2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Hidden Priors for Bayesian Bidirectional Backpropagation

Olaoluwa Adigun[1] and Bart Kosko[2]

***Abstract*— Non-uniform prior probabilities between hidden layers improved deep neural classifiers trained with bidirectional backpropagation. The resulting Bayesian bidirectional backpropagation algorithm jointly maximizes the forward and backward network likelihoods along with the weight priors. The backward direction exploits a hidden regression that ordinary unidirectional backpropagation ignores. Simulations compared Laplacian, Gaussian, Cauchy, and the new *sinc-squared* hidden priors on the CIFAR-10 and CIFAR-100 balanced image data sets. These hidden priors improved the classification accuracy of deep neural classifiers compared with default uniform priors and default unidirectional backpropagation. They did so at little extra computational cost. Sinc-squared and Cauchy multivariate priors often had the best classification accuracy. Cauchy hidden priors gave sparse hidden weights similar to the Laplacian priors associated with sparse lasso regression.**

I. Hidden Priors in Bidirectional Backpropagation

This paper extends the recent likelihood-based bidirectional backpropagation [1] to the deep Bayesian case in Figure 1. It puts prior probabilities on all the synaptic webs that connect the hidden layers in the back-and-forth flow of neural signals.

We also introduce a new multivariate prior probability density called the *sinc-squared* probability density function (pdf). We further study what appears to be the first use of a multivariate Cauchy prior [2] in this deep Bayesian context.

The new sinc-squared pdf f has a median m and a *dispersion* $d > 0$. It lacks a population mean and a finite variance and higher moments as does the Cauchy pdf below. Figure 2d plots the simplest sinc-squared pdf f with zero median $m = 0$ and unit dispersion $d = 1$ for random variable $X \sim f$:

$$f_X(x) = \frac{1}{\pi}\,\mathrm{sinc}^2(x) = \frac{\sin^2(x)}{\pi x^2} \tag{1}$$

for all nonzero real x. The pdf structure follows from the nonnegativity of $\mathrm{sinc}^2(x)$ and the extraordinary fact that

$$\pi = \int_{-\infty}^{\infty} \mathrm{sinc}(x)\; dx = \int_{-\infty}^{\infty} \mathrm{sinc}^2(x)\; dx \tag{2}$$

which follows in turn from integration by parts. The characteristic function $\phi_X(\omega) = \mathbb{E}_X\left[e^{i\omega X}\right]$ is triangular:

$$\phi_X(\omega) = 1 - \frac{|\omega|}{2} \tag{3}$$

if $|\omega| < 2$ and $\phi_X(\omega) = 0$ else. Pdf products give a multivariate sinc-squared pdf $f(x_1, \ldots, x_n) = f_{X_1}(x_1) \cdots f_{X_n}(x_n)$ for independent sinc-squared random variables $X_1, \ldots, X_n$.

Figure 2 shows scalar versions of all priors used. It also shows the contour plots for the log-priors of 2-dimensional random vectors. Sinc-squared and Cauchy priors outperformed the traditional multivariate Gaussian and Laplacian "regularizers" found in respective ridge [3], [4] and lasso regression [5], [6] on the CIFAR-10 and CIFAR-100 image test sets. Figure 2 shows that smaller dispersion values γ in the Cauchy priors produced *soft diamonds* akin to the *hard diamond* for the Laplace prior. This helps explain why the Cauchy priors produced sparse or reduced weights as in lasso regression. Figure 3 shows a 2-D sinc-squared prior.

Ordinary backpropagation (BP) iteratively maximizes the forward likelihood $p_f(y|x, \Theta)$ that maps an input pattern vector x forward through several hidden layers of ReLU or nonvanishing NoVa [7] or other neural units to a target or output vector y. It ignores the backward probability $p_b(x|y, \Theta)$ that describes when a target or output vector y maps back to the input pattern x at the input layer of identity neurons.

The new bidirectional BP algorithm [1] maximizes the network's *joint* likelihood $p_f(y|x, \Theta)p_b(x|y, \Theta)$ of a single forward *and* backward learning epoch. The next section shows that *Bayesian* bidirectional BP (B^3) [8], [9] maximizes the total joint forward-and-backward posterior probability $p_f(y|x, \Theta)h_f(\Theta|x)p_b(x|y, \Theta)h_b(\Theta|y)$.

Figure 1 shows that the identity neurons at the classifier's input layer give rise to a *hidden* backward regression or ordinary function approximation. Feedforward classifiers ignore this identity structure of the input neurons and just treat them as data registers for input patterns. The forward pass defines a one-shot multinomial probability or the roll of a K-sided die because the output neurons are softmax. But the backward pass defines a multivariate normal probability because the input neurons are identity neurons. Taking negative logarithms shows that bidirectional BP maximizes the joint sum of a forward cross-entropy and a backward squared-error. This bidirectional optimization uses the same training data that unidirectional BP uses. The extra computational cost is trivial.

Table I shows the classification benefit of using these non-uniform priors at the hidden layers of the deep neural classifiers that trained on the CIFAR-10 image dataset. Table II shows the same comparative benefit with neural classifiers that trained on the CIFAR-100 image data set with non-uniform hidden priors. Figures 4 and 6 show samples from CIFAR-10 and CIFAR-100. Figures 5 and 7 compare the

[1]Olaoluwa Adigun is a lecturer at the Department of Electrical and Computer Engineering at the University of Southern California, Los Angeles, CA 90089, USA adigun@usc.edu

[2]Bart Kosko is a professor of Electrical and Computer Engineering, and Law, at the University of Southern California, Los Angeles, CA 90089, USA kosko@usc.edu

trained weights after B^3 training and shows similar sparsity for Cauchy and Laplace priors.

