Predictive Complexity Priors: Supplementary Materials

Eric Nalisnick
University of Amsterdam

Jonathan Gordon
University of Cambridge

José Miguel Hernández-Lobato
University of Cambridge

1 ECP FOR LINEAR REGRESSION, CONTINUED

1.1 Varying Prior Parameters

We consider the following priors on the KLD, denoted $\pi(\kappa)$ in the main text:

- Exponential$(x; \lambda) = e^{-x/\lambda}/\lambda$
- Gamma$(x; \boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}) = \frac{x^{\lambda_1 - 1}}{\Gamma(\lambda_1)\lambda_2} e^{-x/\lambda_2}$
- Log-Cauchy$(x; \lambda) = \frac{1}{\pi x}\left(\frac{\lambda}{(\log x)^2 + \lambda^2}\right)$

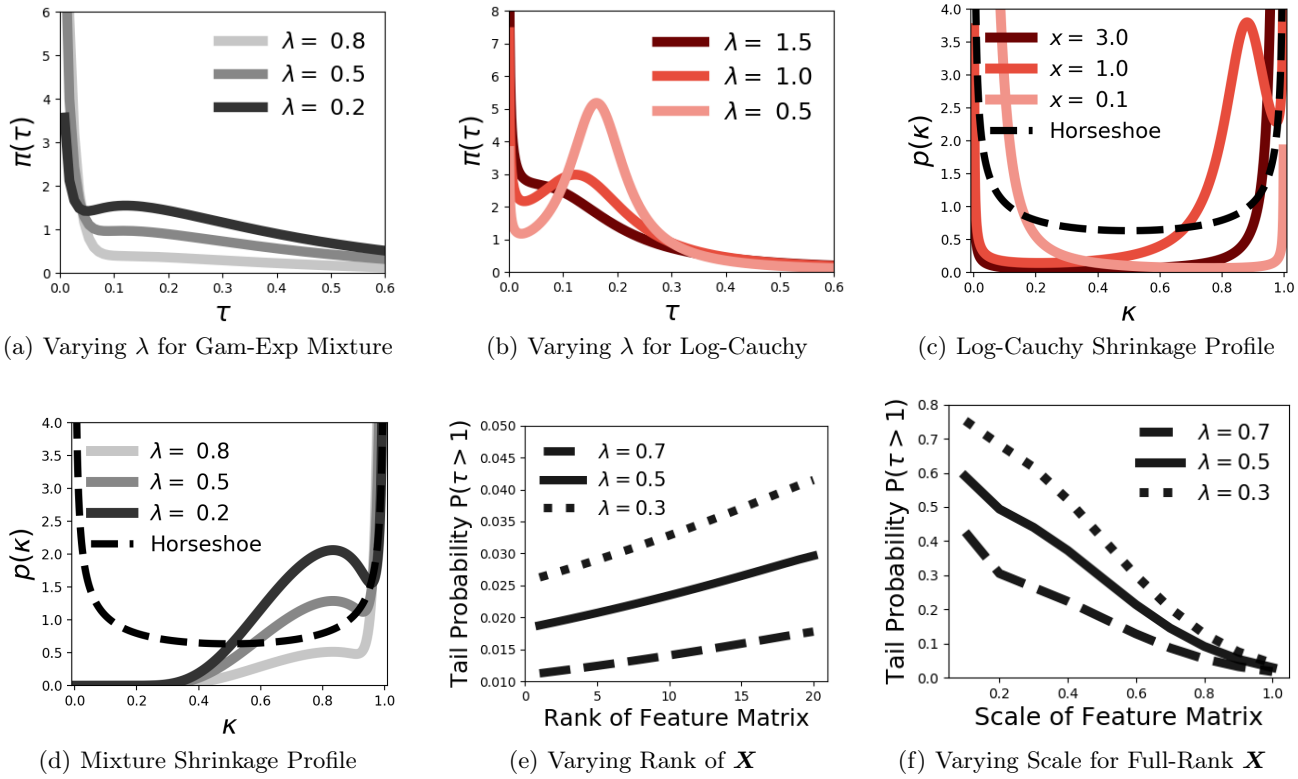
Since the gamma prior favors p_0 and the exponential p_+, we also consider a mixture of the two:

$$\pi(\kappa; \lambda) = \lambda \text{Gamma}\left(\text{KL}\left[p_+ \,||\, p_0\right]\right) + (1 - \lambda)\text{Exponential}\left(\text{KL}\left[p_+ \,||\, p_0\right]\right).$$

This mixture should achieve the same interpolation behavior as the log-Cauchy, balancing preferences for p_0 and p_+. See Figure 1(a) for a visualization of the ECP for the gamma-exponential mixture as the mixing coefficient is varied. Figure 1(b) shows the log-Cauchy ECP as its scale is varied.

1.2 Dynamic Shrinkage Profile

One method for illuminating and comparing the effects of shrinkage priors is through their corresponding *shrinkage profile* (Carvalho et al., 2009). Consider the model: $\mathrm{y} \sim \mathrm{N}(\theta, 1)$, $\theta \sim \mathrm{N}(0, \tau)$, $\tau \sim p(\tau)$. Given one observation $\mathrm{y} = y_0$, the Bayes estimator for θ is: $\mathbb{E}[\theta | y_0, \tau] = \kappa \cdot 0 + (1 - \kappa) \cdot y_0$, $\kappa = 1/(1 + \tau)$. Making the change of variables $p(\kappa) = p(\tau)|\frac{d}{d\kappa} 1/\kappa - 1|$, we can examine the induced prior $p(\kappa)$. If $p(\kappa)$ places high density near $\kappa = 1$, then the prior provides strong regularization since the Bayes estimator would be near zero. Conversely, a strong mode at $\kappa = 0$ means the resulting estimator would be near y_0, implying a tendency to follow the data. Figure 1(c) shows the shrinkage profile for the log-Cauchy ECP for $x \in \{.1, 1, 3\}$, comparing the profiles of the horseshoe's. The horseshoe is an effective prior for robust regression since it is designed to equally favor 0 and 1 ($\kappa \sim$ Beta$(.5, .5)$). The shrinkage profile for the log-Cauchy also can place density at both extremes. However, unlike the horseshoe, the ECP enables *dynamic* shrinkage, being able to adjust the profile as a function of x. We see that for $x = 0.1$, the log-Cauchy ECP actually favors the unshrunk solution whereas for $x = 3$ strong shrinkage is preferred. The mixture ECP's shrinkage profile is shown in Figure 1(d). Due to the exponential not having a heavy tail, the mixture ECP's profile cannot place any significant density at $\kappa = 0$, meaning that the model can never completely 'forget' the shrinkage regularization.



(a) Varying λ for Gam-Exp Mixture (b) Varying λ for Log-Cauchy (c) Log-Cauchy Shrinkage Profile

(d) Mixture Shrinkage Profile (e) Varying Rank of $\boldsymbol{X}$ (f) Varying Scale for Full-Rank $\boldsymbol{X}$

Figure 1: $p(\tau)$ *for Linear Regression.* Subfigures 1(a) and 1(b) show how the mixture and log-Cauchy ECP for τ changes as a function of λ. Subfigures 1(c) and 1(d) show the shrinkage profiles for the log-Cauchy and mixture ECPs (respectively). Subfigures 1(e) and 1(f) show how varying the rank and scale of the design matrix affects the ECP's tail probability.

