Interpreting a Penalty as the Influence of a Bayesian Prior

Pierre Wolinski, Guillaume Charpiat, Yann Ollivier

Interpreting a Penalty as the Influence of a Bayesian Prior

Pierre Wolinski* pierre.wolinski@u-psud.fr
Guillaume Charpiat* guillaume.charpiat@inria.fr
Yann Ollivier† yol@fb.com

February 4, 2020

Abstract

In machine learning, it is common to optimize the parameters of a probabilistic model, modulated by a somewhat ad hoc regularization term that penalizes some values of the parameters. Regularization terms appear naturally in Variational Inference (VI), a tractable way to approximate Bayesian posteriors: the loss to optimize contains a Kullback–Leibler divergence term between the approximate posterior and a Bayesian prior. We fully characterize which regularizers can arise this way, and provide a systematic way to compute the corresponding prior. This viewpoint also provides a prediction for useful values of the regularization factor in neural networks. We apply this framework to regularizers such as $\mathcal{L}^1$ or group-Lasso.

1 Introduction

Adding a penalty term to a loss, in order to make the trained model fit some user-defined property, is very common in machine learning. For instance, penalties are used to improve generalization, prune neurons or reduce the rank of tensors of weights. Therefore, usual penalties are mostly empirical and user-defined, and integrated to the loss as follows:

$$L(\mathbf{w}) = \ell(\mathbf{w}) + r(\mathbf{w}),$$

with $\mathbf{w}$ the vector of all parameters in the network, $\ell(\mathbf{w})$ the error term and $r(\mathbf{w})$ the penalty term.

From a Bayesian point of view, optimizing such a loss L is equivalent to finding the Maximum A Posteriori (MAP) of the parameters $\mathbf{w}$ given the training data and a prior $\alpha \propto \exp(-r)$. Indeed, assuming that the loss ℓ is a log-likelihood loss, namely, $\ell(\mathbf{w}) = -\ln p_{\mathbf{w}}(\mathcal{D})$ with dataset $\mathcal{D}$, then minimizing L is equivalent to minimizing $L_{\text{MAP}}(\mathbf{w}) = -\ln p_{\mathbf{w}}(\mathcal{D}) - \ln(\alpha(\mathbf{w}))$. Thus, within the MAP framework, we can interpret the penalty term r as the influence of a prior α [14].

However, the MAP approximates the Bayesian posterior very roughly, by taking its maximum. Variational Inference (VI) provides a *variational posterior* distribution rather than a single value, hopefully representing the Bayesian posterior much better. VI looks for the best posterior approximation within a family $\beta_{\mathbf{u}}(\mathbf{w})$ of approximate posteriors over $\mathbf{w}$, parameterized

*Inria, Team TAU, Gif-sur-Yvette, France
†Facebook, France

by a vector $\mathbf{u}$. For instance, the weights $\mathbf{w}$ may be drawn from a Gaussian distribution with mean $\mathbf{u}$ and fixed variance. The loss to be minimized over $\mathbf{u}$ is then:

$$L_{\mathrm{VI}}(\beta_{\mathbf{u}}) \;=\; -\mathbb{E}_{\mathbf{w}\sim\beta_{\mathbf{u}}} \ln p_{\mathbf{w}}(\mathcal{D}) \;+\; \mathrm{KL}(\beta_{\mathbf{u}}\|\alpha) \;=\; \underbrace{\mathbb{E}_{\mathbf{w}\sim\beta_{\mathbf{u}}}\ell(\mathbf{w})}_{\text{data fit term}} + \underbrace{\mathrm{KL}(\beta_{\mathbf{u}}\|\alpha)}_{\text{penalty term}}, \tag{1}$$

and is also an upper bound on the Bayesian negative log-likelihood of the data [7]. Here $p_{\mathbf{w}}(\mathcal{D})$ is the likelihood of the full dataset $\mathcal{D}$ given $\mathbf{w}$, $\ell(\mathbf{w}) = -\ln p_{\mathbf{w}}(\mathcal{D})$ is the log-likelihood loss, and α is the Bayesian prior. The posteriors $\beta_{\mathbf{u}}$ that minimize this loss will be concentrated around values of $\mathbf{w}$ that assign high probability to the data, while not diverging too much from the prior α. Thus, the KL divergence term can be seen as a penalty $r(\cdot)$ over the vector $\mathbf{u}$.

Contributions and outline. We start from the following question: given some arbitrary penalty $r(\cdot)$, does it admit such an interpretation? Does there exist a prior α such that for all $\mathbf{u}$, $r(\mathbf{u}) = \mathrm{KL}(\beta_{\mathbf{u}}\|\alpha)$ (up to an additive constant)? If so, is there a systematic way to compute such α?

First, we provide a necessary and sufficient condition (Theorem 1) over the penalty r, that ensures the existence of a prior α such that VI with prior α reproduces the penalty r. The theorem comes with an explicit formula for α. We recover the MAP case as a degenerate case (Section 4.3).

Thus, we are able to determine whether a penalty r makes sense in a Bayesian framework and can be interpreted as the influence of a prior. We find this to be a strong constraint on r (Section 4.2). Here the regularizer r operates on the variational posterior $\beta_{\mathbf{u}}$; for deterministic β this reduces to r directly acting on $\mathbf{w}$, and we recover the traditional MAP correspondence in this case (Section 4.3).

Second (Section 5), we propose a heuristic to predict a priori useful values of the penalty factor λ to be put in front of a penalty r for neural networks, potentially bypassing the usual hyperparameter search on λ. Namely, we posit that the Bayesian prior $\alpha(\mathbf{w})$ corresponding to λr should reasonably match what is known for good a priori values of weights $\mathbf{w}$ in neural networks, namely, that the variance of weights under the prior α should match the Glorot initialization procedure [2]. This usually provides a specific value of λ. Moreover, the penalty size gets automatically adjusted depending on the width of the various layers in the network.

We test this prediction for various penalties (Section 6), including group-Lasso [25], for which per-layer adjustment of the penalty is needed [1]. Experimentally, the predicted value of the regularization factors leads to reasonably good results without extensive hyperparameter search. Still, the optimal penalization factor is found to be systematically about 0.01 to 0.1 times our predicted value, showing that our heuristic provides a usable order of magnitude but not a perfect value, and suggesting that the Bayesian VI viewpoint may over-regularize.

2 Related Work

Interpretations of existing empirical deep learning methods in a Bayesian variational inference framework include, for instance, *variational drop-out* [8], a version of drop-out which fits the Bayesian framework. Further developments of variational drop-out have been made by [18] for weight pruning and by [12] for neuron pruning.

Closer to our present work, the links between a penalized loss and the Bayesian point of view have previously been mentioned by [19] with a few approximations. [14] noted the equivalence of the penalized loss $L(\mathbf{w}) = \ell(\mathbf{w}) + r(\mathbf{w})$ and the MAP loss $L_{\mathrm{MAP}}(\mathbf{w}) = \ell(\mathbf{w}) - \ln(\alpha(\mathbf{w}))$ when $\ell(\mathbf{w})$ is the negative log-likelihood of the training dataset given the weights $\mathbf{w}$, and with a

prior $\alpha(\mathbf{w}) \propto \exp(-r(\mathbf{w}))$. That is, finding the vector $\hat{\mathbf{w}}$ minimizing a loss L can be equivalent to finding the MAP estimator $\hat{\mathbf{w}}_{\mathrm{MAP}}$ by minimizing the loss L_{MAP} with a well-chosen prior distribution α.

However, the MAP framework is not completely satisfying from a Bayesian point of view: instead of returning a distribution over the weights, which contains information about their uncertainty, it returns the most reasonable value. In order to evaluate this uncertainty, [13] proposed a second-order approximation of the Bayesian posterior. In the process, [15] also proposed a complete Bayesian framework and interpretation of neural networks. Still, this approximation of the Bayesian posterior is quite limited.

In the same period, [5] applied the Minimum Description Length (MDL) principle to neural networks. Then, [16] made the link between the MDL principle and variational inference, and [3] applied it to neural networks, allowing for variational approximations of the Bayesian posterior in a tractable way.

3 Variational Inference

We include here a reminder on variational inference for neural networks, following [3].

From a Bayesian viewpoint, we describe the vector of weights $\mathbf{w} \in \mathbb{R}^N$ of a neural network as a random variable. Given a dataset $\mathcal{D}$, we denote by $p_{\mathbf{w}}(\mathcal{D})$ the probability given to $\mathcal{D}$ by the network with parameter $\mathbf{w}$. For instance, with a dataset $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ of n input-output pairs and a model that outputs (log-)probabilities $p_{\mathbf{w}}(y_i|x_i)$ for the outputs, then $\ln p_{\mathbf{w}}(\mathcal{D}) = \sum_{i=1}^{n} \ln p_{\mathbf{w}}(y_i|x_i)$ is the total log-likelihood of the data given the model.

Given the dataset $\mathcal{D}$, the posterior distribution over weights $\mathbf{w}$ is:

$$\pi_{\mathcal{D}}(\mathbf{w}) = \frac{p_{\mathbf{w}}(\mathcal{D})\alpha(\mathbf{w})}{\mathbb{P}(\mathcal{D})}, \qquad \mathbb{P}(\mathcal{D}) = \int_{\mathbf{w}} \alpha(\mathbf{w}) p_{\mathbf{w}}(\mathcal{D})$$

which is analytically intractable for multi-layer nonlinear neural networks. However, the posterior π_D can be approximated by looking for probability distributions β that minimize the loss

$$L_{\mathrm{VI}}(\beta) = -\mathbb{E}_{\mathbf{w}\sim\beta} \ln p_{\mathbf{w}}(\mathcal{D}) + \mathrm{KL}(\beta\|\alpha), \tag{2}$$

where $\mathrm{KL}(\beta\|\alpha) = \int_{\mathbb{R}^N} \ln\left(\frac{\beta(\mathbf{w})}{\alpha(\mathbf{w})}\right) \beta(\mathbf{w})\,\mathrm{d}\mathbf{w}$ is the Kullback–Leibler divergence. Indeed, one has $L_{\mathrm{VI}}(\beta) = -\ln \mathbb{P}(\mathcal{D}) + \mathrm{KL}(\beta\|\pi_{\mathcal{D}})$, which is minimal when $\beta = \pi_{\mathcal{D}}$.

The first term in the loss (2) represents the error made over the dataset $\mathcal{D}$: it is small if β is concentrated around good parameters $\mathbf{w}$. The second term can be seen as a user-defined penalty over β that keeps it from diverging too much from the prior α. Moreover, for any distribution β, the quantity $L_{\mathrm{VI}}(\beta)$ is a bound on the Bayesian log-likelihood of the data: $L_{\mathrm{VI}}(\beta) \geq -\ln \int_{\mathbf{w}} \alpha(\mathbf{w}) p_{\mathbf{w}}(\mathcal{D})$ [7].