Tables I-III show that the Cauchy and sinc-priors performed best in most cases. It also shows the benefit of using both vector Gaussian hidden priors and Laplacian hidden priors to train neural classifiers with B^3 on CIFAR-10.

All four types of hidden priors did better than the default uniform priors. The results also did best when the prior also applied to the synaptic web from the input identity neurons to the first hidden layer. Table III shows the comparative benefit of classifiers after B^3 on the CIFAR-100 image set. Cauchy and Laplace priors produced the most sparse trained synaptic weights but Cauchy priors had better classification accuracy.

II. Bayesian Bidirectional Backpropagation

Bayesian bidirectional backpropagation (B^3) [8], [9] jointly maximizes the neural network's forward posterior $p_f(\Theta|y,x)$ and its backward posterior $p_b(\Theta|x,y)$. Bayes' theorem gives the forward posterior as

$$p_f(\Theta|y,x) = \frac{p_f(y|x,\Theta)h_f(\Theta|x)}{\int p_f(y|x,\Theta)h_f(\Theta|x)\ d\Theta} \tag{4}$$

and $p_f(\Theta|y,x) \propto p_f(y|x,\Theta)h_f(\Theta|x)$ where $p_f(y|x,\Theta)$ is the forward likelihood and $h_f(\Theta|x)$ is the forward prior. The backward posterior $p_b(\Theta|x,y)$ is

$$p_b(\Theta|x,y) = \frac{p_b(x|y,\Theta)h_b(\Theta|y)}{\int p_b(x|y,\Theta)h_b(\Theta|y)\ d\Theta} \tag{5}$$

and $p_b(\Theta|x,y) \propto p_b(x|y,\Theta)h_b(\Theta|y)$ where $p_b(x|y,\Theta)$ is the backward likelihood and $h_b(\Theta|y)$ is the backward prior. Then the B^3 solution Θ^{BMAP} is

$$\Theta^{BMAP} = \arg\max_{\Theta}\ p_f(\Theta|y,x)p_b(\Theta|x,y) \tag{6}$$

$$= \arg\max_{\Theta}\ p_f(y|x,\Theta)h_f(\Theta|x)p_b(x|y,\Theta)h_b(\Theta|y) \tag{7}$$

$$= \arg\max_{\Theta}\ \log p_f(y|x,\Theta) + \log p_b(x|y,\Theta) + \log h(\Theta) \tag{8}$$

with joint bidirectional prior $h(\Theta) = h_f(\Theta|x)h_b(\Theta|y)$.

Bidirectional BP and its Bayesian extension B^3 endows a neural network N with bidirectional mappings between the input space $\mathcal{X}$ and output space $\mathcal{Y}$. Neural signals pass forward through the synaptic weight matrices W and then backward through the transposes W^T of the same weight matrices [10]. The forward pass propagates input vector $\mathbf{x}$ from the input layer to the output layer. The output activation $\mathbf{a}^t$ for the forward pass has the form $\mathbf{a}^t = N(\mathbf{x})$ where the output activation vector $\mathbf{a}^t$ predicts the target $\mathbf{t}$ given input $\mathbf{x}$. The backward pass propagates $\mathbf{t}$ from the output layer back through to the input layer. The input activation $\mathbf{a}^x$ for the backward pass has the form $\mathbf{a}^x = N^T(\mathbf{t})$.

The forward pass of a neural classifier with softmax output activation uses a multinomial or categorical likelihood $p_f(\mathbf{t}|\mathbf{x},\Theta) = \prod_{k=1}^{K}(a_k^t)^{t_k}$ for output target vector $\mathbf{t}$ where a_k^t is the activation of the k^{th} output neuron and t_k is the target of the k^{th} output neuron. The corresponding log-likelihood is

$$\log p_f(\mathbf{t}|\mathbf{x},\Theta) = \sum_{k=1}^{K} t_k\ \log a_k^t. \tag{9}$$

The term a_k^t gives the probability that $\mathbf{x}$ belongs to class k with $0 \leq a_k^t \leq 1$ and $\sum_{k=1}^{K} a_k^t = 1$. The corresponding forward error function $E_f(\Theta)$ is just the cross-entropy between t_k and a_k^t:

$$E_f(\Theta) = -\log\ p_f(\mathbf{t}|\mathbf{x},\Theta) = -\sum_{k=1}^{K} t_k \log\ a_k^t. \tag{10}$$

The backward pass of the neural classifier uses identity input activations. So the input layer has a multivariate-normal backward likelihood $p_b(\mathbf{x}|\mathbf{t},\Theta) = \frac{1}{2\pi^{\frac{L}{2}}} \exp^{-\frac{||\mathbf{a}^x - \mathbf{x}||_2^2}{2}}$:

$$\log\ p_b(\mathbf{x}|\mathbf{t},\Theta) = -\frac{L}{2}\log 2\pi - \frac{1}{2}||\mathbf{a}^x - \mathbf{x}||_2^2 \tag{11}$$

because $p_b(\mathbf{x}|\mathbf{t},\Theta) = \mathcal{N}(\mathbf{x}|\mathbf{a}^x, \mathbf{I})$. The corresponding backward error $E_b(\Theta)$ follows from the negative log-likelihood:

$$E_b(\Theta) = -\log\ p_b(\mathbf{x}|\mathbf{t},\Theta) - \log(2\pi)^{\frac{L}{2}} = \frac{1}{2}||\mathbf{a}^x - \mathbf{x}||_2^2 \tag{12}$$

where $u = -\frac{L}{2}\log 2\pi$ does not depend on parameter vector Θ. The B^3 solution parameter Θ^* for a neural classifier is

$$\Theta^* = \arg\max_{\Theta}\ p_f(\mathbf{t}|\mathbf{x},\Theta)\ p_b(\mathbf{x}|\mathbf{t},\Theta)\ h(\Theta) \tag{13}$$

$$= \arg\max_{\Theta}\ \log p_f(\mathbf{t}|\mathbf{x},\Theta) + \log p_b(\mathbf{x}|\mathbf{t},\Theta) + \log h(\Theta) \tag{14}$$

$$= \arg\min_{\Theta}\ E_f(\Theta) + E_b(\Theta) - \log h(\Theta) \tag{15}$$

where $h(\Theta)$ depends on the choice of prior probability.