1.3 Example: Multivariate Regression

We now examine the case of multivariate regression: $\mathbb{E}[\mathbf{y}|\boldsymbol{X}, \boldsymbol{\beta}] = \boldsymbol{X}\boldsymbol{\beta}$, where $\mathbf{y} \in \mathbb{R}^N$ is an N-dimensional vector of responses, $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ the design matrix, and $\boldsymbol{\beta} \in \mathbb{R}^D$ a vector of parameters. Using the multivariate analogs of the priors from above—$p_0 = \delta(|\boldsymbol{\beta} - \mathbf{0}|)$, $p_+ = \mathrm{N}(\mathbf{0}, \tau\mathbb{I})$—we can derive the following ECP:

$$p(\tau; \boldsymbol{\lambda}, \boldsymbol{X}) = \pi_{\mathrm{KL}}\left(\frac{-1}{2}\log|\mathbb{I} + \tau\sigma_y^{-2}\boldsymbol{X}^T\boldsymbol{X}| + \frac{\tau}{2\sigma_y^2}\mathrm{tr}\left\{\boldsymbol{X}^T\boldsymbol{X}\right\}; \boldsymbol{\lambda}\right)\left|\frac{\partial\ \mathrm{KL}}{\partial\ \tau}\right| \tag{1}$$

where $\mathrm{tr}\{\cdot\}$ denotes the trace operation and $|\cdot|$ the determinant. The KLD is computed between $p(\mathbf{y}|\boldsymbol{X}, \tau)$ and $p_0(\mathbf{y}|\boldsymbol{X})$. We omit the details of the volume term due to space constraints.

Here the KLD is a multidimensional integral that takes into account *correlations* in the model's predictions. Hence, we can explore how characteristics of the design matrix influence the prior. We consider the tail probability $P(\tau > 1)$ since, as probability mass moves away from the origin, the ECP increasingly prefers the extended model. Below we describe simulations using the mixture ECP due to its mixing weight λ being interpretable. First consider the case in which $\boldsymbol{X}$ has a rank of one and all row vectors have a length of one. This means that the data is essentially redundant, generating the same predictions. As the rank of $\boldsymbol{X}$ increases, the predictions start to become varied and the model output becomes responsive to each $\boldsymbol{x}_n$. Thus, we should expect $p(\tau)$ to favor larger values as the rank of $\boldsymbol{X}$ increases. Indeed, this is exactly the behavior we observe in Figure 1(e), plotting the tail probability $P(\tau > 1)$ as we vary the rank from 1 to 20. Another attribute of $\boldsymbol{X}$ we can vary is its scale: $\alpha\boldsymbol{X}$. Subfigure 1(f) shows the tail probability as the scale α increases (for full-rank $\boldsymbol{X}$). We find that scale changes result in more pronounced tail effects than changes in rank.

1.4 Example: ECP for the Linear Regression Coefficient

In the main text, we consider applying the ECP to the scale of the prior on $\boldsymbol{\beta}$, the regression coefficients. Yet, we can also define the ECP on $\boldsymbol{\beta}$ directly. Consider the linear model $\mathbb{E}[\mathrm{y}|x, \boldsymbol{\beta}] = \beta_0 + \beta_1 x, \;\; \beta_0 \sim \mathrm{N}(0, \sigma^2_{\beta_0})$. We wish a define the ECP on β_1 to control deviation from the base model $p_0 : \beta_1 = 0$:

$$\begin{aligned} p(\beta_1; \lambda, x) &= \pi_{\mathrm{KL}} \left(\mathrm{KL} \left[p_+(\mathrm{y}|x, \beta_1) \;||\; p_0(\mathrm{y}|x) \right]; \lambda \right) \left| \frac{\partial \; \mathrm{KL} \left[p_+(\mathrm{y}|x, \beta_1) \;||\; p_0(\mathrm{y}|x) \right]}{\partial \beta_1} \right| \\ &= \frac{1}{2} \; \pi_{\mathrm{KL}} \left(\frac{\beta_1^2 x^2}{2(\sigma_y^2 + \sigma^2_{\beta_0})}; \lambda \right) \left| \frac{\beta_1 x^2}{\sigma_y^2 + \sigma^2_{\beta_0}} \right| \end{aligned} \tag{2}$$

where σ_y^2 is the response noise. Figure 2(a) shows three choices for π_{KL}—exponential (green), gamma (purple), and log-Cauchy (red)—and Figure 2(b) shows the prior each induces on β_1. We see that choice of π_{KL} is significant. If π_{KL} places too little density at zero, the volume term $|\beta_1 x^2/(\sigma_y^2 + \sigma^2_{\beta_0})|$ in the ECP dominates, driving $p(\beta_1)$ to zero at $\beta_1 = 0$. In turn, this drives β_1 from reflecting the behavior of p_0 and suppresses any regularization. We see this behavior from the exponential (green) as its ECP has no density at zero. At the other end of the spectrum is the gamma (purple). It places too much density at zero, forcing $p(\beta_1)$ to preference the base model $\beta_1 = 0$. Lastly, the log-Cauchy (red) has the most interesting behavior: it has strong shrinkage at the origin to reflect p_0 but also significant density away from zero to represent p_+. Hence the log-Cauchy is able to balance preferences for both p_+ and p_0, interpolating between the exponential and gamma's single-mindedness. Lastly, Figure 2(c) shows how $p(\beta_1)$ changes as x is varied—the data adaptive nature mentioned above. The ECP's shrinkage is adjusted to the scale of the features, applying stronger regularization when x is large and relaxing as x decreases. This is sensible since the models predictions cannot change as drastically for small xs as they can for large xs.

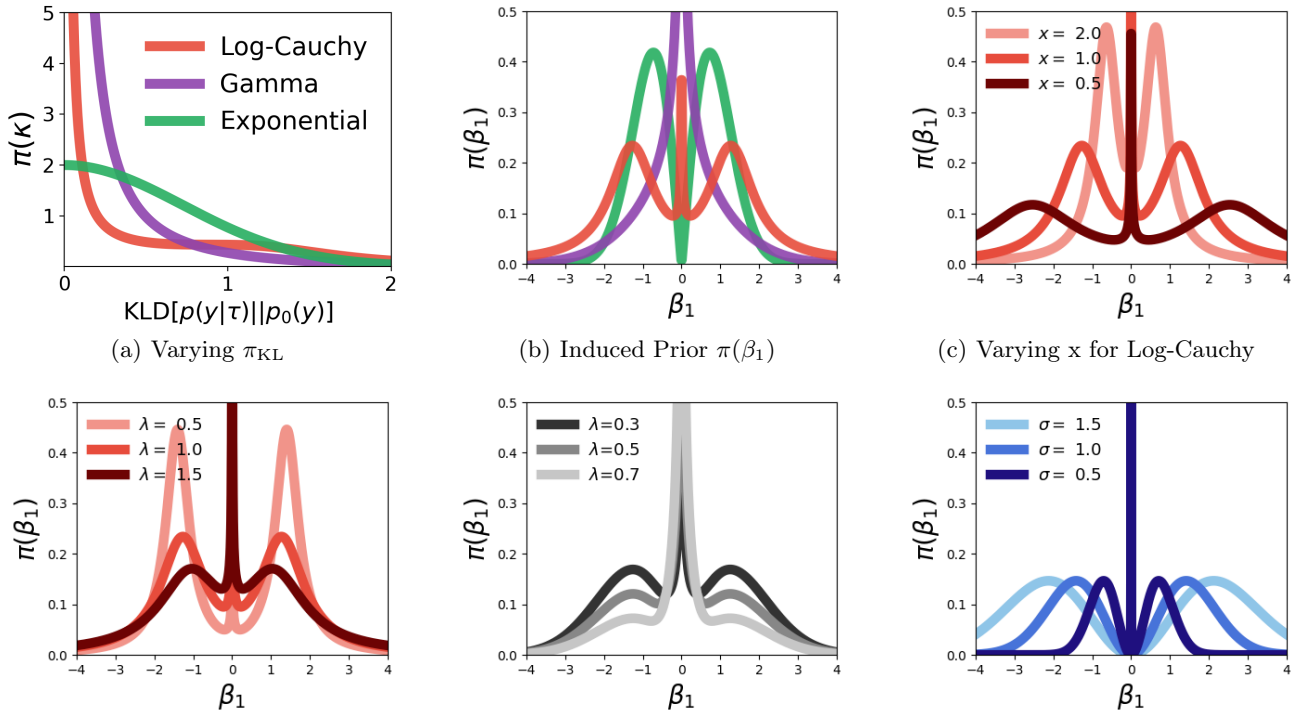
Connection to Non-Local Priors Perhaps the (log-Cauchy and mixture) ECP's most interesting feature is in how it balances between p_0 and p_+ through multi-modality. In addition to the strong mode at zero, there are modes separated from and symmetric about the origin; see Figure 2(d). This is the defining feature of so-called *non-local priors* (Johnson and Rossell, 2010). This class of priors is designed to achieve good convergence rates in Bayesian hypothesis testing by carefully allocating density exactly at the null and distinctly away from the null to represent the alternative. Shin et al. (2018) apply this non-local principle to Bayesian variable selection in high-dimensional regression via the following prior:

$$\beta \sim \mathrm{z}\delta_0 + (1 - \mathrm{z}) p_{\mathrm{NL}}(\beta), \qquad \mathrm{z} \sim \mathrm{Bernoulli}(\rho), \qquad p_{\mathrm{NL}}(\beta) = \frac{\beta^2}{\sigma} \mathrm{N}(\beta; 0, \sigma) \tag{3}$$

where β is the linear model's coefficient and p_{NL} is known as a *product (2nd) moment prior* (Johnson and Rossell, 2010). See Figure 2(f) for a visualization. While the ECP and non-local prior have clear similarities, the connection can be made explicit my considering the Bayes factor $\mathrm{BF}(+|0) = p_+(\mathrm{y}|\boldsymbol{x}, \theta_+)/p_0(\mathrm{y}|\boldsymbol{x})$, which is what we would use to test $\mathrm{H}_0 : \theta_+ = 0$ vs $\mathrm{H}_1 : \theta_+ \neq 0$. The ECP's KLD term contains this exact expression: $\mathrm{KL}\left[p_+ \;||\; p_0\right] = \mathbb{E}_{p_+}\left[\log \mathrm{BF}(+|0)\right]$, which can be interpreted as the (log) Bayes factor expected if the data is truly generated by the extended model.

2 BIJECTIVITY CONDITIONS FOR NEURAL NETWORKS

Below we show that $\mathbb{E}_{\boldsymbol{\theta}|\tau} \mathrm{KL}[p_+||p_0]$ is bijective w.r.t. τ for Gaussian and categorical observation models, ReLU activations, and Gaussian weight priors $\mathbf{W}_l \sim \mathrm{N}(\boldsymbol{\Phi}_l, \tau_l \boldsymbol{\Sigma})$. We assume the base model has weights $\mathbf{W}_l \sim \delta[\boldsymbol{\Phi}_l]$, where $\boldsymbol{\Phi}_l$ is the prior mean of the expanded model. In turn, the change of variables is well-defined and the PredCP is proper (i.e. integrates to 1) when used in the applications given in Section 5. Before diving into the technical details, we point out that invertibility should not be hard to guarantee since $\tau \in \mathbb{R}^+$ is a scalar. If the reader is familiar with the literature on *normalizing flows* (Papamakarios et al., 2019) and invertible architectures (Song et al., 2019), one may have the impression that invertibility is hard to guarantee for neural networks, often requiring constraints on the weights. This is only the case because an invertible architecture must preserve bijectivity w.r.t. the entire input *vector*. We, to the contrary, just have to preserve scalar information. Moreover, we know this scalar τ is strictly positive, which eliminates any symmetry about the origin. Thus, intuitively, the information about τ should be preserved at every hidden layer as long as at least one ReLU unit is active. This brings up the only assumption that we require: *non-degeneracy*. That is, for every hidden layer, there



(a) Varying π_{KL} (b) Induced Prior $\pi(\beta_1)$ (c) Varying x for Log-Cauchy

(d) Varying λ for Log-Cauchy (e) Varying λ for Gam-Exp Mixture (f) Varying σ for Non-Local Prior

Figure 2: $\pi(\beta_1)$ *for Linear Regression.* Subfigure 2(a) shows three choices for π_{KL}: exponential ($\lambda = .5$), gamma ($\lambda = (.2, 2)$), and log-Cauchy ($\lambda = 1$). Subfigure 2(b) shows the priors induced on β_1 for each KLD prior. Subfigure 2(c) shows how the log-Cauchy ECP changes as a function of x. Subfigure 2(d) shows how the log-Cauchy ECP changes when varying its scale parameter ($x = 1$). Subfigure 2(e) shows how the mixture (gamma and exponential) ECP changes when varying the mixture weight λ ($x = 1$). Subfigure 2(f) shows the non-local prior for three scales.

must be at least one active ReLU unit (i.e. a unit evaluated to a positive value). This is an extremely weak assumption, especially for all but the smallest neural networks, and ReLU networks commonly satisfy much stronger non-degeneracy assumptions (Phuong and Lampert, 2020). If dead layers would arise for some reason, initializing the biases to be positive should prevent the pathology. Before moving on to the main results, we first introduce two conventions.

Non-Centered Parameterization: We assume all weights are represented in the observation model $p(\mathbf{y}|\boldsymbol{X}, \mathbf{W})$ using the following non-centered form: $\mathbf{W}_l \overset{d}{=} \boldsymbol{\Phi}_l + \sqrt{\tau_l} \cdot \boldsymbol{\Xi}$, $\boldsymbol{\Xi} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma})$. This parameterization is useful for 'exposing' τ_l so that we can write the KLD as an explicit function of $\boldsymbol{\tau}$.

Linear Representation of ReLU Activations Recall that feedforward networks with ReLU activations are piecewise-linear functions. Thus, it is equivalent to express a dense ReLU layer in terms of a diagonal 'gating' matrix $\tilde{\mathbb{I}}$ (Ma et al., 2018):

$$\texttt{ReLU}(\mathbf{hW}) = \mathbf{hW}\tilde{\mathbb{I}}, \quad \text{where} \quad \tilde{\mathrm{i}}_{j,j} = 1 \quad \text{if} \quad 0 < \sum_{i=1}^{D} \mathrm{w}_{i,j}\mathrm{h}_i, \tag{4}$$

otherwise $\tilde{\mathrm{i}}_{j,j} = 0$. Using this convention, we can then represent the network's linear output as:

$$\mathbf{F}_{out} = \boldsymbol{X}\left(\prod_{l=1}^{L} \mathbf{W}_l \tilde{\mathbb{I}}_l\right) \boldsymbol{W}_{out}$$

where $\mathbf{F}_{out}$ is a matrix containing the network's linear output (i.e. before a softmax is applied, in the classification setting) for all training features $\boldsymbol{X}$.

2.1 Gaussian Observation Model

Let's now consider checking for bijectivity w.r.t. τ_l when the observation model is Gaussian, i.e. $\mathbf{y} \sim \mathrm{N}(\mathbf{f}_{out}, \sigma_y^2\mathbb{I})$. The expected KLD is then:

$$\begin{aligned}\mathbb{E}_{\mathbf{W}_l|\tau_l}\mathrm{KL}[\mathrm{N}(\mathbf{F}^+_{out}, \sigma_y^2\mathbb{I})||\mathrm{N}(\boldsymbol{F}^0_{out}, \sigma_y^2\mathbb{I})] &= \frac{1}{2\sigma_y^2}\sum_{n=1}^{N}\sum_{d=1}^{D}\mathbb{E}_{\mathbf{W}_l|\tau_l}\left[\left(\mathrm{f}^+_{n,d,out} - f^0_{n,d,out}\right)^2\right] \\ &= \frac{1}{2\sigma_y^2}\sum_{n=1}^{N}\sum_{d=1}^{D}\mathbb{E}_{\mathbf{W}_l|\tau_l}[(\mathrm{f}^+_{n,d,out})^2] - 2\mathbb{E}_{\mathbf{W}_l|\tau_l}[\mathrm{f}^+_{n,d,out}]f^0_{n,d,out} + (f^0_{n,d,out})^2\end{aligned} \tag{5}$$

where n indexes the training features and d the output dimensions. Since f^0_{out} does not involve τ_l, we can treat is as a constant. The sum over n and d forms a conic combination. Therefore if all terms have the same monotonicity and are bijective, then the sum will be bijective as well.