In variational inference, a parametric family $\mathcal{B}$ of probability distributions β is fixed, and one looks for the best approximation β^* of the Bayesian posterior in $\mathcal{B}$ by minimizing $L_{\mathrm{VI}}(\beta)$ in this family: the *variational posterior*. Importantly, for some families such as Gaussians with fixed variance, the gradient of $L_{\mathrm{VI}}(\beta)$ can be computed if the gradients of $\ln p_{\mathbf{w}}(\mathcal{D})$ can be computed [3], so that $L_{\mathrm{VI}}(\beta)$ can be optimized by stochastic gradient descent. Thus, this is well-suited for models such as neural networks.

Thus, we consider a parametric family $\mathcal{B} = \{\beta_{\mathbf{u}} : \mathbf{u} \in \mathbb{R}^P\}$ with parameter $\mathbf{u}$, where each $\beta_{\mathbf{u}}$ is a probability distribution over $\mathbf{w} \in \mathbb{R}^N$. Then we learn the parameters $\mathbf{u}$ instead of the weights $\mathbf{w}$. For instance, we can choose one of the following families of variational posteriors.

Example 1. *The family of products of Gaussian distributions over* $\mathbf{w} = (w_1, \cdots, w_N)$*:*

$$\beta_{\mathbf{u}} = \beta_{(\mu_1, \sigma_1^2, \cdots, \mu_N, \sigma_N^2)} = \mathcal{N}(\mu_1, \sigma_1^2) \otimes \cdots \otimes \mathcal{N}(\mu_N, \sigma_N^2).$$

In this case, the weights of the neural network are random and independently sampled from different Gaussian distributions $\mathcal{N}(\mu_k, \sigma_k^2)$*. Instead of learning them directly, the vector of parameters* $\mathbf{u} = (\mu_1, \sigma_1^2, \cdots, \mu_N, \sigma_N^2)$ *is learned to minimize* $L_{\mathrm{VI}}(\beta_{\mathbf{u}})$*.*

Example 2. *The family of products of Dirac distributions over* $\mathbf{w} = (w_1, \cdots, w_N)$*:*

$$\beta_{\mathbf{u}} = \beta_{(\mu_1, \cdots, \mu_N)} = \delta_{\mu_1} \otimes \cdots \otimes \delta_{\mu_N}.$$

In this case, the weights are deterministic: w_k *and* μ_k *are identical for all* k*.*

4 Bayesian Interpretation of Penalties

4.1 When Can a Penalty Be Interpreted as a Prior?

In this section, we provide a necessary and sufficient condition that ensures that a penalty $r(\mathbf{u})$ over the parameters of a variational posterior can be interpreted as a Kullback–Leibler divergence with respect to a prior α; namely, that

$$\exists\, K \in \mathbb{R},\, \forall \mathbf{u}, \qquad r(\mathbf{u}) = \mathrm{KL}(\beta_{\mathbf{u}} \| \alpha) + K$$

(the constant K does not affect optimization). In the process, we give a formula expressing α as a function of $r(\cdot)$.

In a nutshell, we place ourselves in the framework of variational inference: we assume the vector of weights $\mathbf{w}$ of the probabilistic model to be a random variable, drawn from a learned distribution $\beta_{\mathbf{u}}$ parameterized by a vector $\mathbf{u}$. For instance, the vector of weights $\mathbf{w}$ can be drawn from a multivariate normal distribution $\beta_{\mu,\mathbf{\Sigma}} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$.

We use some notions of distribution theory. We provide a reminder in Appendix C.

Notation. In order to approximate the posterior distribution of a vector $\mathbf{w} \in \mathbb{R}^N$, we denote by $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$ the family of variational posteriors over $\mathbf{w}$, parameterized by its mean μ and a vector of additional parameters $\boldsymbol{\nu}$. The basic example is a multivariate Gaussian distribution $\beta_{\mu,\boldsymbol{\nu}}$ parameterized by its mean $\mu \in \mathbb{R}^N$ and its covariance matrix $\boldsymbol{\nu} = \mathbf{\Sigma} \in \mathcal{M}_{N,N}(\mathbb{R})$.

We say that the family $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$ of variational posteriors is *translation-invariant* if $\beta_{\mu,\boldsymbol{\nu}}(\theta) = \beta_{0,\boldsymbol{\nu}}(\theta - \mu)$ for all $\mu, \boldsymbol{\nu}, \theta$.

We denote by $r(\mu, \boldsymbol{\nu}) = r_{\boldsymbol{\nu}}(\mu)$ some penalty, to be applied to the distribution $\beta_{\mu,\boldsymbol{\nu}}$.

We denote by $\mathcal{F}$ the Fourier transform, given by $(\mathcal{F}\varphi)(\xi) := \int_{\mathbb{R}^N} \varphi(x) e^{-i\xi \cdot x}\, \mathrm{d}x$ for $\varphi \in \mathcal{L}^1(\mathbb{R}^N)$. This definition extends to the class of tempered distributions $\mathcal{S}'(\mathbb{R}^N)$ (see Appendix C).

In the sequel, we always restrict ourselves to priors $\alpha \in \mathcal{T}(\mathbb{R}^N) = \{\alpha \text{ s.t. } \ln(\alpha) \in \mathcal{S}'(\mathbb{R}^N)\}$, i.e. log-tempered probability distributions, hence the condition $\ln(\alpha) \in \mathcal{S}'(\mathbb{R}^N)$ in the results below. This provides a reasonable behavior of α at infinity. Common probability distributions belong to this set. As a simple counter-example, one may take $\alpha(\theta) \propto \exp(-\exp|\theta|)$, for which $\ln(\alpha) \notin \mathcal{S}'(\mathbb{R})$.

Definition 1. *We define the following distribution over* $\mathbb{R}^N$*:*

$$A_{\boldsymbol{\nu}} \;:=\; -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\, \mathbb{1} \;-\; \mathcal{F}^{-1}\left[\frac{\mathcal{F} r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right], \tag{3}$$

where $\check{\beta}_{0,\boldsymbol{\nu}}(\theta) := \beta_{0,\boldsymbol{\nu}}(-\theta)$ *and* $\mathbb{1}$ *is the constant function equal to* 1*. We say that*

$$r \text{ fulfills } (\star) \Leftrightarrow \begin{cases} A_{\boldsymbol{\nu}} \text{ does not depend on } \boldsymbol{\nu}, \text{ i.e. } A_{\boldsymbol{\nu}} = A; \\ A \text{ is a function such that } \exp(A) \text{ integrates to } \kappa > 0. \end{cases}$$

Theorem 1. *Let* $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$ *be a translation-invariant family of variational posteriors. Let* $r(\mu, \boldsymbol{\nu}) = r_{\boldsymbol{\nu}}(\mu)$ *be a penalty over* $\beta_{\mu,\boldsymbol{\nu}}$*.*

We assume that $\forall \boldsymbol{\nu}$*, the probability distribution* $\beta_{0,\boldsymbol{\nu}}$ *has finite entropy and lies in the Schwartz class* $\mathcal{S}(\mathbb{R}^N)$*; that* $\mathcal{F}\beta_{0,\boldsymbol{\nu}}$ *is nonzero everywhere; and that* $\mathcal{F}r_{\boldsymbol{\nu}} \in \mathcal{E}'(\mathbb{R}^N)$*, the class of distributions with compact support.*

We are looking for probability distributions α *such that:*

$$\exists\, K \in \mathbb{R} : \forall (\mu, \boldsymbol{\nu}), \qquad r_{\boldsymbol{\nu}}(\mu) = \mathrm{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha) + K. \tag{4}$$

We have the following equivalence:

$$\alpha \text{ is a solution to (4), with } \alpha \in \mathcal{T}(\mathbb{R}^N) \qquad \Leftrightarrow \qquad r \text{ fulfills } (\star) \text{ and } \alpha = \frac{1}{\kappa}\exp(A),$$

where A *is defined in Equation* (3)*.*

The proof is given in Appendix D. It is based on the resolution of a classical integral equation ([20], Section 10.3-1) adapted to the wider framework of distribution theory. This extension is necessary, since the Fourier transform of the widely-used $\mathcal{L}^2$ penalty ($r(x) = x^2$) cannot be expressed as a function.

The preceding result holds under some technical assumptions. Even if some of the technical assumptions fail, the formula is still useful to compute a candidate prior α from a penalty $r(\cdot)$: apply Equation (3) on a penalty r to compute $A_{\boldsymbol{\nu}}$, then check that Condition $(\star)$ holds, define $\alpha = \frac{1}{\kappa}\exp(A)$, and finally compute $\mathrm{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha)$ analytically and compare it to r.

Remark 1. *The assumption* $\mathcal{F}r_{\boldsymbol{\nu}} \in \mathcal{E}'(\mathbb{R}^N)$ *includes the* $\mathcal{L}^2$ *penalty and all* $\mathcal{L}^{2p}$ *penalties (for any positive integer* p*), but not the* $\mathcal{L}^1$ *penalty. Indeed, For* $r_2(x) = x^2$ *one has* $\mathcal{F}r_2 = -2\pi\delta'' \in \mathcal{E}'(\mathbb{R})$*. For* $r_{2p}(x) = x^{2p}$*, one has* $\mathcal{F}r_{2p} = (-1)^p 2\pi\delta^{(2p)} \in \mathcal{E}'(\mathbb{R})$*, but for* $r_1(x) = |x|$*,* $(\mathcal{F}r_1)(t) = 2t^{-2} \notin \mathcal{E}'(\mathbb{R})$*. Still, for any penalty* $r \in \mathcal{S}'(\mathbb{R}^N)$*, it is always possible to find a sequence* $(r_n)_n \in \mathcal{S}'(\mathbb{R}^N)$ *converging to* r *in* $\mathcal{S}'(\mathbb{R}^N)$ *such that* $\mathcal{F}r_n \in \mathcal{E}'(\mathbb{R}^N)$ *for all* n *(see Appendix E).*

Particular case: variational posteriors parameterized by their mean ($\boldsymbol{\nu} = \varnothing$). Theorem 1 provides a method to compute a prior from a penalty and a family of variational posteriors $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$. Still, Eq. (3) returns a distribution $A_{\boldsymbol{\nu}}$ which may or may not satisfy the condition. Here we present a corollary which guarantees that the condition is satisfied; this holds under stricter conditions over the variational posteriors.

