 2025). For pre-trained models we initialize the circuit with a diagonal Laplace approximation and perform additional refinement steps via ELBO optimization if needed.

While computations are independent of the chosen number system representation, the underlying number system strongly affects the approximation results as it dictates the decomposition of the domain. Fig. 4 shows example posteriors demonstrating the differences between fixed-point and float approximations, and the effect of coarsening on these posteriors. We observe that quantization under float approximations obtains more structure around zero, a beneficial property when modelling the posterior of neural network weights.

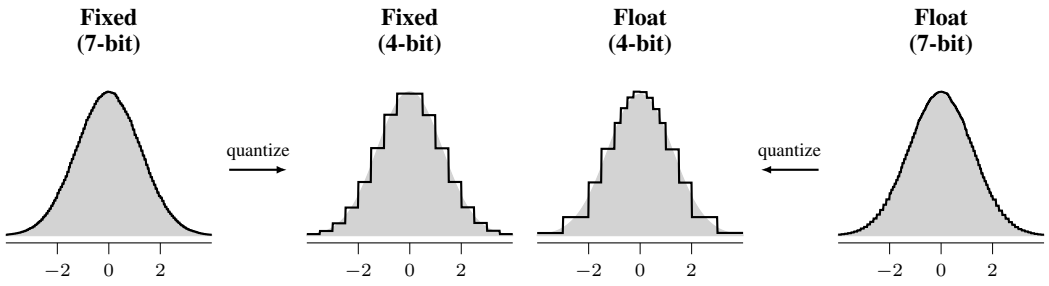

Figure 4: **BitVI on fixed-point and floating-point numbers.** 7-bit approximations of two single-parameter target densities (exact gray, approximation black line), and their quantized 4-bit counterparts. The floating-point representation captures more detail around zero and less far away from zero, which becomes more pronounced as bits decrease.

2.3 QuIL: A MEASURE OF QUANTIZATION-INDUCED INFORMATION LOSS

Bayes' rule, and in consequence the Bayesian posterior, is an efficient information processor; nothing the data tells us is lost (Zellner, 1988). Quantization, however, 'breaks' this conservation and discards part of what we have learned. We propose to measure exactly this loss through **QuIL**.

We introduce **QuIL**, a measure of quantization-induced information loss, which quantifies the additional uncertainty arising when reducing the B-bit posterior $q^B_{\phi,k}$ of weight k to t bits:

$$\textbf{QuIL}: \quad U(t,k) = h(q^t_{\phi,k}) - h(q^B_{\phi,k}) \geq 0 \tag{5}$$

where $t < B$. As a difference between the truncated posterior and the B-bit posterior entropy, **QuIL** measures the posterior structure removed by the chosen coarsening. As we chose $q_{\phi,k}$ to be in the BitVI family, Eq. (5) can be computed efficiently and analytically. However, the accuracy of the measure depends on the fitted posterior approximation, and it is not guaranteed that we recover the exact information loss under the true posterior.

As **QuIL** measures the quantization-induced information loss for the kth weight, it can be beneficial to aggregate measurements over a layer or block of weights for structured quantization, *i.e.*, if a whole layer or block is quantized with the same level. In particular, depending on the setting, one may consider the following aggregation strategies for **QuIL**: *(i)* averaging: $U_{\text{avg}}(t) = {}^1\!/_D \sum_k U(t,k)$, *(ii)* multiplication: $U_{\text{prod}}(t) = \prod_k U(t,k)$, and *(iii)* maximum: $U_{\max}(t) = \max_k U(t,k)$. In this work, we primarily focus on the averaging aggregation and report results based on this strategy for all experiments.

3 THE RELATIONSHIP TO EXISTING QUANTIZATION CRITERIA

QuIL measures the information removed by mass-preserving coarsening of a fitted posterior. Existing quantization criteria usually assess a weight perturbation through its effect on task loss or layer reconstruction error. To relate these views, consider a local Gaussian approximation $q_{\text{L}} = \mathcal{N}(\boldsymbol{\theta}^\star, \boldsymbol{H}^{-1})$, where $\boldsymbol{H} \succ 0$ is the negative-log-posterior Hessian at a stationary point $\boldsymbol{\theta}^\star$. A quantization perturbation $\boldsymbol{\delta} = Q_{B'}(\boldsymbol{\theta}^\star) - \boldsymbol{\theta}^\star$ gives

$$D_{\text{KL}}(q_{\text{L}} \,\|\, q_{\text{L},\boldsymbol{\delta}}) = \tfrac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{H}\boldsymbol{\delta}, \qquad q_{\text{L},\boldsymbol{\delta}} = \mathcal{N}(\boldsymbol{\theta}^\star + \boldsymbol{\delta}, \boldsymbol{H}^{-1}). \tag{6}$$

The same quadratic approximates the increase in the negative log posterior for small perturbations. This gives curvature scores a local KL interpretation, but the operation differs from **QuIL**: translating a Gaussian preserves its entropy, whereas coarsening increases entropy by removing within-cell structure.

HAWQ and HAWQ-V2 HAWQ (Dong et al., 2019) approximates each block Hessian by $\boldsymbol{H}_\ell \approx \lambda_\ell \boldsymbol{I}$ and uses the score $S_\ell = \lambda_\ell / n_\ell$ to determine a relative mixed-precision ordering; for block-wise fine-tuning it uses $\Omega_\ell = \lambda_\ell \|Q(\boldsymbol{W}_\ell) - \boldsymbol{W}_\ell\|_2^2$, where $\boldsymbol{W}_\ell$ denotes the full-precision weight

matrix of a layer. HAWQ-V2 (Dong et al., 2020) replaces the top-eigenvalue proxy by the average curvature, showing that under equal perturbation norms the more sensitive layer is the one with larger $\overline{\mathrm{tr}}(\boldsymbol{H}_\ell) = \mathrm{tr}(\boldsymbol{H}_\ell)/n_\ell$, and it selects candidate bit assignments by minimizing

$$\Omega = \sum_\ell \overline{\mathrm{tr}}(\boldsymbol{H}_\ell)\, \|Q(\boldsymbol{W}_\ell) - \boldsymbol{W}_\ell\|_2^2. \tag{7}$$

These perturbation scores follow, up to a constant, from a block-diagonal quadratic model with isotropic curvature within each block. Interpreting task-loss curvature as posterior precision additionally requires accounting for the loss scaling and prior curvature. HAWQ keeps only the worst curvature direction, whereas HAWQ-V2 keeps its layer average.

OBC/GPTQ OBC (Frantar & Alistarh, 2022) starts from the layerwise reconstruction problem $\min_{\widehat{\boldsymbol{W}}} \|\boldsymbol{W}\boldsymbol{X} - \widehat{\boldsymbol{W}}\boldsymbol{X}\|_F^2$ over quantized weights, whose rowwise Hessian is $\boldsymbol{H} = 2\boldsymbol{X}\boldsymbol{X}^\top$. Let F contain the remaining unquantized coordinates and assume the restricted Hessian $\boldsymbol{H}_F \succ 0$. Quantizing one coordinate while re-optimizing the others yields the OBS-style rule

$$j^\star = \arg\min_{j\in F} \frac{(Q(w_j) - w_j)^2}{[\boldsymbol{H}_F^{-1}]_{jj}}, \qquad \boldsymbol{\delta}_F = \frac{Q(w_{j^\star}) - w_{j^\star}}{[\boldsymbol{H}_F^{-1}]_{j^\star j^\star}} (\boldsymbol{H}_F^{-1})_{:,j^\star}. \tag{8}$$

This is the greedy minimizer of the reconstruction quadratic. GPTQ scales this approach through a shared column order, lazy block updates, and a Cholesky implementation using damped curvature $(2\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I})^{-1}$ (Frantar et al., 2022). Here, curvature describes layer-output reconstruction rather than the full parameter posterior.

These methods and **QuIL** thus address complementary aspects of precision reduction: the local cost of perturbing a point estimate and the posterior structure removed by coarsening. The Gaussian interpretation connects their mathematical forms without identifying their objectives.

4 Experiments

We evaluate **QuIL** along two separate axes: fidelity to information loss under a reference posterior and usefulness for assessing precision reductions. We first quantify posterior-approximation error in a tractable setting, then examine how information loss relates to predictive degradation under full-network coarsening. Next, we test whether **QuIL** distinguishes the effects of local precision changes in transformer weights, using common quantized networks to compare sensitivity criteria. Following Secs. 2 and 3, we use downstream performance as an empirical test of usefulness, distinct from the definition of posterior information loss.

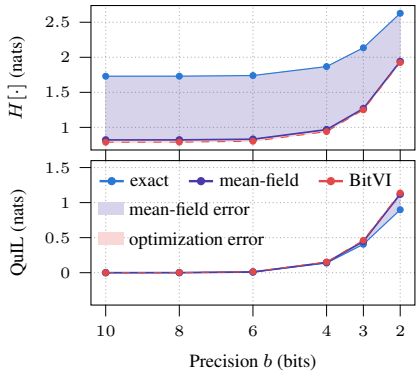

Figure 5: Upper plot shows the entropy of a known 2D posterior (correlation $\rho = 0.9$), its optimal mean field approximation and its BitVI approximation, and the lower plot shows each model's respective **QuIL** for several precision levels. Entropy approximation error is dominated by a mean-field error across precision levels, but it largely cancels out for the BitVI **QuIL**.

4.1 Validating QuIL

We begin with a sanity-check on a toy example that disentangles the stochastic optimization and mean-field assumption induced approximation errors in **QuIL**. We construct a tractable toy example (see Sec. B.5 for details) with the exact full-covariance and mean-field posterior over bit-string representations available, meaning the ground truth **QuIL** value is available. Our results show that **QuIL** tracks the exact ground truth closely at every precision level (from 10 to 2 bits) (Pearson $r \geq 0.85$, all $p < 10^{-10}$) and correlates just as strongly with actual predictive degradation (Pearson $r \in [0.84, 0.88]$). Furthermore, mean-field and optimization error sources have negligible impact on **QuIL** (see Fig. 5). At the most aggressive precision (2 bits) and most extreme misspecification ($\rho = 0.9$) tested, mean-field error is the dominant term (≈ 0.10 nats) and is about $3\times$ larger than the error from BitVI's training stochasticity (≈ 0.03 nats, roughly constant regardless of misspecification severity). In this limited toy model example, the **QuIL** signal is not an artifact of either approximation.

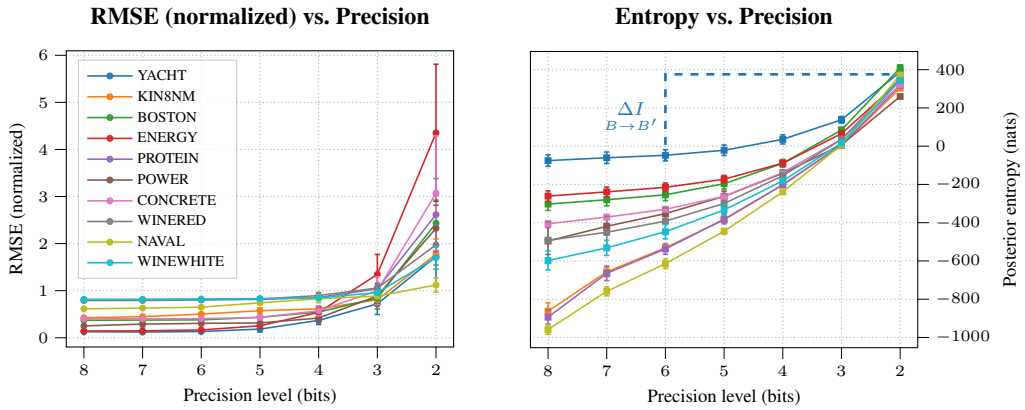

Figure 6: Full-network quantization on UCI regression tasks. Normalized test RMSE and posterior entropy are shown as functions of parameter precision, with error bars denoting s.d. across folds. Both remain stable at moderate precision and increase sharply below roughly 4–5 bits, indicating that the **QuIL** entropy signal tracks the onset of predictive degradation.

4.2 Information loss under full-network coarsening

We validate the new *floating-point* BitVI posterior that can be used by **QuIL** and provide preliminary results that **QuIL** can track quantization-induced performance degradation in a small Bayesian neural network setting. Work by Sladek et al. (2025) evaluated the *fixed-point* BitVI posteriors on the *Bayesian Benchmarks* (Kelly et al., 2025; Salimbeni, 2018) data sets, motivating this as our selection for testing the floating-point variant. We study ten small regression models trained on UCI data sets, where the full network posterior can be learned directly. Each model is a two-hidden-layer MLP with 16 units per layer, layer normalization (Ba et al., 2016), and $\tanh(\cdot)$ activations. We place an independent BitVI posterior over all weights, use a Gaussian likelihood $p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2)$ with fixed variance, and choose the Gaussian prior to match Kaiming initialization (He et al., 2015). Results are averaged over 5-fold cross-validation, with 10% of each training split used for early stopping (see Sec. B.1 for details).

Both floating-point and fixed-point variants of BitVI were trained, and a comparison of their performance is provided in Table A2. These results show that 10-bit and 6-bit BitVI floating-point models perform similarly to the equivalent fixed-point BitVI models. We also visualized the entropy of the floating-point BitVI circuit at various precision levels, shown in Fig. 6. This figure shows that consistently across the UCI data sets, the BitVI entropy increases monotonically as the selected precision level used for all parameters decreases. Degradation is small for larger precision levels, and increases for the more aggressive 2–3 bit quantization levels. A similar trend is shown for RMSE, giving preliminary evidence that our metric could be used to predict the degradation in performance metrics like RMSE as a consequence of the increase in parameter entropy that quantization induces.

4.3 Predicting the effect of local precision changes

To test whether **QuIL** can be scaled for use with large pre-trained vision and language models, we experimented with off-the-shelf ViT-base (Dosovitskiy et al., 2021) and GPT-2 (Radford et al., 2019) backbones, fine-tuned on various classification tasks. For vision, we use CIFAR-10 (Krizhevsky & Hinton, 2009), RESISC (Cheng et al., 2017), and DTD (Cimpoi et al., 2014). For language, we use BoolQ and MRPC from GLUE and SuperGLUE (Wang et al., 2019b;a). We tested **QuIL** by quantizing last-layers of the ViT and GPT-2 models, as well as the inner-layer weights of the ViT model, and compared **QuIL** to other quantization sensitivity metrics.