Bayesian B-BP generalizes to the Bayesian unidirectional BP with zero backward error. So $E_b(\Theta) = 0$. Therefore the Bayesian unidirectional BP training simplifies as follows:

$$\hat{\Theta} = \arg\max_{\Theta}\ p_f(\mathbf{t}|\mathbf{x},\Theta)\ h(\Theta) \tag{16}$$

$$= \arg\min_{\Theta}\ E_f(\Theta) - \log h(\Theta) \tag{17}$$

where $\hat{\Theta}$ is the Bayesian unidirectional BP solution.

III. Prior Layer Structure

Let N be a neural network with J hidden layers. The prior $h(\Theta)$ factors into the product of input prior, hidden priors, and output prior. The input prior $h_x(\Theta|\mathbf{h}_1,\mathbf{x})$ controls the weights that connect the input layer to the first hidden layer $\mathbf{h}_1$. The hidden prior $h(\Theta|\mathbf{h}_{j+1},\mathbf{h}_j)$ controls the weights between any two contiguous hidden layers where $j \in \{1,2,...,J-1\}$. The output prior $h_y(\Theta|\mathbf{y},\mathbf{h}_J)$ controls the synaptic weights between the hidden layer $\mathbf{h}_J$ and the output layer.

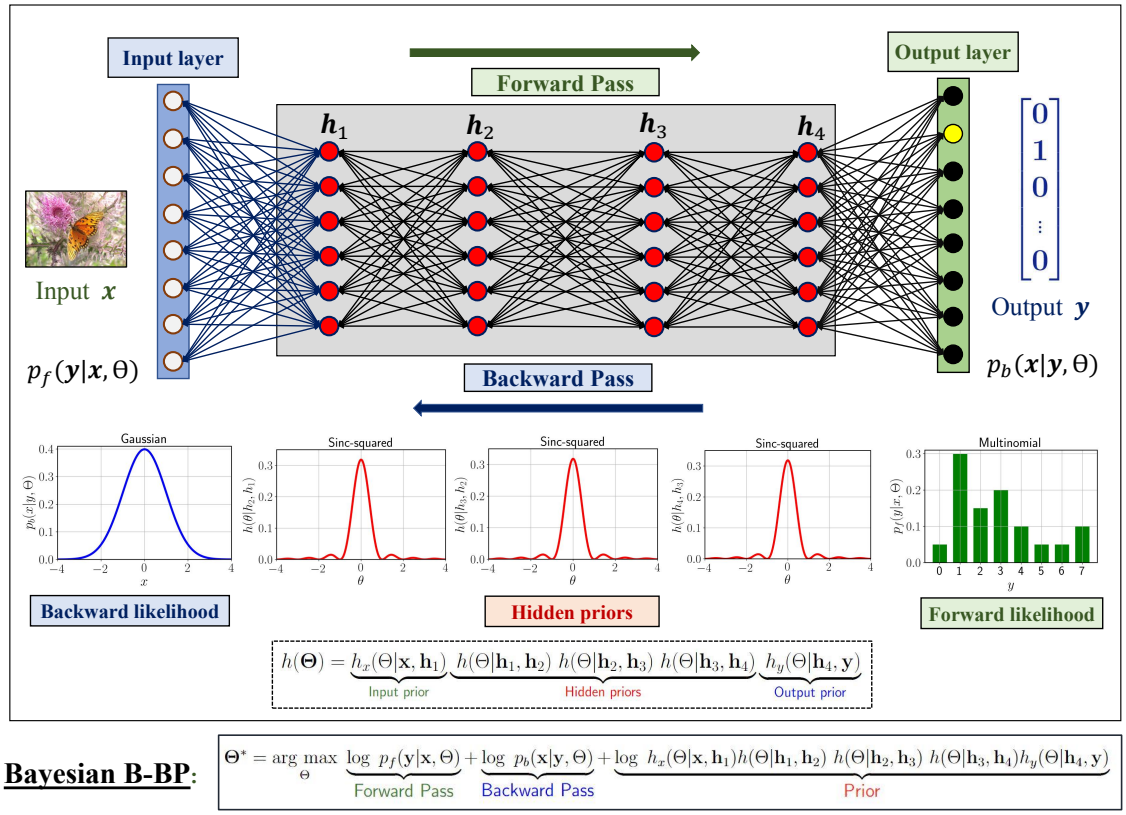


Fig. 1: Bayesian Bidirectional Backpropagation learning: B^3 maximizes the network's *joint* log-posterior or the log-likelihood of the forward network likelihood $p_f(\mathbf{y}|\mathbf{x},\Theta)$, the backward likelihood $p_b(\mathbf{x}|\mathbf{y},\Theta)$, and the weight prior $h(\Theta)$. Here the weight prior splits into one input prior, three hidden priors, and one output prior. The input and output priors are uniform while the hidden priors are the new sinc-squared multivariate priors. This network runs as a classifier in the forward direction and as a regressor or function approximator in the backward direction. The output layer uses softmax neurons and so has a multinomial likelihood and thus a cross-entropy error on the forward pass. The input layer uses identity neurons and so has a vector normal likelihood and thus a squared error on the backward pass.