Corollary 1. *Assume that the family of posterior distributions* $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$ *is only parameterized by their means, that is* $\beta_{\mu,\boldsymbol{\nu}} = \beta_\mu$ *and* $r_{\boldsymbol{\nu}}(\mu) = r(\mu)$*. Assume that* $\beta_0 \in \mathcal{S}(\mathbb{R}^N)$*,* $\mathcal{F}r \in \mathcal{E}'(\mathbb{R}^N)$*,* $\mathcal{F}\beta_0$ *is nonzero everywhere and* β_0 *has a finite entropy. Assume that* $\mathcal{F}^{-1}[\frac{\mathcal{F}r}{\mathcal{F}\check{\beta}_0}]$ *is a function satisfying*

$$\exists\, (a, b, k) \in \mathbb{R}_*^+ \times \mathbb{R}_*^+ \times \mathbb{R} : \forall \theta \in \mathbb{R}^N, \qquad \mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\check{\beta}_0}\right](\theta) \geq a|\theta|^b + k. \tag{5}$$

Then there exists $\kappa > 0$ *such that:*

$$\alpha(\theta) = \frac{1}{\kappa}\exp\left(-\mathrm{Ent}(\beta_0) - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\check{\beta}_0}\right](\theta)\right) = \frac{1}{\kappa}e^{A(\theta)}$$

is a probability density satisfying $r(\mu) = \mathrm{KL}(\beta_\mu \| \alpha)$ *up to a constant, and is the unique such probability density in* $\mathcal{T}(\mathbb{R}^N)$.

Remark 2. *In many cases, condition* (5) *is not a limitation. For instance, with* $\beta_\mu \sim \mathcal{N}(\mu, \sigma^2)$, *where* σ^2 *is a constant, and with the penalty* r *the* $\mathcal{L}^2$ *penalty, Condition* (5) *reads:* $\exists\, (a, b, k) \in \mathbb{R}_*^+ \times \mathbb{R}_*^+ \times \mathbb{R}$ *such that* $\theta^2 - \sigma^2 \geq a|\theta|^b + k$ *for all* $\theta \in \mathbb{R}^N$, *which is satisfied.*

4.2 Example 1: Gaussian Distributions with $\mathcal{L}^2$ Penalty

Let us study the variational posteriors from Example 1: each vector of weights $\mathbf{w} \in \mathbb{R}^N$ is drawn from a Gaussian distribution $\beta_{\mathbf{u}} = \mathcal{N}(\mu_1, \sigma_1^2) \otimes \cdots \otimes \mathcal{N}(\mu_N, \sigma_N^2)$ with parameter $\mathbf{u} = (\mu_1, \sigma_1^2, \cdots, \mu_N, \sigma_N^2)$. We assume that each pair of variational parameters (μ_k, σ_k) is penalized independently by some penalty $r_{a,b}$ depending on two real-valued functions a and b:

$$r_{a,b}(\mu_k, \sigma_k^2) = a(\sigma_k^2) + b(\sigma_k^2)\mu_k^2. \tag{6}$$

The penalty over $\mathbf{u}$ is assumed to be the sum of the penalties $r_{a,b}(\mu_k, \sigma_k)$. Therefore, we study each pair (μ_k, σ_k) independently and we omit the index k.

Corollary 2. *If the penalty* (6) *above corresponds to a prior* α, *then* α *is Gaussian.*

More precisely, let α *be a probability distribution in* $\mathcal{T}(\mathbb{R})$, *and assume that* $r_{a,b}(\mu, \sigma^2) = \mathrm{KL}(\beta_{\mu,\sigma^2} \| \alpha)$, *up to a constant. Then there exists* $\sigma_0^2 > 0$ *such that* $\alpha = \mathcal{N}(0, \sigma_0^2)$. *Moreover, in that case, the penalty is, up to a constant:*

$$r_{a,b}(\mu, \sigma^2) = \frac{1}{2}\left[\frac{\sigma^2 + \mu^2}{\sigma_0^2} + \ln\left(\frac{\sigma_0^2}{\sigma^2}\right) - 1\right].$$

This corollary is an application of Theorem 1. We give the proof in Appendix G.

Thus, the assumption that a penalty r arises from a variational interpretation is a strong constraint over r. Here the penalty r was initially parameterized by a pair of real functions (a, b), and is finally parameterized by a single number σ_0^2.

4.3 Example 2: Deterministic Posteriors and the MAP

Another basic example is to use Dirac functions as variational posteriors (Example 2): $\beta_{\mu,\boldsymbol{\nu}} = \delta_\mu$. Since $\delta_0 \notin \mathcal{S}(\mathbb{R})$, the technical conditions of Theorem 1 are not satisfied. However, it is possible to apply Formula (3) and check that the resulting prior α is consistent with a chosen penalty r.

Applying Formula (3) yields

$$A = -\mathrm{Ent}(\delta_0)\,\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\delta_0}\right] = -\mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathbb{1}}\right] = -r.$$

Thus, if $\exp(-r)$ integrates to $0 < \kappa < \infty$, then we can define $\alpha = \frac{1}{\kappa}\exp(-r)$. Then, we can check that indeed $\mathrm{KL}(\delta_\mu \| \alpha) = r(\mu)$ up to a constant:

$$\mathrm{KL}(\delta_\mu \| \alpha) = -\,\mathrm{Ent}(\delta_0) - \langle \delta_\mu, \ln(\alpha) \rangle = -\ln \alpha(\mu) - \mathrm{Ent}(\delta_0) = r(\mu) + \ln \kappa - \mathrm{Ent}(\delta_0),$$

which confirms that the proposed prior α is consistent with the penalty $r(\cdot)$.

Thus, this formula recovers via variational inference the well-known penalty–prior equivalence in the MAP approximation, $\alpha_{\mathrm{VI}}(\theta) \propto \exp(-r(\theta)) \propto \alpha_{\mathrm{MAP}}(\theta)$ [14].

However, this is somewhat formal: the entropy $\mathrm{Ent}(\delta_0)$ of a Dirac function is technically undefined and is an "infinite constant". In practice, though, with a finite machine precision ε, a Dirac mass can be defined as a uniform distribution over an interval of size ε, and $\mathrm{Ent}(\delta_0)$ becomes the finite constant $\ln \varepsilon$.

5 Application to Neural Networks: Choosing the Penalty Factor

In this section, we compare the prior α arising from a penalty via Theorem 1, to reasonable weight priors for neural networks. In particular, we study how the prior varies when scaling the penalty by a factor λ, namely, $r_\lambda(\cdot) = \lambda\tilde{r}(\cdot)$ for a reference penalty $\tilde{r}$. Requiring that α is comparable to standard priors for neural network weights provides a specific value of the regularization constant λ. Specifically, we compare the prior to the standard initialization of neuron weights [2]: the variance of weights sampled from the prior α should be approximately equal to the inverse of the number of incoming weights to a neuron. This constraint can be used to determine λ. In particular, this suggests different values of λ for different layers, depending on their size.

Possible advantages of being able to predict a good value for λ include avoiding a hyper-parameter search, and better adjustment of the relative penalties of different layers or groups of neurons. For instance, one application of penalties in neural networks is to push neurons or convolutional filters towards zero, allowing for network pruning and reduced computational overhead. Penalties have been developed to remove entire neurons or filters, often based on the Lasso penalty: for instance, group-Lasso [21] and sparse group-Lasso [1]. In the latter work, different penalties are used for different layers, with values of λ determined empirically. We will compare our predicted values of λ to the values used in these works.

Additive penalties and independent priors. Below, we focus on penalties that are expressed as sums over neurons or groups of neurons, such as $\mathcal{L}^2$ or group-Lasso penalties. In the Bayesian setup, this corresponds to the additivity property of Kullback–Leibler divergence over products of distributions, $\mathrm{KL}(\beta_1 \otimes \beta_2 \| \alpha_1 \otimes \alpha_2) = \mathrm{KL}(\beta_1 \| \alpha_1) + \mathrm{KL}(\beta_2 \| \alpha_2)$. Thus, sums of penalties over different neurons or groups of neurons correspond to priors α and variational posteriors β that decompose as independent distributions over these neurons or groups of neurons.

5.1 A Reasonable Condition over the Prior α

Let us consider the variational inference framework applied to one weight w_{lij}, the j-th weight of the i-th neuron in the l-th layer of a neural network. As a default distribution, one could expect α to be usable to initialize the weight w_{lij}.

Therefore, we require α to satisfy the condition given by [2] over the initialization procedure, which we call ($\sharp$):

$$\alpha \text{ fulfills } (\sharp) \quad \Leftrightarrow \quad \mathbb{E}_{w_{lij}\sim\alpha}[w_{lij}] = 0 \text{ and } \mathbb{E}_{w_{lij}\sim\alpha}[w_{lij}^2] = 1/P_l$$

where P_l is the number of incoming weights in one neuron of the layer l. More generally, this condition can be written for the whole set of incoming weights to each neuron: denoting by $\mathbf{w}_{li}$ the vector of all incoming weights of neuron i in layer l, one can define Condition ($\sharp'$):

$$\alpha \text{ fulfills } (\sharp') \quad \Leftrightarrow \quad \mathbb{E}_{\mathbf{w}_{li}\sim\alpha}[\mathbf{w}_{li}] = 0 \text{ and } \mathbb{E}_{\mathbf{w}_{li}\sim\alpha}[\|\mathbf{w}_{li}\|^2] = 1$$

which slightly extends ($\sharp$).

Thus, if the prior α depends on some variable b, then Condition ($\sharp$) is reflected on b. For instance, in Example 1 (Section 4.2), ($\sharp$) is satisfied if and only if $\alpha_{\sigma_0^2}$ is $\mathcal{N}(0, 1/P_l)$. In the end, our suggested recipe for finding reasonable values for parameters b of a penalty r_b is the following:

1. Follow the formulas in Thm. 1 to compute a prior α_b such that $r_b(\mu, \boldsymbol{\nu}) = \mathrm{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha_b) + K$.
2. write Condition ($\sharp$) or ($\sharp'$) (depending on the case) for α_b: this provides a constraint on b.

5.2 Examples: $\mathcal{L}^2$, $\mathcal{L}^1$, and Group-Lasso Penalties

We now review some standard penalties, and apply this criterion to compute the penalty factor λ in front of each penalty.

Here we work with Dirac posterior distributions $\beta_\mu = \delta_\mu$ (Example 2 and Section 4.3). In that case, since each weight w_{lij} is deterministically set to μ_{lij}, we use w_{lij} and μ_{lij} indifferently. Thus, the penalty with penalty factor λ is

$$r_\lambda(w) = r_\lambda(\mu) = \lambda \tilde{r}(\mu) = \mathrm{KL}(\delta_\mu \| \alpha_\lambda) + K$$

for some reference penalty $\tilde{r}$. For each λ, the corresponding prior is $\alpha_\lambda(\theta) \propto e^{-\lambda \tilde{r}(\theta)}$.