Our method was applied to pre-trained models by initializing an 8-bit BitVI posterior via a Laplace approximation, and then fine-tuning it on the ELBO objective (Eq. (4)). Each experiment approximated the posterior of a single layer at a time (assuming independence between layers) whilst keeping the rest of the network parameters at their deterministic full-precision values. We repeated each experiment for 5 random seeds, and report averages and standard deviations across seeds.

Table 1: **Ranking sensitivity at three bits in a ViT-base model.** Spearman correlation between each sensitivity criterion and change in accuracy under quantization of the indicated parameter group. Values are mean±std of correlations. Negative values indicate agreement with observed sensitivity. These are criterion comparisons, not separate quantization algorithms. See Table A9 for results at other quantization levels. *Note that the average Hessian traces for many of the layers were negative.

3-bit quantization Criterion	Block ($n = 7$)	Attention ($n = 7$)	MLP ($n = 7$)	Sublayer ($n = 42$)
QuIL (refined)	$\mathbf{-0.79}_{\pm 0.08}$	$\mathbf{-0.89}_{\pm 0.03}$	$\mathbf{-0.96}_{\pm 0.04}$	$-0.39_{\pm 0.04}$
→Laplace initialization	$\mathbf{-0.79}_{\pm 0.08}$	$\mathbf{-0.88}_{\pm 0.03}$	$\mathbf{-0.96}_{\pm 0.04}$	$-0.39_{\pm 0.04}$
EF-trace score	$-0.77_{\pm 0.13}$	$\mathbf{-0.87}_{\pm 0.04}$	$-0.91_{\pm 0.07}$	$-0.37_{\pm 0.05}$
EF trace only	$\mathbf{-0.80}_{\pm 0.13}$	$-0.78_{\pm 0.05}$	$\mathbf{-0.96}_{\pm 0.04}$	$\mathbf{-0.41}_{\pm 0.04}$
HAWQ-style score*	$+0.15_{\pm 0.75}$	$-0.03_{\pm 0.75}$	$+0.38_{\pm 0.55}$	$+0.18_{\pm 0.46}$
Quant. error	$+0.66_{\pm 0.09}$	$-0.05_{\pm 0.03}$	$+0.93_{\pm 0.04}$	$+0.39_{\pm 0.04}$

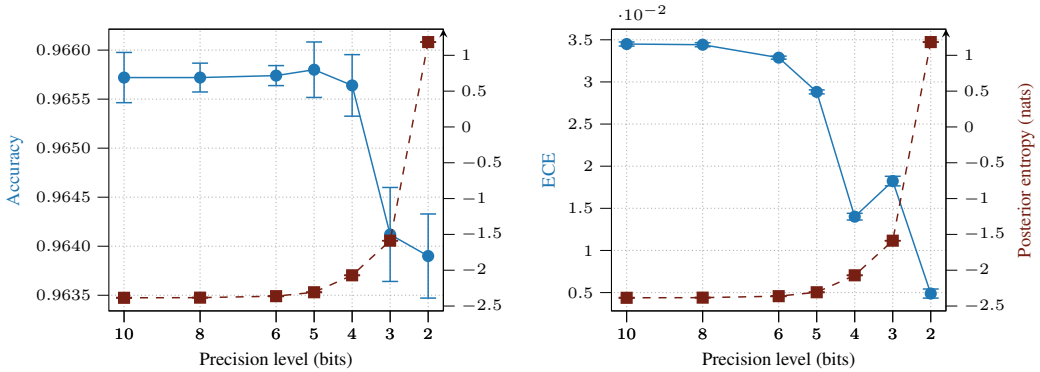

Figure 7: **BitVI entropy tracks quantization degradation in ViT's last layer.** For a ViT-base backbone with a fine-tuned CIFAR-10 classifier head, the posterior entropy used in QuIL (red, right axis) is evaluated from the curvature-initialized BitVI posterior at each bit-width. As precision is reduced, the entropy change remains small until the same low-bit regime where accuracy and calibration begin to change, indicating that QuIL captures the onset of representation-level loss.

For last-layer experiments, predictive performance was measured as the Bayesian model average (BMA) of accuracy, NLPD and ECE (see Sec. A.7 for details). Fig. 7 qualitatively shows how QuIL tracks the onset of predictive degradation in ViT, and Fig. A11 shows the same for GPT-2. See Table A8 for further experiment details, and Sec. A.9 for further last-layer results.

For inner-layer experiments, we focused on the ViT model fine-tuned on CIFAR-10. In total, ViT is composed of seven blocks, each containing six separate layers: four for attention (referred to as attention group, 4×590k parameters each) and two for feed-forward (referred to as MLP group, 2×2.6M parameters each). At evaluation time, all other layers were kept at full precision. Evaluations that aggregate multiple layers (block, attention or MLP), the same precision was applied uniformly to all group parameters. We assume additivity for QuIL across layers, and hence it was averaged across layer QuIL scores. To ensure fair comparison of methods in Table 1, all models were compared against performance metrics computed from the shared starting point, the deterministic model. Table 1 and Fig. 8 show the results of the experiment, with further results provided in Sec. B.4. The results show that QuIL both accurately predicts and is the most consistent metric for predicting the effect of quantization, and that it can be used to identify the most sensitive layers in a transformer model.

5 Discussion and Conclusion

We introduced QuIL as a representation-level view of quantization. The central claim is not that entropy should replace downstream evaluation, but that quantization removes information before this loss is observed through task performance. By placing a tractable posterior over floating-point bit-string representations, we can measure this loss as an entropy change across bit-widths. This turns finite precision from an implementation detail into an explicit modelling object and sheds light on what is happening. QuIL should thus be seen as a unifying view rather than yet another metric.

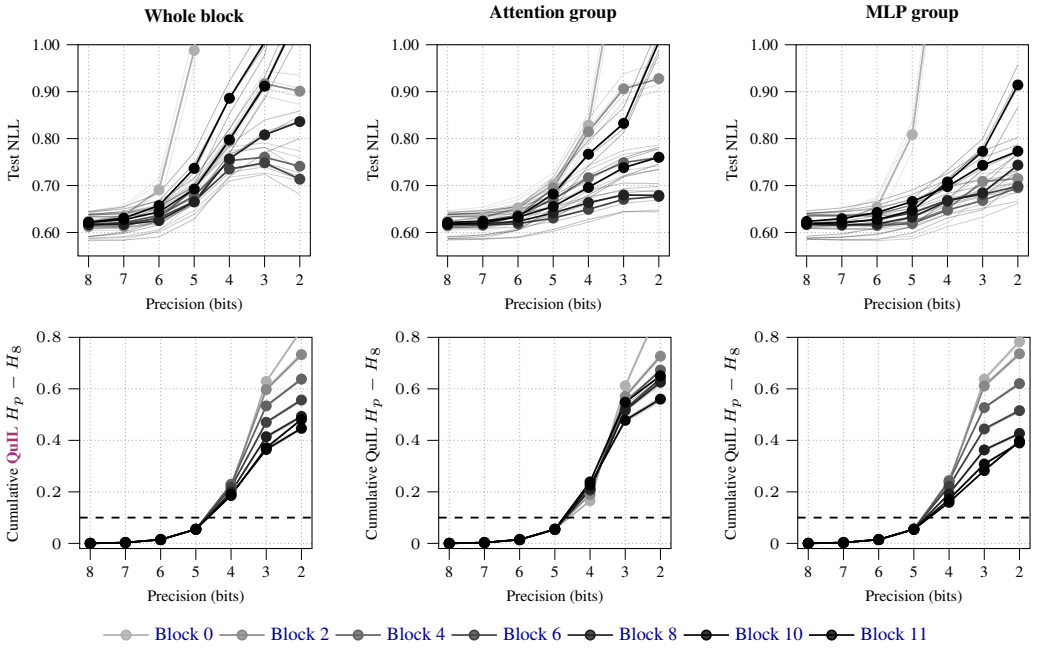

Figure 8: Visualizing the test set negative log-likelihood (top row) and the cumulative **QuIL** for multiple blocks and block components at different bit-widths. Cumulative **QuIL** is effectively the change in entropy from the full-precision model (in this case 8-bit) to the quantized model. The dashed line indicates a hypothetical information loss threshold τ, that could be used to automatically determine optimal quantization bit-width.

This view also clarifies existing quantization criteria. Hessian-based methods such as HAWQ, HAWQ-V2, OBQ, and GPTQ can be read as local quadratic approximations to the same principle: they replace posterior information loss by a curvature-weighted perturbation or reconstruction error. **QuIL** keeps the probabilistic object explicit. The transformer experiments (Sec. 4.3) follow this link directly, using a last-layer Laplace approximation to initialize the bit-string posterior before measuring the entropy change induced by quantization.

Empirically, the results support the modelling assumption. Floating-point BitVI behaves sensibly on regression benchmarks, and full-network quantization shows that **QuIL** changes in the same low-precision regime where RMSE begins to degrade. In pre-trained ViT and GPT-2 models, the curvature-initialized posterior gives the same type of signal for accuracy, calibration, and predictive density. These results suggest that loss of posterior structure in the parameter representation captures a meaningful component of quantization-induced functional degradation. The perspective is closely aligned with probabilistic numerics: numerical representation is not neutral, but constrains the uncertainty that can be expressed about model parameters (Hennig et al., 2015; Cockayne et al., 2019; Hennig et al., 2022). For quantization, this means that the choice of number format and bit-width is part of the model, not merely part of its deployment.

Limitations & future work The current implementation is still limited by scale. For each parameter θ in the model, the deterministic circuit required for **QuIL** has $2^{B+1} - 1$ variational parameters that must be stored and operated on. As such, our method utilizes a mean-field approximation of the posterior, which limits its ability to accurately capture its true functional form. This is, however, not a mandatory assumption and deterministic circuits that share variational parameters between model parameters will be investigated in future work to address this limitation. Future work will also investigate the use of **QuIL** as a metric for guiding targeted *mixed-precision* quantization. In particular, we draw inspiration from Van Baalen et al. (2020) who also propose a Bayesian approach for quantization, and elegantly unify quantization and pruning. Another interesting avenue of research is the investigation of specialized number representations for deep learning, such as bfloats (Wang & Kanwar, 2019) and FP8 (Micikevicius et al., 2022) via **QuIL**.

AI USE STATEMENT

In this work, we used generative AI tools **for assistance in, but not the automation of,** the tasks including: brainstorming, searching for information, implementing methods, creating and editing software code, identifying relevant literature, editing the research paper to improve readability, assisting in modification of scientific figures or images, assisting in the writing of proofs, analyzing existing literature, assisting in refining mathematical claims, and providing feedback on research methodology and experiments.

We have not used generative AI tools for the tasks of: proposing or refining hypotheses, drafting parts of a research paper, formatting references, suggesting a structure for a research paper, suggesting experimental parameters, proposing a title or keywords for the research paper, discovering research topics or identifying gaps, interpreting results or providing critical ingredients for proving mathematical claims.

The tasks of generating synthetic data sets, translation, cleaning and reformatting data sets, supporting qualitative and thematic data analysis, formulating questions for surveys or interviews, creation of artifacts, transcribing recordings of research material are not applicable to this work.

We have extensively reviewed all AI-assisted work. At least one author has reviewed all lines of code used in the research work that was created by generative AI. Mathematical claims, proofs or derivations created with the assistance of AI tools have been verified by at least two authors. We have not included any claims suggested by AI tools without independent verification of the source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not raise any major ethical concerns. The proposed method is a tool for studying machine learning models, and we do not foresee any possible methods for misuse. Improving quantization methods has the potential to reduce memory usage of machine learning models and allowing more efficient computations, which may translate to reduced energy consumption. It may also translate to simply making larger models that use the same amount of energy, or using even more energy by enabling models to be used on a larger variety of machines.

REPRODUCIBILITY STATEMENT

All code and data used in this work will be made available upon publication. Scripts to reproduce all experiments and figures presented in the paper will be included, along with instructions for setting up the necessary environment.

REFERENCES

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, abs/1607.06450, 2016.

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. ACIQ: Analytical clipping for integer quantization of neural networks, 2019.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, University of California, Los Angeles (UCLA), 2020.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Jon Cockayne, Chris J Oates, Timothy John Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Ltd, 2006.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. `LLM.int8()`: 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 30318–30332. Curran Associates, Inc., 2022.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 293–302, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ-v2: Hessian aware trace-weighted quantization of neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 18518–18529. Curran Associates, Inc., 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 4475–4488, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034. IEEE Computer Society, 2015.

Philipp Hennig, Michael A Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

Philipp Hennig, Michael A Osborne, and Hans P Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

IEEE. IEEE standard for floating-point arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository. https://archive.ics.uci.edu, 2025.

Donald E Knuth. *The Art of Computer Programming: Fundamental Algorithms, Volume 1*. Addison-Wesley Professional, 1997.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024.

Lorenzo Loconte, Aleksanteri M Sladek, Stefan Mengel, Martin Trapp, Arno Solin, Nicolas Gillis, and Antonio Vergari. Subtractive mixture models via squaring: Representation and learning. In *12th International Conference on Learning Representations (ICLR)*, 2024.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pp. 3288–3298. Curran Associates, Inc., 2017.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. FP8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1334, 2019.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 7197–7206. PMLR, 2020.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Hugh Salimbeni. Bayesian benchmarks, 2018. URL https://github.com/hughsalimbeni/bayesian_benchmarks.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Aleksanteri Sladek, Martin Trapp, and Arno Solin. Approximate Bayesian inference via bitstring representations. In *Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 286 of *Proceedings of Machine Learning Research*, pp. 3939–3957. PMLR, 2025.

Pat H. Sterbenz. *Floating-point Computation*. Prentice-Hall, 1974.

Mart Van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 5741–5752. Curran Associates, Inc., 2020.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 13189–13201. Curran Associates, Inc., 2021.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates Inc., 2019a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019b.

Shibo Wang and Pankaj Kanwar. Bfloat16: The secret to high performance on cloud TPUs. *Google Cloud Blog*, 4(1), 2019.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023.

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, and Kurt Keutzer. HAWQ-V3: Dyadic neural network quantization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11875–11886. PMLR, 2021.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, volume 35, pp. 27168–27183, 2022.