We first turn toward the expectation in the middle term and expand it using the non-centered parameterization. We drop the indexes to help with notational clutter:

$$\begin{aligned}\mathbb{E}_{\mathbf{W}_l|\tau_l}[f^+_{out}] &= \mathbb{E}_{\mathbf{W}_l|\tau_l}\left[\boldsymbol{x}\left(\prod_{l=1}^{L}\tilde{\mathbb{I}}_l\mathbf{W}_l\right)\boldsymbol{w}_{out}\right] \\ &= \mathbb{E}_{\boldsymbol{\Xi}_l}\left[\boldsymbol{x}\left(\prod_{l=1}^{L}(\boldsymbol{\Phi}_l + \sqrt{\tau_l}\cdot\boldsymbol{\Xi}_l)\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right] \\ &= \mathbb{E}_{\boldsymbol{\Xi}_l}\left[\boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Phi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right] + \mathbb{E}_{\boldsymbol{\Xi}_l}\left[\boldsymbol{x}\left(\prod_{l=1}^{L}\sqrt{\tau_l}\cdot\boldsymbol{\Xi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right].\end{aligned} \tag{6}$$

Pushing through the expectation, we have:

$$= \boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Phi}_l\mathbb{E}_{\boldsymbol{\Xi}_l}\left[\tilde{\mathbb{I}}_l\right]\right)\boldsymbol{w}_{out} + \boldsymbol{x}\left(\prod_{l=1}^{L}\sqrt{\tau_l}\cdot\mathbb{E}_{\boldsymbol{\Xi}_l}\left[\boldsymbol{\Xi}_l\tilde{\mathbb{I}}_l\right]\right)\boldsymbol{w}_{out}.$$

Considering the first term, we have that $\mathbb{E}[\tilde{i}] = 1$ if $\sum_i \phi_{i,j} \mathrm{h}_i > 0$. For the second, $\mathbb{E}_{\boldsymbol{\Xi}_l}\left[\boldsymbol{\Xi}_l \tilde{\mathbb{I}}_l\right] = \mathbf{0}$. Thus, $\mathbb{E}_{\mathbf{W}_l|\tau_l}[f^+_{out}]$ reduces to the first term, and since this term depends only on the prior means, it is equivalent to the output of the base model:

$$\mathbb{E}_{\mathbf{W}_l|\tau_l}[f^+_{out}] = f^0_{out}. \tag{7}$$

Now turning to the other expectation term in Equation 5 and using the expansion from Equation 6, we have:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{W}_l|\tau_l}[(f^+_{out})^2] = \\
&\quad \mathbb{E}_{\boldsymbol{\Xi}_l}\Bigg[\left(\boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Phi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right)^2 + 2\left(\boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Phi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right)\left(\boldsymbol{x}\left(\prod_{l=1}^{L}\sqrt{\tau_l}\cdot\boldsymbol{\Xi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right) \\
&\quad + \left(\boldsymbol{x}\left(\prod_{l=1}^{L}\sqrt{\tau_l}\cdot\boldsymbol{\Xi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right)^2\Bigg].
\end{aligned}$$

The middle term drops out after taking the expectation. We are left with the two remaining terms:

$$\mathbb{E}_{\mathbf{W}_l|\tau_l}[(f^+_{out})^2] = \underbrace{\mathbb{E}_{\boldsymbol{\Xi}_l}\left[\left(\boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Phi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right)^2\right]}_{\gamma_1^{>0}} + \tau_l \cdot \underbrace{\prod_{l'\neq l}\tau_{l'}\cdot\mathbb{E}_{\boldsymbol{\Xi}_l}\left[\left(\boldsymbol{x}\left(\prod_{l=1}^{L}\boldsymbol{\Xi}_l\tilde{\mathbb{I}}_l\right)\boldsymbol{w}_{out}\right)^2\right]}_{\gamma_2^{>0}}.$$

We use the variables γ_1 and γ_2 to denote these two terms from here forward. They have the following three properties that will come in handy below. Firstly, their values are strictly positive due to the square and non-degeneracy assumption ($\mathrm{trace}\{\tilde{\mathbb{I}}_l\} \neq 0 \ \ \forall\, l$). We emphasize this by giving them the superscript > 0. Secondly, we know their derivative w.r.t. τ_l is zero. This is true for γ_1 because it depends on τ_l only through $\tilde{\mathbb{I}}$ and therefore only through the sign function. For γ_2, it does not depend on τ_l; we have factored it out already. Thirdly, γ_1 and γ_2 are bounded w.r.t. τ_l. We know this in the former case because, again, τ_l controls only the 'gates' in γ_1 and $\boldsymbol{\Phi}$ is bounded. The latter case is trivial due to γ_2 not being a function of τ_l. Now substituting back into Equation 5, we have

$$\begin{aligned}
&\mathbb{E}_{\mathbf{W}_l|\tau_l}[(\mathrm{f}^+_{out})^2] - 2\mathbb{E}_{\mathbf{W}_l|\tau_l}[\mathrm{f}^+_{out}]f^0_{out} + (f^0_{out})^2 \\
&\qquad = \gamma_1^{>0} + \tau_l\gamma_2^{>0} - 2f^0_{out}f^0_{out} + (f^0_{out})^2 \\
&\qquad = \gamma_1^{>0} + \tau_l\gamma_2^{>0} - (f^0_{out})^2.
\end{aligned} \tag{8}$$

All that is left to do is to check for injectivity and surjectivity. For the former, we need to verify the derivative has a constant sign, and we find that:

$$\frac{\partial}{\partial\tau_l}\left(\mathbb{E}_{\mathbf{W}_l|\tau_l}[(\mathrm{f}^+_{out})^2] - 2\mathbb{E}_{\mathbf{W}_l|\tau_l}[\mathrm{f}^+_{out}]f^0_{out} + (f^0_{out})^2\right) = \gamma_2^{>0}.$$

Clearly, the derivative is always positive, meaning the function is strictly increasing. Lastly, for surjectivity, it is sufficient to check that

$$\begin{aligned}
\gamma_1^{>0} + \tau_l\gamma_2^{>0} - (f^0_{out})^2 \to 0^+ \quad &\text{as} \quad \tau_l \to 0^+ \\
\gamma_1^{>0} + \tau_l\gamma_2^{>0} - (f^0_{out})^2 \to \infty \quad &\text{as} \quad \tau_l \to \infty.
\end{aligned}$$

As $\tau \to 0$, we have that $\gamma_1 \to (f^0_{out})^2$, therefore causing the first and last terms to cancel. The remaining $\tau_l\gamma_2^{>0}$ term is then forced to zero simply by τ going to zero. As $\tau \to \infty$, the second term is sufficient for the full quantity to go to infinity since all other terms are bounded as a function of τ.

2.2 Categorical Observation Model

In the previous subsection, we showed that bijectivity is preserved at least up through the neural network's linear output $\mathbf{F}^+_{out}$. We now consider the case of a categorical observation model. This case is harder to show explicitly because the expectation cannot be pushed through to individual terms as we did above. Rather, we base the

argument on the fact that compositions of bijections form a bijection. In particular, given that f^+_{out} is bijective, we need the three following operations to preserve that bijectivity: $\mathbb{E}_{\boldsymbol{\Xi}} \circ \text{KLD} \circ \texttt{softmax} \circ f^+_{out}(\tau_l)$.