We will apply Condition ($\sharp$) to find a value for λ, for various penalties. In Table 1 we compare these values to usual ones.

We recall that P_l in the number of incoming weights of a neuron in layer l.

Remark 3. *We recall that $r(\mathbf{w})$ is a full-dataset penalty (see Eq. 1), so that the actual per-minibatch penalty for stochastic gradient is $\frac{1}{n} r(\mathbf{w})$ for individual samples or $\frac{B}{n} r(\mathbf{w})$ for mini-batches of size B.*

The results below apply equally to fully connected and to convolutional networks; in the latter case, "neuron" should read as "individual filter".

$\mathcal{L}^2$ penalty. We have $r_\lambda(w) = \lambda w^2$, thus $\alpha_\lambda(\theta) \sim \mathcal{N}(0, 1/(2\lambda))$. Then Condition ($\sharp$) is equivalent to $\lambda = \frac{P_l}{2}$. Thus, from a Bayesian viewpoint, when using a $\mathcal{L}^2$-penalty, each weight w of a neuron with P_l incoming weights should be penalized by

$$r_\lambda(w) = \frac{P_l}{2} w^2. \tag{7}$$

$\mathcal{L}^1$ penalty. We have $r_\lambda(w) = \lambda |w|$, thus $\alpha_\lambda(\theta) = \frac{\lambda}{2} e^{-\lambda|\theta|}$. Therefore, Condition ($\sharp$) is equivalent to $\lambda = \sqrt{2P_l}$. As a consequence, when penalizing weight w in the l-th layer of a neural network with a $\mathcal{L}^1$-penalty, this weight w should be penalized with the term:

$$r_\lambda(w) = \sqrt{2P_l}|w|. \tag{8}$$

Standard Group-Lasso penalty. The group-Lasso jointly penalizes all the incoming weights of each neuron $\mathbf{w}_{l,i} \in \mathbb{R}^{P_l}$. Thus, we consider a prior and a posterior that are probability distributions on $\mathbb{R}^{P_l}$, and use Condition ($\sharp'$). Denoting by $\mathbf{w} \in \mathbb{R}^{P_l}$ the incoming weights of a neuron, we have

$$r_\lambda(w) = \lambda \|\mathbf{w}\|_2. \tag{9}$$

Then $\alpha_\lambda(\theta) = \frac{\lambda^{P_l}}{S_{P_l - 1}\Gamma(P_l)} e^{-\lambda\|\theta\|_2}$, where Γ is Euler's Gamma function and S_{n-1} is the surface of the $(n-1)$-sphere. After computation of the variance, Condition ($\sharp'$) is equivalent to $\lambda = \sqrt{P_l(P_l+1)}$. As a consequence, when penalizing neuron $\mathbf{w}$ in the l-th layer of a neural network with a group-Lasso penalty, this neuron $\mathbf{w}$ should be penalized with the term:

$$r_\lambda(\mathbf{w}) = \sqrt{P_l(P_l+1)}\|\mathbf{w}\|_2. \tag{10}$$

Table 1: How the regularization constant λ depends on the number of neurons n_l and the number of parameters per neuron P_l in layer l, both for our heuristics ($\lambda_{\text{Bayesian}}$) and for standard settings (λ_{usual}). For details on group-Lasso and reversed group-Lasso penalties, see Equations (9) and (12).

PENALTY	$\mathcal{L}^2$	$\mathcal{L}^1$	GROUP-LASSO	REV. GR.-LASSO
$\lambda_{\text{Bayesian}}$	$P_l/2$	$\sqrt{2P_l}$	$\sqrt{P_l(P_l+1)}$	$\sqrt{P_l(n_l+1)}$
λ_{usual}	1	1	$\sqrt{P_l}$	$\sqrt{n_l}$

This choice differs from [1], who use $\lambda \propto \sqrt{P_l}$, after an intuition proposed in [25]. Their whole-network penalty is

$$L(\mathbf{w}) = \ell(\mathbf{w}) + \sum_{l=1}^{L} \tilde{\lambda}_l \left((1-\gamma)\sqrt{P_l} \sum_{i=1}^{n_l} \|\mathbf{w}_{l,i}\|_2 + \gamma \sum_{i=1}^{n_l} \|\mathbf{w}_{l,i}\|_1 \right), \tag{11}$$

where $\gamma \in [0,1]$ is a fixed constant, L is the number of layers, n_l is the number of neurons in the l-th layer, P_l is the number of parameters in each neuron in the l-th layer, and $\mathbf{w}_{l,i}$ is the set of weights of the i-th neuron of the l-th layer. The per-layer regularization constants $\tilde{\lambda}_l$ are chosen empirically in [1]. On the other hand, our Bayesian reasoning yields a scaling $\sqrt{P_l(P_l+1)}$ instead of $\sqrt{P_l}$ for the penalties $\|\mathbf{w}_{l,i}\|_2$ in (11).

Reversed Group-Lasso penalty. We recall that the standard group-Lasso penalty groups the weights of a layer by neurons, that is output features. It is also possible to group them by input features, namely, to group together the outgoing weights of each neuron. For a fully-connected neural network, the loss penalized by a "reversed" group-Lasso can be written:

$$L(\mathbf{w}) = \ell(\mathbf{w}) + \sum_{l=1}^{L} \lambda_l \sum_{j=1}^{P_l} \|\mathbf{w}_{l,\cdot,j}\|_2, \tag{12}$$

where $\mathbf{w}_{l,\cdot,j} \in \mathbb{R}^{n_l}$ is the vector of weights of the l-th layer linked to the j-th input feature and n_l is the number of neurons in the l-th layer. By computations similar to standard group-Lasso, ($\sharp'$) is equivalent to $\lambda = \sqrt{P_l(n_l+1)}$ where n_l is the number of neurons in layer l, namely,

$$r_\lambda(\mathbf{w}_{l,\cdot,j}) = \sqrt{P_l(n_l+1)}\|\mathbf{w}_{l,\cdot,j}\|_2.$$

6 Experiments

Penalty terms are often used in the literature [21, 1, 4] in order to prune neural networks, that is, to make them sparse. Indeed, penalties such as $\mathcal{L}^1$ or group-Lasso tend to push weights or entire groups of weights towards 0 [24]. The efficiency of such methods can be measured in terms of final accuracy and number of remaining parameters.

We have chosen to compare within this context our heuristic to usual setups for the penalty factor, for different penalties. Our main criterion remains the final accuracy of the pruned

network. According to the results given in Table 1, the global penalty should be decomposed into a sum of penalties over each layer:

$$r(\mathbf{w}) = \lambda \sum_{l=1}^{L} \lambda_l r(\mathbf{w}_l),$$

where λ is a global penalty factor, L is the number of layers, $r(\mathbf{w}_l)$ is the penalty term due to the l-th layer and λ_l is its corresponding penalty factor. According to our heuristic, each λ_l should be set to $\lambda_{\text{Bayesian}}$ given in Table 1 and $\lambda = \lambda_{\text{Th}} = 1/n$, where n is the size of the training dataset (by Remark 3). In the usual setup, each λ_l is set to λ_{usual} given in Table 1, while several values for λ are tested.

We test the quality of our heuristic in two steps. First, we only test the computed partial penalty factors λ_l: they are fixed to $\lambda_{\text{Bayesian}}$, while several values for λ are tested. Second, we test the full heuristic: each λ_l is fixed to its corresponding $\lambda_{\text{Bayesian}}$ and $\lambda = \lambda_{\text{Th}} = 1/n$.

Experimental setup. We consider two neural networks: a version of VGG19 [22] with one fully connected layer of size 512, and CVNN, which is a simple network with two convolutional layers (of respective sizes 100 and 200 with 5×5 patches), each followed by a ReLU and a 2×2 max-pool layer, and two fully-connected layers (of sizes 1000 and 200).

The training and pruning procedure is detailed in Appendix A and the full experimental procedure is detailed in Appendix B. The CIFAR-10 dataset is decomposed into three parts: a training set (42000 images), a validation set (8000 images), and a test set (10000 images). All reported accuracies are computed over the test set, which is never used during each run.

Results. We give the results in Figure 1. Each point gives the final accuracy and number of parameters of a neural network for a given setup, averaged over 3 runs. The red line and the blue line correspond respectively to the usual setup and our setup for per-layer penalty λ_l, for various global factors λ. To check the quality of the heuristics over λ_l, we should compare the maxima of the two lines in each graph. The theoretical value $\lambda = \lambda_{\text{Th}}$ is marked by the grey vertical line, so the blue point on the grey vertical line illustrates the performance of the joint theoretical values on both λ_l and λ.

We report in Table 2 the values of relevant quantities: $\text{acc}^*_{\text{usual}}$, the best (over λ) accuracy of the usual per-layer scaling λ_l; $\text{acc}^*_{\text{Bayesian}}$, the best (over λ) accuracy of the Bayesian per-layer scaling λ_l (we denote by λ^* the optimal λ); and $\text{acc}_{\text{Bayesian}}$, the accuracy of the Bayesian per-layer λ_l together with the Bayesian predicted $\lambda = \lambda_{\text{Th}}$. This would allow us to conclude:

1. if $\text{acc}^*_{\text{Bayesian}} > \text{acc}^*_{\text{usual}}$, then we can conclude that our theoretical estimation for λ_l is better that the usual ones;

2. moreover, we check whether $\text{acc}_{\text{Bayesian}} > \text{acc}^*_{\text{usual}}$. If it is, then we can conclude that our theoretical estimation for λ_l *and* the choice $\lambda = \lambda_{\text{Th}}$ are better than the usual ones;

3. finally, the closer the ratio $\lambda_{\text{Th}}/\lambda^*$ is to 1, the closer we are to the Bayesian theoretical framework.

Regarding Point 1, our theoretical estimation for λ_l leads to accuracies which remain close to accuracies obtained in the usual setup. Our setup leads to slightly better accuracies when training CVNN, and slightly worse for VGG19.

Regarding Points 2 and 3, the usual setup with optimized λ and our setup with Bayesian estimation for $\lambda = \lambda_{\text{Th}}$ perform similarly, though the second one usually performs slightly worse,

due to a systematic mismatch between λ_{Th} and λ^*. Indeed, λ_{Th} is always roughly 10 times greater than λ^*.