Zhongjie Yu, Martin Trapp, and Kristian Kersting. Characteristic circuits. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pp. 34074–34086, 2023.

Arnold Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42 (4):278–280, 1988.

APPENDICES

The appendices detail the bit-string representation and floating-point BitVI circuits, including derivations of entropy, cross-entropy, and KL divergence. We also provide evaluation metrics, experimental settings, additional regression and transformer results, and a tractable toy model for assessing the effects of posterior approximation on **QuIL**.

LIST OF APPENDICES

A Definitions, derivations, and examples **15**

A.1 A probabilistic model of bit-strings 15

A.2 Floating-point bit-string representation in BitVI 17

A.3 BitVI: Probabilistic circuits over bit-strings 18

A.4 Connecting discrete and differential entropy in BitVI 20

A.5 Cross-entropy between BitVI circuits 21

A.6 KL divergence between two BitVI circuits 21

A.7 Performance metric definitions 24

A.8 Additional UCI experiment results 24

A.9 Additional ViT and GPT-2 last-layer experiment results 24

B Additional experiment setup details **28**

B.1 UCI experiment hyperparameters 28

B.2 ViT and GPT-2 experiment hyperparameters 28

B.3 Computational resources 28

B.4 Additional ViT-CIFAR-10 inner-layer experiment results 30

B.5 An exactly-solvable discrete toy model 34

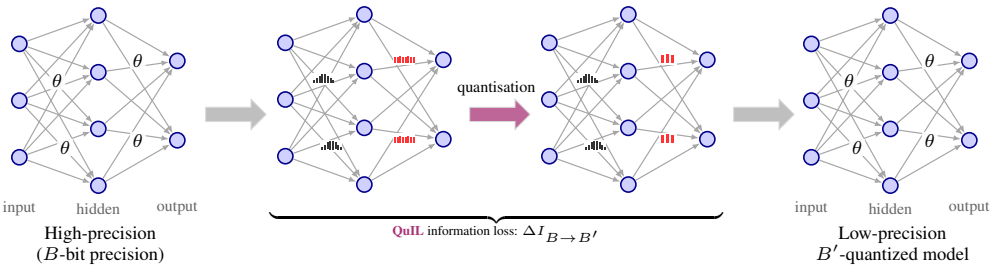

Figure A9: **Quantization as information loss in bit-string space.** **QuIL** models the bit-strings encoding model weights and measures the entropy change induced by reducing precision from B to B' bits. Peaked posteriors ( ) and near-uniform posteriors ( ) respond differently to quantization, giving $\Delta I_{B \to B'}$ as a representation-level loss measure.

A DEFINITIONS, DERIVATIONS, AND EXAMPLES

A.1 A PROBABILISTIC MODEL OF BIT-STRINGS

Let $B_{1:n}$ represent a set of random variables called a *bit-string*, which is defined as the ordered sequence of n binary random variables

$$B_{1:n} = (B_1, B_2, B_3, \ldots, B_n), \quad B_i \in \{0, 1\}.$$

As is typical for random variable notation, we represent values of bits B_i in lowercase as b_i, and a bit-string value as $\mathbf{b}$.

Furthermore, let $P(B_{1:n})$ represent the joint-distribution of $B_{1:n}$,

$$P(B_{1:n}) = P(B_1, B_2, \ldots, B_n) \tag{9}$$

Via the chain-rule of probability, this can be factorized into

$$\begin{aligned} P(B_{1:n}) &= P(B_n \mid B_{1:(n-1)})\, P(B_{1:(n-1)}) \\ &= P(B_n \mid B_{1:(n-1)})\, P(B_{n-1} \mid B_{1:(n-2)})\, P(B_{1:(n-2)}) \\ &= P(B_n \mid B_{1:(n-1)})\, P(B_{n-1} \mid B_{1:(n-2)}) \cdots P(B_2 \mid B_1)\, P(B_1) \\ &= \prod_{k=1}^{n} P(B_k \mid B_{1:(k-1)}), \quad \text{where } P(B_1 \mid B_{1:0}) = P(B_1). \end{aligned} \tag{10}$$

The joint distribution can also be bisected into a prefix and suffix term, where the prefix is defined as $B_{1:k} = (B_1, B_2, \ldots, B_k)$ and the suffix is defined as $B_{(k+1):n} = (B_{k+1}, B_{k+2}, \ldots, B_n)$ for some $k < n$. Together they form,

$$B_{1:n} = B_{1:k} \cup B_{(k+1):n}. \tag{11}$$

By the definition of the joint distribution, this can also be defined in terms of the joint distribution of the prefix $P(B_{1:k})$ and the conditional distribution of the suffix given the prefix $P(B_{(k+1):n} \mid B_{1:k})$ as,

$$P(B_{1:n}) = P(B_{(k+1):n} \mid B_{1:k})\, P(B_{1:k}). \tag{12}$$

Entropy of a bit-string The joint entropy of the bit-string $\mathbf{b} \sim P(B_{1:n})$ is defined as

$$\mathrm{H}\left[B_{1:n}\right] = \mathrm{H}\left[B_1, B_2, \ldots, B_n\right], \tag{13}$$

and, analogous to the chain-rule of probability, the joint entropy decomposes into

$$\begin{aligned} \mathrm{H}\left[B_{1:n}\right] &= \mathrm{H}\left[B_n \mid B_{1:(n-1)}\right] + \mathrm{H}\left[B_{1:(n-1)}\right] \\ &= \mathrm{H}\left[B_n \mid B_{1:(n-1)}\right] + \mathrm{H}\left[B_{n-1} \mid B_{1:(n-2)}\right] + \mathrm{H}\left[B_{1:(n-2)}\right] \\ &= \mathrm{H}\left[B_n \mid B_{1:(n-1)}\right] + \mathrm{H}\left[B_{n-1} \mid B_{1:(n-2)}\right] + \cdots + \mathrm{H}\left[B_2 \mid B_1\right] + \mathrm{H}\left[B_1\right] \\ &= \sum_{k=1}^{n} \mathrm{H}\left[B_k \mid B_{1:(k-1)}\right], \quad \text{where } \mathrm{H}\left[B_1 \mid B_{1:0}\right] = \mathrm{H}\left[B_1\right]. \end{aligned} \tag{14}$$

Furthermore, the entropy of the bit-string can be similarly bisected as follows,

$$\mathrm{H}\left[B_{1:n}\right] = \mathrm{H}\left[B_{1:k}\right] + \mathrm{H}\left[B_{(k+1):n} \mid B_{1:k}\right], \tag{15}$$

where $\mathrm{H}[B_{(k+1):n} \mid B_{1:k}]$ is the **conditional entropy** of the suffix given the prefix. This can be interpreted as the information remaining in the suffix given that the prefix is known.

Hence, the information lost by discarding the suffix $B_{(k+1):n}$ is given by the conditional entropy, which itself can be computed as

$$\mathrm{H}\left[B_{(k+1):n} \mid B_{1:k}\right] = \mathrm{H}\left[B_{1:n}\right] - \mathrm{H}\left[B_{1:k}\right], \tag{16}$$

that is, **the conditional entropy of the suffix given the prefix is a difference of entropies** between the full bit-string and its k-bit prefix.

Connection to information gain This is exactly the quantity that **QuIL**'s Shannon entropy component is computing, *i.e.*,

$$\mathrm{H}\left[B_{(k+1):n} \,|\, B_{1:k}\right] = \mathrm{H}\left[B_{1:n}\right] - \mathrm{H}\left[B_{1:k}\right], \tag{17}$$

A.2 FLOATING-POINT BIT-STRING REPRESENTATION IN BITVI

In this section, we detail the floating-point bit-string representation that a BitVI circuit encodes.

Let $\boldsymbol{b}$ represent a bit-string of length $|\boldsymbol{b}| = B$ with E exponent bits (and hence $M = B - E - 1$ mantissa bits) and is notated as $\boldsymbol{b} = (b_{B-1}, b_{B-2}, \ldots, b_0)$. Starting from the left, the b_{B-1} bit corresponds to the sign-bit, the following E bits $(b_{B-2}, \ldots, b_M)$ are the exponent bits, and the remainder of the bit-string $b_{M-1}, \ldots, b_0$ are the mantissa bits. The indexing highlighted was chosen so that when discarding the bit-string from right to left, the least significant bit is always discarded.

In Definition A.1 we define the mathematical relationship between a number θ and its bit-string representation $\boldsymbol{b}$.

Definition A.1 (The floating-point representation of a bit-string)**.** *(IEEE, 2019)*

$$\theta = f_{\textit{float}}(\mathbf{b}) = \underbrace{(-1)^{b_{B-1}}}_{\textit{sign}} \cdot \underbrace{\left[2^{(\sum_{i=M}^{B-2} b_i 2^{i-M}) - \kappa}\right]}_{\textit{exponent}} \cdot \underbrace{\left(1 + \sum_{j=0}^{M-1} b_j 2^{j-M}\right)}_{\textit{mantissa}} \tag{18}$$

In order to retain precision around zero, we also utilize the sub-normal special case, which is defined as

$$\theta = g_{\textit{sub-normal}}(\mathbf{b}) = \underbrace{(-1)^{b_{B-1}}}_{\textit{sign}} \cdot \underbrace{2^{1-\kappa}}_{\textit{exponent}} \cdot \underbrace{\left(\sum_{j=0}^{M-1} b_j 2^{j-M}\right)}_{\textit{mantissa}}, \tag{19}$$

For the purposes of this work, we exclude special bit-strings defined by IEEE (2019). We exclude the special bit-string that is reserved for representing $-\infty$ and ∞, which is encoded by all exponent bits being 1, all mantissa bits being zero, and varying the sign bit respectively. We also exclude the family of special bit-strings for NaN, which are encoded as sign bit 0 or 1, all exponent bits as 1 and one or more mantissa bits being 1. Furthermore, note that this representation entails a positive and negative zero. We interpret these bit-strings via Eq. (18) and Eq. (19).

A.3 BitVI: Probabilistic circuits over bit-strings

Definition A.2 (Recursive definition of the fixed-point BitVI probabilistic circuit)**.** *(Sladek et al., 2025)*

$$\mathcal{C}(\theta) = \mathsf{N}^{\epsilon}(\theta) = \begin{cases} \mathsf{S}_{\boldsymbol{b}}(\theta) = w_0^{\boldsymbol{b}} \mathsf{N}^{\boldsymbol{b}0}(\theta) + w_1^{\boldsymbol{b}} \mathsf{N}^{\boldsymbol{b}1}(\theta) & \text{if } \mathsf{N} \in \mathcal{N}(\mathcal{T}) \\ \mathsf{L}_{\boldsymbol{b}}(\theta) = U(a_{\boldsymbol{b}}, a_{\boldsymbol{b}} + \Delta_n) & \text{if } \mathsf{N} \in \mathcal{L}(\mathcal{T}) \end{cases} \tag{20}$$

In this work, $\mathcal{T}$ is always a *complete* binary tree of depth n: every sum node has two sum-node children except at depth $n-1$, whose children are leaves. Each sum node splits its interval into two equal halves, so *every* leaf interval has the same width

$$\Delta_n = 2^{-n} \, |\operatorname{supp}(\mathcal{C})|, \tag{21}$$

which is fixed for each leaf node, and depends only on the length of the bit-string n.

In this work, we extend Definition A.2 to the floating-point case, where the leaf intervals are no longer uniform fixed-widths. The sum nodes still split their parent interval into two halves, but the leaf intervals are now determined by the floating-point representation of the bit-string. This is the *de facto* representation for model parameter bit-strings, and allows for more resolution around zero.

Definition A.3 (Recursive definition of the floating-point BitVI probabilistic circuit)**.**

$$\mathcal{C}(\theta) = \mathsf{N}^{\epsilon}(\theta) = \begin{cases} \mathsf{S}_{\boldsymbol{b}}(\theta) = w_0^{\boldsymbol{b}} \mathsf{N}^{\boldsymbol{b}0}(\theta) + w_1^{\boldsymbol{b}} \mathsf{N}^{\boldsymbol{b}1}(\theta) & \text{if } \mathsf{N} \in \mathcal{N}(\mathcal{T}) \\ \mathsf{L}_{\boldsymbol{b}}(\theta) = U(a_{\boldsymbol{b}}, a_{\boldsymbol{b}} + \Delta_{\boldsymbol{b}}) & \text{if } \mathsf{N} \in \mathcal{L}(\mathcal{T}) \end{cases} \tag{22}$$

Unlike in the fixed-point case, the leaf width $\Delta_{\boldsymbol{b}}$ is no longer a single constant shared by every leaf. Instead, it varies leaf-to-leaf according to the scale of the floating-point grid at $\boldsymbol{b}$, which is determined by the exponent bits. Writing $e_{\boldsymbol{b}} = \sum_{i=M}^{B-2} b_i 2^{i-M}$ as the scaling factor encoded by the exponent bits of $\boldsymbol{b}$ (Eq. (18)), the mantissa bits of $\boldsymbol{b}$ divide the grid into 2^{-M} bins, so

$$\Delta_{\boldsymbol{b}} = \begin{cases} 2^{1-\kappa-M} & \text{if } e_{\boldsymbol{b}} = 0 \quad \text{(sub-normal, Eq. (19))} \\ 2^{e_{\boldsymbol{b}}-\kappa-M} & \text{if } e_{\boldsymbol{b}} \geq 1 \quad \text{(normal, Eq. (18))} \end{cases} \tag{23}$$

which depends on $\boldsymbol{b}$ only through its exponent bits, and is constant across the sign bit and the mantissa bits. Comparing Eq. (23) to Eq. (21): the fixed-point leaf width $\Delta_n = 2^{-n} \, |\operatorname{supp}(\mathcal{C})|$ applies a single global scale factor at every leaf, whereas the floating-point leaf width instead scales the fixed mantissa resolution 2^{-M} by a per-leaf, exponent-dependent factor $2^{e_{\boldsymbol{b}}-\kappa}$. In this sense, depth-dependence in Definition A.2 is replaced by exponent-dependence in Definition A.3: the tree still bisects each parent interval at every sum node, but the resulting leaves are a geometric bisection (geometric mean of the end-points) rather than a bisection into equal-width bins (arithmetic mean of the end-points).

