SOAP-Bubbles ○❍
Structured Weight Uncertainty for Neural Networks

Adrian R. Minut[1*] **Nico Carlsen Daheim**[2] **Marco Miani**[3]
Mohammad Emtiyaz Khan[4,5] **Wu Lin**[6] **Thomas Möllenhoff**[5†]

[1]Sapienza University of Rome, Rome, Italy
[2]Ubiquitous Knowledge Processing Lab (UKP Lab),
Department of Computer Science, Technical University of Darmstadt
National Research Center for Applied Cybersecurity ATHENE, Germany
[3]Technical University of Denmark, Lyngby, Denmark
[4]TU Darmstadt & Hessian Center for AI (hessian.AI), Darmstadt, Germany
[5]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
[6]University of Central Florida, Orlando, United States

Abstract

Structured weight-uncertainty can improve many aspects of deep learning, but it remains costly to estimate and difficult to implement. Here, we show that these issues can be addressed by adapting the SOAP optimizer. Our key idea is to run IVON, an existing diagonal-covariance variational method, in the eigenspace of SOAP's preconditioner and then use the preconditioner to transform the diagonal estimate into a non-diagonal covariance. The resulting method has costs similar to those of SOAP and requires no drastic changes to training pipelines. We call the posteriors obtained in this way SOAP-Bubbles and our new optimizer Eigenspace-VON (EVON). We show that, for logistic regression, EVON can match the full Gaussian covariance and that, for language model pretraining, it yields significantly better results than existing diagonal-covariance methods. Our work makes it easier to estimate more expressive posterior distributions for deep learning at scale.
Code is available at: https://github.com/team-approx-bayes/evon.

1 Introduction

Weight uncertainty promises many improvements over current deep learning recipes. For example, it can be useful for exploration in reinforcement learning (Sehnke et al., 2010), to improve model merging (Daheim et al., 2024), to improve language generation by using posterior sampling (Daheim et al., 2025; Gan and Isola, 2026), and to understand model behavior (Nickl et al., 2023). The list of applications is long (Nguyen et al., 2018; Osawa et al., 2019; Wang et al., 2024; Cong et al., 2025; Khan, 2025; Möllenhoff et al., 2025, inter alia), yet weight uncertainty is rarely used in practice.

One reason behind this is that weight-uncertainty can be costly and also increase implementation overheads. Cheaper options do exist, but the quality may not always be acceptable. For example, the IVON optimizer (Shen et al., 2024) obtains Gaussian posteriors without significant increase in the cost compared to Adam, but it can only estimate a diagonal covariance which restricts the expressivity of the posterior. As a result, the quality of the uncertainty estimate might not always be sufficient.

Going beyond diagonal covariances remains an open challenge. There is a lot of work on post-hoc Laplace approximations, for example, with Kronecker-factored structures (Immer et al., 2021;

*Part of this work was done during an internship at RIKEN AIP.
†Correspondence to: thomas.moellenhoff@riken.jp

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

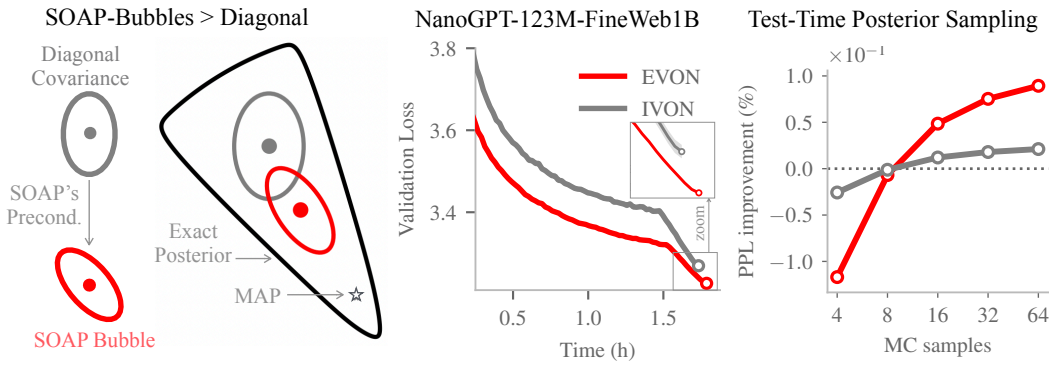


Figure 1: Our key idea is to use SOAP's preconditioner to transform diagonal covariances into non-diagonal ones. We call the resulting posteriors SOAP-Bubbles and show that they provide a much better fit to the exact posterior (left). Our new optimizer EVON runs the state-of-the-art variational method IVON in SOAP's eigenspace and significantly improves both the training speed and final loss for language models (middle). SOAP-Bubble posteriors obtained with EVON also work much better for model averaging during test-time than IVON's diagonal posteriors (right).

Daxberger et al., 2021; Eschenhagen et al., 2023; Dhahri et al., 2024; Hong et al., 2025). Unfortunately, such methods require an additional pass through the training data, which comes with extra costs and is not always feasible. Variational methods with structured covariance (Bae et al., 2018; Zhang et al., 2018; Louizos and Welling, 2016; Mishkin et al., 2018; Lin et al., 2019a, 2021; Fadel et al., 2025) are also difficult to implement and can increase memory and compute overheads. No study exists to date to scale such methods to language models. There is a need for a method to estimate structured weight-uncertainty that is easy to implement, adds little overhead, and scales to large problems.

In this paper, we introduce *SOAP-Bubbles*, a novel family of expressive, non-diagonal posterior distributions. SOAP-Bubbles are transformations of diagonal covariances, where the transformation is a rotation constructed from SOAP's preconditioner (Vyas et al., 2025). By considering a diagonal posterior within a rotated subspace, SOAP-Bubbles can capture block-diagonal covariances in the original space without the prohibitive costs usually associated with structured covariance approximations. We illustrate the main idea in Fig. 1 (left), where the SOAP-Bubble provides a better fit to the exact posterior than an axis-aligned diagonal Gaussian. In our theoretical contribution, we also show that SOAP-Bubbles can recover the optimal full-Gaussian posterior approximation in some cases, for example, for binary logistic regression.

To train such structured posteriors at scale, we propose the Eigenspace Variational Online Newton (EVON) algorithm. EVON entails only a simple modification to the SOAP optimizer (Vyas et al., 2025). SOAP (ShampoO with Adam in the Preconditioner's Eigenbasis) constructs a block-diagonal preconditioner from stochastic minibatch gradients and runs Adam within the resulting eigenspace. EVON builds directly on this structure and operates in the same eigenspace, but replaces Adam with the mean-field variational method IVON (Shen et al., 2024). This maps a diagonal variational method in the eigenspace to a more expressive posterior in the original weight space, achieving the performance of structured variational inference without major changes to standard training pipelines.

In our experiments, we demonstrate that EVON allows SOAP-Bubbles to scale to language model training, yielding consistently better results than IVON under the same computational budget. For example, when training NanoGPT on FineWeb-1B (Penedo et al., 2024), EVON significantly improves validation loss, as shown in Fig. 1 (middle). Furthermore, ensembling models sampled from SOAP-Bubbles substantially outperforms ensembling from IVON's diagonal posterior, see Fig. 1 (right). This confirms that our approach not only improves training optimization dynamics but successfully captures a richer representation of weight uncertainty.

2 Variational Learning and Structured Weight Uncertainty

We start with an introduction of variational learning (VL) and then discuss the challenges of going beyond diagonal covariances in VL. In deep learning we usually minimize a loss function $\ell(\boldsymbol{\theta}) =$

$\sum_{i=1}^{n} \ell_i(\boldsymbol{\theta})/n$ over n data examples with respect to the parameters $\boldsymbol{\theta}$. To model weight uncertainty, VL instead searches for a distribution over the parameters and targets a related objective over $q(\boldsymbol{\theta})$,

$$\min_{q \in \mathcal{Q}} \; \zeta \, \mathbb{E}_{q(\boldsymbol{\theta})}[\ell(\boldsymbol{\theta})] + \mathrm{KL}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})), \tag{1}$$

where $\mathcal{Q}$ is a set of candidate posterior distributions. The Kullback-Leibler term regularizes towards a prior $p(\boldsymbol{\theta})$, here assumed to be an isotropic Gaussian with zero-mean, and is governed by $\zeta^{-1} > 0$. This corresponds to a quadratic regularizer (non-decoupled weight-decay) and in this paper, we will always use prior variance $(\zeta\delta)^{-1}\mathbf{I}$, where $\delta > 0$ is the usual weight-decay parameter.

Variational learning for large deep nets faces several challenges, the most prominent being the choice of the posterior distribution family $\mathcal{Q}$. If $\mathcal{Q}$ is chosen as a family of Gaussian distributions with diagonal covariance, then every parameter in the neural network has a mean and a variance. Naively optimizing both the mean and the variance, as is often done with Adam or SGD in the variational Bayesian literature, doubles the number of optimization variables and is also difficult to implement (Khan et al., 2018).

2.1 Diagonal Variational Methods and IVON

It is possible to formulate algorithms for the variational objective in Eq. 1 which take a similar form to the Adam optimizer and do not introduce additional variables, as shown by Khan et al. (2018); Zhang et al. (2018) and later scaled to larger models by Osawa et al. (2019); Shen et al. (2024). These methods exploit a connection between VL with Gaussian posteriors and second-order optimization. The main idea is that the update for the diagonal posterior precision takes a similar form to the exponential moving average of the scale vector that adapts the learning rate in Adam. Thus, only small modifications to Adam, such as adding the posterior noise to the weights before computing the gradient, are required to target the variational objective. For example, the Variational Online Newton (VON) method of Khan et al. (2018) can be implemented with the following update:

$$\begin{aligned} \widehat{\mathbf{h}} &\leftarrow \widehat{\mathbf{g}} \circ (\boldsymbol{\theta} - \mathbf{m})/\boldsymbol{\sigma}^2, \\ \mathbf{h} &\leftarrow \beta_2 \, \mathbf{h} + (1 - \beta_2) \, \widehat{\mathbf{h}}, \\ \mathbf{m} &\leftarrow \mathbf{m} - \alpha \, (\widehat{\mathbf{g}} + \delta \, \mathbf{m})/(\mathbf{h} + \delta), \end{aligned} \tag{2}$$

where $\widehat{\mathbf{g}}$ is the stochastic gradient evaluated at a sample from the posterior $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{m}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ with variance $\boldsymbol{\sigma}^2 = 1/(\zeta(\mathbf{h} + \delta))$ is the posterior variance. $\widehat{\mathbf{h}}$ is the per-parameter curvature estimate obtained via Stein's identity (element-wise product denoted by $\circ$), $\mathbf{h}$ is the running average of the (positive) curvature with decay rate β_2, $\delta > 0$ is a small prior/regularization constant and α is the learning rate. The mean update for $\mathbf{m}$ uses the curvature as a preconditioner and a weight-decay term $\delta\mathbf{m}$ due to the prior.

IVON adds an additional quadratic correction term (Lin et al., 2020) to the $\mathbf{h}$-update to keep the posterior variance positive, as well as momentum and clipping. Both VON and IVON however maintain only a single scalar curvature estimate per parameter (no extra auxiliary variables beyond the usual running average), inject Gaussian posterior noise via the sampled $\boldsymbol{\theta}$, and target the variational objective in Eq. 1. This makes it a natural baseline for diagonal variational methods, but as all diagonal variational methods, it cannot track inter-parameter correlations in the posterior covariance.

2.2 Non-Diagonal Variational Methods

Extending variational learning to Gaussians with non-diagonal covariances in a scalable manner is non-trivial. There has been earlier work on noisy variants of second-order methods with Kronecker-factored curvature approximations (K-FAC), see (Zhang et al., 2018; Bae et al., 2018). However, K-FAC methods have not yet been shown to work well for modern language model training, and the same is to be expected of their noisy counterparts which target the variational objective. SLANG (Mishkin et al., 2018) maintains a low-rank plus diagonal posterior, but has not been shown to scale to large problems and comes both with implementation and memory overheads. Similarly, the recent VIKING method (Fadel et al., 2025) requires expensive projection steps to optimize non-diagonal covariances, and has not been shown to scale to transformer architectures. Bayesian online natural gradient (BONG) (Jones et al., 2024) tracks non-diagonal covariances but has only been run on small MNIST-scale models. Variational methods for matrix Gaussian posteriors (Louizos and Welling, 2016; Lin

et al., 2019a; van der Ouderaa et al., 2024) have so far also only been shown to work on smaller problems and usually require significant changes to training pipelines. Overall, there is so far no scalable and easy-to-use non-diagonal variational method that is implemented as an optimizer and can train modern neural networks. Our work aims to fill this gap.

2.3 Kronecker-Factored Preconditioners and SOAP

Recently, optimizers with Kronecker-factored preconditioners, including Shampoo (Gupta et al., 2018) and SOAP (Vyas et al., 2025), have gained attention for language model training. For instance, a distributed implementation of Shampoo (Shi et al., 2023) won a recent challenge on optimization for deep learning (Kasimbeg et al., 2025). Here, we build on these advances.

Modern practical versions of the Shampoo optimizer, here written for a single 2D parameter matrix $\mathbf{\Theta}$ inside a neural network, use the following update:

$$\begin{aligned} \mathbf{L} &\leftarrow \beta\mathbf{L} + (1-\beta)\mathbf{G}\mathbf{G}^\top, \\ \mathbf{R} &\leftarrow \beta\mathbf{R} + (1-\beta)\mathbf{G}^\top\mathbf{G}, \\ \mathbf{\Theta} &\leftarrow \mathbf{\Theta} - \alpha\mathbf{L}^{-1/p}\mathbf{G}\mathbf{R}^{-1/p}, \end{aligned} \tag{3}$$

where $\mathbf{G}$ denotes the stochastic minibatch gradient. We do not consider momentum and weight decay for simplicity. While the original Shampoo uses $p = 4$, recent works (Morwani et al., 2025; Vyas et al., 2025; Eschenhagen et al., 2025) suggest using a square root, that is, $p = 2$. Vyas et al. (2025) show a striking similarity of the above update with $p = 2$ to running a step of Adafactor or Adam in the eigenspace obtained from the above running average matrices $\mathbf{L}$ and $\mathbf{R}$:

$$\begin{aligned} \mathbf{Q}_L &\leftarrow \text{Eig}(\mathbf{L}), \\ \mathbf{Q}_R &\leftarrow \text{Eig}(\mathbf{R}), \\ \mathbf{\Theta} &\leftarrow \mathbf{\Theta} - \alpha\mathbf{Q}_L\mathbf{U}\mathbf{Q}_R^\top, \end{aligned} \tag{4}$$

where $\mathbf{U}$ is Adam's preconditioned update, that is, the gradient momentum divided by the square root of the running average of squared gradients. A crucial point is that the momentum and preconditioner are built from the projected gradients $\mathbf{G}^\circ = \mathbf{Q}_L^\top\mathbf{G}\mathbf{Q}_R$. They refer to the above method as "ShampoO with Adam in the Preconditioner's Eigenbasis" (SOAP), and show promising performance across various language model pretraining tasks. We summarize SOAP for a single layer in Alg. 1 (left).

In the following, we show that the SOAP optimizer can be modified to minimize Eq. 1 for a variational family of Gaussian posteriors with structured covariance which we call SOAP-Bubbles. The main idea is to replace the Adam update used in SOAP in the eigenspace spanned by $\mathbf{Q}_L$ and $\mathbf{Q}_R$ from Eq. 4 in order to target the variational objective.

3 SOAP-Bubble: Posteriors within the SOAP Eigenbasis

To formalize this idea, we consider a variational family defined within a transformed coordinate space. Rather than learning a diagonal Gaussian directly in the original parameter space $\boldsymbol{\theta}$, as IVON does, we define a diagonal Gaussian over a transformed variable $\mathbf{w}$:

$$\mathcal{Q}_{\text{diag}} = \left\{\mathcal{N}(\mathbf{w} \mid \mathbf{m}, \text{diag}(\boldsymbol{\sigma}^2)) \mid \mathbf{m} \in \mathbb{R}^P, \boldsymbol{\sigma} \in \mathbb{R}_+^P\right\}, \tag{5}$$

where $\mathbf{w}$ is connected to the original parameter via $\boldsymbol{\theta} = \mathbf{P}\mathbf{w}$ and $\mathbf{P}$ is assumed to be an orthonormal matrix in the following. The corresponding family in the original space has non-diagonal covariances:

$$\mathcal{Q}_{\text{bubble}} = \left\{\mathcal{N}(\boldsymbol{\theta} \mid \mathbf{P}\mathbf{m}, \mathbf{P}\text{diag}(\boldsymbol{\sigma}^2)\mathbf{P}^\top) \mid \mathbf{m} \in \mathbb{R}^P, \boldsymbol{\sigma} \in \mathbb{R}_+^P\right\}. \tag{6}$$

Later, we will pick $\mathbf{P}$ to be a rotation built from a Kronecker product of SOAP's $\mathbf{Q}_R$ and $\mathbf{Q}_L$ matrices, and will therefore refer to elements $q \in \mathcal{Q}_{\text{bubble}}$ as a SOAP-Bubble.