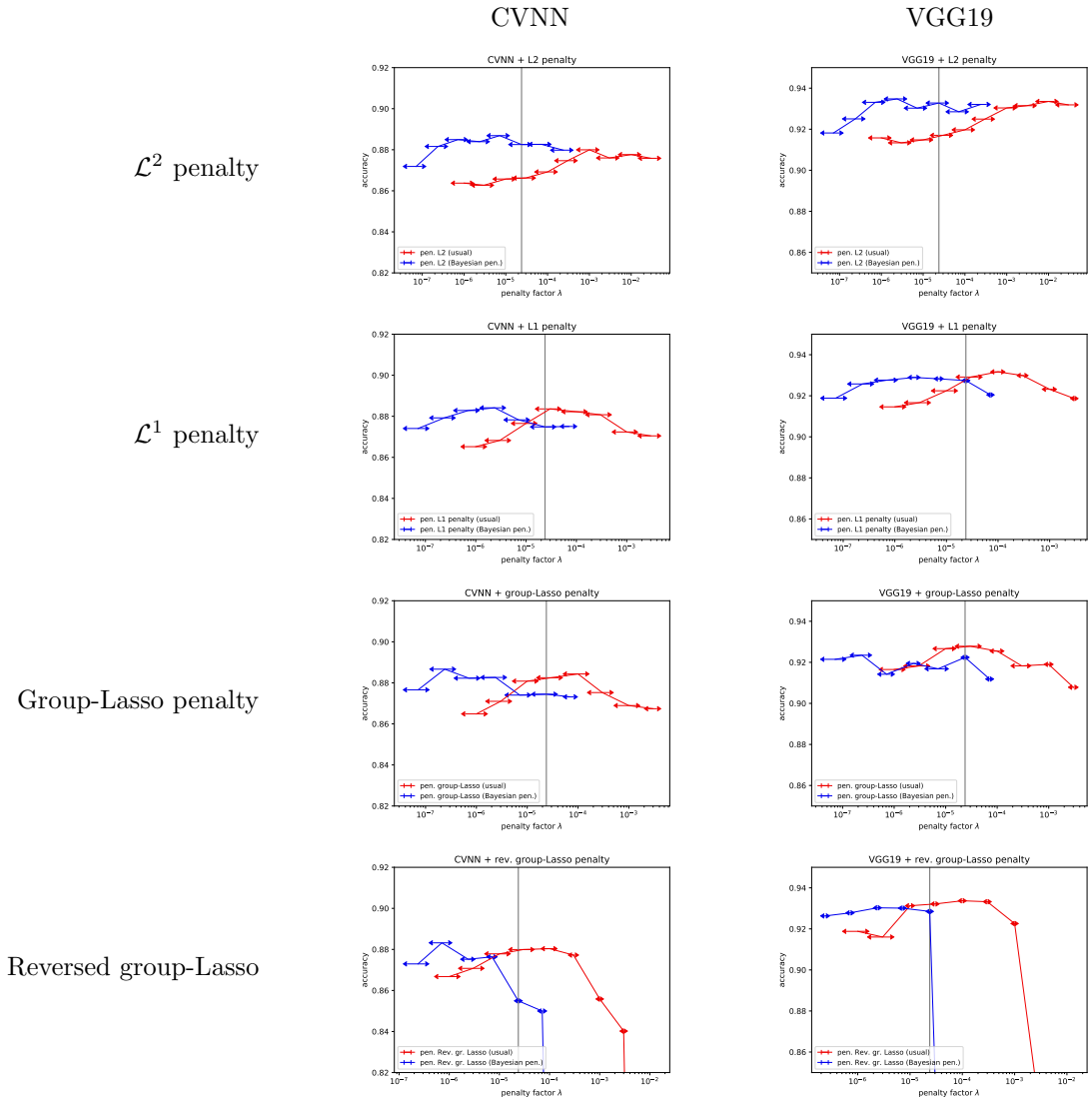

Figure 1: Final performance and number of parameters for various penalties in function of the penalty factor λ. Red line: standard setup for the penalty; blue line: setup provided by the heuristic ($\sharp$) or ($\sharp'$). Each bar corresponds to a final neural network: its abscissa is the penalty factor λ used for training it, its ordinate is its final accuracy, and its width corresponds to its final number of parameters. The vertical grey line is the value of the heuristic for λ.

Table 2: Comparison of the results. We show on the first row the best accuracy $\text{acc}^*_{\text{usual}}$ obtained with the usual setup. Then we show its difference with the best accuracy $\text{acc}^*_{\text{Bayesian}}$ obtained with our setup, its difference with the accuracy $\text{acc}_{\text{Bayesian}}$ obtained with our setup with the Bayesian theoretical value for λ, that is $\lambda_{\text{Th}} = n^{-1}$, and the ratio between λ_{Th} and the factor $\lambda = \lambda^*$, that is the optimal value for λ. Among the results obtained with our methods, those highlighted in blue are better, and those highlighted in red are worse than in the usual setup.

	$\mathcal{L}^2$ pen.		$\mathcal{L}^1$ pen.		group-Lasso pen.		rev. gr.-Lasso pen.	
	CVNN	VGG	CVNN	VGG	CVNN	VGG	CVNN	VGG
$\text{acc}^*_{\text{usual}}$ (%)	88.00 ± .4	93.35 ± .15	88.36 ± .3	93.17 ± .3	88.43 ± .14	92.78 ± .19	88.04 ± .4	93.37 ± .09
$\text{acc}^*_{\text{Bayesian}}$	88.69 ± .12	93.48 ± .09	88.41 ± .3	92.89 ± .2	88.67 ± .09	92.35 ± .18	88.32 ± .16	93.03 ± .15
$\text{acc}_{\text{Bayesian}}$	88.25 ± .3	93.28 ± .17	87.48 ± .08	92.74 ± .19	87.45 ± .17	92.24 ± .14	85.49 ± .3	92.85 ± .06
$\lambda_{\text{Th}}/\lambda^*$	$10^{0.5}$	10^1	10^1	10^1	10^2	10^2	$10^{1.5}$	10^1

Discussion. This systematic overestimation of the penalty factor indicates that some phenomenon is not yet understood. We have two mutually compatible explanations.

First, the choice $\lambda = 1/n$ could be overestimated, as a strict Bayesian setup might be overcautious from the very beginning. Indeed, in other Bayesian approaches to neural networks, such observations have been made. For instance, when using stochastic Langevin dynamics to approximate the posterior, performance is better if the weight of the posterior is arbitrarily decreased by an additional factor n (see footnote 5 in [17], and [9]).

Second, our choice for λ_l, based on Glorot's initialization, could also be overestimated. Indeed, λ_l only depends on P_l (the number of parameters in each neuron in layer l), and not on the location of layer l in the network for instance, while the following observations suggest that the heuristic used to set λ_l should take into account the architecture. We noticed that in most of the pruning experiments we made, the reached level of sparsity is much higher in the second half of the network (layers closer to the output). This is corroborated by [26] which shows that the layers may behave differently from a training point of view, depending on the architecture of the network and their position in it. Notably, they have proven that, in a trained VGG, some of the last convolutional layers can be reset to their initial value without changing much the final accuracy, while this cannot be done for the first layers.

7 Conclusion

We have provided a theorem that bridges the gap between empirical penalties and Bayesian priors when learning the distribution of the parameters of a model. This way, various regularization techniques can be studied in the same Bayesian framework, and be seen as probability distributions. This unified point of view allowed us to take into account well known heuristics (as Glorot's initialization) in order to find reasonable values for hyperparameters of the penalty.

We have checked experimentally that our theoretical framework leads to reasonable results. However, we noticed a constant mismatch (about a factor 10) between our predicted penalty factor λ_{Th} and the best one λ^*. This fact raises interesting questions about a possible overcautiousness of the Bayesian framework and the possible impact of the architecture when penalizing layers, which we plan to investigate further.

References

[1] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[3] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[4] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149*, 2015.

[5] Geoffrey E Hinton and Drew van Camp. Keeping neural networks simple. In *International Conference on Artificial Neural Networks*, pages 11–18. Springer, 1993.

[6] Lars Hörmander. *The Analysis of Linear Partial Differential Operators I*. Springer, 1998.

[7] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[8] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[9] Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1788–1794, 2016.

[10] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[11] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017.

[12] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.

[13] David JC MacKay. Bayesian model comparison and backprop nets. In *Advances in neural information processing systems*, pages 839–846, 1992.

[14] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[15] David JC MacKay. Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995.

[16] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[17] Gaétan Marceau-Caron and Yann Ollivier. Natural langevin dynamics for neural networks. In *International Conference on Geometric Science of Information*, pages 451–459. Springer, 2017.

[18] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[19] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[20] Andrei D Polyanin and Alexander V Manzhirov. *Handbook of integral equations*. CRC press, 1998.

[21] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Elias M Stein and Guido Weiss. *Introduction to Fourier analysis on Euclidean spaces (PMS-32)*, volume 32. Princeton university press, 2016.

[24] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[25] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[26] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019.

A Training and Pruning: Details

Since pruning neurons causes an accuracy drop, we divided the learning into two phases: learning and pruning phase; fine-tuning phase. This trick is widely used in pruning literature [10, 21, 11], in order to achieve better performance.

We define a pruning criterion based on the "norm" of each neuron. The tested penalties, i.e. $\mathcal{L}^2$ penalty, $\mathcal{L}^1$ penalty, group-Lasso penalty, and reversed group-Lasso penalty, can be separated into two categories: penalization of the output features of each layer ($\mathcal{L}^2$, $\mathcal{L}^1$, group-Lasso) and penalization of input features of each layer (reversed group-Lasso).

Pruning with $\mathcal{L}^2$, $\mathcal{L}^1$ and group-Lasso penalties. For each neuron $\mathbf{w}_{l,i}$, we check whether:

$$\|\mathbf{w}_{l,i}\|_2 \leq 0.001.$$

If so, then the neuron is pruned.

Pruning with reversed group-Lasso penalty. For each vector $\mathbf{w}_{l,\cdot,j}$ of the j-th input weights of the neurons in layer l, we check whether:

$$\|\mathbf{w}_{l,\cdot,j}\|_2 \leq 0.001.$$

If so, then the j-th neuron in layer $l-1$ is pruned, because its output is almost not used in layer l.

Pruning and training phase. The penalty is applied and, after each training epoch, pruning is performed over all layers. This phase ends when the number of neurons and the best validation accuracy have not improved for 50 epochs.

Fine-tuning phase. The penalty is removed and the learning rate is decreased by a factor 10 each time the validation accuracy has not improved for 50 epochs, up to 2 times. The third time, training is stopped. No pruning is performed during this phase.