The prior $h(\Theta)$ factors over the neural layers:

$$h(\Theta) = \underbrace{h_x(\Theta|\mathbf{h}_1,\mathbf{x})}_{\text{Input prior}} \underbrace{\prod_{j=1}^{J-1} h(\Theta|\mathbf{h}_{j+1},\mathbf{h}_j)}_{\text{Hidden priors}} \underbrace{h_y(\Theta|\mathbf{y},\mathbf{h}_J)}_{\text{Output prior}}. \tag{18}$$

All experiments used a uniform output prior for the weights from the final hidden layer to the output layer of softmax neurons. So the output prior dropped out of the gradient optimization. This gave the log prior as

$$\log\ h(\Theta) = \log\left(h_x(\Theta|\mathbf{h}_1,\mathbf{x}) \prod_{j=1}^{J-1} h(\Theta|\mathbf{h}_{j+1},\mathbf{h}_j)\right) \tag{19}$$

$$= \log\ h_x(\Theta|\mathbf{h}_1,\mathbf{x}) + \sum_{j=1}^{J-1} \log\ h(\Theta|\mathbf{h}_{j+1},\mathbf{h}_j). \tag{20}$$

IV. Between-Layer Priors

The simulations used four non-uniform hidden priors: Laplacian, Gaussian, Cauchy, and the new sinc-squared prior. These vector priors factor into a product of marginals because we assume that the underlying random variables are independent and identically distributed (i.i.d.).

A. Gaussian Prior:

The Gaussian random vector $\Theta \sim N(\Theta|\mathbf{0},\sigma\mathbf{I})$ gives

$$\log h(\Theta) = \log\left(2\pi\sigma^2\right)^{-\frac{L}{2}} \exp^{-\frac{||\Theta||_2^2}{2\sigma^2}} \tag{21}$$

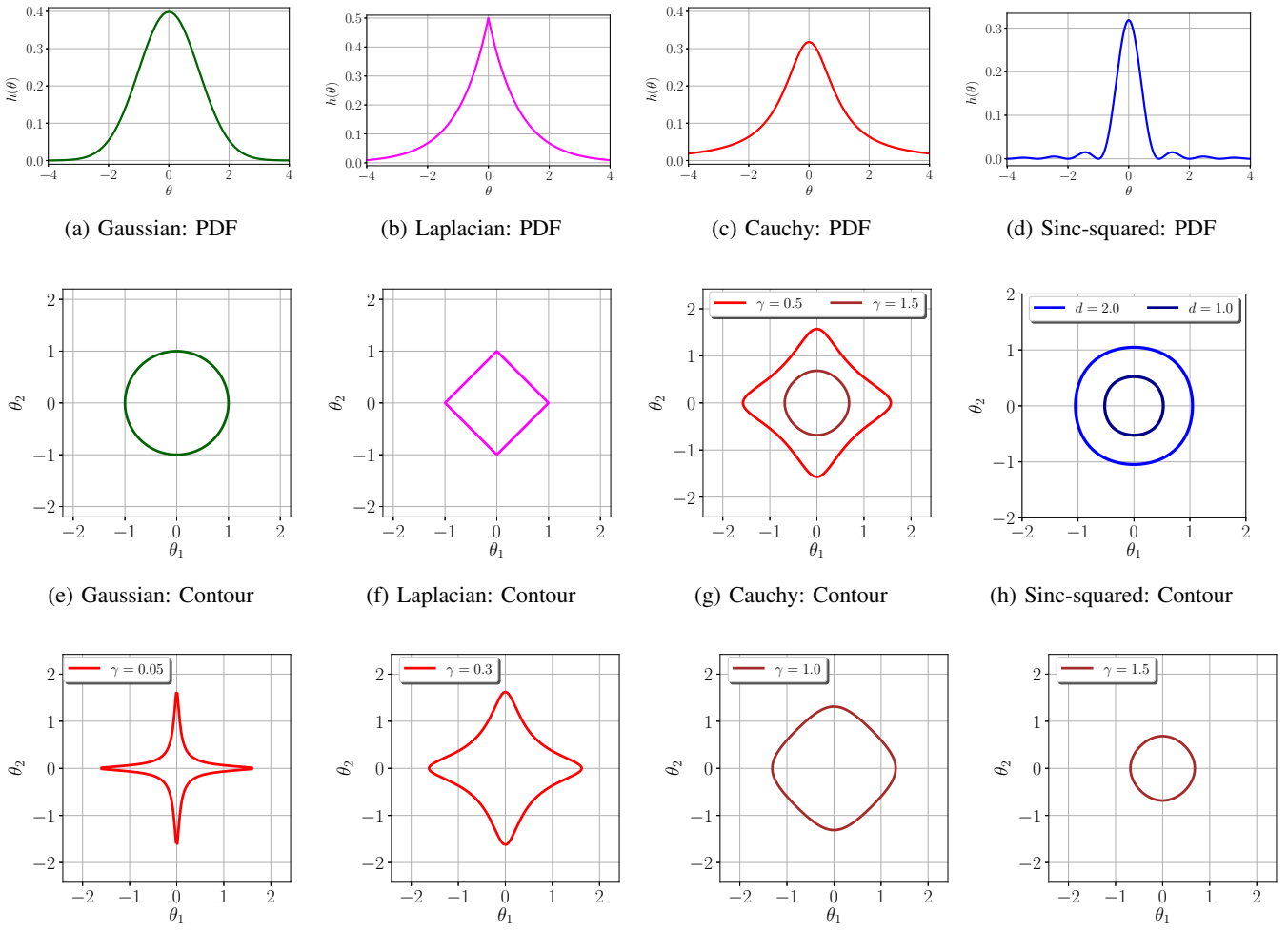
$$= -\frac{L}{2}\log\left(2\pi\sigma^2\right) - \frac{||\Theta||_2^2}{2\tau^2} \tag{22}$$