The main insight to allow for an efficient implementation is that the variational objective for the family $\mathcal{Q}_{\text{bubble}}$ with structured covariance can actually be written over the diagonal family with a transformed loss. The following theorem formalizes this fact.

Algorithm 1 SOAP (Vyas et al., 2025)

No Sampling

Update Preconditioner and Momentum

3: $\mathbf{G} \leftarrow \widehat{\nabla}\ell(\boldsymbol{\Theta})$, $\mathbf{G}^\circ \leftarrow \mathbf{Q}_L^\top \mathbf{G} \mathbf{Q}_R$

4: $\widehat{\mathbf{H}} \leftarrow \mathbf{G}^\circ \odot \mathbf{G}^\circ$

5: $\bar{\mathbf{G}} \leftarrow \beta_1 \bar{\mathbf{G}} + (1-\beta_1)\mathbf{G}^\circ$

6: $\mathbf{H} \leftarrow \beta_2 \mathbf{H} + (1-\beta_2)\widehat{\mathbf{H}}$

Update Parameters

7: $\mathbf{U} \leftarrow \bar{\mathbf{G}} / (\sqrt{\mathbf{H}} + \epsilon)$

8: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \alpha \mathbf{Q}_L \mathbf{U} \mathbf{Q}_R^\top$

Update Eigenbasis

9: $\mathbf{L} \leftarrow \beta_3 \mathbf{L} + (1-\beta_3)\mathbf{G}\mathbf{G}^\top$

10: $\mathbf{R} \leftarrow \beta_3 \mathbf{R} + (1-\beta_3)\mathbf{G}^\top\mathbf{G}$

11: Every T steps:

$\mathbf{Q}_L, \mathbf{Q}_R \leftarrow \text{Eig}(\mathbf{L}), \text{Eig}(\mathbf{R})$

Update momentum $\bar{\mathbf{G}}$ (new basis)

Algorithm 2 EVON (Proposed Method)

Sample Parameters

1: $V_{i,j} \leftarrow 1/(\zeta(H_{i,j} + \delta))$

2: $\boldsymbol{\Theta} \leftarrow \mathbf{M} + \mathbf{Q}_L \mathbf{E} \mathbf{Q}_R^\top$, $E_{i,j} \sim \mathcal{N}(0, V_{i,j})$

Update Hessian and Momentum

3: $\mathbf{G} \leftarrow \widehat{\nabla}\ell(\boldsymbol{\Theta})$, $\mathbf{G}^\circ \leftarrow \mathbf{Q}_L^\top \mathbf{G} \mathbf{Q}_R$

4: $\widehat{\mathbf{H}} \leftarrow \text{clip}(\mathbf{E} \odot \mathbf{G}^\circ / \mathbf{V})$

5: $\bar{\mathbf{G}} \leftarrow \beta_1 \bar{\mathbf{G}} + (1-\beta_1)\mathbf{G}^\circ$

6: $\mathbf{H} \leftarrow \beta_2 \mathbf{H} + (1-\beta_2)\widehat{\mathbf{H}}$
$+\frac{1}{2}(1-\beta_2)^2(\widehat{\mathbf{H}} - \mathbf{H})^{\odot 2}/(\mathbf{H} + \delta)$

Update Posterior Mean

7: $\mathbf{U} \leftarrow (\bar{\mathbf{G}} + \delta \mathbf{Q}_L^\top \mathbf{M} \mathbf{Q}_R)/(\mathbf{H} + \delta)$

8: $\mathbf{M} \leftarrow \mathbf{M} - \alpha\, \text{clip}\big(\mathbf{Q}_L \mathbf{U} \mathbf{Q}_R^\top\big)$

Update Eigenbasis

9: $\mathbf{L} \leftarrow \beta_3 \mathbf{L} + (1-\beta_3)\mathbf{G}\mathbf{G}^\top$

10: $\mathbf{R} \leftarrow \beta_3 \mathbf{R} + (1-\beta_3)\mathbf{G}^\top\mathbf{G}$

11: Every T steps:

$\mathbf{Q}_L, \mathbf{Q}_R \leftarrow \text{Eig}(\mathbf{L}), \text{Eig}(\mathbf{R})$

Update momentum $\bar{\mathbf{G}}$ (new basis)

Figure 2: EVON entails only simple modifications to the SOAP optimizer (highlighted in red). Given an existing implementation of SOAP, it is easy to implement and scales to large problems as SOAP does. $\odot$ denotes element-wise multiplication of matrices, / elementwise division and $^{\odot 2}$ elementwise squaring. As in Vyas et al. (2025), the updates are written here for a single 2D parameter matrix.

Theorem 1. *For the above variational families $\mathcal{Q}_{diag}$ and $\mathcal{Q}_{bubble}$, the following holds:*

$$\begin{aligned}(1) \;&= \min_{q \in \mathcal{Q}_{\text{bubble}}} \zeta\, \mathbb{E}_{q(\boldsymbol{\theta})}[\ell(\boldsymbol{\theta})] + \text{KL}(q(\boldsymbol{\theta}) \,\|\, \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{m}_0, \boldsymbol{\Sigma}_0)) \\ &= \min_{q \in \mathcal{Q}_{\text{diag}}} \zeta\, \mathbb{E}_{q(\mathbf{w})}[\ell(\mathbf{P}\mathbf{w})] + \text{KL}\Big(q(\mathbf{w}) \,\|\, \mathcal{N}(\mathbf{w} \mid \mathbf{P}^\top \mathbf{m}_0, \mathbf{P}^\top \boldsymbol{\Sigma}_0 \mathbf{P})\Big).\end{aligned} \tag{7}$$

The result follows directly from the invariance of the KL under invertible transformations, which is the equality case in the data-processing inequality (Cover and Thomas, 2006). A proof is in Sec. A.

3.1 Eigenbasis Variational Online Newton (EVON)

Using Thm. 1, we can obtain our proposed EVON method shown in Alg. 2. The method is derived by applying IVON (Lin et al., 2020; Shen et al., 2024) to the variational objective in $q(\mathbf{w})$, where we set $\mathbf{P} = \mathbf{Q}_R \otimes \mathbf{Q}_L$ as the Kronecker product of the matrices estimated in SOAP. In particular, we use (Shen et al., 2024, Alg. 1) and rewrite the iteration in terms of the posterior in the original space $\boldsymbol{\theta}$. The variational objective in $q(\mathbf{w})$ then fits exactly the form used in (Shen et al., 2024, Eq. 1) under a transformed loss $\tilde{\ell}(\mathbf{w}) = \ell(\mathbf{P}\mathbf{w})$. A full self-contained derivation is given in Sec. B, and the resulting EVON method is shown in Alg. 2.

The modifications made in red to SOAP in Alg. 2 enable us to estimate a variational posterior with non-diagonal covariance in the original space $\boldsymbol{\theta}$, that is, $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \text{vec}(\mathbf{M}) \quad \text{and} \quad \boldsymbol{\Sigma} = \Big(\mathbf{Q}_R \otimes \mathbf{Q}_L\Big)\, \text{diag}\,[\text{vec}(\mathbf{V})]\, \Big(\mathbf{Q}_R^\top \otimes \mathbf{Q}_L^\top\Big). \tag{8}$$

EVON minimizes Eq. 1 for this family of posteriors, whereas SOAP just targets the regular loss $\ell(\boldsymbol{\theta})$. We will now go over all the modifications that EVON makes to SOAP in Alg. 2, one by one.

(i) Unlike SOAP, EVON uses random samples drawn from the variational posterior (see Eq. 8) in lines 1 and 2 and then computes the gradient at the randomly sampled weights in line 3. This

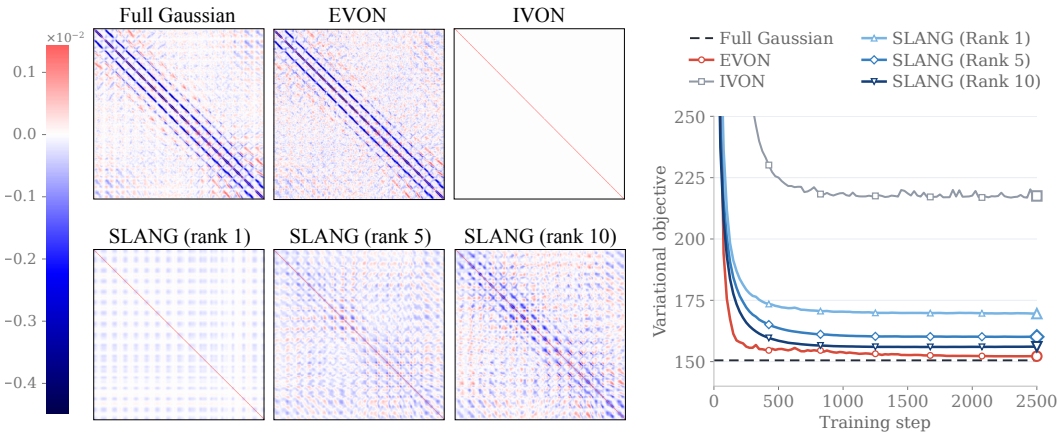


Figure 3: EVON is not only scalable to language models, but also works well for classical variational Bayesian inference. Here, we show that on a binary Bayesian logistic regression, EVON's covariance closely matches the full Gaussian 256×256 posterior covariance matrix on the USPS dataset (16x16 grayscale digits, 3vs5). IVON and SLANG (Mishkin et al., 2018) use diagonal or low-rank plus diagonal approximations which are worse. EVON's variational objective also closely matches the full objective whereas IVON and SLANG need more iterations and converge to a worse value.

yields an unbiased estimator of the gradient of the expected loss $\mathbb{E}_{q(\boldsymbol{\theta})}[\ell(\boldsymbol{\theta})]$ in the variational objective rather than targeting the regular loss $\ell(\boldsymbol{\theta})$ as in SOAP.

(ii) Line 4 changes the squared-gradient to an estimator of the (expected) Hessian. It is known that using squared-gradients approximation to the Hessian within a variational method leads to biased solutions, even on simple logistic regression problems (Khan et al., 2018, Fig. 2a). Due to this change, EVON is able to recover good solutions on simple variational Bayesian problems, though a squared-gradient heuristic may also work well for deep learning as it trades-off variance for bias (Graves, 2011; Osawa et al., 2019). To reduce the variance, we added an optional clipping in line 4, which we found empirically to improve the method when training transformers.

(iii) Line 6 uses the quadratic correction term proposed by Lin et al. (2020), which ensures that the posterior precision remains positive. This is needed, since we departed from the squared-gradient estimator to a Hessian estimator, which may not be positive anymore in nonconvex problems.

(iv) The preconditioned gradient in line 7 now includes the prior, written through the weight-decay parametrization δ, and we precondition with the Hessian rather than the square-root.

(v) Line 8 clips the update, either element-wise as in IVON or Sophia (Liu et al., 2024), or spectrally (Carlson et al., 2015; Jordan et al., 2024b). We use spectral clipping, which performed considerably better in our experiments (Sec. D).

The remaining lines of the two algorithms are completely identical, allowing to implement EVON as a simple modification of SOAP's existing codebase. Since SOAP updates both $\mathbf{Q}_L$ and $\mathbf{Q}_R$ occasionally during the iteration, the variational family that is being optimized also changes during the course of training. When updating $\mathbf{Q}_L$ and $\mathbf{Q}_R$ in line 11, both for SOAP and EVON the momentum $\bar{\mathbf{G}}$ is projected into the new space via $\bar{\mathbf{G}} \leftarrow \mathbf{Q}_L^\top \mathbf{Q}_L' \bar{\mathbf{G}}' (\mathbf{Q}_R')^\top \mathbf{Q}_R$, where the apostrophe denotes the old $\mathbf{Q}$ and $\bar{\mathbf{G}}$. However, both methods do not update $\mathbf{H}$ into the new space.

3.2 Implementation Details

We now discuss several key implementation details which are important to make EVON practical.

Distributed Training. In distributed data-parallel training, using independent noise across GPUs is beneficial as it reduces the variance of these stochastic estimators (Kingma et al., 2015; Osawa et al., 2019). EVON requires a careful implementation beyond a naive distributed data-parallel reduction of gradients to ensure the noisy gradients are aggregated consistently for the Hessian estimator.

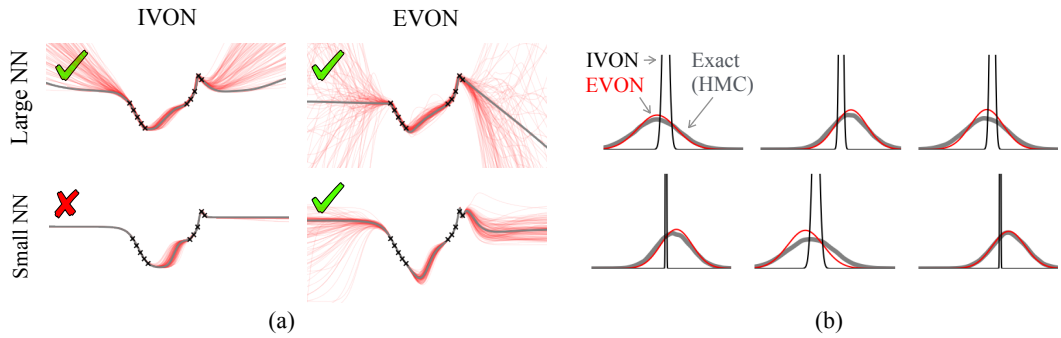


Figure 4: **(a)** We compare IVON and EVON's posterior samples on a small and a large MLP. While both methods work for large MLPs, for small networks, IVON's uncertainty estimates outside of the data are collapsed. In contrast, EVON gives more diverse posterior samples in regions where the data does not fully specify the solution, showing the strength of non-diagonal posterior covariances. **(b)** We show reference posterior marginal distributions (gray) computed with HMC (Hoffman and Gelman, 2014; Carpenter et al., 2017) on multiclass logistic regression on the UCI Iris dataset against IVON's mean-field approximation (black) and EVON's posterior (red). IVON's mean-field approximation underestimates the variance, while EVON better captures the HMC marginals.

Approximate Eigenbasis. Both SOAP and EVON use the QR decomposition as an efficient approximation to the eigendecomposition in line 11, as explained in Vyas et al. (2025). We use `float32` precision for the (approximate) eigenbasis computation, but the rest can be kept in `bfloat16`.

Adaptive Hessian Clipping. Optionally, we clip the Hessian estimator in line 4 adaptively relative to the current moving average of the Hessian. Specifically, if $\mathbf{F}$ is the raw Hessian estimator in line 4 at a given step, we apply element-wise clipping to obtain $\text{clip}(\mathbf{F}, -\gamma(\mathbf{H}+\epsilon), \gamma(\mathbf{H}+\epsilon))$, where $\mathbf{H}$ is the current Hessian estimate, γ is a clip ratio (defaulting to 10), and ϵ is a small constant.

Spectral Clipping. Spectral clipping sets all singular values of the update in line 8 to roughly one, using a quintic Newton–Schulz iteration as in Muon (Jordan et al., 2024b) instead of an SVD (Sec. D).

3.3 Theoretical Guarantees

Originally, we conceived EVON as a scalable way to obtain non-diagonal covariances for large-scale variational Bayesian training of transformer-based language models. In experiments on simple linear models we found that the method is surprisingly accurate and often finds solutions close to the exact posterior among all multivariate Gaussian posteriors. Motivated by these practical findings, shown in Fig. 3, we provide here a theoretical explanation under which conditions exact solutions are possible.

The following theorem and corollary characterize sufficient conditions under which this is the case.