First considering the softmax, the softmax function is defined as:

$$\pi_j = \frac{\exp\{\tau x_j\}}{\sum_{d=1}^{D} \exp\{\tau x_d\}}$$

where $x_j \in \mathbb{R}$, $\tau \in \mathbb{R}^+$, $0 < \pi_j < 1$, and $\sum_{d=1}^{D} \pi_d = 1$. We want to prove that the softmax is invertible w.r.t. the scale τ for fixed $\boldsymbol{x} = \{x_1, \ldots, x_D\}$. Since τ is a scalar, it suffices that only one π_j need be invertible. Below we show bijectity by showing injectivity and then surjectivity. We can show injectivity by examining the softmax's derivative w.r.t. τ and showing it has a constant sign:

$$\pi'_j(\tau) = \underbrace{\pi_j}_{(+)} \underbrace{\left(x_j - \sum_{d=1}^{D} \pi_d x_d \right)}_{(+) \text{ or } (-)}$$

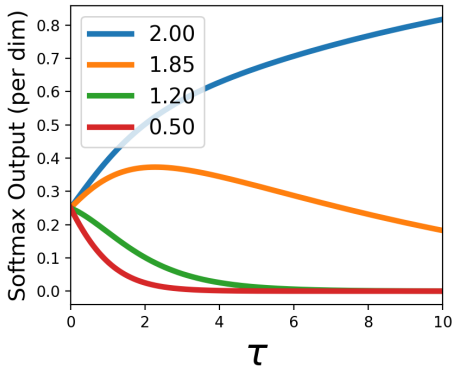
where the first term π_j is positive and the second could be positive or negative, depending on the $\boldsymbol{x}$ values. Denoting the maximum element of $\boldsymbol{x}$ as $x_{\max} = \max_d x_d$, the derivative in this dimension is:

$$\pi'_{\max}(\tau) = \underbrace{\pi_{\max}}_{(+)} \underbrace{\left(x_{\max} - \sum_{d=1}^{D} \pi_d x_d \right)}_{(+)} > 0,$$

and now we know the second term is strictly positive since $\sum_{d=1}^{D} \pi_d x_d$ is a convex combination of $\boldsymbol{x}$. That is, assuming that there's at least one $x_d < x_{\max}$, then $\sum_{d=1}^{D} \pi_d x_d < x_{\max}$. Conversely, for $x_{\min} = \min_d x_d$, we have:

$$\pi'_{\min}(\tau) = \underbrace{\pi_{\min}}_{(+)} \underbrace{\left(x_{\min} - \sum_{d=1}^{D} \pi_d x_d \right)}_{(-)} < 0,$$

since $\sum_{d=1}^{D} \pi_d x_d > x_{\min}$. Thus, we have that at least two dimensions—those corresponding to $x_{\min}$ and $x_{\max}$—of the softmax are injective w.r.t. τ. For surjectivity, it is sufficient to check the limits. For $\tau \to 0^+$, $\pi_j \longrightarrow 1/D$ $(\forall j)$, and for $\tau \to \infty$, we have $\pi_{\min} \longrightarrow 0$ and $\pi_{\max} \longrightarrow 1$. Thus, $\pi_{\min}$ is strictly *decreasing* on $(1/D, 0)$ and that $\pi_{\max}$ is strictly *increasing* on $(1/D, 1)$. In turn, $\texttt{Softmax}{:}\tau \mapsto \pi_{\min/\max}$ is a bijection. The figure to the right shows the softmax outputs for one particular setting of $\boldsymbol{x}$; each line represents an output π_j. We see that the blue ($x = 2.00$) and red ($x = 0.50$) lines, the max and min dimensions respectively, are strictly monotonic. On the other hand, the orange line ($x = 1.85$) is clearly not bijective since it doesn't pass the horizontal line test.



On to the KLD, $\text{KL}[p^+||p^0]$ is strictly convex w.r.t. p^+ for fixed p^0. This is indeed our setting since only p^+ is a function of τ_l. This strict convexity implies that $\text{KL}[p^+||p^0]$ has a unique, global minimum at $\tau = 0$ and is strictly increasing as $\tau \to \infty$—and hence, is bijective. Lastly, the expectation does not have an analytical solution, and thus here (as well as in practice), we consider the Monte Carlo approximation:

$$\mathbb{E}_{\boldsymbol{\Xi}_l} \text{KL}\left[p(\mathbf{y}|\mathbf{X}, \boldsymbol{\Xi}, \tau_l)||p_0(\mathbf{y}|\mathbf{X})\right] \approx \frac{1}{S} \sum_{s=1}^{S} \text{KL}\left[p(\mathbf{y}|\mathbf{X}, \hat{\boldsymbol{\Xi}}_s, \tau_l)||p_0(\mathbf{y}|\mathbf{X})\right]$$

where the approximation becomes exact as $S \to \infty$. The Monte Carlo approximation is a weighted sum of bijective, strictly increasing functions and therefore is also strictly increasing and bijective.

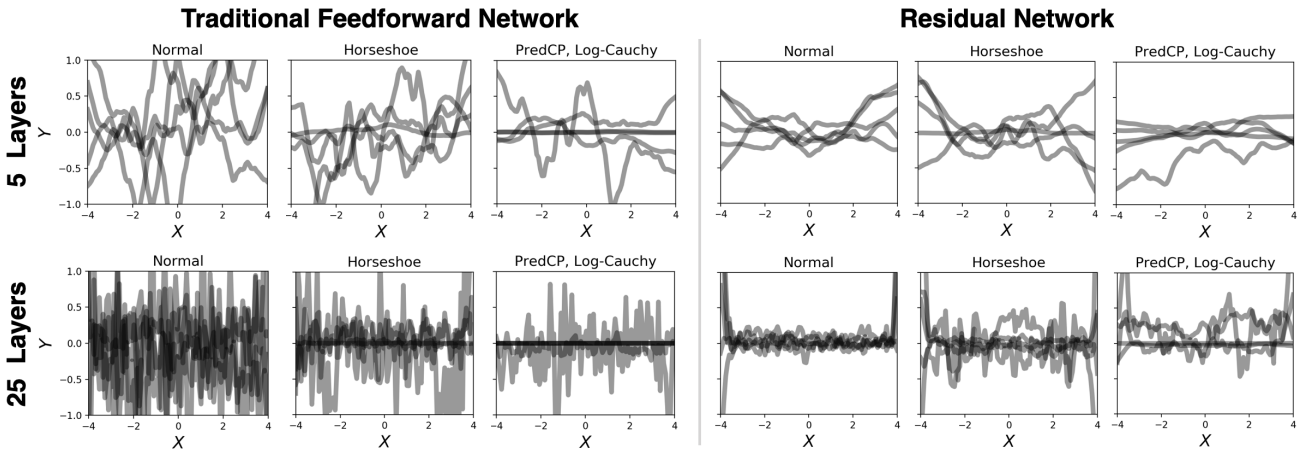


Figure 3: *Induced Prior Over Functions.* We compare a standard normal prior, the horseshoe ($\tau \sim C^+(0,1)$), and the depth-wise PredCP ($\pi(\kappa) = \text{Log-Cauchy}(0,1)$) for 5 and 25 layer NNs.

2.3 Residual Networks

The preceding two sections hold as well for residual networks. In fact, residual networks allow us to relax the non-degeneracy assumption. That is, for invertibility w.r.t. τ_l, we can have $\boldsymbol{h}_j = \mathbf{0}$ for $j > l$ just so long as there is a path of skip (identity) connections from layer l to layer $m > j$. This identity path preserves the information about τ_l that would be lost due to $\boldsymbol{h}_j = \mathbf{0}$.

3 DRAWING SAMPLES FROM THE PREDCP

The user may want to draw samples from the PredCP—for instance, to perform a prior predictive check (Box, 1980). In general, sampling from a reparameterized model can be done via: $\hat{x} \sim p(\mathrm{x})$, $\hat{\theta} = T(\hat{x})$ where $p(\mathrm{x})$ is the base distribution and T is the transformation (Papamakarios et al., 2019). Applying this formula to the PredCP, in theory, we could draw samples via:

$$\hat{\kappa} \sim \pi(\kappa), \quad \hat{\tau} = T^{-1}(\hat{\kappa}) \tag{9}$$

where $\hat{\kappa}$ represents a sampled value of $\mathbb{E}_{\boldsymbol{\theta}|\tau}\mathrm{KL}\left[p_+||p_0\right]$ and T^{-1} represents an inversion of the expected-KLD, yielding τ as a function of κ. Unfortunately, the analytical inverse of $\mathbb{E}_{\boldsymbol{\theta}|\tau}\mathrm{KL}\left[p_+||p_0\right]$ is not available in general. Instead, we use the numerical inversion technique proposed by Song et al. (2019). See Algorithm 1 below.