$$= -\frac{L}{2}\log\left(2\pi\sigma^2\right) - \lambda||\Theta||_2^2 \tag{23}$$

where $\Theta = [\theta_1,\theta_2,...,\theta_L]$. Figure 2a shows the density function of a scalar version of Gaussian prior. The log joint density function $h(\theta_1,\theta_2)$ of two i.i.d. Gaussian and scalar random variables with mean $\mu = 0$ and standard deviation σ is

$$\log h(\theta_1,\theta_2) = -\log 2\pi\sigma^2 - \lambda||\boldsymbol{\theta}||_2^2 - 2\sigma^2 \tag{24}$$

where $\boldsymbol{\theta} = [\theta_1,\theta_2]$ and where θ_1 and θ_2 are the scalar components of the 2-D vector random variable $\boldsymbol{\theta}$.



(a) Gaussian: PDF (b) Laplacian: PDF (c) Cauchy: PDF (d) Sinc-squared: PDF

(e) Gaussian: Contour (f) Laplacian: Contour (g) Cauchy: Contour (h) Sinc-squared: Contour

(i) Cauchy: $\gamma = 0.05$ (j) Cauchy: $\gamma = 0.3$ (k) Cauchy: $\gamma = 1$ (l) Cauchy: $\gamma = 1.5$

Fig. 2: Scalar probability density functions of the non-uniform priors and the contour plots of their log-priors. Non-uniform priors: Gaussian, Laplacian, Cauchy, and sinc-squared priors. These contours result from taking the logarithm of 2-D i.i.d. densities and dropping the constant terms. (a) density function of a Gaussian prior with $\mu = 0$ and $\sigma = 1$. The Gaussian bell curve has thin exponential tails. (b) density function of a Laplacian prior with $\mu = 0$ and $b = 1$. (c) density function of a Cauchy prior with $m = 0$ and $\gamma = 1$. The Cauchy bell curve has thicker power-law tails and so rare events are more frequent. (d) density function of a sinc-squared prior with $d = 1$. (e) contour plot for a Gaussian prior with equation $\theta_1^2 + \theta_2^2 = 1$. (f) contour plot for a Laplacian prior with equation $|\theta_1| + |\theta_2| = 1$. (g) contour plots for Cauchy priors with equation $\log(\gamma^2 + \theta_1^2) + \log(\gamma^2 + \theta_2^2) - 2\log\gamma = 1$. The case $\gamma = 0.5$ gives a "soft diamond" compared with the Laplacian's hard diamond. (h) contour plots for sinc-squared priors with equation $\log \text{sinc}^2\left(\frac{\theta_1}{d}\right) + \log \text{sinc}^2\left(\frac{\theta_2}{d}\right) = -1$. (i) Cauchy "soft diamond" solution contour for $\gamma = 0.05$ and $c = 1$. (j) Cauchy "soft diamond" solution contour for $\gamma = 0.3$ and $c = 1$. (k) Cauchy "soft diamond" solution contour for $\gamma = 1$ and $c = 1$. (l) Cauchy solution contour for $\gamma = 1.5$ and $c = 1$.

The corresponding contour equation for this log-prior with respect to θ_1 and θ_2 is

$$\lambda||\boldsymbol{\theta}||_2^2 = \lambda({\theta_1}^2 + {\theta_2}^2) = c \tag{25}$$

where $c \geq 0$. Figure 2e shows the corresponding contour plot with $c = 1$ and $\lambda = 1$.

B. *Laplacian Prior:*

The Laplacian parameter vector $\Theta \sim \text{Laplace}(\Theta|\mathbf{0}, \tau\mathbf{I})$ gives

$$\log h(\Theta) = \log\left((2\tau)^{-L} \exp^{-\frac{||\Theta||_1}{\tau}}\right) \tag{26}$$

$$= -L\log(2\tau) - \frac{||\Theta||_1}{\tau} \tag{27}$$

$$= -L\log(2\tau) - \lambda||\Theta||_1 \tag{28}$$

with $\tau > 0$ and $\Theta = [\theta_1, \theta_2, ..., \theta_L]$. The log density function $h(\theta_1, \theta_2)$ of two i.i.d. Laplacian random variables with parameter τ is

$$\log h(\theta_1, \theta_2) = -2\log(2\tau) - \lambda||\boldsymbol{\theta}||_1 \tag{29}$$

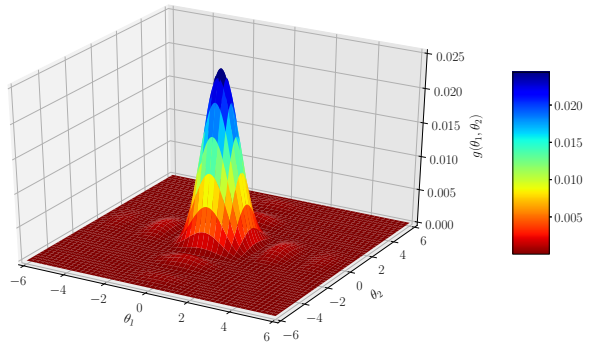


Fig. 3: Sinc-squared multivariate prior: Joint distribution of independent and identically distributed random variables θ_1 and θ_2 with the new sinc-squared probability density function for dispersion $d = 2.5$.

where $\boldsymbol{\theta} = [\theta_1, \theta_2]$, and where θ_1 and θ_2 are the scalar components of the 2-D vector random variable $\boldsymbol{\theta}$. The corresponding contour equation for this log-prior is

$$\lambda||\boldsymbol{\theta}||_1 = \lambda(|\theta_1| + |\theta_2|) = c \tag{30}$$

where $c \geq 0$. Figure 2f shows the corresponding contour plot with $c = 1$ and $\lambda = 1$.