Theorem 2. *Assume a linear model f, loss function c, an isotropic prior and a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$. If the matrices $\mathbf{L}$ and $\{\mathbf{x}_i\mathbf{x}_i^\top\}_{i=1,\dots,n}$ and the matrices $\mathbf{R}$ and $\{\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla_f^2 c(y_i, f(\boldsymbol{\Theta}, \mathbf{x}_i))]\}_{i=1,\dots,n}$ are simultaneously diagonalizable, then the optimal posterior q among all multivariate Gaussians is a SOAP-Bubble, that is, in EVON's variational family.*

Corollary 1. *Assume a linear model, and an isotropic prior. Assume datapoints/features $\mathbf{x}_i$ are pairwise orthogonal or parallel, and fix the optimizer batch size to 1. For binary logistic regression and for general linear regression, the exact full Gaussian posterior among all multivariate Gaussians is a SOAP-Bubble, that is, in EVON's variational family.*

4 Experiments

4.1 Illustrative Examples

We illustrate the intuition behind EVON on a simple logistic regression problem with two-dimensional features in Fig. 1. IVON uses a diagonal posterior covariance, visualized as an ellipse in the figure. This forces the ellipse to be axis aligned which causes the posterior mean to move far away from the

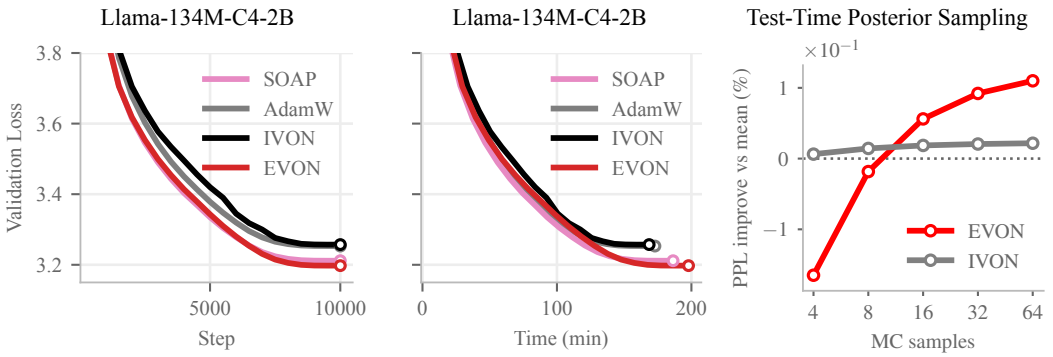


Figure 5: EVON also works well on Llama models, improving significantly over IVON training, similar to how SOAP improves over AdamW. EVON's final validation loss is lower than SOAP's and training with EVON provides a variational posterior on top for free. Averaging next-token probabilities of random models drawn from EVON's SOAP-Bubble before predicting the next token decreases the test-loss more than using IVON's posterior.

mode. EVON's rotated space aligns well with the shape of the posterior, allowing for a rotated ellipse that captures more of the posterior mass.

In Fig. 4 (left), we run IVON and EVON on one large and one small fully-connected neural network on a simple regression task. Due to the limited capacity of the small neural network, the uncertainty estimates from IVON's diagonal posterior lack diversity. EVON's non-diagonal posterior finds highly diverse solutions in regions where there is little data which highlights the benefits of non-diagonal posterior covariances for neural networks. Details for both of these experiments are found in Sec. E.1.

4.2 Bayesian Logistic Regression

To study the expressiveness of EVON's variational family, we consider a binary Bayesian logistic regression problem on the USPS digit classification task next. The experiment follows the setup of Mishkin et al. (2018, Fig. 1). We compare EVON to an exact full Gaussian posterior and show the results in Fig. 3. We find that both the covariance structure and variational bound attained match those of the full Gaussian posterior. These results provide empirical evidence of accurate Gaussian approximation beyond our sufficient theoretical conditions in Thm. 2. A diagonal posterior used in IVON leads to a worse variational bound and slower convergence. SLANG (Mishkin et al., 2018) can get closer to the full Gaussian posterior when the rank is sufficiently high but has so far not been shown to scale to larger problems. Rank 10 SLANG still performs worse than EVON.

We also consider a multiclass Bayesian logistic regression problem on the UCI Iris dataset on the right hand side of Fig. 4. There, we compute reference posterior marginals using sampling (Hoffman and Gelman, 2014). EVON's Gaussian marginals closely match these reference marginals, whereas IVON's marginals are highly concentrated. We report details for these experiments in Sec. E.2.

4.3 Language Model Pre-Training

Our goal is to show that EVON can provide good performance on larger scales, for example, for language model pretraining. To show this, we pretrain language models using NanoGPT (Jordan et al., 2024a) (123M parameters) in Fig. 1 (middle) and the LLaMA-based architecture from Glentis et al. (2025) (134M parameters) with EVON, following the same setup as Lin et al. (2026) in Fig. 5, and compare it to AdamW, IVON, and SOAP. We fix training hyperparameters like the token budget, batch size, learning-rate schedule, and data pipeline across all methods and only swap the optimizer. We report details for these experiments in Sec. E.3.

In Fig. 1, we compare IVON and EVON on NanoGPT pre-training on a 1B subset of FineWeb (Penedo et al., 2024). EVON demonstrates superior performance by quickly reaching a lower loss in the initial phase of the optimization, and maintains a large gap up until the end of training. We additionally report the perplexity (PPL) improvement relative to each method's posterior mean as a function of the number of Monte Carlo samples used for Bayesian model-averaging on a held-out validation set.

Metric	Optimizer	CARS	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
Acc. ↑	AdamW	$84.03_{2.75}$	$81.20_{1.29}$	$98.32_{0.44}$	$98.97_{0.11}$	$99.51_{0.13}$	$95.61_{0.25}$	$77.85_{0.22}$	$96.76_{0.37}$	91.53
	IVON	$87.11_{0.39}$	$80.48_{0.43}$	$98.71_{0.12}$	$98.71_{0.11}$	$99.52_{0.12}$	$95.17_{0.32}$	$77.21_{0.11}$	$96.98_{0.34}$	91.74
	—"— @32	$87.32_{0.53}$	$80.60_{0.39}$	$98.78_{0.07}$	$98.74_{0.08}$	$99.53_{0.12}$	$95.23_{0.25}$	$77.27_{0.11}$	$97.01_{0.32}$	91.81
	SOAP	$81.39_{4.05}$	$\mathbf{82.98}_{0.89}$	$\mathbf{98.98}_{0.08}$	$98.40_{0.14}$	$99.55_{0.09}$	$95.74_{0.03}$	$77.51_{0.09}$	$96.92_{0.15}$	91.43
	+BayesVLM	$85.82_{5.19}$	$82.70_{0.61}$	$98.96_{0.05}$	$98.36_{0.31}$	$99.55_{0.03}$	$\mathbf{95.96}_{0.18}$	$77.61_{0.13}$	$96.94_{0.27}$	91.99
	EVON	$88.11_{0.45}$	$82.15_{0.36}$	$98.72_{0.13}$	$99.02_{0.13}$	$99.60_{0.06}$	$95.31_{0.82}$	$78.84_{0.24}$	$97.35_{0.19}$	92.39
	—"— @32	$\mathbf{88.51}_{0.47}$	$81.99_{0.26}$	$98.87_{0.11}$	$\mathbf{99.11}_{0.15}$	$\mathbf{99.63}_{0.05}$	$95.79_{0.22}$	$\mathbf{79.28}_{0.16}$	$\mathbf{97.41}_{0.16}$	**92.57**
NLL ↓	AdamW	$0.564_{0.086}$	$0.813_{0.043}$	$0.055_{0.012}$	$0.046_{0.010}$	$0.031_{0.005}$	$0.156_{0.007}$	$0.854_{0.018}$	$0.133_{0.011}$	0.33
	IVON	$\mathbf{0.409}_{0.010}$	$0.731_{0.006}$	$0.042_{0.006}$	$0.053_{0.007}$	$0.033_{0.007}$	$0.183_{0.008}$	$0.791_{0.006}$	$0.132_{0.015}$	0.30
	—"— @32	$0.411_{0.013}$	$0.670_{0.013}$	$\mathbf{0.034}_{0.003}$	$0.051_{0.007}$	$0.032_{0.007}$	$0.164_{0.009}$	$0.775_{0.003}$	$0.129_{0.014}$	0.28
	SOAP	$0.675_{0.160}$	$0.696_{0.014}$	$0.042_{0.005}$	$0.070_{0.010}$	$0.035_{0.004}$	$0.161_{0.009}$	$0.832_{0.013}$	$0.141_{0.005}$	0.33
	+BayesVLM	$0.492_{0.176}$	$0.672_{0.010}$	$0.041_{0.003}$	$0.068_{0.018}$	$0.031_{0.003}$	$0.145_{0.007}$	$0.841_{0.001}$	$0.135_{0.012}$	0.30
	EVON	$0.415_{0.010}$	$0.787_{0.013}$	$0.044_{0.005}$	$0.059_{0.010}$	$0.027_{0.002}$	$0.180_{0.010}$	$0.758_{0.004}$	$0.129_{0.008}$	0.30
	—"— @32	$\mathbf{0.409}_{0.008}$	$\mathbf{0.633}_{0.005}$	$0.035_{0.002}$	$\mathbf{0.035}_{0.005}$	$\mathbf{0.026}_{0.001}$	$\mathbf{0.123}_{0.007}$	$\mathbf{0.695}_{0.003}$	$\mathbf{0.117}_{0.005}$	**0.26**
ECE ↓ (×100)	AdamW	$4.92_{0.60}$	$9.21_{0.69}$	$0.89_{0.12}$	$\mathbf{0.29}_{0.06}$	$1.01_{0.02}$	$2.21_{0.09}$	$9.03_{0.57}$	$0.78_{0.20}$	3.54
	IVON	$\mathbf{1.67}_{0.41}$	$4.93_{0.40}$	$0.61_{0.15}$	$0.59_{0.09}$	$1.07_{0.18}$	$2.50_{0.20}$	$4.84_{0.24}$	$\mathbf{0.69}_{0.15}$	2.11
	—"— @32	$5.76_{0.54}$	$\mathbf{2.47}_{0.52}$	$\mathbf{0.43}_{0.08}$	$0.60_{0.11}$	$1.09_{0.18}$	$1.64_{0.22}$	$3.75_{0.18}$	$0.79_{0.14}$	**2.07**
	SOAP	$5.46_{1.10}$	$7.04_{0.78}$	$0.71_{0.08}$	$0.75_{0.14}$	$0.85_{0.04}$	$2.48_{0.15}$	$7.57_{0.68}$	$\mathbf{0.69}_{0.08}$	3.19
	+BayesVLM	$2.73_{1.06}$	$4.94_{0.55}$	$0.73_{0.03}$	$0.47_{0.05}$	$\mathbf{0.40}_{0.05}$	$2.11_{0.20}$	$7.54_{0.12}$	$0.85_{0.39}$	2.47
	EVON	$3.89_{0.22}$	$8.86_{0.26}$	$0.57_{0.10}$	$0.58_{0.13}$	$0.95_{0.03}$	$2.70_{0.18}$	$6.54_{0.20}$	$0.98_{0.12}$	3.13
	—"— @32	$6.39_{0.61}$	$3.53_{0.32}$	$0.67_{0.05}$	$0.86_{0.15}$	$1.49_{0.04}$	$\mathbf{0.51}_{0.17}$	$\mathbf{1.30}_{0.18}$	$2.73_{0.30}$	2.19
Brier ↓ (×10)	AdamW	$2.35_{0.38}$	$2.92_{0.16}$	$0.26_{0.06}$	$0.15_{0.02}$	$0.09_{0.02}$	$0.69_{0.03}$	$3.30_{0.05}$	$0.52_{0.06}$	1.3
	IVON	$1.88_{0.07}$	$2.91_{0.04}$	$0.20_{0.02}$	$0.18_{0.02}$	$0.08_{0.02}$	$0.77_{0.04}$	$3.27_{0.01}$	$0.49_{0.06}$	1.2
	—"— @32	$1.91_{0.08}$	$2.82_{0.06}$	$0.18_{0.01}$	$0.18_{0.02}$	$0.08_{0.02}$	$0.73_{0.04}$	$3.24_{0.01}$	$0.48_{0.06}$	1.2
	SOAP	$2.71_{0.56}$	$2.63_{0.08}$	$\mathbf{0.17}_{0.01}$	$0.24_{0.03}$	$0.08_{0.01}$	$0.67_{0.02}$	$3.29_{0.02}$	$0.51_{0.02}$	1.3
	+BayesVLM	$2.07_{0.71}$	$\mathbf{2.58}_{0.05}$	$\mathbf{0.17}_{0.01}$	$0.24_{0.05}$	$0.08_{0.00}$	$0.64_{0.02}$	$3.30_{0.00}$	$0.50_{0.04}$	1.2
	EVON	$\mathbf{1.77}_{0.05}$	$2.81_{0.02}$	$0.20_{0.01}$	$0.17_{0.02}$	$\mathbf{0.07}_{0.01}$	$0.71_{0.03}$	$3.08_{0.01}$	$0.45_{0.03}$	1.2
	—"— @32	$1.82_{0.05}$	$2.60_{0.02}$	$0.18_{0.00}$	$\mathbf{0.13}_{0.02}$	$\mathbf{0.07}_{0.00}$	$\mathbf{0.62}_{0.03}$	$\mathbf{2.96}_{0.02}$	$\mathbf{0.43}_{0.02}$	**1.1**

Table 1: We finetune CLIP ViT-B/16 on 8 classification datasets. EVON is highlighted in gray, the best scores are **bolded**, and the second best scores are in red. When compared to IVON, EVON often provides improvements in terms of Accuracy, Negative Log-Likelihood (NLL), Expected Calibration Error (ECE) and Brier scores. For variational optimizers, we can improve scores by averaging the predictions of 32 model samples (BMA). EVON's gains are often bigger compared to IVON's, suggesting that the estimated posterior is more informative. We also compare to BayesVLM (Baumann et al., 2026), applied post-hoc on SOAP checkpoints (denoted by "+").

For very small sample counts, averaging is not yet beneficial, but with more samples both methods improve over their posterior means. EVON delivers larger gains and scales better with the number of samples, whereas IVON's improvements quickly saturate. This indicates that EVON's non-diagonal posterior provides more useful uncertainty estimates that could be employed further for test-time scaling, where IVON's posterior has already provided improvements (Cong et al., 2025).

In Fig. 5 (left, middle), we compare IVON, SOAP, AdamW and EVON on LLaMA training. We train all models on the same 2B token subset of C4 (Raffel et al., 2020). Just as SOAP improves upon AdamW, EVON improves over IVON and reaches an overall lower loss than SOAP, while also providing a variational posterior. The variational posterior is immediately useful to get better results with more inference compute, and Fig. 5 shows that averaging softmax predictions of posterior ensembles for predicting the next token reduces the overall validation loss. We observe a bigger reduction in test loss for EVON than for IVON, again suggesting that the non-diagonal posterior contains better uncertainty estimates.

To verify that these findings hold at larger scale, we additionally pretrain a 350M LLaMA model under the same protocol (Fig. 8 in the appendix). EVON again achieves the lowest final validation loss, confirming that the advantage over SOAP and IVON is preserved at increased model scale.

4.4 Fine-Tuning CLIP Models

Finally, we use EVON to finetune CLIP models (Radford et al., 2021) on several downstream tasks in the visual domain. We initialize the model with the OpenAI ViT-B/16 backbone pre-trained using the AdamW optimizer. We then train the linear layers of the ViT model (99% of parameters) on 8 visual classification benchmarks. We list these benchmarks in Sec. E.4 along with further experimental details. We evaluate all methods in terms of downstream accuracy, negative log-likelihood (NLL), expected calibration error (ECE, Naeini et al. (2015)), and Brier score (Brier, 1950) and compare EVON to AdamW, IVON, SOAP, and BayesVLM (Baumann et al., 2026).

Results are summarized in Table 1. EVON performs on par with the best performing optimizers and sometimes outperforms them, especially in terms of NLL, ECE and Brier score. To test the effectiveness of the estimated posteriors, we perform Bayesian Model Averaging, which we report as EVON@32 in the table (indicating that we use 32 Monte Carlo samples). Same applies to IVON@32. We also compare to the post-hoc probabilistic method BayesVLM (Baumann et al., 2026), applied on top of SOAP checkpoints.