Sampling process and bit-string notation. Sampling from $\mathcal{C}(\theta)$ generates a random bit-string

$$B_{1:n} = (B_1, \ldots, B_n), \qquad B_k \,|\, B_{1:k-1} \sim \text{Bernoulli}\big(w_1^{B_{1:k-1}}\big), \tag{24}$$

where bit B_k is drawn at the depth-k sum node reached by the prefix $B_{1:k-1}$ (with $B_{1:0} = \epsilon$ the root). The leaf indexed by $B_{1:n}$ then emits

$$\theta \,|\, B_{1:n} = \boldsymbol{b} \sim U\left(a_{\boldsymbol{b}},\, a_{\boldsymbol{b}} + \Delta_{\boldsymbol{b}}\right). \tag{25}$$

We follow the convention that uppercase B denotes the *random* bit-string generated by sampling, while lowercase $\boldsymbol{b}$ denotes a fixed bit-string used as a node address (the path from the root to $\mathsf{N}_{\boldsymbol{b}}$). The two are related through events: the possible values of the random prefix $B_{1:d}$ are exactly the depth-d node addresses, so $\{B_{1:d} = \boldsymbol{b}\}$ is the event that the sample's path passes through $\mathsf{N}_{\boldsymbol{b}}$, and the weight applied at step k of Eq. (24) is the fixed parameter $w_1^{\boldsymbol{b}}$ whose address $\boldsymbol{b} = B_{1:k-1}$ is selected at random by the path so far. For a node $\mathsf{N}_{\boldsymbol{b}}$ with $|\boldsymbol{b}| = d$, we write $B_{\boldsymbol{b}} = B_{d+1:n} \,|\, (B_{1:d} = \boldsymbol{b})$ for the *suffix* of the bit-string generated below $\mathsf{N}_{\boldsymbol{b}}$; in particular $B_\epsilon = B_{1:n}$, and $B_{\boldsymbol{b}}$ is empty (zero entropy) when $\mathsf{N}_{\boldsymbol{b}}$ is a leaf. Note that $B_{1:n}$ is in bijection with the quantization cells of $\mathcal{C}$, so writing X^{Δ} for the quantized version of X as in Definition A.5,

$$\mathrm{H}\left[B_{1:n}\right] = \mathrm{H}\left[X^{\Delta}\right]. \tag{26}$$

BitVI exact entropy For completeness, we restate the definition of the BitVI deterministic circuit entropy from Sladek et al. (2025). Please refer to the original paper for a more detailed discussion of the derivation and properties of this quantity.

Definition A.4 (Recursive Definition of BitVI Entropy)**.** *The entropy of a deterministic probabilistic circuit $\mathcal{C}$ is defined as*

$$\mathrm{H}\left[\mathcal{C}(\theta)\right] = \begin{cases} \mathrm{H}[\mathsf{S}_{\boldsymbol{b}}(\theta)] & \text{if } \mathsf{N} \in \mathcal{N}(\mathcal{T}) \\ \mathrm{H}[\mathsf{L}_{\boldsymbol{b}}(\theta)] & \text{if } \mathsf{N} \in \mathcal{L}(\mathcal{T}) \end{cases} \tag{27}$$

where N denotes the root node of $\mathcal{C}$. Furthermore,

$$\begin{aligned} \mathrm{H}\left[\mathsf{S}_{\boldsymbol{b}}(\theta)\right] = &-w_0^{\boldsymbol{b}} \log(w_0^{\boldsymbol{b}}) + w_0^{\boldsymbol{b}} \mathrm{H}\left[\mathsf{N}_{\boldsymbol{b}0}(\theta)\right] \\ &- w_1^{\boldsymbol{b}} \log(w_1^{\boldsymbol{b}}) + w_1^{\boldsymbol{b}} \mathrm{H}\left[\mathsf{N}_{\boldsymbol{b}1}(\theta)\right] \end{aligned} \tag{28}$$

and

$$\mathrm{H}\left[\mathsf{L}_{\boldsymbol{b}}(\theta)\right] = \log \Delta \tag{29}$$

where $\mathsf{N}_{\boldsymbol{b}0}$ and $\mathsf{N}_{\boldsymbol{b}1}$ are the child nodes of $\mathsf{N}_{\boldsymbol{b}}$ (in the case that $\mathsf{N}_{\boldsymbol{b}}$ is a sum node S). $\mathcal{N}(\mathcal{T})$ and $\mathcal{L}(\mathcal{T})$ denote the set of sum nodes and leaf nodes in the circuit, respectively.

Since sibling subtrees of $\mathcal{C}$ have disjoint supports, the circuit is deterministic, and the recursion in Definition A.4 computes the differential entropy of the density represented by $\mathcal{C}$ exactly (Vergari et al., 2021).

Optimizing BitVI BitVI circuit parameters are optimized w.r.t. the Evidence Lower BOund (ELBO) loss function. The ELBO decomposes into two terms: (1) a Monte Carlo estimate of the expected log-joint $\mathbb{E}_{\boldsymbol{\theta}\sim q_\phi}[\log p(\boldsymbol{\theta})]$, and (2) the entropy $H(q_\phi(\boldsymbol{\theta}))$.

Term (1) is estimated by drawing samples $\boldsymbol{\theta} \sim q_\phi$, made differentiable through an inverse-CDF reparameterization. We sample a CDF value $\mathbf{u} \sim U(0,1)$ and evaluate $\boldsymbol{\theta} = F^{-1}_{q_\phi}(\mathbf{u})$ by a single top-down pass through the circuit. At each sum node, the value of $\mathbf{u}$ selects a branch and applies the corresponding affine rescaling of $\mathbf{u}$ (Sladek et al., 2025, Eq. 6). On reaching a leaf, the rescaled value is passed through the leaf's uniform inverse-CDF to yield the final (continuous) sample. The whole map is a composition of affine operations in the sum-node weights, so it is differentiable and isolates randomness into u. Branch selection makes the map piecewise, but it is differentiable within each piece, so gradients flow through the selected path into the circuit weights. At this stage, the sample is still continuous and can be discretized to the closest bin-edge via rounding + the straight-through estimator to keep the operation differentiable. Alternatively, a continuous relaxation can be adopted and the continuous sample viewed as the discrete bin edge + random noise proportional to the bin-width, which is the approach we adopted for our experiments.

The resulting sample $\boldsymbol{\theta}$ is then evaluated under the model to obtain the log-likelihood, which for the NN models is differentiable by standard backpropagation.

Term (2) is available in closed form for a deterministic PC and is computed exactly (linear in the number of circuit edges), so its gradient w.r.t. the circuit parameters requires no sampling.

A.4 Connecting discrete and differential entropy in BitVI

Given that this paper deals with quantization of continuous random variables, that is, continuous random variables being transformed into discrete random variables, it is important to make explicit the relationship of the discrete entropy of a quantized random variable to the differential entropy of the continuous random variable.

Definition A.5 (Relationship between discrete entropy and differential entropy)**.** *(Cover & Thomas, 2006, Theorem 8.3.1) If the density $f(x)$ of the random variable X is Riemann integrable, then the discrete entropy $H(X^\Delta)$ of the quantized version of X, notated as X^Δ, is related to the differential entropy $h(X)$ as follows:*

$$H(X^\Delta) + \log \Delta \to h(X) \quad \text{as } \Delta \to 0 \tag{30}$$

Meaning that the discrete entropy of X^Δ converges to the differential entropy of X as the quantization step Δ approaches zero, and the n-bit quantization of continuous RV X is approximately $h(X) + n$ (Cover & Thomas, 2006, Equation 8.30).

Restated, the differential entropy of a continuous random variable X can be approximated by the discrete entropy of its quantized version X^Δ as follows:

$$h(X) \approx H(X^\Delta) + \log \Delta \tag{31}$$

A.4.1 BitVI circuit entropy decomposition

First recall that the density function of the circuit is given as: $p(x) = \frac{p(g^{-1}(x))}{\Delta_{g^{-1}(x)}}$ where g^{-1} maps from the real line to bit-strings $\boldsymbol{b}$ and is implicitly defined by the choice of number system representation. Here, $\Delta_{g^{-1}(x)} = \Delta_{\boldsymbol{b}}$ denotes the area under the uniform distribution at $\boldsymbol{b}$.

Now, computing the differential entropy with respect to the Lebesgue measure is given as:

$$\begin{aligned}
h(p(x)) &= -\int_{x\in\mathbb{R}} \frac{p(g^{-1}(x))}{\Delta_{g^{-1}(x)}} \log\left(\frac{p(g^{-1}(x))}{\Delta_{g^-1(x)}}\right) \mathrm{d}x && (32)\\
&= -\int_{x\in\mathbb{R}} \frac{p(g^{-1}(x))}{\Delta_{g^{-1}(x)}} \log p(g^{-1}(x))\mathrm{d}x + \int_{x\in\mathbb{R}} \frac{p(g^{-1}(x))}{\Delta_{g^{-1}(x)}} \log \Delta_{g^-1(x)}\mathrm{d}x && (33)\\
&= -\sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log p(\boldsymbol{b}) \underbrace{\int_{a_{\boldsymbol{b}}}^{a_{\boldsymbol{b}}+\Delta_{\boldsymbol{b}}} {}^1\!/_{\Delta_{g^-1(x)}}\mathrm{d}x}_{=1} + \sum_{\boldsymbol{b}} p(\boldsymbol{b}) \underbrace{\int_{a_{\boldsymbol{b}}}^{a_{\boldsymbol{b}}+\Delta_{\boldsymbol{b}}} \frac{\log \Delta_{g^-1(x)}}{\Delta_{g^-1(x)}}\mathrm{d}x}_{=\log \Delta_{\boldsymbol{b}}} && (34)\\
&= -\sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log p(\boldsymbol{b}) + \sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log \Delta_{\boldsymbol{b}}, && (35)
\end{aligned}$$

where the second step is because $g^-1(x) = \boldsymbol{b}$ for every $x \in [a_{\boldsymbol{b}}, a_{\boldsymbol{b}} + \Delta_{\boldsymbol{b}}]$.

Entropy for fixed-point BitVI For fixed-point representations, each quantization cell is of equal width, $\Delta_{\boldsymbol{b}} = \Delta_n$. In terms of BitVI's leaf nodes, the uniform distribution width at every leaf node is the same. Consequently, the above equation further simplifies,

$$\begin{aligned}
h(p(x)) &= -\sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log p(\boldsymbol{b}) + \sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log \Delta_n \\
&= -\sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log p(\boldsymbol{b}) + \log \Delta_n \underbrace{\sum_{\boldsymbol{b}} p(\boldsymbol{b})}_{=1} \\
&= -\sum_{\boldsymbol{b}} p(\boldsymbol{b}) \log p(\boldsymbol{b}) + \log \Delta_n,
\end{aligned} \tag{36}$$

which is effectively equivalent to Eq. (31).

Entropy for floating-point BitVI We strongly suspect that the decomposition of the floating-point BitVI circuits can also be simplified similarly, given the regularity of the floating-point partitioning function. Showing this equivalence however remains future work.

A.5 Cross-entropy between BitVI circuits

In this section we derive the cross-entropy between two structurally identical BitVI circuits, denoted as $\mathcal{C}^P(x)$ and $\mathcal{C}^Q(x)$.

Note that this is not a novel contribution but rather a straightforward application of already known properties of deterministic probabilistic circuits. See (Vergari et al., 2021) for more details on the properties of deterministic PCs. This section is included for clarity and completeness.

Recall that the BitVI circuit structure is a full binary tree of sum nodes and leaf nodes. Each circuit root is a weighted sum of child nodes: $\mathcal{C}^P(x) = \sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x)$. The cross-entropy is:

$$\int_{-\infty}^{\infty} \mathcal{C}^P(x) \log(\mathcal{C}^Q(x))\,\mathrm{d}x = \int_{-\infty}^{\infty} \left(\sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x)\right) \log\left(\sum_{j=1}^{2} w_j^Q \mathsf{C}_j^Q(x)\right)\,\mathrm{d}x \tag{37}$$

Because the circuits satisfy the **determinism** property (meaning the supports of all child distributions at tree depth d are mutually exclusive), the logarithm of the sum simplifies to the sum of the logarithms for non-zero terms:

$$= \sum_{i=1}^{2} \int_{-\infty}^{\infty} w_i^P \mathsf{C}_i^P(x) \log(w_i^Q \mathsf{C}_i^Q(x))\,\mathrm{d}x \tag{38}$$

Expanding the logarithm inside the integral yields:

$$= \sum_{i=1}^{2} \int_{-\infty}^{\infty} w_i^P \mathsf{C}_i^P(x)[\log(w_i^Q) + \log(\mathsf{C}_i^Q(x))]\,\mathrm{d}x \tag{39}$$

Distributing the terms and pulling constants out of the integrals:

$$= \sum_{i=1}^{2} \left[w_i^P \log(w_i^Q) \int_{-\infty}^{\infty} \mathsf{C}_i^P(x)\,\mathrm{d}x + w_i^P \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^Q(x))\,\mathrm{d}x \right] \tag{40}$$

Because $\mathsf{C}_i^P(x)$ is a normalized probability distribution, its integral over all space equals 1 (*i.e.*, $\int \mathsf{C}_i^P(x)dx = 1$). Substituting this back in gives the final simplified cross-entropy:

$$= \sum_{i=1}^{2} w_i^P \log(w_i^Q) + \sum_{i=1}^{2} w_i^P \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^Q(x))\,\mathrm{d}x \tag{41}$$

This decomposition can then be continued recursively down the tree structure for each new cross-entropy term to the child node pair $\mathsf{C}_i^P(x)$ and $\mathsf{C}_i^Q(x)$ until the leaf nodes are reached.

A.6 KL divergence between two BitVI circuits

Decomposing the KL

The derivation begins with the standard definition of Kullback-Leibler (KL) divergence between two probability distributions $p(x)$ and $q(x)$:

$$D_{\mathrm{KL}}(p \,\|\, q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x \tag{42}$$

Using the properties of logarithms, this expands to:

$$= \int_{-\infty}^{\infty} p(x)[\log(p(x)) - \log(q(x))]\,\mathrm{d}x \tag{43}$$

Splitting the integral gives us two distinct terms:

$$= \int_{-\infty}^{\infty} p(x) \log(p(x)) \, \mathrm{d}x - \int_{-\infty}^{\infty} p(x) \log(q(x)) \, \mathrm{d}x \tag{44}$$

The first term is the negative differential entropy of $p(x)$, $-\mathbb{H}(p) = -\mathbb{E}_{p(x)}[\log(p(x))]$. The second term is the expected cross-entropy between $p(x)$ and $q(x)$, $\mathbb{E}_{p(x)}[\log(q(x))]$.