C. Cauchy Prior:

The Cauchy random vector Θ also has a density that factors into marginals for the infinite-variance random variables $\theta_l \sim \text{Cauchy}(\Theta_l|0, \gamma)$ for $l = \{1, 2, ..., L\}$ where $\Theta = \{\Theta_l\}_{l=1}^L$. Then

$$\log\ h(\Theta) = \log \prod_{l=1}^{L} \frac{\gamma}{\pi(\theta_l^2 + \gamma^2)} = \sum_{l=1}^{L} \log \frac{\gamma}{\pi(\theta_l^2 + \gamma^2)}. \tag{31}$$

The log of Cauchy joint-prior for $L = 2$ is

$$\log h(\theta_1, \theta_2) = 2\log\gamma - 2\log\pi - \sum_{l=1}^{2} \log(\theta_l + \gamma^2) \tag{32}$$

where θ_1 and θ_2 are the respective realizations of Θ_1 and Θ_2. The corresponding contour plot for this log-prior with respect to θ_1 and θ_2 is

$$-2\log\gamma + \sum_{l=1}^{2} \log({\theta_l}^2 + \gamma^2) = c\ . \tag{33}$$

Figure 2g shows two Cauchy prior contour plots for $c = 1.0$ with $\gamma = 0.5$ and $\gamma = 1.4$.

D. Sinc-squared Prior:

The probability density function h of a scalar sinc-squared random variable θ with zero median $m = 0$ is

$$h(\theta) = \frac{1}{\pi d}\,\text{sinc}^2\left(\frac{\theta}{d}\right) = \frac{d}{\pi\theta^2}\sin^2\left(\frac{\theta}{d}\right) \tag{34}$$

for dispersion $d > 0$. The density structure follows from

$$\int_{-\infty}^{\infty} \text{sinc}^2\left(\frac{x}{d}\right)\,dx = \pi d = \int_{-\infty}^{\infty} \text{sinc}\left(\frac{x}{d}\right)\,dx \tag{35}$$

where $d > 0$. Therefore $h(\theta) \geq 0$ and $\int h(\theta)\ d\theta = 1$ for $\theta \in (-\infty, \infty)$. Figure 2d shows the density function for $d = 1$. The corresponding derivative of this prior is

$$h'(\theta) = \frac{d}{d\theta}\frac{1}{\pi d}\,\text{sinc}^2\left(\frac{\theta}{d}\right) \tag{36}$$

$$= \frac{2}{\pi d}\,\text{sinc}\left(\frac{\theta}{d}\right)\frac{d}{d\theta}\,\text{sinc}\left(\frac{\theta}{d}\right) \tag{37}$$

$$= \frac{2}{\pi d}\,\text{sinc}\left(\frac{\theta}{d}\right)\left(\frac{\theta\cos\left(\frac{\theta}{d}\right) - d\sin\left(\frac{\theta}{d}\right)}{\theta^2}\right) \tag{38}$$

$$= \frac{2\,\text{sinc}\left(\frac{\theta}{d}\right)\left(\theta\cos\left(\frac{\theta}{d}\right) - d\sin\left(\frac{\theta}{d}\right)\right)}{\pi d\theta^2}\ . \tag{39}$$

The sinc-squared random vector Θ has a joint density function that factors into the product of marginals of sinc-squared random variables $\Theta_l \sim \text{sinc-squared}(\Theta_l|m = 0, d)$ for $l \in \{1, 2, ..., L\}$ where $\Theta = \{\Theta_l\}_{l=1}^L$. Figure 3 shows the joint probability density function for a sinc-squared random vector with $L = 2$ and $d = 2.5$. The corresponding log-likelihood for the random vector Θ is

$$\log h(\Theta) = \log \prod_{l=1}^{L} \frac{1}{\pi d}\,\text{sinc}^2\left(\frac{\theta_l}{d}\right) \tag{40}$$

$$= \sum_{l=1}^{L} \log\frac{1}{d} - \log\pi + \log\left(\text{sinc}^2\left(\frac{\theta_l}{d}\right)\right) \tag{41}$$

$$= -L(\log d + \log\pi) + \sum_{l=1}^{L} \log\left(\text{sinc}^2\left(\frac{\theta_l}{d}\right)\right) \tag{42}$$

where θ_l is a realization of Θ_l for all l.

The log joint-prior for $L = 2$ is

$$\log h(\theta_1, \theta_2) = -2(\log d + \log\pi) + \sum_{l=1}^{2} \log\left(\text{sinc}^2\left(\frac{\theta_l}{d}\right)\right). \tag{43}$$

The corresponding contour equation for this log-prior with respect to θ_1 and θ_2 is

$$\log\left(\text{sinc}^2\left(\frac{\theta_1}{d}\right)\right) + \log\left(\text{sinc}^2\left(\frac{\theta_2}{d}\right)\right) = c \tag{44}$$

where $c \leq 0$. Figure 2h shows two sinc-squared contour plots for $d = 2.0$ and $d = 1$.