As shown, variational optimizers provide comparable or better performance, even with pre-trained checkpoints obtained from a non-variational optimizer (AdamW). EVON is especially effective at improving over the best accuracy of the non-variational counterparts. At the same time, when using Bayesian Model Averaging, calibration and uncertainty quantification performance remains strong. For IVON, averaging predictions only brings marginal improvements compared to the mean, perhaps due to lack of expressiveness of the diagonal covariances. EVON@32 achieves the highest average accuracy and the lowest average NLL and Brier score among the evaluated methods. Calibration results are mixed: IVON@32 has the lowest average ECE, and Bayesian model averaging does not improve ECE on every dataset. Following Baan et al. (2022), these deviations may stem from the reliance on hard labels or finite-sample bias. We find that ECE alone can be misleading when assessing calibration: posterior averaging reduces maximum-class confidence, which can worsen standard ECE even when proper scoring rules such as NLL and Brier score improve (Guo et al., 2017). Implementation details can be found in Sec. E.4.

5 Discussion and Limitations

Scaling variational learning with structured weight uncertainty to large neural networks has traditionally been a hard problem. Our proposed EVON method offers a simple solution, requiring only a few modifications to the contemporary second-order deep learning optimizer SOAP. Our results further confirm a fundamental connection between second-order optimization and Gaussian variational Bayesian methods (Opper and Archambeau, 2009; Khan et al., 2018; Lin, 2023). Thus, further advances in second-order optimization are expected to lead to better variational learning methods, and vice versa, variational Bayesian principles can help design better optimizers.

Despite being scalable to large problems, EVON also works surprisingly well on classical Bayesian inference problems, and we have provided some theoretical justifications for that in Thm. 2 and Cor. 1. Still, these rely on simultaneous-diagonalization and orthogonality assumptions that are unlikely to hold exactly in practical deep-learning settings. We view these results primarily as intuition: they show that EVON's structured covariance family is expressive enough to closely match the optimal full-Gaussian variational posterior in controlled settings, and they help explain the strong empirical performance observed in larger-scale problems where the assumptions no longer apply. A general characterization of the objective under a moving eigenbasis remains an open question.

While our focus is on scaling variational Bayesian training to large deep-learning models, the non-deep-learning experiments we conducted suggest that EVON often approaches a reference full-Gaussian solution. Applying EVON-like methods to classical Bayesian models, such as hierarchical models, Gaussian processes, or state-space models, is a promising direction for future work.

For language model training, EVON can even surpass SOAP's final validation loss while offering additional SOAP-Bubble posteriors for free. These can be used to improve validation loss during test-time and could also be useful in downstream applications such as continual learning (Nguyen et al., 2018), model understanding (Nickl et al., 2023), language generation (Daheim et al., 2025) or exploration and data collection in open-ended settings (Silver and Sutton, 2025). These are all promising avenues for future research.

In distributed settings, EVON requires a careful implementation to minimize computational overhead compared to Adam. Since EVON is structurally similar to SOAP, this can be addressed by building on top of improvements over SOAP or Shampoo (Shi et al., 2023; Lin et al., 2026). Like any other variational learning method, EVON also introduces an extra hyperparameter, ζ, from the variational objective that must be tuned carefully. Setting ζ too large causes the posterior to collapse, while setting it too small leads to unstable training. Developing scaling laws to predict how to optimally choose ζ for a given model and dataset size remains an important direction for future work.

Acknowledgements

We thank Cong Bai for helpful discussions and help with implementing an early version of EVON. This work is supported by the Bayes duality project, JST CREST Grant Number JPMJCR2112. Thomas Moellenhoff acknowledges the support of JSPS KAKENHI Grant Number 26H02541. This research work has been funded by the German Federal Ministry of Research, Technology and Space and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. Adrian Robert Minut acknowledges support from Sapienza University of Rome, under Startup Research project TVT (Task Vector Transfer across heterogeneous models).

Author Contribution

Authors: Adrian R. Minut (ARM), Nico Carlsen Daheim (ND), Marco Miani (MM), Mohammad Emtiyaz Khan (MEK), Wu Lin (WL), Thomas Moellenhoff (TM).

MEK and TM discussed the possibility of variational training in a subspace. This led TM to develop a first version of the EVON algorithm and run some proof-of-concept experiments. With feedback from WL, ND and TM, ARM refined the method and implemented a scalable variant. ARM conducted all the experiments on Llama, nanoGPT and ViT with help from ND, WL and TM. TM did the small-scale illustrative experiments with suggestions from MEK. MM derived the theoretical results, with feedback from TM. All authors contributed to writing the paper, as well as proof-reading.

References

Baan, J., Aziz, W., Plank, B., and Fernandez, R. (2022). Stop measuring calibration when humans disagree. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1892–1915, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bae, J., Zhang, G., and Grosse, R. (2018). Eigenvalue corrected noisy natural gradient. In *Neural Information Processing Systems (Bayesian Deep Learning Workshop)*.

Baumann, A., Li, R., Klasson, M., Mentu, S., Karthik, S., Akata, Z., Solin, A., and Trapp, M. (2026). Post-hoc probabilistic vision-language models. In *International Conference on Learning Representations (ICLR)*.

Brier, G. W. (1950). Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1–3.

Carlson, D. E., Collins, E., Hsieh, Y.-P., Carin, L., and Cevher, V. (2015). Preconditioned spectral descent for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, 76:1–32.

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

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cong, B., Daheim, N., Shen, Y., Yokota, R., Khan, M. E., and Möllenhoff, T. (2025). Improving LoRA with variational learning. *arXiv:2506.14280*.

Cover, T. M. and Thomas, J. A. (2006). *Elements of information theory*. John Wiley & Sons.

Daheim, N., Meister, C., Möllenhoff, T., and Gurevych, I. (2025). Uncertainty-aware decoding with minimum Bayes risk. In *International Conference on Learning Representations (ICLR)*.

Daheim, N., Möllenhoff, T., Ponti, E., Gurevych, I., and Khan, M. E. (2024). Model merging by uncertainty-based gradient matching. In *International Conference on Learning Representations (ICLR)*.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021). Laplace redux–effortless Bayesian deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Dhahri, R., Immer, A., Charpentier, B., Günnemann, S., and Fortuin, V. (2024). Shaving weights with Occam's razor: Bayesian sparsification for neural networks using the marginal likelihood. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Eschenhagen, R., Defazio, A., Lee, T.-H., Turner, R. E., and Shi, H.-J. M. (2025). Purifying Shampoo: Investigating Shampoo's heuristics by decomposing its preconditioner. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Eschenhagen, R., Immer, A., Turner, R., Schneider, F., and Hennig, P. (2023). Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Fadel, S. G., Roy, H., Krämer, N., Zainchkovskyy, Y., Syrota, S., Mahou, A. V., Ek, C. H., and Hauberg, S. (2025). VIKING: Deep variational inference with stochastic projections. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gan, Y. and Isola, P. (2026). Neural thickets: Diverse task experts are dense around pretrained weights. *arXiv:2603.12228*.

Glentis, A., Li, J., Han, A., and Hong, M. (2025). A minimalist optimizer design for LLM pretraining. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*.

Gupta, V., Koren, T., and Singer, Y. (2018). Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*.

Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226.

Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Hong, D. M., Mlodozeniec, B. K., Eschenhagen, R., and Turner, R. E. (2025). Better hessians matter: Studying the impact of curvature approximations in influence functions. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

Huggins, J., Kasprzak, M., Campbell, T., and Broderick, T. (2020). Validated variational inference via practical posterior error bounds. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan Mohammad, E. (2021). Scalable marginal likelihood estimation for model selection in deep learning. In *International Conference on Machine Learning (ICML)*.

Jones, M., Chang, P., and Murphy, K. (2024). Bayesian online natural gradient (BONG). In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jordan, K., Bernstein, J., Rappazzo, B., @fernbear.bsky.social, Boza, V., You, J., Cesista, F., Koszarsky, B., and @Grad62304977 (2024a). modded-nanoGPT: Speedrunning the NanoGPT baseline.

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. (2024b). Muon: An optimizer for hidden layers in neural networks.

Kasimbeg, P., Schneider, F., Eschenhagen, R., Bae, J., Sastry, C. S., Saroufim, M., Feng, B., Wright, L., Yang, E. Z., Nado, Z., et al. (2025). Accelerating neural network training: An analysis of the AlgoPerf competition. In *International Conference on Learning Representations (ICLR)*.

Khan, M. E. (2025). Knowledge adaptation as posterior correction. *arXiv:2506.14262*.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In *International Conference on Machine Learning (ICML)*.

Khan, M. E. and Rue, H. (2023). The Bayesian learning rule. *J. Mach. Learn. Res. (JMLR)*, 24(281):1–46.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3D object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*.

Lin, W. (2023). *Computationally efficient geometric methods for optimization and inference in machine learning*. PhD thesis, University of British Columbia.

Lin, W., Khan, M. E., and Schmidt, M. (2019a). Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *International Conference on Machine Learning (ICML)*.

Lin, W., Khan, M. E., and Schmidt, M. (2019b). Stein's lemma for the reparameterization trick with exponential family mixtures.

Lin, W., Lowe, S. C., Dangel, F., Eschenhagen, R., Xu, Z., and Grosse, R. B. (2026). Understanding and improving Shampoo and SOAP via Kullback-Leibler minimization. In *International Conference on Learning Representations (ICLR)*.

Lin, W., Nielsen, F., Emtiyaz, K. M., and Schmidt, M. (2021). Tractable structured natural-gradient descent using local parameterizations. In *International Conference on Machine Learning (ICML)*.

Lin, W., Schmidt, M., and Khan, M. E. (2020). Handling the positive-definite constraint in the Bayesian learning rule. In *International Conference on Machine Learning (ICML)*.

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. (2024). Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations (ICLR)*.

Louizos, C. and Welling, M. (2016). Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning (ICML)*.

Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M., and Khan, M. E. (2018). SLANG: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Möllenhoff, T., Swaroop, S., Doshi-Velez, F., and Khan, M. E. (2025). Federated ADMM from Bayesian duality. In *International Conference on Learning Representations (ICLR)*.

Morwani, D., Shapira, I., Vyas, N., Malach, E., Kakade, S. M., and Janson, L. (2025). A new perspective on Shampoo's preconditioner. In *International Conference on Learning Representations (ICLR)*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Naeini, M. P., Cooper, G., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 29.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations (ICLR)*.

Nickl, P., Xu, L., Tailor, D., Möllenhoff, T., and Khan, M. E. (2023). The memory-perturbation equation: Understanding model's sensitivity to data. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Opper, M. and Archambeau, C. (2009). The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. (2019). Practical deep learning with Bayesian principles. *Advances in Neural Information Processing Systems (NeurIPS)*.

Penedo, G., Kydlíček, H., Lozhkov, A., Mitchell, M., Raffel, C., Von Werra, L., Wolf, T., et al. (2024). The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. In *J. Mach. Learn. Res. (JMLR)*, volume 21, pages 1–67.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559.

Shen, Y., Daheim, N., Cong, B., Nickl, P., Marconi, G. M., Bazan, C., Yokota, R., Gurevych, I., Cremers, D., Khan, M. E., and Möllenhoff, T. (2024). Variational learning is effective for large deep networks. In *International Conference on Machine Learning (ICML)*.

Shi, H.-J. M., Lee, T.-H., Iwasaki, S., Gallego-Posada, J., Li, Z., Rangadurai, K., Mudigere, D., and Rabbat, M. (2023). A distributed data-parallel PyTorch implementation of the distributed Shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*.

Silver, D. and Sutton, R. S. (2025). Welcome to the era of experience. *Google AI*, 1:11.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332.

van der Ouderaa, T. F., van der Wilk, M., and De Haan, P. (2024). Noether's razor: Learning conserved quantities. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Vyas, N., Morwani, D., Zhao, R., Shapira, I., Brandfonbrener, D., Janson, L., and Kakade, S. M. (2025). SOAP: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations (ICLR)*.

Wang, Y., Shi, H., Han, L., Metaxas, D., and Wang, H. (2024). Blob: Bayesian low-rank adaptation by backpropagation for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2014). Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119:3–22.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2018). Noisy natural gradient as variational inference. In *International Conference on Machine Learning (ICML)*.

A Proof of Thm. 1

Theorem 1. *For the above variational families $\mathcal{Q}_{diag}$ and $\mathcal{Q}_{bubble}$, the following holds:*

$$\begin{aligned}(1) \;&= \min_{q\in\mathcal{Q}_{\text{bubble}}} \zeta\,\mathbb{E}_{q(\boldsymbol{\theta})}[\ell(\boldsymbol{\theta})] + \mathrm{KL}(q(\boldsymbol{\theta})\,\|\,\mathcal{N}(\boldsymbol{\theta}\mid\mathbf{m}_0,\boldsymbol{\Sigma}_0)) \\ &= \min_{q\in\mathcal{Q}_{\text{diag}}} \zeta\,\mathbb{E}_{q(\mathbf{w})}[\ell(\mathbf{P}\mathbf{w})] + \mathrm{KL}\Big(q(\mathbf{w})\,\|\,\mathcal{N}(\mathbf{w}\mid\mathbf{P}^\top\mathbf{m}_0,\mathbf{P}^\top\boldsymbol{\Sigma}_0\mathbf{P})\Big).\end{aligned} \tag{7}$$

Proof. The main idea is that any $q\in\mathcal{Q}_{\text{bubble}}$ can be written as a linear transformation (pushforward) $\mathbf{P}_\sharp q'$ for a distribution $q'\in\mathcal{Q}_{\text{diag}}$. To see this, recall that a linear transformation of a Gaussian is another Gaussian, that is, $\mathbf{P}_\sharp\mathcal{N}(\mathbf{m}',\boldsymbol{\Sigma}') = \mathcal{N}(\mathbf{P}\mathbf{m}',\mathbf{P}\boldsymbol{\Sigma}'\mathbf{P}^\top)$. Since any member of $\mathcal{Q}_{\text{bubble}}$ is obtained that way, we can rewrite the variational objective in Eq. 1 over the structured family as a minimization over the pushforward of the diagonal family:

$$(1) \;= \min_{q'\in\mathcal{Q}_{\text{diag}}} \zeta\,\mathbb{E}_{\boldsymbol{\theta}\sim\mathbf{P}_\sharp q'}[\ell(\boldsymbol{\theta})] + \mathrm{KL}(\mathbf{P}_\sharp q'\,\|\,\mathcal{N}(\mathbf{m}_0,\boldsymbol{\Sigma}_0)). \tag{9}$$

Using the fact that the pushforward is adjoint to the pullback, we can rewrite the expected loss as $\mathbb{E}_{\boldsymbol{\theta}\sim\mathbf{P}_\sharp q'}[\ell(\boldsymbol{\theta})] = \mathbb{E}_{\mathbf{w}\sim q'}[\ell(\mathbf{P}\mathbf{w})]$, which gives us the first part of the statement. For the KL-term, we use the fact that the KL-divergence is invariant under invertible transformations $T_\sharp$ of both measures, that is: $\mathrm{KL}(\mathbf{P}_\sharp q'\,\|\,\mathcal{N}(\mathbf{m}_0,\boldsymbol{\Sigma}_0)) = \mathrm{KL}(\mathbf{T}_\sharp\mathbf{P}_\sharp q'\,\|\,\mathbf{T}_\sharp\mathcal{N}(\mathbf{m}_0,\boldsymbol{\Sigma}_0))$. This is the equality case in the data-processing inequality, when all information is preserved due to invertibility (Cover and Thomas, 2006). The result also holds for more general divergences, and for self-contained proof, see for example Huggins et al. (2020, Lemma 4.1). Setting $\mathbf{T} = \mathbf{P}^{-1} = \mathbf{P}^\top$ and using $\mathbf{T}_\sharp\mathbf{P}_\sharp = \mathrm{id}_\sharp$, we get,

$$\mathrm{KL}(\mathbf{T}_\sharp\mathbf{P}_\sharp q'\,\|\,\mathbf{T}_\sharp\mathcal{N}(\mathbf{m}_0,\boldsymbol{\Sigma}_0)) = \mathrm{KL}\Big(\mathrm{id}_\sharp q'\,\|\,\mathbf{P}_\sharp^\top\mathcal{N}(\mathbf{m}_0,\boldsymbol{\Sigma}_0)\Big) \tag{10}$$

$$= \mathrm{KL}\Big(q'\,\|\,\mathcal{N}(\mathbf{P}^\top\mathbf{m}_0,\mathbf{P}^\top\boldsymbol{\Sigma}_0\mathbf{P})\Big), \tag{11}$$

which concludes the proof. □

B Derivation of the EVON Algorithm

In this appendix, we derive the proposed EVON method and show how it leads to the SOAP-like updates of Alg. 2. To derive EVON, we start from the variational online Newton (VON) method in Khan and Rue (2023, Eq. 12) applied to the variational objective in $q(\mathbf{w})$, which is Eq. 7 in the main text. The VON method directly targets Eq. 7. For a generic Gaussian family $\mathcal{N}(\boldsymbol{\theta}\mid\mathbf{m},\mathrm{diag}(\mathbf{s})^{-1})$ with diagonal covariance and isotropic Gaussian prior, $\mathbf{m}_0 = 0$ and $\boldsymbol{\Sigma}_0 = s_0^{-1}\mathbf{I}$ the method reads, with $\mathbf{1}$ the all-ones vector:

$$\mathbf{s} \leftarrow (1-\rho)\mathbf{s} + \rho\left[\zeta\mathbb{E}_{q(\boldsymbol{\theta})}[\mathrm{diag}(\nabla^2\ell(\boldsymbol{\theta}))] + s_0\mathbf{1}\right], \tag{12}$$

$$\mathbf{m} \leftarrow \mathbf{m} - \rho\mathrm{diag}(\mathbf{s})^{-1}\left(\zeta\mathbb{E}_{q(\boldsymbol{\theta})}[\nabla\ell(\boldsymbol{\theta})] + s_0\mathbf{m}\right), \tag{13}$$

where $\rho > 0$ is a learning rate.