KL DIVERGENCE BETWEEN TWO CIRCUITS

In this section we derive the KL divergence between two structurally identical BitVI circuits, denoted as $\mathcal{C}^P(x)$ and $\mathcal{C}^Q(x)$.

Note that this is not a novel contribution but rather a straightforward application of already known properties of deterministic probabilistic circuits. See (Vergari et al., 2021) for more details on the properties of deterministic PCs. This section is included for clarity and completeness.

$$D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) = \int_{-\infty}^{\infty} \mathcal{C}^P(x) \log \mathcal{C}^P(x) \, \mathrm{d}x - \int_{-\infty}^{\infty} \mathcal{C}^P(x) \log(\mathcal{C}^Q(x)) \, \mathrm{d}x \tag{45}$$

This is expanded out with the definitions of $\mathcal{C}^P$ and $\mathcal{C}^Q$, recalling that the circuit structure in this paper is a full binary tree of sum nodes and leaf distribution nodes. Each circuit root is a weighted sum of child nodes: $\mathcal{C}^P(x) = \sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x)$. The KL divergence is then:

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) = &\int_{-\infty}^{\infty} \sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x) \log \left(\sum_{j=1}^{2} w_j^P \mathsf{C}_j^P(x) \right) \mathrm{d}x \\ &- \int_{-\infty}^{\infty} \sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x) \log \left(\sum_{j=1}^{2} w_j^Q \mathsf{C}_j^Q(x) \right) \mathrm{d}x \end{aligned} \tag{46}$$

Using the formula for the entropy derived in Sladek et al. (2025), the first term simplifies to:

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) = &\sum_{i=1}^{2} \left[w_i^P \log(w_i^P) + w_i^P \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^P(x)) \, \mathrm{d}x \right] \\ &- \int_{-\infty}^{\infty} \sum_{i=1}^{2} w_i^P \mathsf{C}_i^P(x) \log \left(\sum_{j=1}^{2} w_j^Q \mathsf{C}_j^Q(x) \right) \mathrm{d}x \end{aligned} \tag{47}$$

Furthermore, using the derivation for the cross-entropy between two circuits from Eq. (41), the second term simplifies to:

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) = &\sum_{i=1}^{2} \left[w_i^P \log(w_i^P) + w_i^P \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^P(x)) \, \mathrm{d}x \right] \\ &- \sum_{i=1}^{2} \left[w_i^P \log(w_i^Q) + w_i^P \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^Q(x)) \, \mathrm{d}x \right] \end{aligned} \tag{48}$$

Taking w_i^P as a common factor,

$$= \sum_{i=1}^{2} w_i^P \left[\log(w_i^P) + \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^P(x)) \, \mathrm{d}x - \log(w_i^Q) - \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^Q(x)) \, \mathrm{d}x \right]. \tag{49}$$

Grouping terms and simplifying,

$$
\begin{aligned}
D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) &= \sum_{i=1}^{2} w_i^P \left[\log(w_i^P) - \log(w_i^Q)\right] \\
&\quad + \sum_{i=1}^{2} w_i^P \left[\int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^P(x))\,\mathrm{d}x - \int_{-\infty}^{\infty} \mathsf{C}_i^P(x) \log(\mathsf{C}_i^Q(x))\,\mathrm{d}x\right] \\
&= \sum_{i=1}^{2} w_i^P \log\left(\frac{w_i^P}{w_i^Q}\right) + w_i^P D_{\mathrm{KL}}(\mathsf{C}_i^P \,\|\, \mathsf{C}_i^Q).
\end{aligned} \tag{50}
$$

This can then be recursively applied down the tree structure for each child node pair's KL divergence until the leaf nodes are reached. Written compactly in a recursive formula:

$$
D_{\mathrm{KL}}[\mathsf{N}^P(x) \,\|\, \mathsf{N}^Q(x)] = \begin{cases} \sum_{i=1}^{2} w_i^P \log\left(\frac{w_i^P}{w_i^Q}\right) + w_i^P D_{\mathrm{KL}}[\mathsf{C}_i^P \,\|\, \mathsf{C}_i^Q] & \text{if } \mathsf{N} \text{ is a sum node} \\ 0 & \text{if } \mathsf{N} \text{ is a leaf node} \end{cases} \tag{51}
$$

EXAMPLE: KL DIVERGENCE BETWEEN TWO SIMPLE CIRCUITS WITH UNIFORM LEAF DISTRIBUTIONS

Consider a simple case where both circuits, $\mathcal{C}^P$ and $\mathcal{C}^Q$, consist of a single root sum node with exactly two leaf nodes. The structural definitions for the circuits are:

$$
\mathcal{C}^P(x) = w_1^P \mathsf{L}_1^P(x) + w_2^P \mathsf{L}_2^P(x) \tag{52}
$$

$$
\mathcal{C}^Q(x) = w_1^Q \mathsf{L}_1^Q(x) + w_2^Q \mathsf{L}_2^Q(x) \tag{53}
$$

Based on the derivations above, the KL divergence is the difference between the negative entropy of $\mathcal{C}^P$ and the cross-entropy between $\mathcal{C}^P$ and $\mathcal{C}^Q$. By substituting $\mathcal{C}^Q$ with $\mathcal{C}^P$ in our cross-entropy equation, the negative entropy of $\mathcal{C}^P$ is:

$$
\int_{-\infty}^{\infty} \mathcal{C}^P(x) \log(\mathcal{C}^P(x))\,\mathrm{d}x = \sum_{i=1}^{2} w_i^P \log(w_i^P) + \sum_{i=1}^{2} w_i^P \int_{-\infty}^{\infty} \mathsf{L}_i^P(x) \log(\mathsf{L}_i^P(x))\,\mathrm{d}x \tag{54}
$$

Subtracting the cross-entropy from this negative entropy yields the full KL divergence:

$$
\begin{aligned}
D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) &= \sum_{i=1}^{2} \left[w_i^P \log(w_i^P) + w_i^P \int_{-\infty}^{\infty} \mathsf{L}_i^P(x) \log(\mathsf{L}_i^P(x))\,\mathrm{d}x\right] \\
&\quad - \sum_{i=1}^{2} \left[w_i^P \log(w_i^Q) + w_i^P \int_{-\infty}^{\infty} \mathsf{L}_i^P(x) \log(\mathsf{L}_i^Q(x))\,\mathrm{d}x\right]
\end{aligned} \tag{55}
$$

By factoring out w_i^P and grouping the discrete weight terms and the continuous integral terms together, we get:

$$
D_{\mathrm{KL}}(\mathcal{C}^P \,\|\, \mathcal{C}^Q) = \sum_{i=1}^{2} w_i^P \left(\log(w_i^P) - \log(w_i^Q)\right) + \sum_{i=1}^{2} w_i^P \int_{-\infty}^{\infty} \mathsf{L}_i^P(x) \left(\log(\mathsf{L}_i^P(x)) - \log(\mathsf{L}_i^Q(x))\right)\,\mathrm{d}x \tag{56}
$$

This cleanly simplifies into a sum of the KL divergence of the sum node weights and the weighted KL divergences of the respective leaf distributions:

$$
D_{\mathrm{KL}}[\mathcal{C}^P \,\|\, \mathcal{C}^Q] = \sum_{i=1}^{2} w_i^P \log\left(\frac{w_i^P}{w_i^Q}\right) + \sum_{i=1}^{2} w_i^P D_{\mathrm{KL}}[\mathsf{L}_i^P \,\|\, \mathsf{L}_i^Q] \tag{57}
$$

Since the leaf distributions for both circuits match exactly ($\mathsf{L}_i^P = \mathsf{L}_i^Q$), the KL divergence between the corresponding leaves becomes zero ($D_{\mathrm{KL}}[\mathsf{L}_i^P \,\|\, \mathsf{L}_i^Q] = 0$).

The final formula reduces entirely to the KL divergence of the sum node weights:

$$
D_{\mathrm{KL}}[\mathcal{C}^P \,\|\, \mathcal{C}^Q] = w_1^P \log\left(\frac{w_1^P}{w_1^Q}\right) + w_2^P \log\left(\frac{w_2^P}{w_2^Q}\right) \tag{58}
$$

A.7 PERFORMANCE METRIC DEFINITIONS

Let $D = \{(x_i, y_i)\}_{i=1}^N$ be a data set of N input-output pairs, and let $p(\theta|D)$ be the posterior distribution over model parameters θ given the data. We are interested in evaluating the predictive performance of our model under the posterior distribution over parameters. Computing the expectation w.r.t a parameter posterior distribution is generally intractable in our setting, and hence will be approximated via Monte Carlo (MC) sampling. Let $\{\theta^{(s)}\}_{s=1}^S$ be S samples where $\theta^{(s)} \sim p(\theta|D)$.

The Expected Log-Likelihood (ELL) is defined as:

$$\text{NLL} = \mathbb{E}_{p(\theta \mid D)} \left[\sum_{i=1}^{N} \log(p(y_i|\theta)) \right] \tag{59}$$

which is approximated via MC sampling as:

$$\widehat{\text{NLL}} = \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{N} \log(p(y_i|\theta^{(s)})) \tag{60}$$

The Negative Log Point-wise Predictive Density (NLPD) is defined as:

$$\text{NLPD} = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\mathbb{E}_{p(\theta \mid D)}[p(y_i|\theta)] \right) \tag{61}$$

which is approximated via MC sampling as:

$$\widehat{\text{NLPD}} = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{1}{S} \sum_{s=1}^{S} p(y_i|\theta^{(s)}) \right) \tag{62}$$

A.8 ADDITIONAL UCI EXPERIMENT RESULTS

Table A2: Average testing set RMSE scores ($\pm$ s.d.) from a 5-fold CV experiment on UCI benchmark data sets. Methods are bolded based on a paired t-test ($p = 5\%$) with the best performing method, and show that floating-point BitVI performs similarly or better than fixed-point BitVI.

Dataset	(n, d)	10-bit BITVI (fixed)	10-bit BITVI (float)	6-bit BITVI (fixed)	6-bit BITVI (float)
YACHT	(308, 6)	$\mathbf{1.7797 \pm 0.6636}$	$\mathbf{1.7018 \pm 0.6175}$	2.0008 ± 0.5833	$\mathbf{1.9141 \pm 0.6273}$
KIN8NM	(8192, 8)	$\mathbf{0.0755 \pm 0.0030}$	$\mathbf{0.0781 \pm 0.0022}$	0.1177 ± 0.0037	0.1190 ± 0.0026
BOSTON	(506, 13)	$\mathbf{3.4884 \pm 0.4263}$	$\mathbf{3.4529 \pm 0.3189}$	$\mathbf{3.4500 \pm 0.2917}$	$\mathbf{3.7481 \pm 0.4802}$
ENERGY	(768, 8)	$\mathbf{0.8836 \pm 0.0476}$	$\mathbf{0.9750 \pm 0.0575}$	1.3458 ± 0.0858	1.5155 ± 0.1149
PROTEIN	(45730, 9)	$\mathbf{4.3651 \pm 0.0359}$	4.4250 ± 0.0395	4.8528 ± 0.0394	4.9367 ± 0.0308
POWER	(9568, 4)	$\mathbf{4.0617 \pm 0.0716}$	$\mathbf{4.0783 \pm 0.0889}$	4.3936 ± 0.0860	4.3988 ± 0.1080
CONCRETE	(1030, 8)	$\mathbf{5.9547 \pm 0.1760}$	$\mathbf{6.0300 \pm 0.3611}$	6.6486 ± 0.5015	6.7065 ± 0.4417
WINERED	(1599, 11)	$\mathbf{0.6291 \pm 0.0368}$	$\mathbf{0.6321 \pm 0.0404}$	$\mathbf{0.6322 \pm 0.0371}$	$\mathbf{0.6364 \pm 0.0346}$
NAVAL	(11934, 14)	$\mathbf{0.0012 \pm 0.0001}$	0.0018 ± 0.0002	0.0095 ± 0.0003	0.0096 ± 0.0003
WINEWHITE	(4898, 11)	$\mathbf{0.7094 \pm 0.0190}$	$\mathbf{0.7049 \pm 0.0106}$	$\mathbf{0.7132 \pm 0.0127}$	$\mathbf{0.7255 \pm 0.0126}$

A.9 ADDITIONAL VIT AND GPT-2 LAST-LAYER EXPERIMENT RESULTS

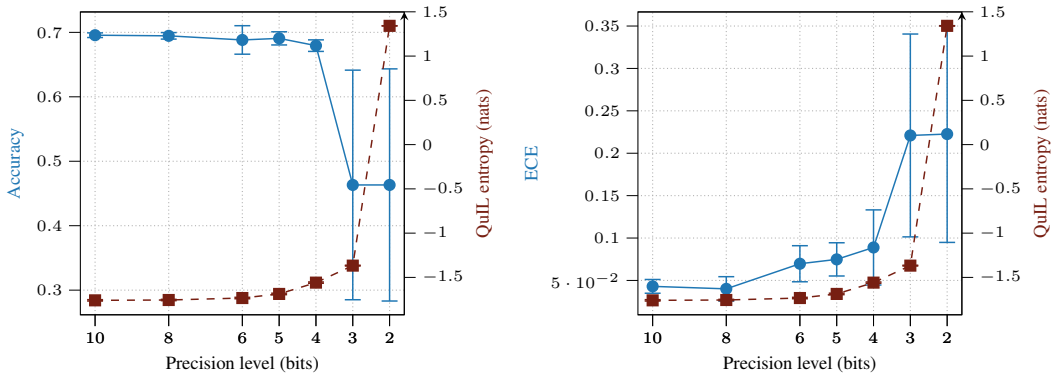

Figure A10: **Experiments on GPT-2:** The **QuIL** entropy shows clear signal on diminishing accuracy and ECE on MRPC.