Algorithm 1 Sampling from the PredCP

Input: Prior $\pi(\kappa)$, number of iterations T, step size α
Sample $\hat{\kappa} \sim \pi(\kappa)$
Initialize $\tau_0 \leftarrow \hat{\kappa}$
For $t = [1, T]$:
  Compute $\mathbb{D}(\tau_{t-1}) = \mathbb{E}_{\boldsymbol{\theta}|\tau_{t-1}}\mathrm{KL}\left[p_+||p_0\right]$
  Compute $\partial\mathbb{D}(\tau_{t-1})/\partial\tau_{t-1}$
  Update $\tau_t \leftarrow \tau_{t-1} - \alpha\left(\partial\mathbb{D}(\tau_{t-1})/\partial\tau_{t-1}\right)^{-1}\left(\mathbb{D}(\tau_{t-1}) - \hat{\kappa}\right)$
Return τ_T

We used Algorithm 1 to sample from the depth-wise PredCP and ancestral sampling to draw functions from the NN. We considered both residual and non-residual architectures, both having batch normalization applied at each hidden layer. The observation model is Gaussian with $\sigma_y = 1$. We used a log-Cauchy$(0,1)$ KLD prior, $\alpha = 5 \times 10^{-5}$, $T = 20$, and 5 Monte Carlo samples for the $\boldsymbol{\theta}|\tau$ expectation. In Figure 3, we show 5 function samples for the standard Normal, horseshoe ($\tau \sim C^+(0,1)$), and PredCP for 5 and 25 hidden layers. The PredCP's samples are notably simpler.

4 MODULAR PRIOR SPECIFICATION FOR META-LEARNING

We provide details for specifying and evaluating *modular* priors for meta-learning. Following Chen et al. (2019), we split $\boldsymbol{\theta}$ into M distinct modules (e.g., layers), such that $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_M^T]^T$. Our goal is then to place a separate prior on each module, and allow the modules to adapt differently to new tasks. Denoting $\boldsymbol{\phi}_m$ and τ_m as the global parameters and shrinkage parameter of module m, respectively, we have that $\boldsymbol{\theta}_{t,m} \sim \mathrm{N}(\boldsymbol{\phi}_m, \tau_m \mathbb{I})$ for the local parameters of task t at module m. By specifying $p_m(\tau_m)$, we can control how much each module is allowed to deviate from the global model. For example, Chen et al. (2019) place an improper flat prior on τ_m and perform MAP estimation. Importantly, to perform MAP estimation for $\{\tau_m\}_{m=1}^M$, we need only evaluate the log density of the priors $p_m(\tau_m)$, and add these terms to the outer-loop optimization objective (Chen et al., 2019).

Defining a PredCP Prior for τ In this setting, It is natural to consider the global parameters as defining the base model for the PredCP prior. We can achieve this by specifying the prior to be $p_{0,m}(\boldsymbol{\theta}_{t,m}) = \delta(\boldsymbol{\theta}_{t,m} - \boldsymbol{\phi}_m)$. Denoting $p_0(\boldsymbol{\theta}) = \prod_{m=1}^M p_{0,m}(\boldsymbol{\theta}_m)$, we then have

$$p_0(\mathbf{y}|\boldsymbol{x}) = \int p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\theta}) p_0(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = p_{\boldsymbol{\phi}}(\mathbf{y}|\boldsymbol{x}),$$

where we denote $p_{\boldsymbol{\phi}}(\mathbf{y}|\boldsymbol{x})$ as the predictions made by the model using only global parameters. Next, for every module m, we can define the extended model

$$p_m(\mathbf{y}|\boldsymbol{x}, \tau_m) = \int p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta}_m) p_m(\boldsymbol{\theta}_m|\tau_m) \mathrm{d}\boldsymbol{\theta}_m,$$

where we use the notation $p(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta}_m)$ to denote the model that uses $\boldsymbol{\phi}$ for all but module m, which uses $\boldsymbol{\theta}_m$. Using the notation from the main text, we further denote $p_0(D_t) = \prod_{\boldsymbol{x},\mathbf{y} \in D_t} p_0(\mathbf{y}|\boldsymbol{x})$ and $p_m(D_t|\boldsymbol{\theta}_{t,m}) = \prod_{\boldsymbol{x},\mathbf{y} \in D_t} p_m(\mathbf{y}|\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta}_m)$. Recall that using the KLD upper bound, the PredCP prior for this setting can be expressed as

$$p(\tau_m) = \pi \left(\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\theta}_{t,m}|\tau_m} \mathrm{KL}\left[p_m(\mathcal{D}_t|\boldsymbol{\theta}_{t,m}) \;||\; p_0(\mathcal{D}_t) \right] \right) \left| \frac{1}{T} \sum_t \frac{\partial\, \mathbb{E}_{\boldsymbol{\theta}_t|\tau} \mathrm{KL}_t}{\partial\, \tau} \right|.$$

We can approximate the intractable term inside $\pi(\kappa)$ without bias via Monte-Carlo sampling:

$$\mathbb{E}_{\boldsymbol{\theta}_{t,m}|\tau_m} \mathrm{KL}\left[p_m(\mathcal{D}_t|\boldsymbol{\theta}_{t,m}) \;||\; p_0(\mathcal{D}_t) \right] \approx \frac{1}{L} \sum_{l=1}^{L} \mathrm{KL}\left[p_m(\mathcal{D}_t|\boldsymbol{\theta}_{t,m}^{(l)}) \;||\; p_0(\mathcal{D}_t) \right],$$

where $\boldsymbol{\theta}_{t,m} \sim \mathrm{N}(\boldsymbol{\phi}_m, \tau_m \mathbb{I})$. Note that the KLD term itself is a sum over the KLD terms for each $(\boldsymbol{x}, \mathbf{y})$ in the support set. In turn, each of these terms is a KLD between categorical likelihoods, which is easily computed. As detailed in the main text, we divide the sum by the number of points in the set to ease the reasoning about the parameters of π_{KL}. We can further achieve an unbiased estimator to the term inside π_{KL} using stochastic mini-batches of tasks, as is standard in the few-shot classification literature.

Finally, assuming a factorization of the τ_m's, we can simply evaluate the log-density for each prior, and compute their sum. In practice, we use single-sample estimators of the KL term, and compute the prior terms over batches of tasks to reduce the number of forward passes through the network required to compute the objective. The procedure for evaluating the PredCP prior for a batch of B tasks is detailed in Algorithm 2. Here we use the notation $\mathrm{CNN}(\boldsymbol{x}; \cdot)$ to denote a forward pass through a convolutional neural network using a set of parameters, applied to an input. We treat the output of such a call as the logits of a categorical distribution.

5 LOGISTIC REGRESSION EXPERIMENTAL DETAILS

For the logistic regression experiment in Section 6 (Table 1), we used the probabilistic programming language `Stan` (Carpenter et al., 2017) for the implementation of both Markov chain Monte Carlo (MCMC) and variational inference (VI) (Kucukelbir et al., 2017). In both cases, we performed inference for the full posterior $p(\boldsymbol{\beta}, \boldsymbol{\lambda}, \tau | \boldsymbol{X}, \boldsymbol{y})$. Following Piironen and Vehtari (2017)'s implementation, we used a non-centered parameterization: $\beta_d = \lambda_d \cdot \tau \cdot \xi_d$, $\xi_d \sim \mathrm{N}(0, 1)$. For MCMC, we left `Stan` at its default settings. We evaluated the predictive log-likelihood using 4000 posterior samples (the default output). For VI, `Stan` uses a mean-field Normal posterior, appropriately transforming

Algorithm 2 Single sample evaluation of π_τ for modular meta-learning

Input: Global parameters $\{\boldsymbol{\phi}_m\}_{m=1}^M$, network architecture CNN
Input: Prior $\pi(\kappa)$, current values $\{\tau_m\}_{m=1}^M$
Input: Inputs from all tasks in batch $\{\{\boldsymbol{x}_{n,b}\}_{n=1}^N : b = 1, \ldots, B\}$