Now, we apply the above VON updates to the reparametrized problem in $\mathbf{w}$ from Thm. 1, which leads to the following updates:

$$\mathbf{s}' \leftarrow (1-\rho)\mathbf{s}' + \rho\left(\zeta\mathbb{E}_{q(\mathbf{w})}[\mathrm{diag}(\mathbf{P}^\top\nabla^2\ell(\mathbf{P}\mathbf{w})\mathbf{P})] + s_0\mathbf{1}\right), \tag{14}$$

$$\mathbf{m}' \leftarrow \mathbf{m}' - \rho\,\mathrm{diag}(\mathbf{s}')^{-1}\left(\zeta\mathbf{P}^\top\mathbb{E}_{q(\mathbf{w})}[\nabla\ell(\mathbf{P}\mathbf{w})] + s_0\mathbf{m}'\right). \tag{15}$$

Here, we used the chain-rule for gradient and Hessian for the $\ell(\mathbf{P}\mathbf{w})$. We will now use Price's theorem (Lin et al., 2019b, Theorem 4) to write the Hessian in terms of gradients. Also using stochastic gradients, both for minibatches and the expectations, and writing $\mathbf{h} = (\mathbf{s}' - s_0)/\zeta$, $s_0 = \zeta\delta$, and using short-hand notation $\mathbf{g}^\circ$ for the projected gradient, the above can be simplified:

$$\mathbf{g} \leftarrow \widehat{\nabla}\ell(\mathbf{P}(\mathbf{m}'+\boldsymbol{\epsilon})), \mathbf{g}^\circ \leftarrow \mathbf{P}^\top\mathbf{g}, \quad \boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon}\mid 0,\boldsymbol{\sigma}^2) \tag{16}$$

$$\mathbf{h} \leftarrow (1-\rho)\mathbf{h} + \rho\mathbf{g}^\circ\boldsymbol{\epsilon}/\boldsymbol{\sigma}^2, \tag{17}$$

$$\mathbf{m}' \leftarrow \mathbf{m}' - \rho\,\mathrm{diag}(\mathbf{h}+\delta)^{-1}\left(\mathbf{g}^\circ + \delta\mathbf{m}'\right). \tag{18}$$

Multiplying the last line from the left with $\mathbf{P}$, we can bring the update to the original $\boldsymbol{\theta}$-space:

$$\mathbf{m} \leftarrow \mathbf{m} - \rho\,\mathbf{P}\text{diag}(\mathbf{h}+\delta)^{-1}\left(\mathbf{g}^\circ + \delta\mathbf{P}^\top\mathbf{m}\right). \tag{19}$$

To arrive at the form shown in Alg. 2, we apply the method to neural networks with L layers, where the weights $\boldsymbol{\theta} = \text{cat}(\text{vec}(\boldsymbol{\Theta}_1), \ldots, \text{vec}(\boldsymbol{\Theta}_L))$ are assumed to be a concatenation of matrix parameters $\boldsymbol{\Theta}_l$ flattened into vectors via $\text{vec}(\cdot)$. The inverse of $\text{vec}(\cdot)$ is denoted by $\text{mat}(\cdot)$. We also assume $\mathbf{P}$ to be a block-diagonal matrix with blocks $\mathbf{P}_1, \ldots, \mathbf{P}_L$ such that $\mathbf{P}_l\boldsymbol{\theta}_l = \text{vec}(\mathbf{Q}_L^l\boldsymbol{\Theta}_l(\mathbf{Q}_R^l)^\top)$ with $\boldsymbol{\theta}_l = \text{vec}(\boldsymbol{\Theta}_l)$, that is, $\mathbf{P}_l = \mathbf{Q}_R^l \otimes \mathbf{Q}_L^l$ has Kronecker-factored structure.

To simplify notation, following (Vyas et al., 2025), we write the method for a single matrix parameter and drop the index l. Capital letters denote matrix version of the parameter that was vectorized before.

$$\mathbf{G} \leftarrow \widehat{\nabla}\ell(\mathbf{M} + \mathbf{Q}_L\mathbf{E}\mathbf{Q}_R^\top), \quad \mathbf{G}^\circ \leftarrow \mathbf{Q}_L^\top\mathbf{G}\mathbf{Q}_R, \quad E_{i,j} \sim \mathcal{N}(0, V_{i,j}) \tag{20}$$

$$\mathbf{H} \leftarrow (1-\rho)\mathbf{H} + \rho\mathbf{G}^\circ \odot \mathbf{E}/\mathbf{V}, \tag{21}$$

$$\mathbf{M} \leftarrow \mathbf{M} - \rho\mathbf{Q}_L\left[\left(\mathbf{G}^\circ + \delta\mathbf{Q}_L^\top\mathbf{M}\mathbf{Q}_R\right)/(\mathbf{H}+\delta)\right]\mathbf{Q}_R^\top, \tag{22}$$

where $V_{i,j} = 1/\zeta(H_{i,j}+\delta)$ are the entries of $\mathbf{V} = \text{mat}(\boldsymbol{\sigma}^2)$. Importantly, the matrix $\mathbf{V}$ is not to be confused with the posterior covariance in the original space. It is the diagonal variance $\boldsymbol{\sigma}^2$ in the rotated space, reshaped in matrix form. The posterior distribution in the original space for that layer is written in Eq. 8 in the main text. The posterior covariance over all parameters is a block-diagonal matrix with one such block per layer. Each block $(\mathbf{Q}_R \otimes \mathbf{Q}_L)\,\text{diag}(\text{vec}(\mathbf{V}))\,(\mathbf{Q}_R \otimes \mathbf{Q}_L)^\top$ has a Kronecker-factored eigenbasis, but is itself not a Kronecker product in general: it is one only if $\mathbf{V} = \mathbf{a}\mathbf{b}^\top$ has rank one, in which case it equals $(\mathbf{Q}_R\text{diag}(\mathbf{b})\mathbf{Q}_R^\top) \otimes (\mathbf{Q}_L\text{diag}(\mathbf{a})\mathbf{Q}_L^\top)$. Since the eigenvalues $\mathbf{V}$ are unrestricted, SOAP-Bubbles strictly contain the Kronecker-factored covariances with the same eigenbasis.

To arrive at EVON (Alg. 2), we introduce a few more tricks. We use a different learning rate β_2 for the Hessian and introduce a gradient momentum $\bar{\mathbf{G}}$, which leads to the following updates:

$$\mathbf{G} \leftarrow \widehat{\nabla}\ell(\mathbf{M} + \mathbf{Q}_L\mathbf{E}\mathbf{Q}_R^\top), \quad \mathbf{G}^\circ \leftarrow \mathbf{Q}_L^\top\mathbf{G}\mathbf{Q}_R, \quad E_{i,j} \sim \mathcal{N}(0, V_{i,j}) \tag{23}$$

$$\widehat{\mathbf{H}} \leftarrow \mathbf{G}^\circ \odot \mathbf{E}/\mathbf{V}, \tag{24}$$

$$\bar{\mathbf{G}} \leftarrow \beta_1\bar{\mathbf{G}} + (1-\beta_1)\mathbf{G}^\circ, \tag{25}$$

$$\mathbf{H} \leftarrow \beta_2\mathbf{H} + (1-\beta_2)\widehat{\mathbf{H}} + \tfrac{1}{2}(1-\beta_2)^2(\widehat{\mathbf{H}} - \mathbf{H})^{\odot 2}/(\mathbf{H}+\delta), \tag{26}$$

$$\mathbf{M} \leftarrow \mathbf{M} - \alpha\mathbf{Q}_L\left[\left(\bar{\mathbf{G}} + \delta\mathbf{Q}_L^\top\mathbf{M}\mathbf{Q}_R\right)/(\mathbf{H}+\delta)\right]\mathbf{Q}_R^\top. \tag{27}$$

In the Hessian update, we also added the quadratic correction term used in IVON (Shen et al., 2024). These updates give lines 1–8 in Alg. 2, where we also added clipping to stabilize the algorithm in practice. Lines 9, 10, 11 are identical to SOAP and update the matrix $\mathbf{P}$ during training.

C Theoretical Guarantees on Full Posterior Recovery

In this section, we prove Thm. 2 and Cor. 1 from the main paper. In order to do so, we first clarify some notation, we give analytical expressions for some quantities like the Hessian and the matrices $\mathbf{L}$ and $\mathbf{R}$, and we provide three technical Lemmas.

C.1 Notation and Preliminaries

We consider a linear model $f : \mathbb{R}^{d\times o} \times \mathbb{R}^d \to \mathbb{R}^o$ with parameter $\boldsymbol{\Theta} \in \mathbb{R}^{d\times o}$ and data $\mathbf{x} \in \mathbb{R}^d$

$$f(\boldsymbol{\Theta}, \mathbf{x}) = \boldsymbol{\Theta}^\top\mathbf{x} \tag{28}$$

for which the Jacobian is

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\Theta}, \mathbf{x}) = \mathbf{I}_o \otimes \mathbf{x}^\top \in \mathbb{R}^{o\times do} \tag{29}$$

where $\mathbf{I}_o$ is the identity matrix of size o and we keep the notation from the main paper of $\boldsymbol{\theta} = \text{vec}(\boldsymbol{\Theta})$. The number of parameters is equal to the product of data and label dimensionality $P = do$.

Given some cost function $c(y, f(\mathbf{\Theta}, \mathbf{x}))$ that measures the offset between label $\mathbf{y}$ and model prediction $f(\mathbf{\Theta}, \mathbf{x})$, the loss evaluated over a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$ is $\ell(\boldsymbol{\theta}) = \sum_{i=1}^n c(y_i, f(\mathbf{\Theta}, \mathbf{x}_i))$. Unlike the main text, where ℓ is the average loss, in this section ℓ denotes the summed loss, i.e., n times the loss in Eq. 1. The variational objective then reads $(\zeta/n)\,\mathbb{E}_q[\ell(\boldsymbol{\theta})] + \mathrm{KL}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta}))$, so ζ is replaced by ζ/n in the stationarity condition below. Positive scalar factors change neither the eigenvectors of $\mathbf{L}$, $\mathbf{R}$ and the expected Hessian, nor whether matrices are simultaneously diagonalizable, so the results below hold under either convention. By chain rule, its gradient is

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} f(\mathbf{\Theta}, \mathbf{x}_i)^\top \cdot \nabla_{f_i} c(y_i, f_i) \tag{30}$$

$$= \sum_{i=1}^n (\mathbf{I}_o \otimes \mathbf{x}_i) \cdot \nabla_{f_i} c(y_i, f_i) = \sum_{i=1}^n \mathrm{vec}\left(\mathbf{x}_i \nabla_{f_i} c(y_i, f_i)^\top\right) \in \mathbb{R}^{do} \tag{31}$$

Thus, the gradient in matrix form is a sum of outer products

$$\nabla_{\mathbf{\Theta}} \ell(\boldsymbol{\theta}) = \sum_{i=1}^n \mathbf{x}_i \nabla_{f_i} c(y_i, f_i)^\top \quad \in \mathbb{R}^{d \times o} \tag{32}$$

where f_i is a shorthand for $f(\mathbf{\Theta}, \mathbf{x}_i)$.

The expected Hessian under a posterior distribution $q(\boldsymbol{\theta})$ is equal to

$$\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})] = \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} f(\mathbf{\Theta}, \mathbf{x}_i)^\top \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i} c(y_i, f_i)] \nabla_{\boldsymbol{\theta}} f(\mathbf{\Theta}, \mathbf{x}_i) \tag{33}$$

$$= \sum_{i=1}^n (\mathbf{I}_o \otimes \mathbf{x}_i) \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i} c(y_i, f_i)] (\mathbf{I}_o \otimes \mathbf{x}_i^\top) \tag{34}$$

$$= \sum_{i=1}^n \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i} c(y_i, f_i)] \otimes (\mathbf{x}_i \mathbf{x}_i^\top) \quad \in \mathbb{R}^{od \times od} \tag{35}$$

that is a sum of Kronecker products.

The matrices $\mathbf{L}$ and $\mathbf{R}$ are the Exponential Moving Averages (EMA) of outer products of the gradient in matrix form, as in Eq. 32. Assuming to use full-batch gradients, T steps of the optimizer will follow some trajectory $\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_T$. The Exponential Moving Average computed iteratively (lines 9 and 10 of Alg. 2) is an exponentially weighted sum over the trajectory: with $\mathbf{L}$ and $\mathbf{R}$ initialized at zero, for any function g the following holds:

$$\mathrm{EMA}_{\boldsymbol{\theta}}[g(\boldsymbol{\theta})] = (1 - \beta_3) \sum_{t=0}^T \beta_3^{T-t} g(\boldsymbol{\theta}_t). \tag{36}$$

The matrix $\mathbf{L}$ is a weighted sum of outer products of data

$$\mathbf{L} = \mathrm{EMA}_{\boldsymbol{\theta}}\left[\nabla_{\mathbf{\Theta}} \ell(\boldsymbol{\theta}) \cdot \nabla_{\mathbf{\Theta}} \ell(\boldsymbol{\theta})^\top\right] \tag{37}$$

$$= \mathrm{EMA}_{\boldsymbol{\theta}}\left[\sum_{i,j=1}^n \mathbf{x}_i \cdot \nabla_{f_i} c(y_i, f_i)^\top \cdot \nabla_{f_j} c(y_j, f_j) \cdot \mathbf{x}_j^\top\right] \tag{38}$$

$$= \sum_{i,j=1}^n \underbrace{\mathrm{EMA}_{\boldsymbol{\theta}}\left[\langle \nabla_{f_i} c(y_i, f_i), \nabla_{f_j} c(y_j, f_j) \rangle\right]}_{\in \mathbb{R}} \underbrace{\mathbf{x}_i \mathbf{x}_j^\top}_{\in \mathbb{R}^{d \times d}}, \tag{39}$$

where the scalar coefficients that weight the sum are given by the EMA of the scalar product between residuals. Similarly, the matrix $\mathbf{R}$ is a weighted sum of outer products of residuals

$$\mathbf{R} = \text{EMA}_{\boldsymbol{\theta}}\left[\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\theta})^\top \cdot \nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\theta})\right] \tag{40}$$

$$= \text{EMA}_{\boldsymbol{\theta}}\left[\sum_{i,j=1}^{n} \nabla_{f_i} c(y_i, f_i) \cdot \mathbf{x}_i^\top \cdot \mathbf{x}_j \cdot \nabla_{f_j} c(y_j, f_j)^\top\right] \tag{41}$$

$$= \sum_{i,j=1}^{n} \underbrace{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}_{\in \mathbb{R}} \underbrace{\text{EMA}_{\boldsymbol{\theta}}\left[\nabla_{f_i} c(y_i, f_i) \cdot \nabla_{f_j} c(y_j, f_j)^\top\right]}_{\in \mathbb{R}^{o\times o}}, \tag{42}$$

where the scalar coefficients that weight the sum are given by the scalar product between data.