B Experimental Procedure

For each combination of neural network (VGG19 or CVNN) and penalty ($\mathcal{L}^2$, $\mathcal{L}^1$, group-Lasso, or reversed group Lasso), we have tested two setups: our Bayesian heuristic for the penalty factor λ_l of each layer l, and the usual setup for them (as described in Table 1). For each setup, we have planned to plot the final accuracy and final number of parameters in function of the global penalty factor λ. We have proceeded as follows:

1. we define a set Λ of λ we want to use into our experiments. Typically, we have chosen $\Lambda = (1/n) \cdot \{10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}, 10^{0.5}\}$ in our setup (where n is the size of the training set), and $\Lambda = \{10^{-6}, 10^{-5.5}, 10^{-5}, 10^{-4.5}, 10^{-4}, 10^{-3.5}, 10^{-3}, 10^{-2.5}\}$ in the usual setup;
2. for each $\lambda \in \Lambda$, we test several learning rates $\eta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, and we select the learning rate η_λ which led to the best accuracy;
3. for each $\lambda \in \Lambda$, we run 2 more experiments with the selected learning rate η_λ. Thus, we are able to average our results over 3 runs.

C Reminder of Distribution Theory

In order to explain the main result, we recall some basic concepts of distribution theory. We use three functional spaces: the Schwartz class $\mathcal{S}(\mathbb{R}^N)$, the space of tempered distributions $\mathcal{S}'(\mathbb{R}^N)$, and the space of distributions with compact support $\mathcal{E}'(\mathbb{R}^N)$.

Above all, we recall the definition of the space of distributions $\mathcal{D}'(\mathbb{R}^N)$. We denote by $C^\infty(\mathbb{R}^N)$ the space of infinitely derivable functions mapping $\mathbb{R}^N$ to $\mathbb{R}$, and by $C_c^\infty(\mathbb{R}^N) \subset C^\infty(\mathbb{R}^N)$ the subspace of functions with compact support, that is $\varphi \in C_c^\infty(\mathbb{R}^N)$ if, and only if:

$$\varphi \in C^\infty(\mathbb{R}^N) \text{ and } \exists K \subset \mathbb{R}^N \text{ compact s.t.: } \{x \in \mathbb{R}^N : \varphi(x) \neq 0\} \subseteq K.$$

The set $\{x \in \mathbb{R}^N : \varphi(x) \neq 0\}$ is also denoted by $\mathrm{supp}(\varphi)$.

Space of distributions $\mathcal{D}'(\mathbb{R}^N)$. The space of distributions $\mathcal{D}'(\mathbb{R}^N)$ is defined as the space of continuous linear forms over $C_c^\infty(\mathbb{R}^N)$. For any *distribution* $T \in \mathcal{D}'(\mathbb{R}^N)$, we denote by $\langle T, \phi \rangle$ the value of T at a given *test function* $\varphi \in C_c^\infty(\mathbb{R}^N)$.

More formally, $T \in \mathcal{D}'(\mathbb{R}^N)$ if, and only if, for all compact set K of $\mathbb{R}^N$, there exists $p \in \mathbb{N}$ and $C > 0$ such that:

$$\forall \varphi \in C_c^\infty(\mathbb{R}^N) \text{ with } \mathrm{supp}(\varphi) \subseteq K, \qquad |\langle T, \varphi \rangle| \leq C \sup_{|\alpha| \leq p} \|\partial^\alpha \varphi\|_\infty.$$

Distributions are easier to visualize in specific cases. For instance, if a function $f : \mathbb{R}^N \to \mathbb{R}$ is integrable on every compact set $K \in \mathbb{R}^N$, then we can define a distribution T_f by:

$$\forall \varphi \in C_c^\infty(\mathbb{R}^N), \qquad \langle T_f, \varphi \rangle = \int f\varphi.$$

Therefore, distributions are often called "generalized functions", since most functions can be seen as distributions. In fact, by abuse of notation, $\langle f, \varphi \rangle$ stands for $\langle T_f, \varphi \rangle$.

Another classic example of distribution is the Dirac at zero δ, defined as follows:

$$\forall \varphi \in C_c^\infty(\mathbb{R}^N), \qquad \langle \delta, \varphi \rangle = \varphi(0).$$

Schwartz class $\mathcal{S}(\mathbb{R}^N)$. A function φ belongs to $\mathcal{S}(\mathbb{R}^N)$ if, and only if $\varphi \in C^\infty(\mathbb{R}^N)$ and:

$$\forall p \in \mathbb{N}, \exists C_p > 0 : \sup_{|\alpha| \leq p, |\beta| \leq p} \|x^\alpha \partial^\beta \varphi(x)\|_\infty \leq C_p.$$

In few words, $\mathcal{S}(\mathbb{R}^N)$ contains smooth and rapidly decreasing functions. For instance, any Gaussian density function belongs to $\mathcal{S}(\mathbb{R}^N)$.

We can easily define the Fourier transform on $\mathcal{S}(\mathbb{R}^N)$. Let $\varphi \in \mathcal{S}(\mathbb{R}^N)$:

$$(\mathcal{F}\varphi)(\xi) = \int_{\mathbb{R}^N} \varphi(x) e^{-i\xi x} \, \mathrm{d}x.$$

Thus, $\mathcal{F}^{-1} = (2\pi)^{-N} \bar{\mathcal{F}}$, where $\bar{\mathcal{F}}\varphi = \mathcal{F}\check{\varphi}$ and $\check{\varphi}(x) = \varphi(-x)$.

Space of tempered distributions $\mathcal{S}'(\mathbb{R}^N)$. Let $T \in \mathcal{D}'(\mathbb{R}^N)$ be a distribution. T belongs to $\mathcal{S}'(\mathbb{R}^N)$ if, and only if, there exists $p \in \mathbb{N}$ and $C > 0$ such that:

$$\forall \varphi \in C_c^\infty(\mathbb{R}^N), \qquad |\langle T, \varphi \rangle| \leq \sup_{|\alpha| \leq p, |\beta| \leq p} \|x^\alpha \partial^\beta \varphi\|_\infty.$$

The Fourier transform is defined on $\mathcal{S}'(\mathbb{R}^N)$ by duality. For any $T \in \mathcal{S}'(\mathbb{R}^N)$, $\mathcal{F}T \in \mathcal{S}'(\mathbb{R}^N)$ and is defined by:

$$\forall \varphi \in \mathcal{S}(\mathbb{R}^N), \qquad \langle \mathcal{F}T, \varphi \rangle = \langle T, \mathcal{F}\varphi \rangle.$$

Notably, this definition allows us to compute the Fourier transform of functions that do not lie in $\mathcal{L}^2$. This is very useful in the applications of Theorem 1, where we need the Fourier transform of $f : x \mapsto x^2$, which is $\mathcal{F}f = -2\pi\delta'' \in \mathcal{S}'(\mathbb{R})$ (where δ'' is the second derivative of the Dirac, defined below).

Space of distributions with compact support $\mathcal{E}'(\mathbb{R}^N)$. Let $T \in \mathcal{D}'(\mathbb{R}^N)$. The support of T is defined by:

$$\mathrm{supp}(T) = \mathbb{R}^N \backslash \{x \in \mathbb{R}^N : \exists \omega \text{ neighborhood of } x \text{ s.t. } T_{|\omega} = 0\}.$$

Thus, T is said to have a compact support if, and only if, $\mathrm{supp}(T)$ is contained into a compact subset of $\mathbb{R}^N$. As fundamental property of $\mathcal{E}'(\mathbb{R}^N)$, one should notice that $\mathcal{E}'(\mathbb{R}^N) \subset \mathcal{S}'(\mathbb{R}^N)$. That is, the Fourier transform is defined on $\mathcal{E}'(\mathbb{R}^N)$.

For instance, the Dirac at zero δ and its derivatives $\delta^{(k)}$ have support $\{0\}$, which is compact:

$$\begin{aligned} \langle \delta, \varphi \rangle &= \varphi(0) \\ \langle \delta^{(k)}, \varphi \rangle &= (-1)^k \varphi^{(k)}(0), \end{aligned}$$

for any test function $\varphi \in C_c^\infty(\mathbb{R}^N)$.

D Proof of Theorem 1

Proof. The proof can be split into three parts:

1. according to Definition 1, we have:

$$A_{\boldsymbol{\nu}} := -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right],$$

   and we prove that, for all $(\mu, \boldsymbol{\nu})$:

$$\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}}) - \langle A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}} \rangle = r_{\boldsymbol{\nu}}(\mu). \tag{13}$$

2. we prove that, if $\exp \circ A_{\boldsymbol{\nu}}$ is a function integrating to $\kappa > 0$, then there exists $K \in \mathbb{R}$ such that:

$$\forall (\mu, \boldsymbol{\nu}), \qquad r(\mu, \boldsymbol{\nu}) = \mathrm{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha_{\boldsymbol{\nu}}) + K, \tag{14}$$

   where $\alpha_{\boldsymbol{\nu}} = \frac{1}{\kappa} \exp \circ A_{\boldsymbol{\nu}}$.

   At that point, if Condition $(\star)$ is satisfied, then r can be written as a KL-divergence with respect to a prior. Thus, Condition $(\star)$ is a sufficient condition to ensure the existence of a solution to Equation (4).

3. we prove that, if α is a solution to Equation (4), then $\ln \circ \alpha$ is equal to $A_{\boldsymbol{\nu}}$ (up to a constant) and $A_{\boldsymbol{\nu}}$ satisfies Condition $(\star)$.

In the proof, and notably in Appendix D.3, we need the following proposition.

Proposition 3. *Let $T \in \mathcal{S}'(\mathbb{R}^N)$ and $\varphi \in \mathcal{S}(\mathbb{R}^N)$. Then $T * \varphi \in \mathcal{S}'(\mathbb{R}^N)$ and:*

$$\mathcal{F}[T * \varphi] = (\mathcal{F}T) \cdot (\mathcal{F}\varphi).$$

Proof. This statement is directly given in the proof of Theorem 3.18 [23]. □

D.1 Proof of Equation (13)

The distribution $A_{\boldsymbol{\nu}}$ is defined by:

$$A_{\boldsymbol{\nu}} = -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right].$$

Since $\mathbb{1} \in \mathcal{S}'(\mathbb{R}^N)$ and $\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}} \in \mathcal{S}'(\mathbb{R}^N)$, then $A_{\boldsymbol{\nu}} \in \mathcal{S}'(\mathbb{R}^N)$ by stability of $\mathcal{S}'$ by Fourier transform ([6], Lemma 7.1.3).

We compute:

$$\begin{aligned}
&-\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}}) - \langle A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}}\rangle \\
&= -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}}) + \left\langle \mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathbb{1} + \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right], \beta_{\mu,\boldsymbol{\nu}}\right\rangle \\
&= \left\langle \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right], \beta_{\mu,\boldsymbol{\nu}}\right\rangle,
\end{aligned}$$

since $\int \beta_{\mu,\boldsymbol{\nu}} = 1$.