Compute $p_0^{n,b} \leftarrow \mathrm{CNN}(\boldsymbol{x}_{n,b}; \boldsymbol{\phi})$
For $m = [1, M]$:
  Sample $\boldsymbol{\theta}_m \sim \mathrm{N}(\boldsymbol{\phi}_m, \tau_m \mathbb{I})$
  Compute $p_m^{n,b} \leftarrow \mathrm{CNN}(\boldsymbol{x}_{n,b}; \boldsymbol{\phi}, \boldsymbol{\theta}_m)$
  Compute $\mathrm{KL}_m \leftarrow \frac{1}{NB} \sum_{n,b} \mathrm{KLD}(p_m^{n,b}, p_0^{n,b})$
  Compute $\log \pi_m \leftarrow \log \pi_{\mathrm{KL}}(\mathrm{KL}_m) + \log \left| \frac{\partial \mathrm{KL}_m}{\partial \tau_m} \right|$
Return $\sum_{m=1}^M \log \pi_m$

all variables so that their support is $\mathbb{R}$. Again we left all hyper-parameters at their defaults. The predictive log-likelihood was evaluated with 1000 samples drawn from the approximate posterior. For the PredCP, we used 10 samples to evaluate the Monte Carlo expectation over $\boldsymbol{\theta}|\boldsymbol{\tau}$ for `colon` and `breast`. For `allaml`, we used only one Monte Carlo sample due to the data set being larger and requiring more time to run the MCMC. The `allaml` and `colon` data sets were downloaded from `http://featureselection.asu.edu/datasets.php`. `breast` was downloaded from `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra`. We standardized the features using the z-transform $(x - \hat{\mu})/\hat{\sigma}$, which we found to improve the speed at which the MCMC converged. We made 20 80% - 20% train-test splits for all three data sets. We left out 10% of the training set for each as a validation set that we ultimately did not use.

6 RESNET EXPERIMENTAL DETAILS

For the resnet regression experiment in Section 6 (Table 2), we followed the experimental framework of Nalisnick et al. (2019)[1] (which followed Gal and Ghahramani (2016) and Hernández-Lobato and Adams (2015)). All networks had two hidden layers, ReLU activations, and no batch or layer normalization. The posterior was approximated as:

$$p(\{\mathbf{W}_l, \boldsymbol{\lambda}_l, \tau_l\}_{l=1}^3 | \boldsymbol{y}, \boldsymbol{X}) \approx \prod_{l=1}^3 q(\mathbf{W}_l)\ q(\boldsymbol{\lambda}_l)\ q(\tau_l) = \prod_{l=1}^3 \mathrm{N}(\mathbf{W}_l; \boldsymbol{\mu}_l, \mathrm{diag}\{\boldsymbol{\Sigma}_l\})\ \delta[\boldsymbol{\lambda}_l]\ \delta[\tau_l]. \tag{10}$$

The weight approximation (Bayes-by-backprop (Blundell et al., 2015), fully factorized Gaussian) was optimized using Adam (Kingma and Ba, 2014) with a learning rate of 1×10^{-3} (other parameters left at Tensorflow defaults), using mini-batches of size 32, and run for 4500 epochs. The Monte Carlo expectations in the ELBO and PredCP both used 10 samples and *flipout* (Wen et al., 2018) for decorrelation. The ARD scales $\boldsymbol{\lambda}$ could be updated in closed-form for all models. The PredCP does not allow for a closed-form τ update (for ADD) and so we used an iterative maximization step. We used only one step per update so that the PredCP's training time was comparable to the other models'. The UCI data sets were standardized and divided into 20 90% - 10% train-test splits, following Hernández-Lobato and Adams (2015). The test set RMSE was calculated using 500 samples from the $\mathrm{N}(\mathbf{W}_l)$ posterior. For the fixed scale model, we selected the better performing of $\tau_0 = \{.1, 1\}$. For the PredCP, we used the log-Cauchy$(0, 1)$ KLD prior, finding it worked well in the logistic regression experiment and generated sensible sample functions. In Algorithm 3, we provide pseudo-code for evaluating the depth-wise (log) PredCP.

7 FEW-SHOT CLASSIFICATION: EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

We provide experimental details and results for our few-shot classification experiments.

[1] `https://github.com/enalisnick/dropout_icml2019`

Algorithm 3 Evaluating the Depth-Wise Log-PredCP

Input: Scales $\tau_1, \ldots, \tau_L$, prior $\pi(\kappa)$, number of Monte Carlo samples S, feature matrix $\boldsymbol{X}$
Initialize $\pi \leftarrow 0$
Initialize $p_0 \leftarrow p(\mathbf{y}|\boldsymbol{X}, \mathbf{W}_{\text{in}}, \mathbf{W}_{\text{out}})$
For $l = [1, L]$:
  Sample $\hat{\boldsymbol{W}}_{l,1}, \ldots, \hat{\boldsymbol{W}}_{l,S} \sim p(\mathbf{W}_l|\tau_l)$
  For $s = [1, S]$:
    Compute $p_{+,s} = p(\mathbf{y}|\boldsymbol{X}, \mathbf{W}_{\text{in}}, \{\hat{\boldsymbol{W}}_{j,s}\}_{j=1}^{l}, \mathbf{W}_{\text{out}})$
  Compute $\hat{\kappa}_l = \frac{1}{S}\sum_{s=1}^{S} \text{KL}[p_{+,s}||p_{0,s}]$
  Update $\pi \leftarrow \pi + \log \pi(\hat{\kappa}_l) + \log|\partial\hat{\kappa}_l/\partial\tau_l|$
  Update $\{p_{0,1}, \ldots, p_{0,S}\} \leftarrow \{p_{+,1}, \ldots, p_{+,S}\}$
Return π (*log-PredCP*)

7.1 Data Details

We use two standard few-shot classification benchmarks for our experiments: mini-ImageNet (Vinyals et al., 2016) and few-shot CIFAR100 (FC; (Oreshkin et al., 2018)). For mini-ImageNet, images are first down-sampled to 84x84, and then normalized. For FC, the images are classified in their 32x32 format after normalization. We use the standard split as suggested by Vinyals et al. (2016) for mini-ImageNet, containing 64 training classes, 16 validation classes, and 20 test classes. For FC, we follow the protocol proposed by Oreshkin et al. (2018), using the CIFAR100 super-classes to split the data. See Oreshkin et al. (2018) supplementary for full details. An $N-way$, $K-shot$ task is randomly sampled according to the following procedure:

- Sample N classes from the appropriate set uniformly at random.
- For each class, sample K examples uniformly at random for the context / support set.
- For each class, sample 15 examples uniformly at random for the target / query set.

Evaluation is conducted by randomly sampling 600 tasks from the test set. Average accuracy and standard errors are reported.

7.2 Network Architectures

For all tasks, we use the standard convolutional architecture proposed by Finn et al. (2017). Each network is comprised of four convolutional blocks, followed by a linear classier. For mini-ImageNet, we use a standard 3x3 convolution with 32 channels, followed by a max-pool (stride 2), a ReLU non-linearity, and a batch normalization layer. We flatten the output of the final layer, leading to an 800d representation, which is then passed through the linear classifier.

For FC, we employ the same architecture, but with 64 channels, and no max-pooling. A global-average pooling is applied to the output of the final convolutional layer, resulting in a 64d representation, which is then passed through the linear classifier.

We used the standard hyper-parameters proposed by Finn et al. (2017), without any tuning. In particular, we use 5 gradient steps for the inner loop during training, and 10 at test time. The inner learning rate is fixed to 0.01, and the meta-learning rate is 1e-3. We use a meta-batch size of 4, and train all models for 60,000 iterations.

7.3 Hyper-Priors and Additional Results

For the modular shrinkage model, we experiment with four hyper-priors: Half-Cauchy, log-Cauchy, a mixture of Gamma and Exponential (GEM), and standard Exponential. Each of these is experimented with as a standard shrinkage prior, as well as the base prior for the PredCP. The parameters of the hyper-priors are fixed throughout experimentation, and are given as follows:

- Half-Cauchy: $p(\sigma), \pi_{\text{KL}} = \text{half-Cauchy}(1.0)$

Table 1: Complete results for few-shot classification with few-shot CIFAR100. Considering four priors for σMAML and four base priors for PredCP.