Neglecting the QR approximation for simplicity, the rotated eigenbase is defined as

$$\mathbf{P} = \text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L}) \in \mathbb{R}^{od\times od} \tag{43}$$

And EVON's variational family (SOAP-Bubbles) is

$$\mathcal{Q}_{\text{bubble}} = \left\{ \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{m}, \mathbf{P}\text{diag}(\boldsymbol{\sigma}^2)\mathbf{P}^\top) \mid \mathbf{m} \in \mathbb{R}^P, \boldsymbol{\sigma} \in \mathbb{R}_+^P \right\}. \tag{44}$$

Is this flexible enough? We study under which conditions this variational family does contain the optima among multivariate Gaussians. To see this, we first need to prove a few technical Lemmas.

C.2 Technical Lemmas

Lemma 1. *Let $q \sim \mathcal{N}(\mu, \Sigma)$ be a local minima of the variational problem in Eq. 1 among the family of multivariate Gaussians , then*

$$q \in \mathcal{Q}_{\textit{bubble}} \qquad \iff \qquad \mathbf{P}^\top \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]\mathbf{P} \textit{ is diagonal} \tag{45}$$

Proof. First, given that q is a local minimum, it must satisfy $\Sigma^{-1} = (\zeta/n)\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})] + \Sigma_0^{-1}$. The mode of the Gaussians can always be equated. Then $q \in \mathcal{Q}_{\text{bubble}}$ if and only if its covariance can be expressed by the variational family, i.e. if and only if there exists $\boldsymbol{\sigma} \in \mathbb{R}_+^P$ such that $\Sigma = \mathbf{P}\text{diag}(\boldsymbol{\sigma}^2)\mathbf{P}^\top$. This, in turn, is true if and only if $\Sigma^{-1} = \mathbf{P}\text{diag}(\boldsymbol{\sigma}^{-2})\mathbf{P}^\top$, and such $\boldsymbol{\sigma}$ exists as long as $\mathbf{P}^\top\Sigma^{-1}\mathbf{P}$ is diagonal. An isotropic prior $\Sigma_0^{-1} = \alpha\mathbf{I}$ remains isotropic under any change of basis $\mathbf{P}^\top\Sigma_0^{-1}\mathbf{P} = \alpha\mathbf{I}$. Consequently, $\mathbf{P}^\top\Sigma^{-1}\mathbf{P} = \mathbf{P}^\top\left((\zeta/n)\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})] + \Sigma_0^{-1}\right)\mathbf{P} = (\zeta/n)\mathbf{P}^\top\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]\mathbf{P} + \alpha\mathbf{I}$ is diagonal if and only if $\mathbf{P}^\top\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]\mathbf{P}$ is diagonal. □

In Eq. 43 multiple eigendecompositions may exist. For example, for the identity matrix, any basis is a valid set of eigenvectors. Once one is fixed, the condition in the RHS of Eq. 45 may not hold. This is quite restrictive, and in fact it makes more sense to ask whether *there exists* an eigendecomposition such that the matrix in the RHS of Eq. 45 is diagonal. This existence condition is studied in the next lemma, where $\mathbf{M}$ plays the role of the expected hessian.

Lemma 2. *Let $\mathbf{M}$ be a symmetric matrix. Then the following holds*

$$\textit{There exist an eigendecomposition } \textit{Eig}(\mathbf{R}) \textit{ and an eigendecomposition } \textit{Eig}(\mathbf{L}) \textit{ such that } \mathbf{P}^\top\mathbf{M}\mathbf{P} \textit{ is diagonal} \iff \mathbf{R}\otimes\mathbf{L} \textit{ and } \mathbf{M} \textit{ are simultaneously diagonalizable with } \mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2 \textit{ that is both } \mathbf{U}^\top\mathbf{M}\mathbf{U} \textit{ and } \mathbf{U}^\top(\mathbf{R}\otimes\mathbf{L})\mathbf{U} \textit{ are diagonal}$$

Proof idea: By defining $\mathbf{P} = \text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L})$ we are choosing $\mathbf{P}$ to be an eigenbasis of $\mathbf{R}\otimes\mathbf{L}$. An orthogonal matrix ($\mathbf{P}$ in this case) diagonalizes a matrix $\mathbf{M}$ if it is made by an $\mathbf{M}$-eigenbasis. Then $\mathbf{R}\otimes\mathbf{L}$ and $\mathbf{M}$ have the same eigenbasis $\mathbf{P}$.

Proof. First recall some known linear algebra properties: an eigendecomposition is an orthogonal matrix that diagonalizes a matrix, and product of orthogonal matrices is orthogonal. The Kronecker

product of orthogonal matrices is orthogonal.
$\boxed{\Rightarrow}$ Let $\text{Eig}(\mathbf{R})$ and $\text{Eig}(\mathbf{L})$ be the eigendecompositions in the LHS of the statement, then we have

$$\mathbf{R} = \text{Eig}(\mathbf{R})\mathbf{D}_R\text{Eig}(\mathbf{R})^\top \qquad \mathbf{L} = \text{Eig}(\mathbf{L})\mathbf{D}_L\text{Eig}(\mathbf{L})^\top \tag{46}$$

for two diagonal matrices $\mathbf{D}_R$ and $\mathbf{D}_L$. Consequently, we have

$$\mathbf{R} \otimes \mathbf{L} = (\text{Eig}(\mathbf{R})\mathbf{D}_R\text{Eig}(\mathbf{R})^\top) \otimes (\text{Eig}(\mathbf{L})\mathbf{D}_L\text{Eig}(\mathbf{L})^\top) \tag{47}$$

$$= \underbrace{(\text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L}))}_{=\mathbf{P}}(\mathbf{D}_R \otimes \mathbf{D}_L)\underbrace{(\text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L}))^\top}_{=\mathbf{P}^\top} \tag{48}$$

where $\mathbf{D}_R \otimes \mathbf{D}_L$ is diagonal and $\mathbf{P} = \text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L})$ diagonalizes the matrix $\mathbf{R} \otimes \mathbf{L}$. Since $\mathbf{P}$ diagonalizes also M by hypothesis, $\mathbf{R} \otimes \mathbf{L}$ and M are simultaneously diagonalizable, by definition.

$\boxed{\Leftarrow}$ Let $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$ be the orthogonal matrix that simultaneously diagonalizes $\mathbf{R} \otimes \mathbf{L}$ and M. First let's see that both $\mathbf{U}_1$ and $\mathbf{U}_2$ must be orthogonal matrices. $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$ implies that

$$(\mathbf{U}_1^\top\mathbf{U}_1) \otimes (\mathbf{U}_2^\top\mathbf{U}_2) = (\mathbf{U}_1^\top \otimes \mathbf{U}_2^\top)(\mathbf{U}_1 \otimes \mathbf{U}_2) = (\mathbf{U}_1 \otimes \mathbf{U}_2)^\top(\mathbf{U}_1 \otimes \mathbf{U}_2) = \mathbf{U}^\top\mathbf{U} = \mathbf{I} = \mathbf{I} \otimes \mathbf{I} \tag{49}$$

and two Kronecker products are equal if and only if both factors are equal up to a scalar constant. This implies that there exists $c \in \mathbb{R}_+$ such that $\mathbf{U}_1^\top\mathbf{U}_1 = c\mathbf{I}$ and $\mathbf{U}_2^\top\mathbf{U}_2 = c^{-1}\mathbf{I}$, so the two matrices are indeed orthogonal, up to a non-zero constant.

The fact that $\mathbf{U}_1 \otimes \mathbf{U}_2$ diagonalizes $\mathbf{R} \otimes \mathbf{L}$, assumed by hypothesis, means that for some diagonal matrix $\mathbf{D}$ it holds

$$\mathbf{D} = (\mathbf{U}_1 \otimes \mathbf{U}_2)^\top(\mathbf{R} \otimes \mathbf{L})(\mathbf{U}_1 \otimes \mathbf{U}_2) = (\mathbf{U}_1^\top\mathbf{R}\mathbf{U}_1) \otimes (\mathbf{U}_2^\top\mathbf{L}\mathbf{U}_2). \tag{50}$$

A Kronecker product is diagonal only if both its factors are diagonal, and non-zero. Which implies that there exists two diagonal matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ such that

$$\mathbf{D}_1 = \mathbf{U}_1^\top\mathbf{R}\mathbf{U}_1 \qquad \mathbf{D}_2 = \mathbf{U}_2^\top\mathbf{L}\mathbf{U}_2 \qquad \mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2 \tag{51}$$

Then $c^{-\frac{1}{2}}\mathbf{U}_1$ is an eigenbasis for $\mathbf{R}$ and similarly $c^{\frac{1}{2}}\mathbf{U}_2$ is an eigenbasis for $\mathbf{L}$. Then

$$\mathbf{P} = \text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L}) = (c^{-\frac{1}{2}}\mathbf{U}_1) \otimes (c^{\frac{1}{2}}\mathbf{U}_2) = \mathbf{U} \tag{52}$$

so $\mathbf{P} = \mathbf{U}$ and since $\mathbf{U}$ diagonalises $\mathbf{M}$ by hypothesis, also $\mathbf{P}$ diagonalises $\mathbf{M}$. Which is equivalent to $\mathbf{P}^\top\mathbf{M}\mathbf{P}$ being diagonal, which concludes the proof. □

We then consider the RHS condition of the previous Lemma in our setting, that is, when the matrix $\mathbf{M}$ is the expected Hessian $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$. Then we can use the equivalent Hessian formulation from Eq. 35 to prove the following Lemma. Note that this is a sufficient but not necessary condition.

Lemma 3. *The matrices $\mathbf{R} \otimes \mathbf{L}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$ are simultaneously diagonalizable if*

(1) $\mathbf{L}$, $\mathbf{x}_1\mathbf{x}_1^\top$, …, $\mathbf{x}_n\mathbf{x}_n^\top$ are simultaneously diagonalizable,

(2) $\mathbf{R}$, $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_1}c(y_1, f_1)]$, …, $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_n}c(y_n, f_n)]$ are simultaneously diagonalizable.

Moreover, the orthogonal matrix $\mathbf{U}$ that diagonalizes them is also Kronecker factorized.

Proof. Notice that $\mathbf{R} \otimes \mathbf{L}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$ are both symmetric, since they are defined as sum and Kronecker products of symmetric matrices. Consequently, if $\mathbf{R} \otimes \mathbf{L}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$ are simultaneously diagonalizable if and only if they commute, formally

$$(\mathbf{R} \otimes \mathbf{L})\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})](\mathbf{R} \otimes \mathbf{L}). \tag{53}$$

Following the expression for the expected Hessian in Eq. 35, this is equivalent to

$$(\mathbf{R} \otimes \mathbf{L})\left(\sum_{i=1}^n \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i, f_i)] \otimes (\mathbf{x}_i\mathbf{x}_i^\top)\right) = \left(\sum_{i=1}^n \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i, f_i)] \otimes (\mathbf{x}_i\mathbf{x}_i^\top)\right)(\mathbf{R} \otimes \mathbf{L}) \tag{54}$$

which can be equivalently rearranged as

$$\sum_{i=1}^{n}(\mathbf{R}\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)])\otimes(\mathbf{L}\mathbf{x}_i\mathbf{x}_i^\top)=\sum_{i=1}^{n}(\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\mathbf{R})\otimes(\mathbf{x}_i\mathbf{x}_i^\top\mathbf{L}). \tag{55}$$

This equality can be satisfied in various ways, and this is the reason why this Lemma only provides a sufficient condition. One possibility (but not the only one) is that the equality holds for each added $i=1,\dots,n$ in the sum individually, i.e.

$$(\mathbf{R}\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)])\otimes(\mathbf{L}\mathbf{x}_i\mathbf{x}_i^\top)=(\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\mathbf{R})\otimes(\mathbf{x}_i\mathbf{x}_i^\top\mathbf{L}) \text{ for all } i=1,\dots,n \tag{56}$$

In fact if we sum Eq. 56 for all $i=1,\dots,n$ then we obtain exactly Eq. 55.

We can equate each factor in the Kronecker products in Eq. 56 and obtain

$$\mathbf{R}\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]=\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\mathbf{R}\quad\text{and}\quad\mathbf{L}\mathbf{x}_i\mathbf{x}_i^\top=\mathbf{x}_i\mathbf{x}_i^\top\mathbf{L} \text{ for all } i=1,\dots,n \tag{57}$$

Note that Eq. 57 implies Eq. 56, as needed, but the converse only holds up to scalar factors.

Eq. 57 contains exactly hypotheses (1) and (2) of the Lemma. Then as we saw Eq. 57 $\Rightarrow$ Eq. 56 $\Rightarrow$ Eq. 55 $\Leftrightarrow$ Eq. 54 $\Leftrightarrow$ Eq. 53 $\Leftrightarrow$ $\mathbf{R}\otimes\mathbf{L}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$ are simultaneously diagonalizable.

This concludes the first part of the proof. We now show that the orthogonal matrix $\mathbf{U}$ that diagonalizes them is also Kronecker factorized. By hypothesis (1), there exists an orthogonal matrix $\mathbf{U}_2$ and $n+1$ diagonal matrices $\{\mathbf{D}_2^{(i)}\}_{i=0,\dots,n}$ such that

$$\mathbf{D}_2^{(0)}=\mathbf{U}_2^\top\mathbf{L}\mathbf{U}_2\qquad\mathbf{D}_2^{(i)}=\mathbf{U}_2^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{U}_2\qquad\text{for every } i=1,\dots,n \tag{58}$$

Similarly, by hypothesis (2), there exists an orthogonal matrix $\mathbf{U}_1$ and $n+1$ diagonal matrices $\{\mathbf{D}_1^{(i)}\}_{i=0,\dots,n}$ such that

$$\mathbf{D}_1^{(0)}=\mathbf{U}_1^\top\mathbf{R}\mathbf{U}_1\qquad\mathbf{D}_1^{(i)}=\mathbf{U}_1^\top\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\mathbf{U}_1\qquad\text{for every } i=1,\dots,n \tag{59}$$

We argue that $\mathbf{U}=\mathbf{U}_1\otimes\mathbf{U}_2$ is the matrix that diagonalizes both $\mathbf{R}\otimes\mathbf{L}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]$ and concludes the proof. We show these conditions explicitly, respectively

$$\begin{aligned}
\mathbf{U}^\top(\mathbf{R}\otimes\mathbf{L})\mathbf{U}&=(\mathbf{U}_1\otimes\mathbf{U}_2)^\top(\mathbf{R}\otimes\mathbf{L})(\mathbf{U}_1\otimes\mathbf{U}_2)=(\mathbf{U}_1^\top\mathbf{R}\mathbf{U}_1)\otimes(\mathbf{U}_2^\top\mathbf{L}\mathbf{U}_2) &(60)\\
&=\mathbf{D}_1^{(0)}\otimes\mathbf{D}_2^{(0)} &(61)\\
\mathbf{U}^\top\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})]\mathbf{U}&=(\mathbf{U}_1\otimes\mathbf{U}_2)^\top\left(\sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\otimes(\mathbf{x}_i\mathbf{x}_i^\top)\right)(\mathbf{U}_1\otimes\mathbf{U}_2) &(62)\\
&=\sum_{i=1}^{n}(\mathbf{U}_1\otimes\mathbf{U}_2)^\top\left(\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\otimes(\mathbf{x}_i\mathbf{x}_i^\top)\right)(\mathbf{U}_1\otimes\mathbf{U}_2) &(63)\\
&=\sum_{i=1}^{n}\left(\mathbf{U}_1^\top\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i}c(y_i,f_i)]\mathbf{U}_1\right)\otimes\left(\mathbf{U}_2^\top(\mathbf{x}_i\mathbf{x}_i^\top)\mathbf{U}_2\right) &(64)\\
&=\sum_{i=1}^{n}\mathbf{D}_1^{(i)}\otimes\mathbf{D}_2^{(i)} &(65)
\end{aligned}$$

and both Eq. 61 and Eq. 65 are diagonal matrices. This concludes the proof. □

C.3 Proof of the Main Results

With these Lemmas in place, we can finally prove Thm. 2. We restate it below for convenience.

Theorem 2. *Assume a linear model f, loss function c, an isotropic prior and a dataset $\mathcal{D}=\{(\mathbf{x}_i,y_i)\}_{i=1,\dots,n}$. If the matrices $\mathbf{L}$ and $\{\mathbf{x}_i\mathbf{x}_i^\top\}_{i=1,\dots,n}$ and the matrices $\mathbf{R}$ and $\{\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_f c(y_i,f(\boldsymbol{\Theta},\mathbf{x}_i))]\}_{i=1,\dots,n}$ are simultaneously diagonalizable, then the optimal posterior q among all multivariate Gaussians is a SOAP-Bubble, that is, in EVON's variational family.*

Proof. By hypothesis, both conditions (1) and (2) of Lemma 3 hold, thus implying that $\mathbf{R} \otimes \mathbf{L}$ and $\mathbf{M} = \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})]$ are simultaneously diagonalizable with some $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$, that is, both $\mathbf{U}^\top \mathbf{M} \mathbf{U}$ and $\mathbf{U}^\top (\mathbf{R} \otimes \mathbf{L}) \mathbf{U}$ are diagonal.