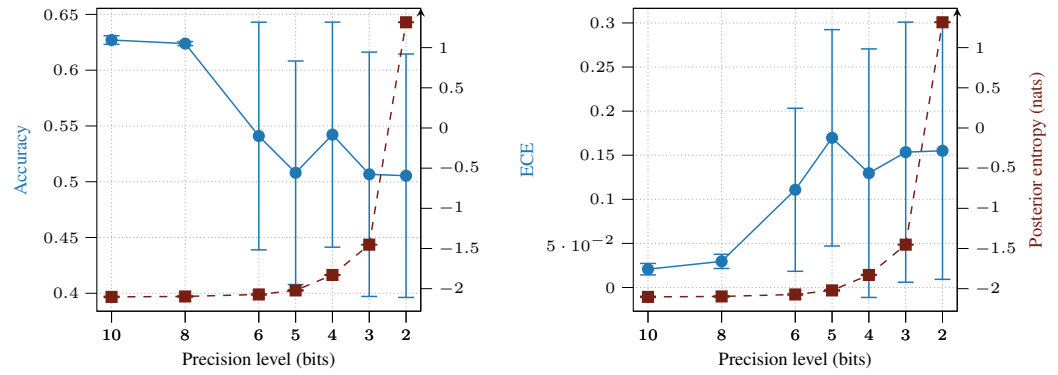

Figure A11: **QuIL tracks quantization degradation in GPT-2.** For a GPT-2 backbone with a fine-tuned BoolQ classifier head, the **QuIL** entropy signal (red, right axis) is computed from the curvature-initialized BitVI posterior at each bit-width. As precision is reduced, the entropy change remains small until the same low-bit regime where accuracy and calibration begin to degrade, indicating that **QuIL** captures the onset of representation-level loss.

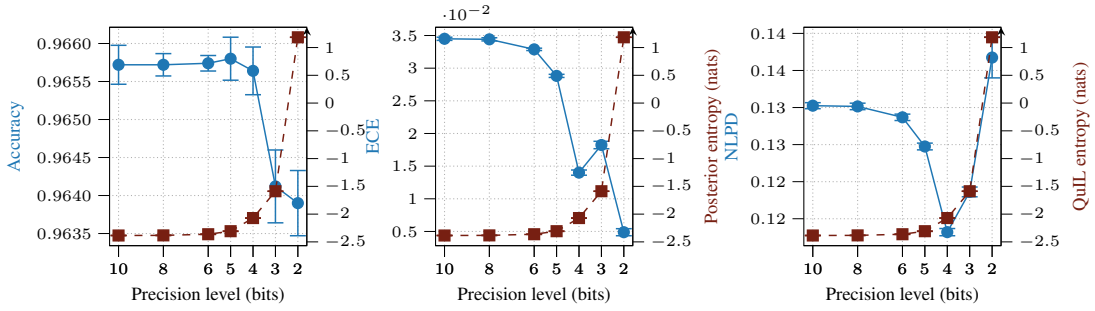

Figure A12: **ViT-base on CIFAR-10:** **QuIL** entropy vs. accuracy, ECE, and NLPD across bit-widths.

Table A3: Average testing set metrics (± s.d.) from a 5-seed experiment on GPT-2+MRPC. Methods are bolded based on a paired t-test ($p = 5\%$) with the best performing method. Metrics for the BitVI method are similar to the Laplace method.

		Precision (bits)				
Metric	Model	32	10	8	4	2
Accuracy (↑)	Laplace	**0.6951±0.0050**	**0.7005±0.0116**	**0.6951±0.0110**	**0.6466±0.0698**	**0.4623±0.1789**
	BitVI	-	**0.6956±0.0036**	**0.6946±0.0050**	**0.6794±0.0090**	**0.4632±0.1801**
NLPD (↓)	Laplace	**0.6085±0.0145**	**0.6093±0.0137**	**0.6113±0.0149**	**0.6436±0.0301**	**0.7627±0.1051**
	BitVI	-	**0.5965±0.0059**	**0.5976±0.0057**	**0.6323±0.0236**	**0.7542±0.1070**
ECE (↓)	Laplace	**0.0533±0.0188**	**0.0571±0.0228**	**0.0539±0.0281**	**0.0766±0.0360**	**0.2264±0.1287**
	BitVI	-	**0.0430±0.0081**	**0.0401±0.0143**	**0.0889±0.0443**	**0.2226±0.1278**

Table A4: Average testing set metrics (± s.d.) from a 5-seed experiment on ViT+DTD. Methods are bolded based on a paired t-test ($p = 5\%$) with the best performing method. Accuracy and NLPD Metrics for the BitVI method are consistently better than the Laplace method, suggesting it may be a more accurate approximation of the parameter posterior distribution.

		Precision (bits)				
Metric	Model	32	10	8	4	2
Accuracy (↑)	Laplace	**0.2280±0.0144**	0.2619 ± 0.0089	0.2641 ± 0.0072	0.2622 ± 0.0080	0.2472 ± 0.0098
	BitVI	-	**0.6477±0.0044**	**0.6480±0.0061**	**0.6380±0.0049**	**0.6257±0.0038**
NLPD (↓)	Laplace	**3.3640±0.0282**	3.3428 ± 0.0251	3.3416 ± 0.0249	3.3080 ± 0.0263	3.3200 ± 0.0271
	BitVI	-	**1.8948±0.0057**	**1.8959±0.0059**	**1.9478±0.0073**	**2.0424±0.0078**
ECE (↓)	Laplace	**0.1631±0.0132**	**0.2016±0.0074**	**0.2037±0.0061**	**0.1967±0.0066**	**0.1799±0.0083**
	BitVI	-	0.3818 ± 0.0046	0.3824 ± 0.0070	0.3887 ± 0.0055	0.4026 ± 0.0053

Table A5: Average testing set metrics (± s.d.) from a 5-seed experiment on ViT+RESISC. Methods are bolded based on a paired t-test ($p = 5\%$) with the best performing method. Accuracy and NLPD metrics for the BitVI method are consistently better than the Laplace method, suggesting it may be a more accurate approximation of the parameter posterior distribution.

		Precision (bits)				
Metric	Model	32	10	8	4	2
Accuracy (↑)	Laplace	**0.3519±0.1416**	0.3571 ± 0.1455	0.3576 ± 0.1455	0.3548 ± 0.1446	0.3405 ± 0.1409
	BitVI	-	**0.7840±0.0025**	**0.7839±0.0023**	**0.7806±0.0018**	**0.7720±0.0027**
NLPD (↓)	Laplace	**2.9808±0.3218**	2.9776 ± 0.3210	2.9767 ± 0.3214	2.8391 ± 0.3826	2.6622 ± 0.4985
	BitVI	-	**0.9622±0.0060**	**0.9619±0.0058**	**0.8999±0.0058**	**0.8658±0.0070**
ECE (↓)	Laplace	**0.2727±0.1309**	**0.2791±0.1343**	**0.2791±0.1348**	**0.2568±0.1255**	**0.2121±0.0835**
	BitVI	-	**0.2422±0.0014**	**0.2419±0.0012**	**0.2003±0.0017**	**0.1533±0.0029**

Table A6: **Curvature-initialized BitVI on the last-layer posterior.** Average test metrics (± s.d.) over five seeds for GPT-2 on BoolQ. Laplace gives a diagonal inverse-Hessian posterior. Bold values are tied with the best method under a paired t-test at $p = 5\%$.

		Precision (bits)				
Metric	Model	32	10	8	4	2
Accuracy (↑)	Laplace	**0.6223±0.0061**	0.6229 ± 0.0065	**0.6207±0.0025**	**0.5390±0.1025**	**0.5001±0.1068**
	BitVI	-	**0.6270±0.0039**	**0.6239±0.0018**	**0.5422±0.1009**	**0.5054±0.1091**
NLPD (↓)	Laplace	**0.6535±0.0041**	0.6535 ± 0.0042	**0.6558±0.0049**	**0.7415±0.1436**	**0.7714±0.1536**
	BitVI	-	**0.6472±0.0011**	**0.6491±0.0013**	**0.7428±0.1535**	**0.7696±0.1545**
ECE (↓)	Laplace	**0.0182±0.0079**	**0.0172±0.0067**	**0.0321±0.0133**	**0.1220±0.1411**	**0.1532±0.1471**
	BitVI	-	**0.0208±0.0065**	**0.0296±0.0080**	**0.1296±0.1409**	**0.1550±0.1458**

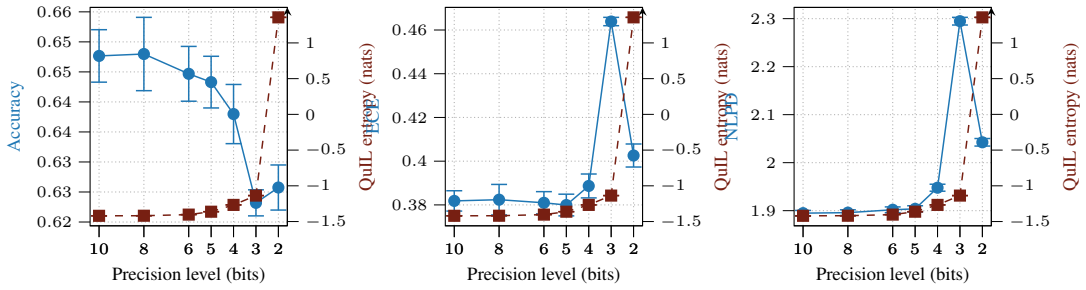

Figure A13: **ViT-base on DTD: QuIL** entropy vs. accuracy, ECE, and NLPD across bit-widths.

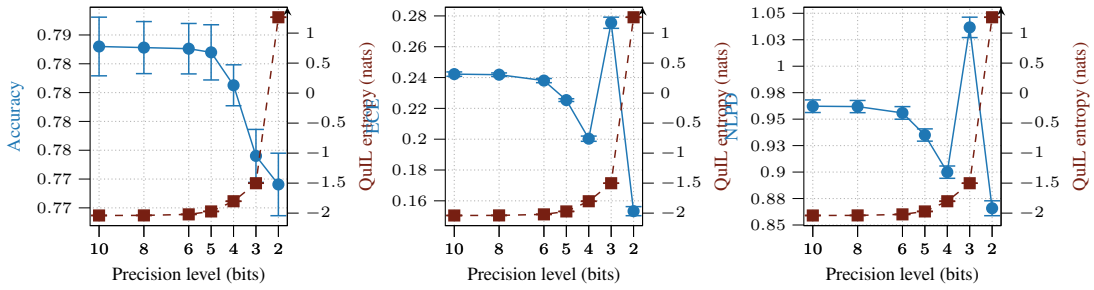

Figure A14: **ViT-base on RESISC: QuIL** entropy vs. accuracy, ECE, and NLPD across bit-widths.

B ADDITIONAL EXPERIMENT SETUP DETAILS

B.1 UCI EXPERIMENT HYPERPARAMETERS

Table A7: Experimental setup for the UCI fixed- versus floating-point BitVI comparison in Table A2. Settings spanning all four columns are shared across arms. The two 10-bit arms are identical except for the encoding, as are the two 6-bit arms except for the exponent/integer split noted below.

Setting	**10-bit (fixed)**	**10-bit (float)**	**6-bit (fixed)**	**6-bit (float)**
Data and protocol				
Data sets	10 UCI regression benchmarks (see Table A2), loaded via `bayesian_benchmarks`			
Cross-validation	5-fold, split seed 42			
Validation split	20% of each training fold			
Normalisation	z-score (inputs and targets); RMSE in original target units			
Batch size	32 ($n < 2{,}000$), 256 ($2{,}000 \leq n < 20{,}000$), 512 (PROTEIN)			
Architecture				
Network	MLP, mean-field variational linear layers			
Hidden layers	32–32	32–32	16–16	16–16
Activation	tanh			
Normalisation layer	LayerNorm			
Output	Mean only (homoskedastic)			
Weight posterior (BitVI circuit)				
Distribution	Mean-field over weights			
Parameters per sum node	2			
Initialisation	Kaiming			
MC samples K (train)	64			
Encoding	Fixed	Float	Fixed	Float
Total bits B	10	10	6	6
Integer / exponent bits	3	3	3	2
Bit layout	1+3+6	1+3+6	1+3+2	1+2+3
	(sign + integer + fraction, or sign + exponent + mantissa)			
Exponent bias	—	3	—	1
Representable range	± 7.98	$[0.125, 32]$	± 7.75	$[0.5, 8]$
Resolution (LSB)	0.0156	—	0.25	—
Optimisation				
Optimiser	Adam			
Learning rate	1×10^{-2} (constant; no schedule)			
Epochs	2000	2000	3000	3000
Likelihood tempering	Sinusoidal, 5 cycles, period 200 epochs			
Seed	33 (one seed; variation reported across folds)			
Prior and likelihood				
Weight / bias prior variance	1.0			
Observation noise σ	0.1 (fixed)			
Model selection and evaluation				
Selection criterion	Lowest validation RMSE, evaluated every 10 epochs			
Early stopping	Disabled			
Reported statistic	Test RMSE, mean $\pm$ s.d. over the 5 folds			
Significance test	Paired t-test against the best arm, $p = 0.05$			

B.2 VIT AND GPT-2 EXPERIMENT HYPERPARAMETERS

Hyperparameter listing for the large-scale **QuIL** experiments are shown in Table A8.

B.3 COMPUTATIONAL RESOURCES

All experiments were performed on our institution's computational cluster. Experiments were mainly conducted on NVIDIA 5090 (32GB VRAM), NVIDIA V100 (32 GB VRAM) and NVIDIA H200

Table A8: Hyperparameter settings for ViT and GPT-2 last-layer experiments. Shared settings apply to both backbones; dataset-specific values are listed where they differ.

Hyperparameter	**ViT**	**GPT-2**
Fine-tuning		
Backbone	ViT-Base/16 (pretrained)	GPT-2 (pretrained)
Optimiser	AdamW	AdamW
Learning rate	1×10^{-5} – 5×10^{-5}	1×10^{-4}
Weight decay	1×10^{-5}	1×10^{-5}
LR schedule	Linear with 10% warmup	
Epochs	5 CIFAR-10, 8 (RESISC), 10 (DTD)	10 (BoolQ), 20 (MRPC)
Batch size	64	16
Laplace approximation		
Subset of weights	Last layer (diagonal)	
Prior precision	Optimised via marginal likelihood	
Prediction type	GLM, probit link	
Posterior refinement (BitVI)		
Optimiser	Adam	
Learning rate	1×10^{-3}	
Epochs	30	
Circuit / distribution		
Distribution	Deterministic Mean Field Circuit	
MC samples K	64	
Total bits B	10	
Exponent bits	4	
Float bias	7	
Encoding	Float	
Parameters per sum node	2	
Prior variance	Laplace-initialised	
Evaluation		
Precision levels	10, 8, 6, 5, 4, 3, 2 bits	
Seeds	5	
MC Samples	1024	

(141GB VRAM) GPUs. Each large-scale experiment took roughly 6–15 hours to complete on the compute cluster.