By definition of the Fourier transform over $\mathcal{S}'(\mathbb{R}^N)$ ([6], Definition 7.1.9), we have:

$$\begin{aligned}
&-\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}}) - \langle A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}}\rangle \\
&= \left\langle \frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}, \mathcal{F}^{-1}\beta_{\mu,\boldsymbol{\nu}}\right\rangle. \qquad (15)
\end{aligned}$$

We compute $\mathcal{F}^{-1}\beta_{\mu,\boldsymbol{\nu}}$:

$$\begin{aligned}
(\mathcal{F}^{-1}\beta_{\mu,\boldsymbol{\nu}})(\xi) &= (2\pi)^{-N}\langle e^{i\xi\cdot}, \beta_{\mu,\boldsymbol{\nu}}(\cdot)\rangle \\
&= (2\pi)^{-N}\langle e^{i\xi\cdot}, \beta_{0,\boldsymbol{\nu}}(\cdot - \mu)\rangle \\
&= (2\pi)^{-N}\langle e^{i\xi\cdot}, \check{\beta}_{0,\boldsymbol{\nu}}(\mu - \cdot)\rangle,
\end{aligned}$$

using the assumptions over the family $(\beta_{\mu,\boldsymbol{\nu}})_{\mu,\boldsymbol{\nu}}$.

Thus:

$$\begin{aligned}
(\mathcal{F}^{-1}\beta_{\mu,\boldsymbol{\nu}})(\xi) &= (2\pi)^{-N}\langle e^{i\xi(\mu-\cdot)}, \check{\beta}_{0,\boldsymbol{\nu}}(\cdot)\rangle \\
&= (2\pi)^{-N}e^{i\xi\mu}\langle e^{-i\xi\cdot}, \check{\beta}_{0,\boldsymbol{\nu}}(\cdot)\rangle \\
&= (2\pi)^{-N}e^{i\xi\mu}(\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}})(\xi).
\end{aligned}$$

By injecting the result into Equation (15), we have:

$$\begin{aligned}
&- \operatorname{Ent}(\beta_{0,\boldsymbol{\nu}}) - \langle A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}} \rangle \\
&= (2\pi)^{-N} \left\langle \frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}, e^{i\mu\cdot} (\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}})(\cdot) \right\rangle \\
&= (2\pi)^{-N} \left\langle \mathcal{F}r_{\boldsymbol{\nu}}, e^{i\mu\cdot} \frac{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}} \right\rangle \\
&= (2\pi)^{-N} \left\langle \mathcal{F}r_{\boldsymbol{\nu}}, e^{i\mu\cdot} \right\rangle .
\end{aligned}$$

Since $\mathcal{F}r_{\boldsymbol{\nu}} \in \mathcal{E}'(\mathbb{R}^N)$, we can apply Theorem 7.1.14 [6]:

$$\begin{aligned}
(2\pi)^{-N} \left\langle \mathcal{F}r_{\boldsymbol{\nu}}, e^{i\mu\cdot} \right\rangle &= (\mathcal{F}^{-1}\mathcal{F}r_{\boldsymbol{\nu}})(\mu) \\
&= r_{\boldsymbol{\nu}}(\mu),
\end{aligned}$$

which achieves the proof of Equation (13).

D.2 Proof of Equation (14)

Now, we assume that $\exp \circ A_{\boldsymbol{\nu}}$ is a function integrating to $\kappa > 0$. We define $\alpha_{\boldsymbol{\nu}}$:

$$\alpha_{\boldsymbol{\nu}} = \frac{1}{\kappa} \exp \circ A_{\boldsymbol{\nu}},$$

which is non-negative and integrates to 1. Thus, $\alpha_{\boldsymbol{\nu}}$ is a density of probability.

We compute the KL-divergence between $\beta_{\mu,\boldsymbol{\nu}}$ and $\alpha_{\boldsymbol{\nu}}$:

$$\begin{aligned}
\operatorname{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha_{\boldsymbol{\nu}}) &= \int_{\mathbb{R}^N} \ln \left(\frac{\beta_{\mu,\boldsymbol{\nu}}(\theta)}{\alpha_{\boldsymbol{\nu}}(\theta)} \right) \beta_{\mu,\boldsymbol{\nu}}(\theta) \, \mathrm{d}\theta \\
&= -\operatorname{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \langle \ln \circ \alpha_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}} \rangle \\
&= -\operatorname{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \langle -\ln(\kappa) \mathbb{1} + A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}} \rangle \\
&= -\operatorname{Ent}(\beta_{\mu,\boldsymbol{\nu}}) + \ln(\kappa) \langle \mathbb{1}, \beta_{\mu,\boldsymbol{\nu}} \rangle - \langle A_{\boldsymbol{\nu}}, \beta_{\mu,\boldsymbol{\nu}} \rangle \\
&= r_{\boldsymbol{\nu}}(\mu) + \ln \kappa,
\end{aligned}$$

since $\beta_{\mu,\boldsymbol{\nu}}$ integrates to 1 and $\operatorname{Ent}(\beta_{\mu,\boldsymbol{\nu}})$ does not depend on μ. Therefore, using Equation (13):

$$\operatorname{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha_{\boldsymbol{\nu}}) = r(\mu, \boldsymbol{\nu}) + \ln \kappa.$$

Moreover, if $A_{\boldsymbol{\nu}}$ does not depend on $\boldsymbol{\nu}$, then:

$$\operatorname{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha) = r(\mu, \boldsymbol{\nu}) + \ln \kappa.$$

Therefore, Condition ($\star$) is a sufficient condition to ensure the existence of a solution α to Equation (4).

D.3 Uniqueness of the Solution

We assume that α is a probability distribution in $\mathcal{T}(\mathbb{R}^N)$ and:

$$r_{\boldsymbol{\nu}}(\mu) = \operatorname{KL}(\beta_{\mu,\boldsymbol{\nu}} \| \alpha) + K,$$

where $K \in \mathbb{R}$ is a constant.

Thus:

$$\begin{aligned}
r_{\boldsymbol{\nu}}(\mu) &= \int_{\mathbb{R}^N} \ln\left(\frac{\beta_{\mu,\boldsymbol{\nu}}(\theta)}{\alpha(\theta)}\right) \beta_{\mu,\boldsymbol{\nu}}(\theta)\,\mathrm{d}\theta + K \\
&= -\mathrm{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \int_{\mathbb{R}^N} \left[\ln(\alpha(\theta)) - K\right] \beta_{\mu,\boldsymbol{\nu}}(\theta)\,\mathrm{d}\theta \\
&= -\mathrm{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \int_{\mathbb{R}^N} \hat{A}(\theta)\beta_{\mu,\boldsymbol{\nu}}(\theta)\,\mathrm{d}\theta \qquad (\text{where } \hat{A}(\theta) := \ln(\alpha(\theta)) - K) \\
&= -\mathrm{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \int_{\mathbb{R}^N} \hat{A}(\theta)\beta_{0,\boldsymbol{\nu}}(\theta - \mu)\,\mathrm{d}\theta \\
&= -\mathrm{Ent}(\beta_{\mu,\boldsymbol{\nu}}) - \int_{\mathbb{R}^N} \hat{A}(\theta)\check{\beta}_{0,\boldsymbol{\nu}}(\mu - \theta)\,\mathrm{d}\theta \\
&= -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}}) - (\hat{A} * \check{\beta}_{0,\boldsymbol{\nu}})(\mu),
\end{aligned}$$

since the convolution between $\hat{A} \in \mathcal{S}'(\mathbb{R}^N)$ and $\check{\beta}_{0,\boldsymbol{\nu}} \in \mathcal{S}(\mathbb{R}^N)$ is well-defined ([23], Theorem 3.13), and $\mathrm{Ent}(\beta_{\mu,\boldsymbol{\nu}}) = \mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})$.

Then, we can apply the Fourier transform:

$$\begin{aligned}
\mathcal{F}r_{\boldsymbol{\nu}} &= -2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\delta - \mathcal{F}(\hat{A} * \check{\beta}_{0,\boldsymbol{\nu}}) \\
&= -2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\delta - (\mathcal{F}\hat{A}) \cdot (\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}),
\end{aligned}$$

by applying Proposition 3. Since $\mathcal{F}\beta_{0,\boldsymbol{\nu}}$ is supposed to be nonzero everywhere, then:

$$\mathcal{F}\hat{A} = \frac{-2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\delta - \mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}.$$

From now, we just have to compute the inverse Fourier transform of $\mathcal{F}\hat{A}$ to get $\hat{A}$:

$$\begin{aligned}
\hat{A} &= \mathcal{F}^{-1}\left[\frac{-2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\delta - \mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] \\
&= -\mathcal{F}^{-1}\left[\frac{2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\delta}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] \\
&= -2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathcal{F}^{-1}\left[\frac{\delta}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right].
\end{aligned}$$

We compute the first term, which is a tempered distribution. For all $\varphi \in \mathcal{S}(\mathbb{R}^N)$, we have:

$$\begin{aligned}
\left\langle \frac{\delta}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}, \varphi \right\rangle &= \left\langle \delta, \frac{\varphi}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}} \right\rangle \\
&= \frac{\varphi(0)}{(\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}})(0)} \\
&= \varphi(0),
\end{aligned}$$

since $(\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}})(0)$ is equal to $\int \check{\beta}_{0,\boldsymbol{\nu}} = \int \beta_{0,\boldsymbol{\nu}} = 1$. Thus, $\frac{\delta}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}} = \delta$.

Therefore:

$$\begin{aligned}\hat{A} &= -2\pi\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathcal{F}^{-1}\delta - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] \\ &= -\mathrm{Ent}(\beta_{0,\boldsymbol{\nu}})\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{\boldsymbol{\nu}}}{\mathcal{F}\check{\beta}_{0,\boldsymbol{\nu}}}\right] \\ &= A_{\boldsymbol{\nu}}.\end{aligned}$$

Recalling that $\hat{A}(\theta) := \ln(\alpha(\theta)) - nK$, we have:

$$\ln(\alpha(\theta)) - K = A_{\boldsymbol{\nu}}(\theta),$$

thus $A_{\boldsymbol{\nu}}$ does not depend on $\boldsymbol{\nu}$, from which we deduce that r fulfills Condition $(\star)$ and $\alpha \propto \exp\circ A$. □

E Note on Remark 1

We show that, for all $r \in \mathcal{S}'(\mathbb{R}^N)$, there exists a sequence $(r_n)_n \in \mathcal{S}'(\mathbb{R}^N)$ converging to r in $\mathcal{S}'(\mathbb{R}^N)$ such that $\mathcal{F}r_n \in \mathcal{E}'(\mathbb{R}^N)$ for all n.