That is exactly the assumption on the RHS of Lemma 2, and we can apply it to deduce that there exists an eigendecomposition $\text{Eig}(\mathbf{R})$ and an eigendecomposition $\text{Eig}(\mathbf{L})$ such that the orthogonal matrix $\mathbf{P} = \text{Eig}(\mathbf{R}) \otimes \text{Eig}(\mathbf{L})$, as in Eq. 43, is an eigenbasis for the expected loss Hessian, that is, that the matrix $\mathbf{P}^\top \mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})] \mathbf{P}$ is diagonal.

Finally, once established that the matrix $\mathbf{P}$ can be an eigenbasis, Lemma 1 implies that the stationary points of the variational problem in Eq. 1 are a SOAP-Bubble. Equivalently, EVON's variational family contains the optimal multivariate Gaussian posterior. This concludes the proof. □

We now derive some sufficient conditions for each of the requirements of the Theorem, two related to the matrix $\mathbf{R}$ and one related to the matrix $\mathbf{L}$. These allow to prove Cor. 1, which we restate for convenience.

Corollary 1. *Assume a linear model, and an isotropic prior. Assume datapoints/features $\mathbf{x}_i$ are pairwise orthogonal or parallel, and fix the optimizer batch size to 1. For binary logistic regression and for general linear regression, the exact full Gaussian posterior among all multivariate Gaussians is a SOAP-Bubble, that is, in EVON's variational family.*

Proof. **Sufficient conditions for $\mathbf{R}$ and $\mathbb{E}_{\boldsymbol{\theta}\sim q}[\nabla^2_{f_i} c(y_i, f_i)]$**
If we are solving a regression task, it means that $c(y_i, f_i) = \|y_i - f_i\|^2$, or some scalar multiple of it. Thus, its Hessian $\nabla^2_{f_i} c(y_i, f_i)$ and consequently the expectation is equal to the identity, or some scalar multiple of it. The hypothesis of Thm. 2 then follows since any matrix $\mathbf{R}$ is simultaneously diagonalizable with the identity.

Alternatively, if we are solving a binary logistic regression task, it means that the output dimensionality $o = 1$. Consequently, the expected Hessian matrix is simply a scalar and trivially any (scalar) matrix $\mathbf{R}$ is simultaneously diagonalizable with any scalar.

Sufficient conditions for $\mathbf{L}$ and $x_i x_i^\top$
If the optimizer batch size is fixed to 1, then the gradients used to accumulate $\mathbf{L}$ are computed on a single datapoint. Let i_t denote the datapoint used at step t, which can be selected in any order. Then the gradient at step t is $\nabla_{\boldsymbol{\theta}} c(y_{i_t}, f(\boldsymbol{\theta}_t, \mathbf{x}_{i_t})) = \mathbf{x}_{i_t} \cdot \nabla_{f_{i_t}} c(y_{i_t}, f_{i_t})^\top$, where $f_{i_t} = f(\boldsymbol{\theta}_t, x_{i_t})$, evaluated at an arbitrary parameter $\boldsymbol{\theta}_t$. Consequently, the expression of $\mathbf{L}$ in Eq. 32 reduces to

$$\begin{aligned}
\mathbf{L} &= \text{EMA}_t \left[\nabla_{\boldsymbol{\Theta}} c(y_{i_t}, f(\boldsymbol{\theta}_t, \mathbf{x}_{i_t})) \cdot \nabla_{\boldsymbol{\Theta}} c(y_{i_t}, f(\boldsymbol{\theta}_t, \mathbf{x}_{i_t}))^\top \right] && (66)\\
&= \text{EMA}_t \left[\mathbf{x}_{i_t} \cdot \nabla_{f_{i_t}} c(y_{i_t}, f_{i_t})^\top \cdot \nabla_{f_{i_t}} c(y_{i_t}, f_{i_t}) \cdot \mathbf{x}_{i_t}^\top \right] && (67)\\
&= \text{EMA}_t \left[\|\nabla_{f_{i_t}} c(y_{i_t}, f_{i_t})\|^2 \mathbf{x}_{i_t} \mathbf{x}_{i_t}^\top \right] && (68)\\
&= \sum_{j=1}^{n} \underbrace{\alpha_j}_{\in \mathbb{R}} \underbrace{\mathbf{x}_j \mathbf{x}_j^\top}_{\in \mathbb{R}^{d\times d}}, && (69)
\end{aligned}$$

for some scalar values α_i, and where, similarly as before, $\text{EMA}_t[g(t)] = (1-\beta_3) \sum_{t=0}^{T} \beta_3^{T-t} g(t)$. If the datapoints are orthogonal, that is, $\langle x_j, x_k \rangle = 0$ for $j \neq k$, then, using the expression of $\mathbf{L}$ in Eq. 69, we have that for every $i = 1, \ldots, n$

$$\mathbf{L}\mathbf{x}_i = \sum_{j=1}^{n} \alpha_j \mathbf{x}_j \mathbf{x}_j^\top \mathbf{x}_i = \|\mathbf{x}_i\|^2 \alpha_i \mathbf{x}_i \tag{70}$$

and similarly that for every $i = 1, \ldots, n$

$$\mathbf{x}_i^\top \mathbf{L} = \mathbf{x}_i^\top \sum_{j=1}^{n} \alpha_j \mathbf{x}_j \mathbf{x}_j^\top = \|\mathbf{x}_i\|^2 \alpha_i \mathbf{x}_i^\top \tag{71}$$

which we can use to prove that the condition holds, as

$$\mathbf{L}\mathbf{x}_i \mathbf{x}_i^\top = \|\mathbf{x}_i\|^2 \alpha_i \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{x}_i \mathbf{x}_i^\top \mathbf{L} \quad \text{for all } i = 1, \ldots, n \tag{72}$$

The same equality holds if some datapoints are parallel, rather than orthogonal, that is, if for any $j \neq k$, either $\langle \mathbf{x}_j, \mathbf{x}_k \rangle = 0$ or $\mathbf{x}_j = \mathbf{x}_k$. Thus the proof also extends to repeated datapoints. □

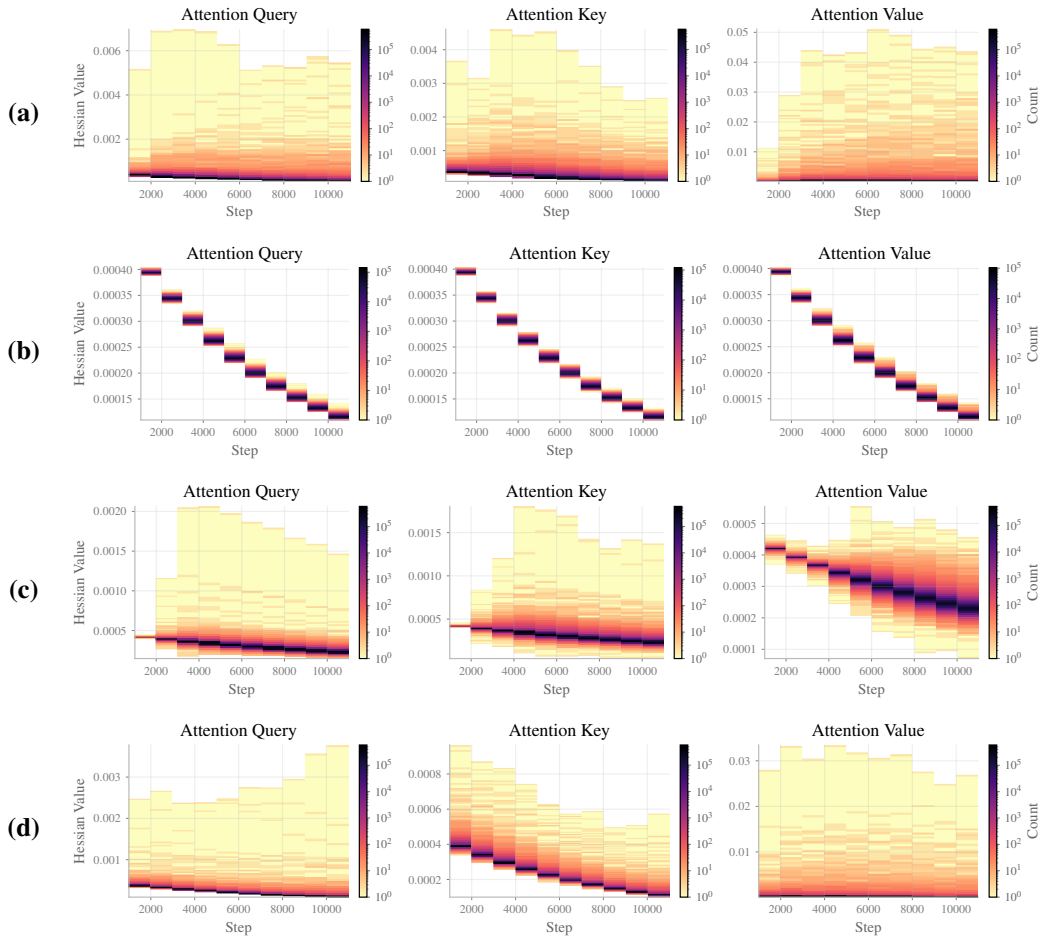


Figure 6: **Evolution of the EVON Hessian distribution in SOAP's eigenspace** (plotted every 1000 training steps) for the attention matrices in the sixth block of NanoGPT. We compare the baseline (Plain) optimizer (a) as a reference against other variants. Tight Hessian adaptive clipping (HC) (b) results in a much narrower range of smaller Hessian values, which may overestimate the posterior variance. Phased Gradients (PG) (c) also yield a narrower distribution of smaller Hessian values, whereas spectral clipping (SC) (d) produces a distribution much closer to the baseline's, maintaining similar optimization dynamics.

D Investigating Potential Improvements to EVON

We investigate potential improvements to the EVON optimizer on the NanoGPT pretraining task, reporting our results in Fig. 7 and Table 2. We fix the hyperparameters of the baseline optimizer and enable or disable the following components: Phased Gradients (PG), Hessian adaptive clipping (HC) with tight ratio (1.5) and spectral clipping (SC) of the update. The table presents the final test loss for each optimizer variant, and the effectiveness of the posterior is studied by performing Bayesian Model Averaging (BMA) with 20 Monte Carlo samples.

Spectral clipping (SC). Let $\mathbf{D} = \mathbf{Q}_L \mathbf{U} \mathbf{Q}_R^\top \in \mathbb{R}^{m \times n}$ be the preconditioned update in line 8 of Alg. 2, with singular value decomposition $\mathbf{D} = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^\top$. Spectral clipping replaces $\mathbf{D}$ by its orthogonal polar factor $\mathbf{A}\mathbf{B}^\top = \mathbf{D}(\mathbf{D}^\top\mathbf{D})^{-1/2}$, so the mean update becomes $\mathbf{M} \leftarrow \mathbf{M} - \alpha\,\mathbf{A}\mathbf{B}^\top$. The preconditioner $1/(\mathbf{H} + \delta)$ still determines the singular directions $\mathbf{A}$ and $\mathbf{B}$, but the step length along each of them is set by α alone; the Hessian estimate and the posterior variance $\mathbf{V}$ (lines 1–6) are unaffected. We approximate the polar factor with the quintic Newton–Schulz iteration of Muon (Jordan et al., 2024b):

starting from $\mathbf{X}_0 = \mathbf{D}/\|\mathbf{D}\|_F$ (transposed if $m > n$), we run five steps of

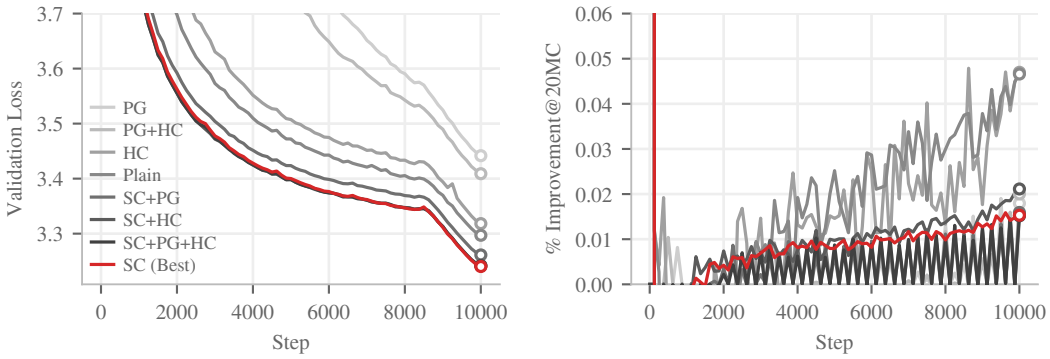
$$\mathbf{X}_{k+1} = a\mathbf{X}_k + \big(b\,\mathbf{X}_k\mathbf{X}_k^\top + c\,(\mathbf{X}_k\mathbf{X}_k^\top)^2\big)\mathbf{X}_k, \qquad (a, b, c) = (3.4445,\ -4.7750,\ 2.0315),$$

in `bfloat16`. These coefficients trade accuracy for speed: after five steps the singular values lie roughly between 0.5 and 1.5 rather than exactly at one. Parameters with more than two dimensions are reshaped into a matrix with the first dimension as rows, and one-dimensional parameters are normalized to unit ℓ_2 norm. Unlike Muon, we do not rescale the result by $\sqrt{\max(1, m/n)}$. Alg. 2 also admits element-wise clipping of the update as in IVON; we found spectral clipping to perform considerably better in our experiments, so the released implementation only supports spectral clipping.

Phased Gradients (PG). Phased Gradients alternates every training step between two phases: one phase computes gradients at perturbed parameter values for Hessian estimation, while the other computes gradients at the posterior mean to update the $\mathbf{L}$ and $\mathbf{R}$ preconditioners. Both phases update the posterior mean.

Our findings indicate that Phased Gradients (PG) can accelerate initial convergence when combined with other variants (SC and HC). Phased Gradients and tight Hessian adaptive clipping degrade the overall performance of the optimizer in isolation. Conversely, spectral clipping of the update yields notable improvements in stability and final test loss; the posterior estimation is not significantly affected when we compare the final loss of the posterior mean and BMA (Table 2).

To further understand these behaviors, we examine the effects of these modifications on the estimated Hessian. Figs. 6a to 6d show the histograms of the estimated Hessian in Shampoo's eigenspace for each variant. We observe that both Hessian adaptive clipping (Fig. 6b) and Phased Gradients (Fig. 6c) result in narrower distributions with much smaller Hessian values. This behavior may indicate an overestimate of the posterior variance. The spectral clipping variant (Fig. 6d) preserves a Hessian distribution and training behavior very close to the baseline's.



(a) Evaluation loss curves.

(b) BMA loss improvement.

Figure 7: Evaluation loss curve and BMA test loss improvement for different EVON variants during NanoGPT pretraining. (a) EVON with spectral clipping (SC) achieves the lowest test loss, demonstrating the effectiveness of spectrally clipping the update. In contrast, variants utilizing only Phased Gradients (PG) and Hessian adaptive clipping (HC) underperform both the baseline (Plain) and EVON with spectral clipping. (b) Percentage improvement in test loss when using BMA with 20 Monte Carlo samples compared to the posterior mean.

E Details on the Experiments and Hyperparameters

To ensure reproducibility, we report all hyperparameters for each dataset in Table 6. EVON as a PyTorch optimizer package can be found at `https://github.com/team-approx-bayes/evon` and the code to reproduce all figures and tables can be found at `https://github.com/team-approx-bayes/evon-experiments`.

Config	Loss@mean	Loss@20MC
SC	3.241	3.240 (-0.015%)
SC+PG+HC	3.242	3.241 (-0.015%)
SC+HC	3.243	3.242 (-0.021%)
SC+PG	3.262	3.261 (-0.016%)
Plain	3.299	3.297 (-0.047%)
HC	3.320	3.318 (-0.047%)
PG+HC	3.410	3.409 (-0.020%)
PG	3.442	3.442 (-0.018%)

Table 2: Performance evaluation of EVON variants in the NanoGPT pretraining task. Results include the final test loss, measuring both the loss of the posterior mean, and the loss at 20 Monte Carlo samples after Bayesian Model Averaging (BMA). We highlight the relative change over the mean in green.