B.4 Additional ViT-CIFAR-10 inner-layer experiment results

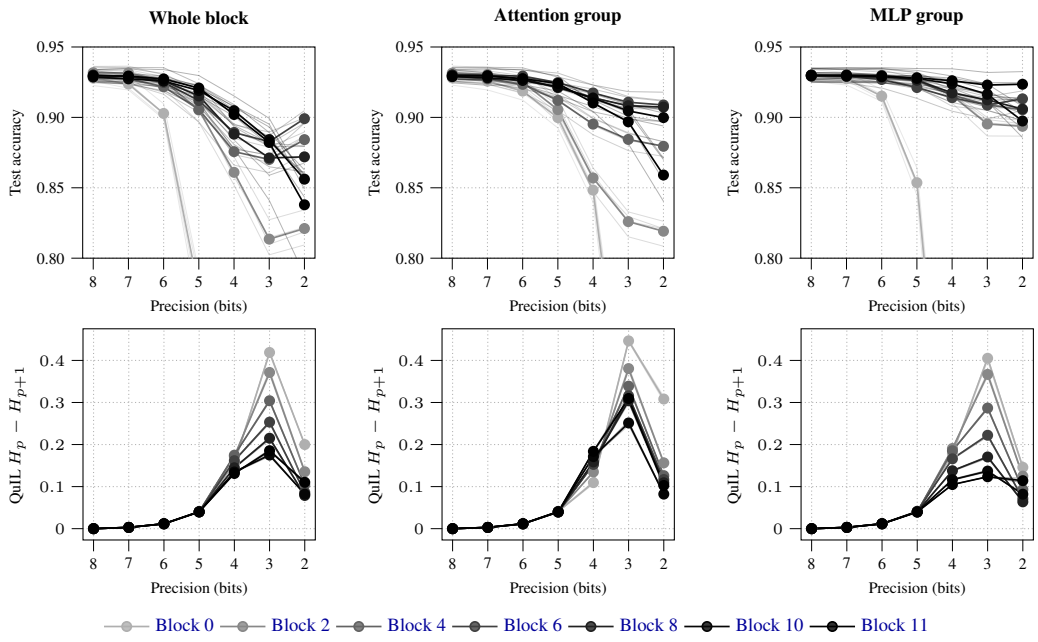

Figure A15: Deterministic test accuracy (top row; y-axis restricted to $[0.80, 0.95]$ to resolve the onset of degradation, so trajectories that fall further leave the panel) and **QuIL** as the step-wise entropy change $H_p - H_{p+1}$ of the fine-tuned BitVI circuit between consecutive precision levels (bottom row; zero at 8 bits), when jointly quantizing a whole transformer block (left column), only its attention sublayers (middle column), or only its MLP sublayers (right column). Shades of gray encode block depth (light = early, dark = late); bold lines are means over four seeds, faint lines individual seeds.

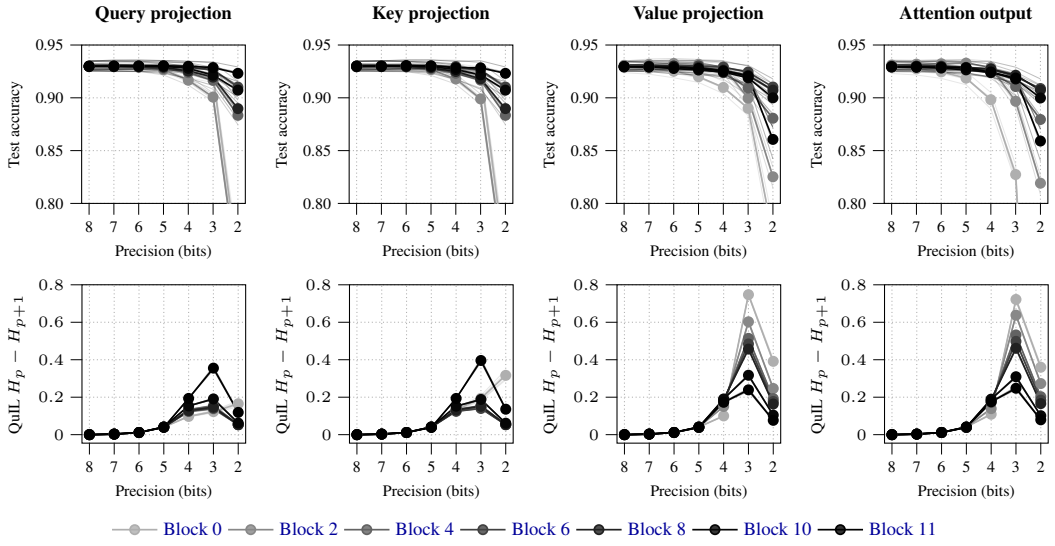

Figure A16: As Fig. A15, quantizing one attention sublayer at a time with every other sublayer held at full precision: the query, key and value projections and `attention.output.dense` (columns), accuracy above **QuIL**.

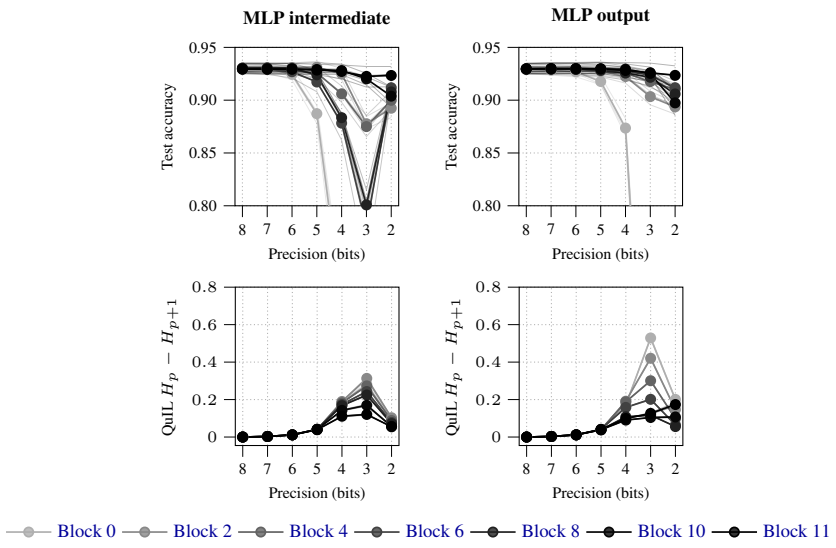

Figure A17: As Fig. A16, for the two MLP sublayers `intermediate.dense` and `output.dense`.

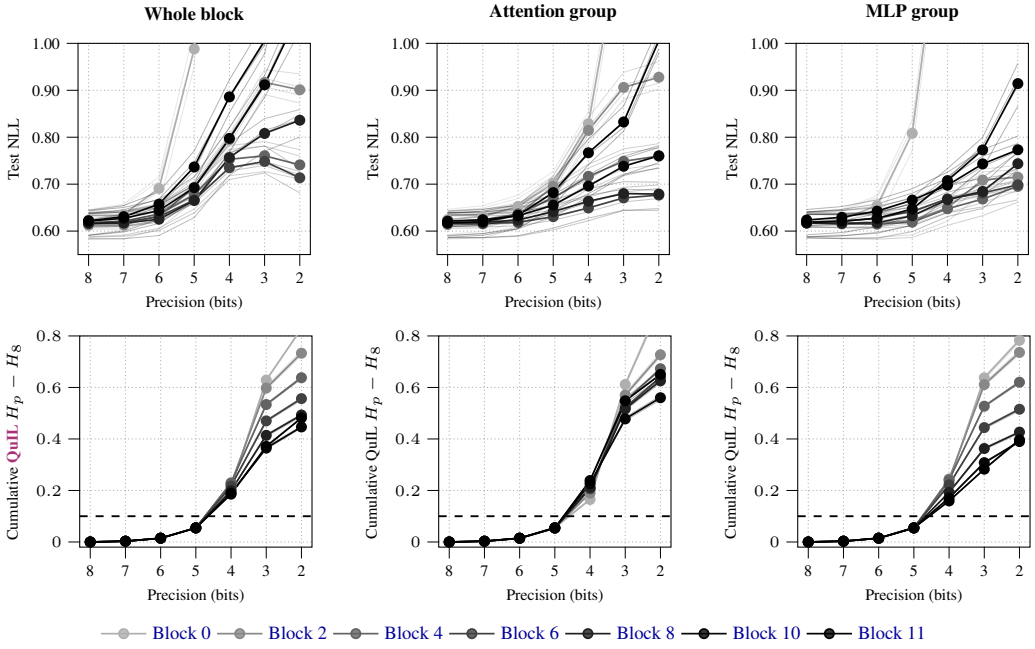

Figure A18: As Fig. A15, with the test negative log-likelihood (NLL; the mean cross-entropy of the point-quantized network's softmax on the true class, a single forward pass with no posterior sampling) in place of accuracy (top row; y-axis restricted to $[0.55, 1.0]$: the 8-bit baseline lies at 0.58–0.65 nats and the NLL reaches about 0.96 where accuracy crosses 0.80, so trajectories that rise further leave the panel).

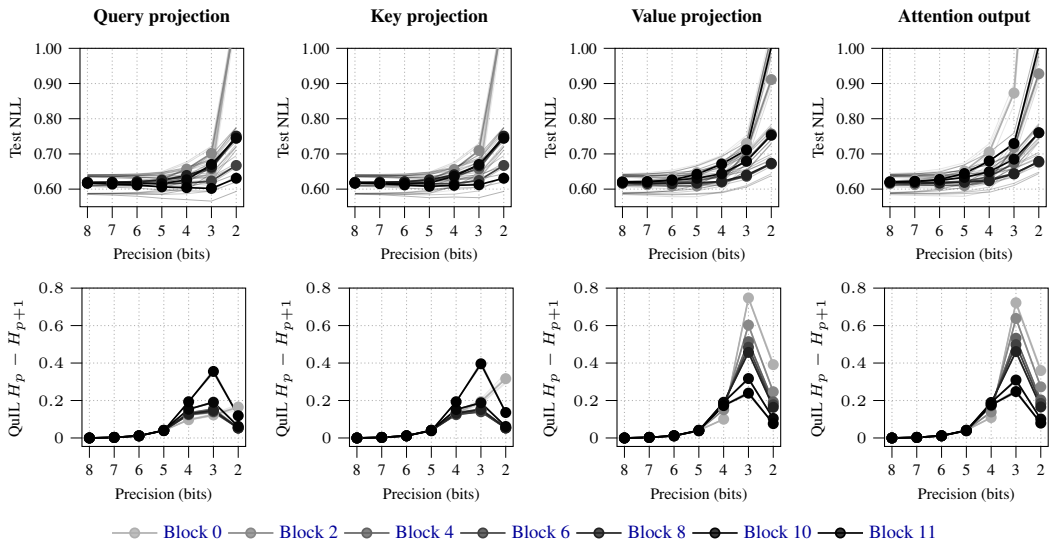

Figure A19: As Fig. A16, with test NLL in place of accuracy.

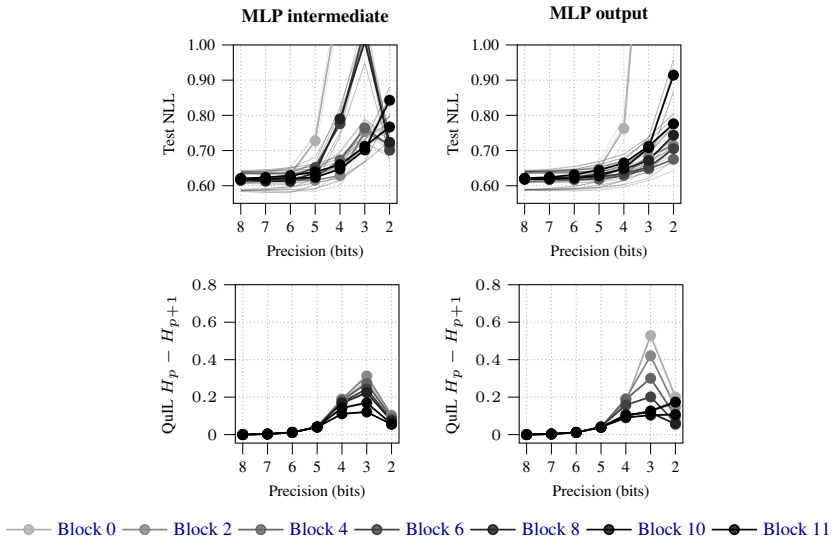

Figure A20: As Fig. A17, with test NLL in place of accuracy.

Table A9: Table showing the correlation between different performance metrics and precision for different parameter groups. Large negative correlations indicate that layers with a large sensitivity metric value correspond to layers with a large negative change in accuracy (accuracy degradation). Spearman correlations are averages across seeds, reported with standard deviation σ. Sensitivity metrics include the average Hessian trace (via Hutchinson and empirical Fisher estimators) with and without weighting quantization error Ω (a HAWQ-style metric). Two variants of **QuIL** are included, with Laplace **QuIL** indicating the Laplace initialized, non-fine-tuned version.