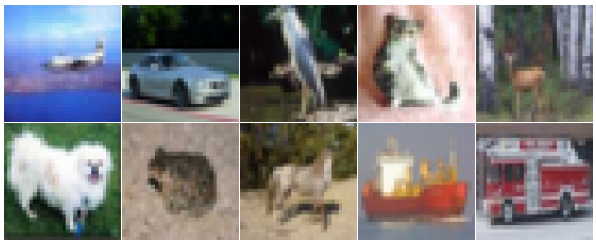
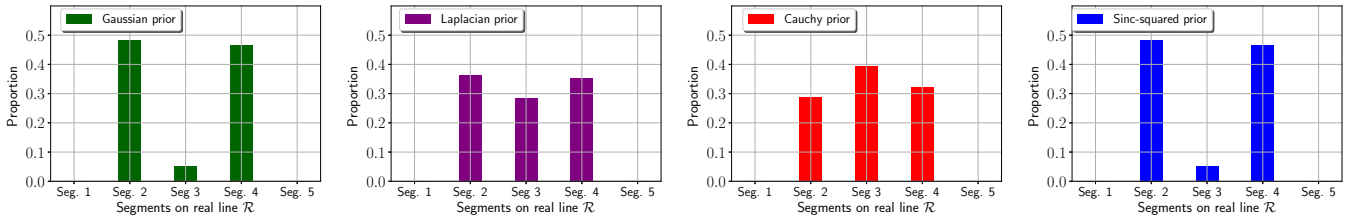


Fig. 4: CIFAR-10 images: 10 samples from the CIFAR-10 dataset that contains 10 pattern classes and a total of 60,000 sample images with 6,000 images per class.

TABLE I: Classification accuracy of Bayesian bidirectional backpropagation training with non-uniform hidden priors on the CIFAR-10 image dataset for Gaussian, Laplacian, Cauchy, and sinc-squared priors. The neural classifiers used uniform input and output priors. They also used 1,000 ReLU neurons per hidden layer.

Network structure	**Unidirectional BP**	**Bayesian Bidirectional BP** (Hidden Priors)				
		Uniform Prior	Gaussian Prior	Laplacian Prior	Cauchy Prior	Sinc-squared Prior
2 hidden layers	59.88%	60.61%	61.69%	61.70%	61.66%	**62.18%**
4 hidden layers	61.36%	61.11%	61.63%	61.52%	61.39%	**62.38%**
6 hidden layers	60.48%	61.07%	61.13%	61.14%	61.13%	**62.41%**
8 hidden layers	59.74%	60.08%	61.46%	61.25%	61.59%	**62.21%**
10 hidden layers	60.03%	60.46%	60.67%	60.10%	60.27%	**60.94%**



(a) Gaussian weights (b) Laplacian weights (c) Cauchy weights (d) Sinc-squared weights

Fig. 5: Weight sparsity with Bayesian bidirectional backpropagation training on CIFAR-10: The figures show the weight distribution of the deep-neural classifier after it trained over 100 epochs. The classifiers used 8 hidden layers with 1,000 ReLU neurons *per* hidden layer. The plots divide the real line $\mathcal{R}$ into 5 segments: Seg. 1 denotes $(-\infty, -0.2)$, Seg. 2 denotes $[-0.2, -0.002)$, Seg. 3 denotes $[-0.002, 0.002)$, Seg. 4 denotes $[0.002, 0.2)$, and Seg. 5 denotes $[0.2, \infty)$. The Cauchy prior used $\gamma = 0.3$.

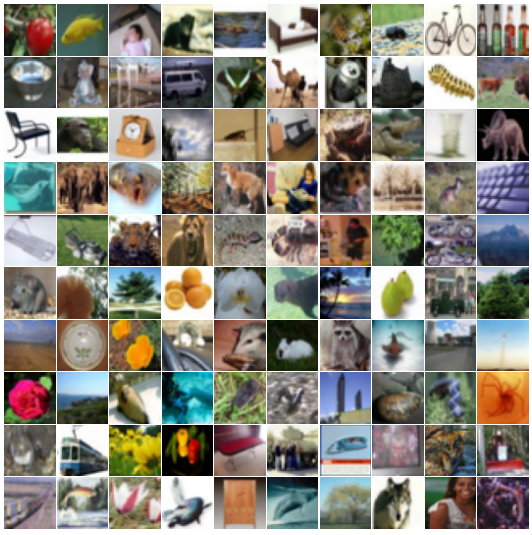


Fig. 6: CIFAR-100 images: This figure shows 100 samples from the CIFAR-100 dataset that contains 100 pattern classes with 600 images per class.

V. Neural Network Simulations

The simulated deep classifiers used two image datasets. The first was the CIFAR-10 dataset [11] and the second was the CIFAR-100 dataset [11].

The CIFAR-10 test set consists of 60,000 color images from 10 categories ($K = 10$). Each image has size $32 \times 32 \times 3$. The 10 pattern categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck [11]. Figure 4 shows sample images with one image per class.

CIFAR-100 dataset is a set of 60,000 color images with image size $32 \times 32 \times 3$. The images are from 100 pattern classes with 600 images per class. Each class is made up of 500 training images and 100 testing images. Figure 6 shows sample images with one image per class.

The deep neural classifiers trained on CIFAR-10 and CIFAR-100 with the B^3 algorithm. Each classifier network used 1000 neurons *per* hidden layer and used K softmax or Gibbs output activations. The hidden neurons used rectified linear unit (ReLU) activation. The input layer used identity neurons. We varied the size of the hidden layers and the choice of hidden prior. The competing hidden priors were uniform, Laplacian, Gaussian, Cauchy, and sinc-squared. B^3 minimized the cross entropy in the forward direction and minimized the squared-error in the backward direction.

Tables I-III show that the non-uniform hidden priors improved the performance of neural classifiers and allowed deeper neural classifiers. Specifically: The Cauchy and sinc-squared priors outperformed others. This held for both the CIFAR-10 and the CIFAR-100 balanced datasets.