Table 3: Hyperparameters for the illustrative examples and logistic-regression experiments. See https://github.com/team-approx-bayes/evon-experiments/tree/master/toyexamples for additional details.

Experiment	**Method**	ζ	α	h_0	δ	β_1	β_2	clip-radius
Neural network regression, Fig. 4(a)								
Small MLP	EVON	3×10^4	10^{-2}	5×10^{-2}	5×10^{-7}	0.9	0.999	–
Small MLP	IVON	10^4	10	5×10^{-2}	10^{-5}	0.9	0.999	10^{-1}
Large MLP	EVON	10^4	10^{-1}	5×10^{-2}	10^{-4}	0.9	0.999	–
Large MLP	IVON	10^5	1	10^{-2}	10^{-5}	0.9	0.9995	10^{-1}
UCI Iris multiclass logistic regression, Fig. 4(b)								
LogReg	EVON	n	5×10^{-2}	10^{-2}	$10^{-4}/n$	0.9	0.99985	–
LogReg	IVON	n	1	5×10^{-2}	$10^{-4}/n$	0.9	0.9995	10^{-1}
USPS 3vs5 binary logistic regression, Fig. 3								
LogReg	EVON	n	$0.03 \to 0.001$	$\frac{1-10^{-5}}{n}$	$10^{-5}/n$	0.9	$0.99 \to 0.999$	–
LogReg	IVON	n	5×10^{-3}	$\frac{1-10^{-5}}{n}$	$10^{-5}/n$	0.9	$0.995 \to 0.9995$	10^{-1}

E.1 Illustrative Examples

Illustrative experiment in Fig. 1. The experiment uses a linearly separable binary classification dataset with 2D features, from the Bayesian logistic regression chapter of Murphy (2012). Both EVON and IVON are run with small learning rate for many steps to ensure convergence to their respective solution, which are shown as the ellipses in the figure.

Neural network regression experiments in Fig. 4 (a). We follow the setup from Fadel et al. (2025, Figure 5), where our small MLP has the same 1-8-8-1 architecture with $\tanh$-activations. The large MLP has an 1-64-64-64-64-1 architecture with ELU activations. We train using full-batch gradients, as the dataset only consists of 10 data points. IVON uses elementwise clipping, EVON uses whitening with Hessian clipping. The hyperparameters are in Table 3.

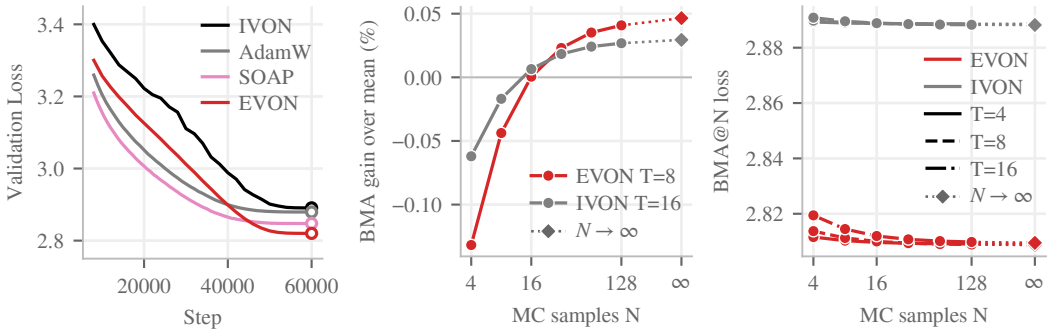
E.2 Bayesian Logistic Regression

Experiment in Fig. 3. We follow the setup of Mishkin et al. (2018, Fig. 1), which is binary logistic regression on the USPS dataset, considering the 3vs5 classification case. The hyperparameters are in Table 3.

Experiment in Fig. 4 (b). The experiment is multiclass logistic regression on the UCI Iris dataset. The hyperparameters are in Table 3. For EVON, the eigenbasis is updated every 25 steps and no Hessian clipping is used. We use Stan (Carpenter et al., 2017) via the CmdStanPy package to compute the reference posterior marginals via Hamiltonian Monte Carlo (HMC) and the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014).

Table 4: Hyperparameters for NanoGPT and Llama. EVON uses spectral clipping (Sec. D) (γ is the adaptive Hessian clipping parameter). IVON uses element-wise clipping with the listed radius.

Setup	ζ	h_0	α	β_1	$1-\beta_2$	δ	β_3	clipping
EVON (NanoGPT)	5.27×10^{6}	5.82×10^{-1}	1.72×10^{-2}	0.943	9.12×10^{-4}	1×10^{-6}	0.995	$\gamma = 1.5$
IVON (NanoGPT)	4.66×10^{7}	1.80×10^{-2}	1.42×10^{2}	0.890	1.23×10^{-4}	1×10^{-6}	–	10^{-4}
EVON (Llama)	7.61×10^{7}	9.31×10^{-1}	1.52×10^{-2}	0.881	3.46×10^{-4}	8.83×10^{-7}	0.995	$\gamma = 1.5$
IVON (Llama)	5.06×10^{8}	1.95×10^{-3}	1.18×10^{3}	0.826	7.29×10^{-5}	3.93×10^{-7}	–	2.49×10^{-4}



(a) Validation loss. (b) BMA gain over the mean. (c) BMA loss for all T.

Figure 8: 350M LLaMA pretraining on C4. (a) Validation loss: EVON achieves the lowest final validation loss, outperforming SOAP, AdamW, and IVON under the same protocol. (b) Relative improvement of the validation loss of Bayesian Model Averaging (BMA) with N Monte Carlo samples over the posterior mean, and (c) the BMA loss itself, for the final checkpoints. Samples are drawn with the posterior covariance scaled by a temperature $T \in \{4, 8, 16\}$; (c) shows all temperatures, (b) the per-method T with the lowest BMA loss at $N = 128$. The $N \to \infty$ limit (diamonds) is estimated by Richardson extrapolation, $2L_{128} - L_{64}$, assuming the BMA loss L_N approaches its limit as $1/N$. As for NanoGPT, averaging only becomes beneficial with enough samples ($N \geq 16$); beyond that, EVON's gain grows faster than IVON's and reaches a larger limit.

E.3 Language Model Pretraining

The experimental setup for NanoGPT (Jordan et al., 2024a) and Llama (Glentis et al., 2025) follows the experiments shown in Lin et al. (2026, Fig. 1), who provide tuned baselines and hyperparameters for AdamW and SOAP. NanoGPT experiments are run on a single B200 GPU and Llama experiments on a single H100 GPU, all with batch size 512. For NanoGPT, a linear warmup, constant step size and linear cooldown is used. Llama has linear warmup with cosine decay. Both SOAP and EVON update the preconditioner every 10 steps using QR decomposition. For both experiments, EVON and IVON's hyperparameters were tuned using equal budget of 500 runs in a random search over all hyperparameters. All optimizers use AdamW for the embedding layer and 1D tensors, where AdamW uses the same hyperparameters as Lin et al. (2026). The best hyperparameters (wrt final validation loss) found by the random search are summarized in Table 4.

E.4 CLIP Finetuning

We train CLIP models on CARS (Krause et al., 2013), DTD (Describable Textures Dataset) (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (German Traffic Sign Recognition Benchmark) (Stallkamp et al., 2012), MNIST, RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2014), and SVHN (Netzer et al., 2011). For each dataset, we tune the hyperparameters of the optimizers using a random search over the parameter ranges defined in Table 6, running 50 iterations for each optimizer and dataset. We then test the best hyperparameters using 5 different random seeds. We set early stopping patience to 3 epochs for all optimizers, monitoring the validation loss. In Table 7 we report the best hyperparameter values found for each optimizer and dataset.

E.5 Memory Comparison

We report peak GPU memory usage for NanoGPT training in Table 5. EVON and SOAP have similar memory footprints, since both maintain the same Kronecker-factored preconditioner and perform the same QR-based basis updates. EVON's additional cost comes mainly from random-number sampling for posterior perturbation, which does not require additional persistent memory. IVON and AdamW have similar memory footprints, as reported by Shen et al. (2024, App.).

Optimizer	**Peak Memory (MiB)**
AdamW	17,011
SOAP	19,312
EVON	19,887

Table 5: Peak GPU memory usage for NanoGPT training on a single H100 GPU. EVON and SOAP have similar memory footprints. IVON/AdamW use less memory since they do not store L, R kronecker factors and their eigenbases.

Optimizer	Hyperparameter	Distribution	Range / Values
AdamW	Learning rate α	Log-uniform	$[10^{-6},\ 5\times10^{-4}]$
	Weight decay λ	Log-uniform	$[10^{-4},\ 10^{-1}]$
SGD	Learning rate α	Log-uniform	$[10^{-4},\ 10^{-1}]$
	Weight decay δ	Log-uniform	$[10^{-6},\ 10^{-3}]$
	Momentum μ	Uniform	$[0.8,\ 0.99]$
IVON	Learning rate α	Log-uniform	$[10^{-4},\ 5\times10^{-2}]$
	Weight decay δ	Log-uniform	$[10^{-7},\ 10^{-2}]$
	ζ	Grid	$\{10^{7},\ 5\times10^{7},\ 10^{8},\ 5\times10^{8},\ 10^{9},\ 10^{10}\}$
	Hessian init h_0	Grid	$\{10^{-3},\ 10^{-2},\ 10^{-1}\}$
	β_1	Grid	$\{0.8,\ 0.9,\ 0.95,\ 0.99\}$
	$1-\beta_2$	Grid	$\{10^{-3},\ 5\times10^{-4},\ 10^{-4},\ 5\times10^{-5},\ 10^{-5}\}$
SOAP	Learning rate α	Log-uniform	$[10^{-6},\ 5\times10^{-4}]$
	Weight decay δ	Log-uniform	$[10^{-4},\ 2\times10^{-1}]$
	$(\beta_1,\ 1-\beta_2)$	Grid	$\{0.85, 0.9, 0.95\}\times\{5\times10^{-2}, 10^{-2}\}$ (6 pairs)
	β for $\mathbf{L}$ and $\mathbf{R}$	Grid	$\{0.9,\ 0.95,\ 0.99\}$
EVON	Learning rate α	Log-uniform	$[10^{-4},\ 10^{-1}]$
	Weight decay δ	Log-uniform	$[10^{-7},\ 10^{-2}]$
	ζ	Grid	$\{10^{7},\ 5\times10^{7},\ 10^{8},\ 5\times10^{8},\ 10^{9},\ 10^{10}\}$
	Hessian init h_0	Grid	$\{10^{-3}, 3\times10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$
	$(\beta_1,\ 1-\beta_2)$	Grid	$\{0.8, 0.9, 0.95, 0.99\}$ $\times\{10^{-3}, 5\times10^{-4}, 10^{-4}, 10^{-5}\}$ $\cup\ \{(0.95,\ 5\times10^{-2})\}$ (17 pairs)

Table 6: **Hyperparameter sweep ranges for CLIP finetuning.** For continuous hyperparameters we use a log-uniform (learning rate, weight decay) or uniform (momentum) distribution. For discrete hyperparameters we search over the listed grid values.

Optimizer	Hyperparameter	CARS	DTD	EuroSAT	GTSRB
AdamW	α	3.04×10^{-5}	1.7×10^{-5}	7.04×10^{-6}	9.93×10^{-6}
	δ	8.91×10^{-3}	0.0144	2.56×10^{-3}	1.19×10^{-4}
IVON	α	3.54×10^{-4}	1.12×10^{-4}	1.43×10^{-3}	2.38×10^{-3}
	δ	1.01×10^{-3}	3.14×10^{-6}	6.89×10^{-6}	5.15×10^{-4}
	ζ	5×10^{7}	10^{8}	5×10^{8}	10^{9}
	Hessian init h_0	0.01	10^{-3}	0.1	0.01
	β_1	0.99	0.9	0.9	0.8
	$1 - \beta_2$	10^{-4}	10^{-4}	10^{-4}	10^{-4}
SOAP	α	7.43×10^{-5}	3.71×10^{-5}	9.43×10^{-6}	1.56×10^{-5}
	δ	0.04575	1.67×10^{-4}	4.81×10^{-3}	2.49×10^{-4}
	$(\beta_1, 1 - \beta_2)$	(0.9, 0.01)	(0.85, 0.01)	(0.95, 0.01)	(0.95, 0.01)
	β for $\mathbf{L}$ and $\mathbf{R}$	0.9	0.95	0.99	0.99
EVON	α	0.06263	3.74×10^{-3}	0.07636	0.01957
	δ	1.87×10^{-7}	1.13×10^{-4}	5.98×10^{-4}	1.4×10^{-6}
	ζ	10^{7}	10^{8}	5×10^{7}	10^{7}
	Hessian init h_0	0.01	5×10^{-3}	10^{-3}	0.01
	β_1	0.95	0.95	0.95	0.95
	$1 - \beta_2$	0.05	0.05	0.05	0.05

Optimizer	Hyperparameter	MNIST	RESISC45	SUN397	SVHN
AdamW	α	1.98×10^{-6}	5.1×10^{-6}	1.4×10^{-5}	1.46×10^{-5}
	δ	1.7×10^{-4}	1.08×10^{-3}	4×10^{-4}	0.01245
IVON	α	6.75×10^{-3}	2.78×10^{-3}	5.28×10^{-3}	8.09×10^{-4}
	δ	3.39×10^{-3}	1.67×10^{-3}	3.51×10^{-4}	2.88×10^{-3}
	ζ	5×10^{8}	5×10^{8}	10^{8}	5×10^{8}
	Hessian init h_0	10^{-3}	0.01	10^{-3}	0.1
	β_1	0.95	0.99	0.95	0.99
	$1 - \beta_2$	5×10^{-4}	10^{-4}	10^{-3}	5×10^{-4}
SOAP	α	5.73×10^{-6}	1.27×10^{-5}	7.5×10^{-6}	9.2×10^{-5}
	δ	2.57×10^{-3}	7.33×10^{-3}	3.83×10^{-4}	1.89×10^{-4}
	$(\beta_1, 1 - \beta_2)$	(0.9, 0.05)	(0.85, 0.01)	(0.9, 0.01)	(0.95, 0.05)
	β for $\mathbf{L}$ and $\mathbf{R}$	0.99	0.99	0.95	0.95
EVON	α	0.04798	0.09509	0.03325	0.08859
	δ	9.45×10^{-6}	1.33×10^{-6}	1.08×10^{-7}	3.08×10^{-6}
	ζ	10^{7}	10^{8}	5×10^{8}	10^{10}
	Hessian init h_0	0.01	10^{-3}	10^{-3}	0.01
	β_1	0.95	0.95	0.95	0.95
	$1 - \beta_2$	0.05	0.05	0.05	0.05

Table 7: **Best hyperparameters for CLIP finetuning.** For each dataset and optimizer, we report the best hyperparameter values found during the random search. In these experiments, we enable adaptive hessian clipping (with a loose 2.0 ratio) and spectral clipping of the update for EVON.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract claims that we can do effective variational learning with non-diagonal posteriors for large models by modifying a few lines of the SOAP optimizer. The experiments demonstrate that our method can accurately recover exact posteriors in controlled variational Bayesian settings, while improving over previous state-of-the-art method IVON on language model experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of the method in the conclusion section of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide proofs for all theorems in the appendix, and proofs have been carefully checked for correctness.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide pseudo-code of the algorithm, along with all used hyperparameters. Code is available at https://github.com/team-approx-bayes/evon/ and https://github.com/team-approx-bayes/evon-experiments/.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All data sets are openly available. Code is available at https://github.com/team-approx-bayes/evon/ and https://github.com/team-approx-bayes/evon-experiments/.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and test details are described in the experiments section and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to computational constraints, and as is common practice in the language model pre-training setting, only runs for a single seed are shown. We did not find the result to significantly vary for different seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We consider standard training setups for language model pretraining. All experiments are run on a node with one H100 or B200 GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: The work is concerned with core machine learning and variational Bayesian methods for deep learning, with no immediately clear societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not train models of industry-level scale, so no safe-guards are required.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the dataset the models are trained on.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce new datasets or assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not perform any crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve any study participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLMs were used for the original submission. For the camera-ready version of this paper, we used GPT-6.1 Sol for proof-reading and coding assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.