Average of per-seed Spearman correlation with Δaccuracy (deterministic-network ground truth), $p \in \{5, 4, 3, 2\}$

Precision	Metric	Block ($n = 7$)	Attention ($n = 7$)	MLP ($n = 7$)	Sublayer ($n = 42$)
$p = 5$	QuIL	-0.04 (σ=0.10)	-0.44 (σ=0.20)	-0.32 (σ=0.16)	-0.14 (σ=0.07)
	Laplace-QuIL	-0.03 (σ=0.10)	-0.38 (σ=0.23)	-0.28 (σ=0.18)	-0.15 (σ=0.06)
	$\overline{tr}(H) \cdot \Omega$ (Hutch)	+0.38 (σ=0.62)	+0.13 (σ=0.74)	+0.52 (σ=0.34)	+0.12 (σ=0.19)
	$\overline{tr}(H)$ (Hutch)	+0.23 (σ=0.75)	+0.08 (σ=0.78)	+0.37 (σ=0.31)	+0.12 (σ=0.20)
	$\overline{tr}(H) \cdot \Omega$ (Fisher)	-0.65 (σ=0.20)	-0.20 (σ=0.10)	-0.19 (σ=0.22)	-0.17 (σ=0.08)
	$\overline{tr}(H)$ (Fisher)	-0.97 (σ=0.02)	-0.78 (σ=0.12)	-0.60 (σ=0.31)	-0.17 (σ=0.06)
	Ω (Quant. error)	+0.87 (σ=0.03)	-0.06 (σ=0.14)	+0.55 (σ=0.32)	+0.09 (σ=0.04)
$p = 4$	QuIL	-0.68 (σ=0.03)	+0.57 (σ=0.08)	-0.66 (σ=0.15)	-0.05 (σ=0.08)
	Laplace-QuIL	-0.68 (σ=0.03)	+0.57 (σ=0.08)	-0.66 (σ=0.15)	-0.04 (σ=0.08)
	$\overline{tr}(H) \cdot \Omega$ (Hutch)	+0.30 (σ=0.66)	-0.02 (σ=0.76)	+0.62 (σ=0.32)	+0.12 (σ=0.31)
	$\overline{tr}(H)$ (Hutch)	+0.21 (σ=0.75)	-0.07 (σ=0.81)	+0.37 (σ=0.44)	+0.09 (σ=0.33)
	$\overline{tr}(H) \cdot \Omega$ (Fisher)	-0.78 (σ=0.05)	-0.63 (σ=0.06)	-0.46 (σ=0.20)	-0.16 (σ=0.03)
	$\overline{tr}(H)$ (Fisher)	-0.95 (σ=0.03)	-0.78 (σ=0.08)	-0.84 (σ=0.11)	-0.21 (σ=0.02)
	Ω (Quant. error)	+0.72 (σ=0.05)	-0.36 (σ=0.11)	+0.82 (σ=0.11)	+0.16 (σ=0.04)
$p = 3$	QuIL	-0.79 (σ=0.08)	-0.89 (σ=0.03)	-0.96 (σ=0.04)	-0.39 (σ=0.04)
	Laplace-QuIL	-0.79 (σ=0.08)	-0.88 (σ=0.03)	-0.96 (σ=0.04)	-0.39 (σ=0.04)
	$\overline{tr}(H) \cdot \Omega$ (Hutch)	+0.15 (σ=0.75)	-0.03 (σ=0.75)	+0.38 (σ=0.55)	+0.18 (σ=0.46)
	$\overline{tr}(H)$ (Hutch)	+0.15 (σ=0.75)	-0.08 (σ=0.80)	+0.19 (σ=0.68)	+0.16 (σ=0.48)
	$\overline{tr}(H) \cdot \Omega$ (Fisher)	-0.77 (σ=0.13)	-0.87 (σ=0.04)	-0.91 (σ=0.07)	-0.37 (σ=0.05)
	$\overline{tr}(H)$ (Fisher)	-0.80 (σ=0.13)	-0.78 (σ=0.05)	-0.96 (σ=0.04)	-0.41 (σ=0.04)
	Ω (Quant. error)	+0.66 (σ=0.09)	-0.05 (σ=0.03)	+0.93 (σ=0.04)	+0.39 (σ=0.04)
$p = 2$	QuIL	-0.32 (σ=0.06)	-0.84 (σ=0.02)	-0.48 (σ=0.12)	-0.23 (σ=0.06)
	Laplace-QuIL	-0.32 (σ=0.06)	-0.84 (σ=0.02)	-0.53 (σ=0.10)	-0.23 (σ=0.06)
	$\overline{tr}(H) \cdot \Omega$ (Hutch)	-0.16 (σ=0.53)	-0.16 (σ=0.70)	+0.11 (σ=0.51)	+0.01 (σ=0.57)
	$\overline{tr}(H)$ (Hutch)	-0.22 (σ=0.56)	-0.19 (σ=0.72)	-0.05 (σ=0.61)	+0.03 (σ=0.56)
	$\overline{tr}(H) \cdot \Omega$ (Fisher)	-0.32 (σ=0.11)	-0.33 (σ=0.10)	-0.80 (σ=0.02)	-0.24 (σ=0.02)
	$\overline{tr}(H)$ (Fisher)	-0.28 (σ=0.08)	-0.62 (σ=0.04)	-0.64 (σ=0.11)	-0.16 (σ=0.04)
	Ω (Quant. error)	+0.10 (σ=0.08)	-0.46 (σ=0.04)	+0.48 (σ=0.12)	+0.02 (σ=0.04)

B.5 AN EXACTLY-SOLVABLE DISCRETE TOY MODEL

As with many weight-space posterior approximation methods for neural networks, **QuIL** relies on a mean-field assumption weights to remain tractable and performs a stochastic optimization procedure for finding the locally optimal mean-field solution. To gain intuition on the impact of this assumption in the approximation error for **QuIL**, we constructed a tractable $D = 2$ linear regression $y = [\psi(\boldsymbol{b})_1, \psi(\boldsymbol{b})_2]^\top \boldsymbol{x} + \epsilon$ (where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{b}_i \in \mathbb{B}^B$) toy example, s.t. $y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \sigma^2)$ and $\boldsymbol{w} = \psi(\cdot)$ is the floating-point mapping from bit-strings $\boldsymbol{b}_i$ to reals $w \in \mathbb{R}$. We define a prior $p(\boldsymbol{b}_i) = \frac{1}{2^B}, i \in \{1, 2\}$ as uniform over all bit-string permutations, and arrive at the posterior via Bayes' rule:

$$p(\boldsymbol{b}_1, \boldsymbol{b}_2 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{b}_1, \boldsymbol{b}_2) p(\boldsymbol{b}_1) p(\boldsymbol{b}_2)}{p(\boldsymbol{x})}.$$

Usually, the evidence $p(\boldsymbol{x})$ is intractable, but in this simple case, it can be computed exactly. Since there are just 2^B possible bit-strings for each weight, the sum over all combinations is tractable. Thus, we compute

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{b}_1, \boldsymbol{b}_2} p(\boldsymbol{x} \mid \boldsymbol{b}_1, \boldsymbol{b}_2) p(\boldsymbol{b}_1) p(\boldsymbol{b}_2)$$

is the evidence that to then obtain the exact posterior $p(\boldsymbol{b}_1, \boldsymbol{b}_2 \mid \boldsymbol{x})$.

We can also compute the optimal mean-field approximation via coordinate ascent VI (CAVI). This allows us to disentangle the error due to the mean-field assumption. Then, we can compare the optimal mean-field approximation to the BitVI approximation to disentangle the error due to the stochastic optimization procedure. Effectively,

$$\text{Total error} = \underbrace{\text{Mean-field error}}_{\text{CAVI vs. exact posterior}} + \underbrace{\text{Optimization error}}_{\text{BitVI vs. CAVI}}. \tag{63}$$

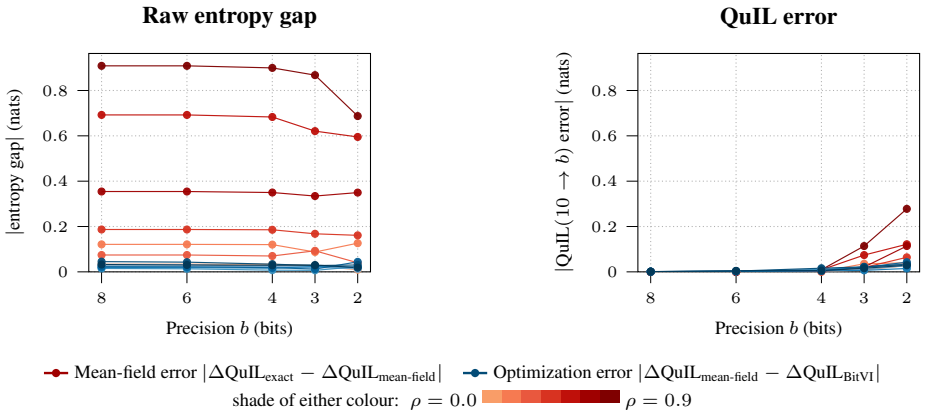

Figure A21: Figure showing the behavior of entropy approximation error in the CAVI mean-field approximation wrt the exact posterior entropy and the approximation error between the CAVI mean-field entropy and the BitVI entropy (left). These are compared wrt the error in their respective **QuIL** values under quantization (right). The plot shows that though the mean-field error is large, it remains fairly constant across precision levels and exact posterior correlations ρ. Optimization error is negligible between the CAVI and BitVI entropies and **QuIL** values

Table A10: Table showing the error decomposition of **QuIL** in nats across $N = 5$ experiments. Total error in **QuIL** between the BitVI approximation and the exact posterior is decomposed into the mean-field and optimization errors for several choices of posterior correlation ρ. Results indicate that the mean-field error dominates at low-precision levels (2-3 bits), and mean-field error increases as ρ increases.

ρ	Precision level	N	Mean-field error	Optimization error	Total error
0.0	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.0	6 bits	5	0.000 ± 0.000	0.003 ± 0.001	0.003 ± 0.001
0.0	4 bits	5	0.000 ± 0.001	0.011 ± 0.005	0.011 ± 0.006
0.0	3 bits	5	0.009 ± 0.012	0.021 ± 0.007	0.027 ± 0.017
0.0	2 bits	5	0.027 ± 0.045	0.025 ± 0.017	0.052 ± 0.052
0.15	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.15	6 bits	5	0.000 ± 0.000	0.003 ± 0.001	0.003 ± 0.001
0.15	4 bits	5	0.001 ± 0.001	0.006 ± 0.004	0.007 ± 0.004
0.15	3 bits	5	0.036 ± 0.041	0.010 ± 0.011	0.046 ± 0.041
0.15	2 bits	5	0.040 ± 0.047	0.028 ± 0.022	0.061 ± 0.039
0.3	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.3	6 bits	5	0.000 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
0.3	4 bits	5	0.004 ± 0.006	0.010 ± 0.007	0.013 ± 0.011
0.3	3 bits	5	0.022 ± 0.031	0.021 ± 0.010	0.032 ± 0.025
0.3	2 bits	5	0.034 ± 0.041	0.034 ± 0.019	0.059 ± 0.027
0.45	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.45	6 bits	5	0.000 ± 0.000	0.002 ± 0.001	0.002 ± 0.001
0.45	4 bits	5	0.001 ± 0.002	0.003 ± 0.003	0.004 ± 0.003
0.45	3 bits	5	0.019 ± 0.020	0.007 ± 0.004	0.023 ± 0.022
0.45	2 bits	5	0.064 ± 0.080	0.015 ± 0.011	0.064 ± 0.081
0.6	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.6	6 bits	5	0.000 ± 0.000	0.002 ± 0.001	0.003 ± 0.001
0.6	4 bits	5	0.009 ± 0.007	0.006 ± 0.004	0.008 ± 0.005
0.6	3 bits	5	0.074 ± 0.081	0.020 ± 0.012	0.089 ± 0.087
0.6	2 bits	5	0.122 ± 0.149	0.043 ± 0.024	0.158 ± 0.148
0.75	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.75	6 bits	5	0.000 ± 0.000	0.003 ± 0.001	0.003 ± 0.001
0.75	4 bits	5	0.004 ± 0.005	0.016 ± 0.006	0.020 ± 0.011
0.75	3 bits	5	0.021 ± 0.017	0.022 ± 0.017	0.038 ± 0.027
0.75	2 bits	5	0.114 ± 0.079	0.033 ± 0.021	0.108 ± 0.081
0.9	8 bits	5	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
0.9	6 bits	5	0.000 ± 0.000	0.004 ± 0.001	0.004 ± 0.001
0.9	4 bits	5	0.009 ± 0.007	0.005 ± 0.007	0.014 ± 0.009
0.9	3 bits	5	0.114 ± 0.095	0.016 ± 0.012	0.129 ± 0.098
0.9	2 bits	5	0.278 ± 0.250	0.029 ± 0.014	0.307 ± 0.263
Pooled (all ρ)	8 bits	35	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
Pooled (all ρ)	6 bits	35	0.000 ± 0.000	0.003 ± 0.001	0.003 ± 0.001
Pooled (all ρ)	4 bits	35	0.004 ± 0.006	0.008 ± 0.007	0.011 ± 0.009
Pooled (all ρ)	3 bits	35	0.042 ± 0.063	0.017 ± 0.012	0.055 ± 0.066
Pooled (all ρ)	2 bits	35	0.097 ± 0.146	0.029 ± 0.021	0.116 ± 0.151

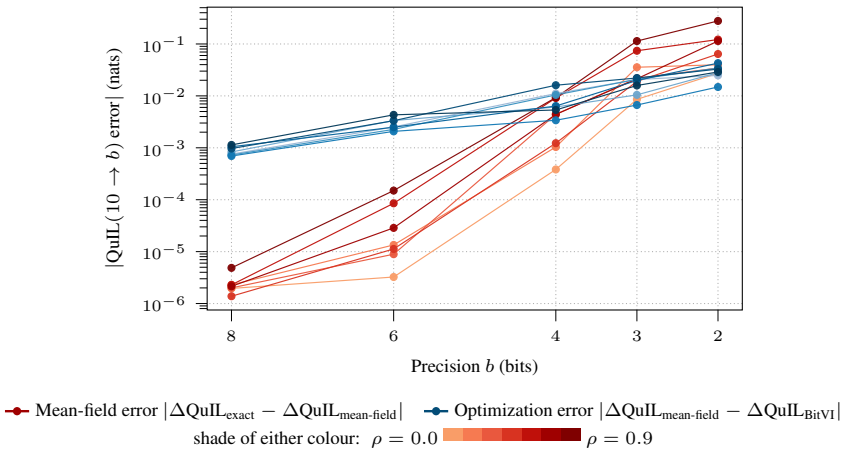

Figure A22: Figure showing how the **QuIL** error for quantizing from 10-bits to b-bits changes across precision level and posterior correlation ρ. Optimization error overwhelms the mean-field error at higher precision, until it is overtaken by the mean-field error at low-bit-widths.