Figures 5 and 7 show that the Cauchy hidden priors increased the weight sparsity of the deep neural classifiers. Most of the weight parameters were close to 0 in this case. Small dispersion value increased the sparsity of the weights. Cauchy hidden priors achieved the highest level of sparsity.

VI. Conclusions

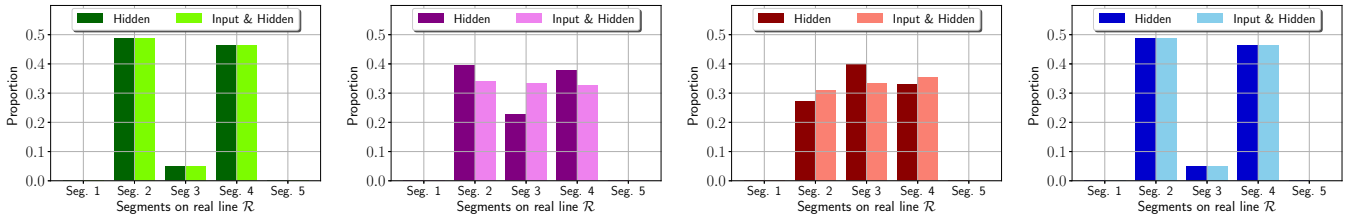
Non-uniform hidden priors improved the classification accuracy of Bayesian bidirectional backpropagation. The benefit of using non-hidden priors was most pronounced in very deep neural classifiers where uniform priors performed

TABLE II: Classification accuracy of Bayesian bidirectional backpropagation training with non-uniform hidden priors on the CIFAR-100 image dataset for Gaussian, Laplacian, Cauchy, and sinc-squared priors on the CIFAR-100 image dataset. The neural classifiers used uniform input and output priors. They also used 1,000 ReLU neurons per hidden layer

Network structure	**Unidirectional BP**	**Bayesian Bidirectional BP** (Hidden Priors)				
		Uniform Prior	Gaussian Prior	Laplacian Prior	Cauchy Prior	Sinc-squared Prior
2 hidden layers	30.75%	30.54%	**35.59%**	31.96%	35.09%	35.44%
4 hidden layers	31.87%	32.00%	32.55%	33.01%	32.36%	**34.02%**
6 hidden layers	31.64%	31.02%	31.88%	32.07%	32.26%	**32.44%**
8 hidden layers	30.50%	30.32%	31.13%	30.36%	**31.23%**	31.10%
10 hidden layers	26.93%	27.30%	**27.71%**	27.35%	27.44%	27.58%

TABLE III: Bayesian bidirectional backpropagation training with non-uniform hidden and input priors on the CIFAR-100 image dataset for Gaussian, Laplacian, Cauchy, and sinc-squared priors. The neural classifiers used uniform output priors. We compared non-uniform input *and* hidden priors on classification accuracy. The neural classifiers used 1,000 ReLU neurons per hidden layer.

Network structure	**Unidirectional BP**	**Bayesian Bidirectional BP** (Input & Hidden Priors)				
		Uniform Prior	Gaussian Prior	Laplacian Prior	Cauchy Prior	Sinc-squared Prior
2 hidden layers	30.75%	30.54%	35.89%	32.23%	**36.00%**	35.97%
4 hidden layers	31.87%	32.00%	34.26%	33.51%	34.17%	**34.34%**
6 hidden layers	31.64%	31.02%	32.55%	**33.07%**	32.92%	32.72%
8 hidden layers	30.50%	30.32%	30.67%	30.74%	**31.08%**	30.83%
10 hidden layers	26.93%	27.30%	27.45%	27.31%	**27.95%**	27.77%



(a) Gaussian weights (b) Laplacian weights (c) Cauchy weights (d) Sinc-squared weights

Fig. 7: Weight sparsity with Bayesian bidirectional backpropagation training on the CIFAR-100 image dataset: The figures show the weight distribution of the deep-neural classifier after it trained over 100 epochs. The classifiers used 8 hidden layers with 1,000 ReLU neurons *per* hidden layer. The plots divide the real line $\mathcal{R}$ into 5 segments: Seg. 1 denotes $(-\infty, -0.2)$, Seg. 2 denotes $[-0.2, -0.002)$, Seg. 3 denotes $[-0.002, 0.002)$, Seg. 4 denotes $[0.002, 0.2)$, and Seg. 5 denotes $[0.2, \infty)$. The plots show that non-uniform hidden priors promote sparsity. Cauchy hidden priors gave the sparsest weights.

poorly. Cauchy and sinc-squared priors often did best. The non-uniform priors also promoted weight sparsity. Cauchy "soft diamond" hidden priors produced the sparsest weights.

REFERENCES

[1] O. Adigun and B. Kosko, "Bidirectional backpropagation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 5, pp. 1982–1994, 2019.

[2] A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su, "A weakly informative default prior distribution for logistic and other regression models," 2008.

[3] T. Hastie, "Ridge regularization: An essential concept in data science," *Technometrics*, vol. 62, no. 4, pp. 426–433, 2020.

[4] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proceedings of the Twenty-first International Conference on MachineLlearning*, 2004, p. 78.

[5] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[6] T. Park and G. Casella, "The Bayesian Lasso," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 681–686, 2008.

[7] O. Adigun and B. Kosko, "Deeper neural networks with non-vanishing logistic hidden units: NoVa vs. ReLU neurons," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 1407–1412.

[8] ——, "Bayesian bidirectional backpropagation learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–7.

[9] B. Kosko, "Bidirectional associative memories: Unsupervised Hebbian learning to bidirectional backpropagation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 103–115, 2021.

[10] ——, "Bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

[11] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.