Beta	Half-Cauchy		Log-Cauchy		Gem		Exponential	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
PredCP								
1e-0	33.2 ± 1.8	51.7 ± 0.9	37.7 ± 1.7	51.8 ± 0.9	37.9 ± 1.9	52.0 ± 0.97	38.9 ± 1.8	51.4 ± 1.0
1e-1	39.6 ± 1.6	51.4 ± 0.9	37.9 ± 1.9	50.6 ± 0.7	40.9 ± 1.8	51.7 ± 0.9	39.5 ± 1.9	51.1 ± 0.8
1e-2	38.2 ± 1.8	50.9 ± 0.8	40.1 ± 1.9	52.5 ± 0.9	37.8 ± 1.8	51.6 ± 0.8	40.3 ± 1.9	49.1 ± 1.1
1e-3	39.7 ± 1.9	**52.9 ± 0.9**	40.2 ± 1.8	50.6 ± 1.0	38.8 ± 1.8	51.3 ± 0.9	38.5 ± 1.8	50.8 ± 0.9
1e-4	37.4 ± 1.7	52.7 ± 0.9	37.7 ± 1.8	50.5 ± 0.9	35.8 ± 1.7	51.5 ± 0.9	**41.2 ± 1.8**	50.9 ± 0.9
Shrinkage Priors								
1e-0	36.9 ± 1.8	51.4 ± 0.9	39.5 ± 1.9	52.2 ± 1.1	39.8 ± 1.8	52.4 ± 0.9	39.4 ± 1.8	52.6 ± 0.7
1e-1	39.8 ± 1.9	**52.7 ± 0.8**	**40.9 ± 1.9**	**52.7 ± 0.9**	36.6 ± 1.8	52.5 ± 0.8	38.7 ± 1.9	51.6 ± 1.1
1e-2	38.2 ± 1.8	52.0 ± 0.9	38.8 ± 1.7	51.2 ± 0.8	36.4 ± 1.8	51.4 ± 0.9	39.1 ± 1.9	52.2 ± 0.9
1e-3	39.3 ± 1.7	51.3 ± 0.7	35.9 ± 1.8	50.8 ± 0.9	40.8 ± 1.9	51.9 ± 1.0	37.5 ± 1.9	52.2 ± 0.8
1e-4	38.6 ± 1.7	51.5 ± 0.9	40.9 ± 1.8	50.1 ± 0.9	38.8 ± 1.8	51.8 ± 0.9	40.1 ± 1.8	51.4 ± 0.9
MAML (flat θ_t prior)	35.6 ± 1.8	50.3 ± 0.9						
σMAML (flat τ prior)	39.3 ± 1.8	51.0 ± 1.0						

Table 2: Complete results for few-shot classification with mini-ImageNet. Considering four priors for σMAML and four base priors for PredCP.

Beta	Half-Cauchy		log-Cauchy		GEM		Exponential	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
PredCP								
1e-0	33.4 ± 1.7	60.7 ± 0.8	28.6 ± 1.6	59.8 ± 0.8	28.7 ± 1.6	59.1 ± 0.8	30.2 ± 1.7	59.1 ± 0.8
1e-1	47.9 ± 1.7	60.4 ± 0.9	**49.3** ± 1.7	60.4 ± 0.7	47.5 ± 1.9	61.4 ± 0.8	47.7 ± 1.7	60.6 ± 0.9
1e-2	47.0 ± 1.8	60.3 ± 0.7	45.9 ± 1.6	60.7 ± 0.8	46.7 ± 1.8	61.7 ± 0.7	47.2 ± 1.9	**61.9** ± 0.9
1e-3	46.4 ± 1.7	61.3 ± 0.8	48.1 ± 1.8	61.2 ± 0.9	47.9 ± 1.7	60.5 ± 0.8	46.9 ± 1.7	60.5 ± 0.8
1e-4	47.7 ± 1.8	60.4 ± 0.9	47.7 ± 1.8	60.3 ± 0.9	48.1 ± 1.9	60.1 ± 0.9	48.3 ± 1.8	60.4 ± 0.9
Shrinkage Priors								
1e-0	42.9 ± 1.8	57.3 ± 0.9	**48.5** ± 1.7	**60.9** ± 0.9	24.7 ± 1.5	29.0 ± 0.7	44.5 ± 1.8	57.8 ± 0.9
1e-1	46.5 ± 1.8	59.2 ± 0.9	47.2 ± 1.8	59.7 ± 0.9	45.8 ± 1.8	58.7 ± 0.8	47.0 ± 1.8	59.6 ± 0.9
1e-2	46.8 ± 1.7	59.3 ± 0.8	46.3 ± 1.9	59.3 ± 0.8	46.7 ± 1.7	60.1 ± 0.7	46.8 ± 1.8	59.1 ± 0.9
1e-3	47.3 ± 1.9	59.2 ± 0.9	47.7 ± 1.7	59.3 ± 0.9	47.2 ± 1.9	60.1 ± 0.9	47.5 ± 1.9	59.5 ± 1.0
1e-4	46.7 ± 1.8	59.2 ± 0.9	47.4 ± 1.7	60.2 ± 0.7	48.0 ± 1.5	59.5 ± 0.9	47.8 ± 1.7	58.6 ± 0.9
MAML (flat θ_t prior)	45.6 ± 1.8	58.4 ± 0.9						
σMAML (flat τ prior)	47.4 ± 1.8	60.1 ± 0.9						

- Log-Cauchy: $p(\sigma), \pi_{\text{KL}} = \text{log-Cauchy}(2.0)$
- GEM: $p(\sigma), \pi_{\text{KL}} = 0.5\,(\Gamma(0.2, 2.0) + \text{Exp}(0.5))$
- Exponetial: $p(\sigma), \pi_{\text{KL}} = \text{Exp}(0.5)$

Additionally, we experimented with a vague (flat) prior, which recovers the model proposed by Chen et al. (2019). In addition, we also add a constant weight β, which multiplies the prior on σ in the outer-loop objective for both the regular shrinkage and PredCP. For each prior, we experiment with $\beta \in \{1e-0, 1e-1, 1e-2, 1e-3, 1e-4\}$.

Tables 1 and 2 provide our complete results for these experiments. For both datasets, and for both 1- and 5- shot, we observe that a PredCP prior with appropriate weighting on the prior term provides the best performance in terms of accuracy on the test set.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

George E. P. Box. Sampling and Bayes' Inference in Scientific Modelling and Robustness. *Journal of the Royal Statistical Society: Series A*, 1980.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 2017.

Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. Handling Sparsity via the Horseshoe. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, 2009.

Yutian Chen, Abram L. Friesen, Feryal Behbahani, David Budden, Matthew W. Hoffman, Arnaud Doucet, and Nando de Freitas. Modular Meta-Learning with Shrinkage. *NeurIPS Workshop on Meta-Learning*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Valen E. Johnson and David Rossell. On the Use of Non-Local Prior Densities in Bayesian Hypothesis Tests. *Journal of the Royal Statistical Society: Series B*, 2010.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, 2014.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic Differentiation Variational Inference. *The Journal of Machine Learning Research*, 2017.

Fangchang Ma, Ulas Ayaz, and Sertac Karaman. Invertibility of Convolutional Generative Networks from Partial Measurements. In *Advances in Neural Information Processing Systems*, 2018.

Eric Nalisnick, José Miguel Hernández-Lobato, and Padhraic Smyth. Dropout as a Structured Shrinkage Prior. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: Task Dependent Adaptive Metric for Improved Few-Shot Learning. In *Advances in Neural Information Processing Systems*, 2018.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *ArXiv e-Prints*, 2019.

Mary Phuong and Christoph H. Lampert. Functional vs. Parametric Equivalence of ReLU Networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Juho Piironen and Aki Vehtari. On the Hyperprior Choice for the Global Shrinkage Parameter in the Horseshoe Prior. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Minsuk Shin, Anirban Bhattacharya, and Valen E. Johnson. Scalable Bayesian Variable Selection Using Nonlocal Prior Densities in Ultrahigh-Dimensional Settings. *Statistica Sinica*, 2018.

Yang Song, Chenlin Meng, and Stefano Ermon. MintNet: Building Invertible Neural Networks with Masked Convolutions. In *Advances in Neural Information Processing Systems*, 2019.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, and Daan Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems*, 2016.

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. In *Proceedings of the International Conference on Learning Representations*, 2018.