For all $r \in \mathcal{S}'(\mathbb{R}^N)$, $\mathcal{F}r \in \mathcal{S}'(\mathbb{R}^N)$. Let us build the sequence of distributions :

$$q_n = (\mathcal{F}r)\mathbb{1}_{[-n,n]^N}.$$

The sequence $(q_n)_n \in \mathcal{E}'(\mathbb{R}^N)$ converges to $\mathcal{F}r$ in $\mathcal{S}'(\mathbb{R}^N)$. Since $\mathcal{F}^{-1}$ is continuous, we have:

$$\mathcal{F}^{-1}q_n \to r \qquad \text{in } \mathcal{S}'(\mathbb{R}^N).$$

Therefore, we can pose $r_n = \mathcal{F}^{-1}q_n$, which is appropriate.

F Proof of Corollary 1

Proof. We apply Formula (3):

$$A = -\mathrm{Ent}(\beta_0)\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\beta_0}\right].$$

Thus:

$$\exp(A(\theta)) = e^{-\mathrm{Ent}(\beta_0)}\exp\left(-\mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\beta_0}\right](\theta)\right).$$

Assuming there exists $a > 0, b > 0$ and $k \in \mathbb{R}$ such that:

$$\mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\beta_0}\right](\theta) \geq a|\theta|^b + k,$$

we can write:

$$0 \leq \exp(A(\theta))e^{\mathrm{Ent}(\beta_0)} = \exp\left(-\mathcal{F}^{-1}\left[\frac{\mathcal{F}r}{\mathcal{F}\beta_0}\right](\theta)\right) \leq \exp\left(-\left[a|\theta|^b + k\right]\right).$$

Since $\int \exp\left(-\left[a|\theta|^b + k\right]\right)\mathrm{d}\theta$ converges, then $\int \exp(A(\theta))\,\mathrm{d}\theta$ converges. Thus, according to Theorem 1, there exists $\kappa > 0$ such that $\alpha = \frac{1}{\kappa}\exp\circ A$ verifies:

$$\mathrm{KL}(\beta_\mu \| \alpha) = r(\mu),$$

up to a constant. □

G Proof of Corollary 2

We want to apply Theorem 1 in this case. We assume that α is a solution to Equation (4) with $r_{a,b}(\mu_k, \sigma_k^2) = a(\sigma_k^2) + b(\sigma_k^2)\mu_k^2$, such that $\alpha \in \mathcal{T}(\mathbb{R})$. That is, for some constant K:

$$r_{a,b}(\mu_k, \sigma_k^2) = \mathrm{KL}(\beta_{\mu_k,\sigma_k^2} \| \alpha) + K.$$

We check the hypotheses:

- for all $(\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}^+$, $\beta_{\mu,\sigma^2}(\cdot) = \beta_{0,\sigma^2}(\cdot - \mu)$ and $\beta_{0,\sigma^2} = \check{\beta}_{0,\sigma^2}$;
- $\beta_{\mu,\sigma^2} \in \mathcal{S}(\mathbb{R})$ and $\mathcal{F}\beta_{\mu,\sigma^2}$ is nonzero everywhere;
- $\mathcal{F}r_{a,b,\sigma^2} = 2\pi \left[a(\sigma^2)\delta - b(\sigma^2)\delta''\right] \in \mathcal{E}'(\mathbb{R})$.

Then, we apply Theorem 1, which ensures that r fulfills $(\star)$, that is A_{a,b,σ^2} does not depend on σ^2. First, we compute A_{a,b,σ^2} by using its definition, given in Equation (3). A first calculus (see Appendix G.1) leads to:

$$\frac{\mathcal{F}r_{a,b,\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}} = 2\pi \left[\left(a(\sigma^2) - \sigma^2 b(\sigma^2)\right)\delta - b(\sigma^2)\delta''\right], \tag{16}$$

and, according to the calculus made in Appendix G.2, we have:

$$A_{a,b,\sigma^2}(\theta) = -\ln(2\pi e\sigma^2) + \left[-a(\sigma^2) + \sigma^2 b(\sigma^2) - b(\sigma^2)\theta^2\right]. \tag{17}$$

Second, we prove in Appendix G.3 that A_{a,b,σ^2} does not depend on σ^2 if, and only if:

$$a(\sigma^2) = \sigma^2 b_0 - a_0 - \frac{\ln(2\pi e\sigma^2)}{2} \tag{18}$$

$$b(\sigma^2) = b_0, \tag{19}$$

where a_0 and b_0 are real constants. Thus, the loss $r_{a,b}$ takes the following form:

$$r_{a,b}(\mu, \sigma^2) = \sigma^2 b_0 - a_0 - \frac{\ln(2\pi e\sigma^2)}{2} + b_0\mu^2 + K.$$

In that case, $A_{a,b,\sigma^2}(\theta) = A_{a,b}(\theta) = a_0 - b_0\theta^2$. Thus $\alpha_{a,b} = \frac{1}{\kappa}\exp \circ A_{a,b}$ is a probability distribution, where:

$$\kappa = e^{a_0}\sqrt{\frac{2\pi}{2b_0}}$$

$$\alpha_{a,b}(\theta) = \frac{1}{\sqrt{2\pi\frac{1}{2b_0}}}\exp\left(-\frac{\theta^2}{2\frac{1}{2b_0}}\right).$$

Thus, $\alpha_{a,b}$ can be seen as the density of the Gaussian distribution $\mathcal{N}(0, \frac{1}{2b_0})$.

Therefore $\alpha_{a,b}$ is Gaussian and the penalty is necessarily the Kullback–Leibler divergence between two Gaussian distributions $\beta_{\mu,\sigma^2} \sim \mathcal{N}(\mu, \sigma^2)$ and $\alpha_{a,b} = \alpha_{\sigma_0^2} \sim \mathcal{N}(0, \sigma_0^2)$:

$$\begin{aligned} r_{\sigma_0^2}(\mu, \sigma^2) &= \frac{\sigma^2}{2\sigma_0^2} + \frac{\ln(2\pi\sigma_0^2)}{2} - \frac{\ln(2\pi e\sigma^2)}{2} + \frac{1}{2\sigma_0}\mu^2 + K \\ &= \frac{1}{2}\left[\frac{\sigma^2 + \mu^2}{\sigma_0^2} + \ln\left(\frac{\sigma_0^2}{\sigma^2}\right) - 1\right] + K. \end{aligned}$$

G.1 Proof of Equation (16)

We compute $\frac{\mathcal{F}r_{a,b,\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}}$. We recall that $\mathcal{F}r_{a,b,\sigma^2} = 2\pi\left[a(\sigma^2)\delta - b(\sigma^2)\delta''\right]$. Thus we have, for each $\varphi \in \mathcal{S}(\mathbb{R})$:

$$\begin{aligned}
\left\langle \frac{\mathcal{F}r_{a,b,\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}}, \varphi \right\rangle &= \left\langle \frac{2\pi\left[a(\sigma^2)\delta - b(\sigma^2)\delta''\right]}{\mathcal{F}\beta_{0,\sigma^2}}, \varphi \right\rangle \\
&= 2\pi \left\langle a(\sigma^2)\delta - b(\sigma^2)\delta'', \frac{\varphi}{\mathcal{F}\beta_{0,\sigma^2}} \right\rangle \\
&= 2\pi a(\sigma^2)\varphi(0) - 2\pi b(\sigma^2) \left\langle \delta'', \varphi(x) \exp\left(\frac{\sigma^2 x^2}{2}\right) \right\rangle_x \\
&= 2\pi a(\sigma^2)\varphi(0) + 2\pi b(\sigma^2) \left\langle \delta', \left(\varphi'(x) + \varphi(x)\sigma^2 x\right) \exp\left(\frac{\sigma^2 x^2}{2}\right) \right\rangle_x \\
&= 2\pi a(\sigma^2)\varphi(0) \\
&\quad - 2\pi b(\sigma^2) \left\langle \delta, \left(\varphi''(x) + 2\varphi'(x)\sigma^2 x + \varphi(x)\sigma^2(1+\sigma^2 x^2)\right) \exp\left(\frac{\sigma^2 x^2}{2}\right) \right\rangle_x \\
&= 2\pi\left[a(\sigma^2)\varphi(0) - b(\sigma^2)\left(\varphi''(0) + \sigma^2\varphi(0)\right)\right]
\end{aligned}$$

Thus, $\frac{\mathcal{F}r_{a,b,\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}} = 2\pi\left[\left(a(\sigma^2) - \sigma^2 b(\sigma^2)\right)\delta - b(\sigma^2)\delta''\right]$.

G.2 Proof of Equation (17)

Theorem 1 defines the following distribution:

$$A_{a,b,\sigma^2} = -\mathrm{Ent}(\beta_{0,\sigma^2})\mathbb{1} - \mathcal{F}^{-1}\left[\frac{\mathcal{F}r_{a,b,\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}}\right]$$

We have:

$$\begin{cases} \mathrm{Ent}(\beta_{0,\sigma^2}) & = \frac{1}{2}\ln(2\pi e\sigma^2) \\ \frac{\mathcal{F}r_{a,b\sigma^2}}{\mathcal{F}\beta_{0,\sigma^2}} & = 2\pi\left[\left(a(\sigma^2) - \sigma^2 b(\sigma^2)\right)\delta - b(\sigma^2)\delta''\right] \end{cases}$$

Besides, $2\pi\mathcal{F}^{-1}\delta = \mathbb{1}$ and $2\pi(\mathcal{F}^{-1}\delta'')(\theta) = -\theta^2$.

Thus:

$$\begin{aligned}
A_{a,b,\sigma^2}(\theta) &= -\frac{1}{2}\ln(2\pi e\sigma^2) - \left[a(\sigma^2) - \sigma^2 b(\sigma^2) + b(\sigma^2)\theta^2\right] \\
&= -\frac{1}{2}\ln(2\pi e\sigma^2) + \left[-a(\sigma^2) + \sigma^2 b(\sigma^2) - b(\sigma^2)\theta^2\right]
\end{aligned}$$

G.3 Proof of Conditions (18) and (19)

We have:

$$A_{a,b,\sigma^2}(\theta) = -\frac{1}{2}\ln(2\pi e\sigma^2) + \left[-a(\sigma^2) + \sigma^2 b(\sigma^2) - b(\sigma^2)\theta^2\right].$$

The polynomial A_{a,b,σ^2} does not depend on σ^2 if, and only if, its coefficients do not depend on σ^2. That is:

$$\begin{cases} b(\sigma^2) &= b_0 \\ -\frac{1}{2}\ln(2\pi e\sigma^2) - a(\sigma^2) + \sigma^2 b(\sigma^2) &= a_0 \end{cases},$$

that is:

$$\begin{cases} b(\sigma^2) &= b_0 \\ a(\sigma^2) &= \sigma^2 b_0 - a_0 - \frac{\ln(2\pi e\sigma^2)}{2} \end{cases}.$$