Bayesian Bidirectional Backpropagation Learning

Olaoluwa Adigun
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, CA 90089-2564.
adigun@usc.edu

Bart Kosko
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, CA 90089-2564.
kosko@usc.edu

***Abstract*—We show that training neural classifiers with *Bayesian* bidirectional backpropagation improves the performance of the network. Bidirectional backpropagation trains a deep network for both forward and backward recall through the same layers of neurons and with the same weights. It maximizes the network's joint forward and backward likelihood. Bayesian bidirectional backpropagation combines prior probabilities at the input and output layers with the likelihood structure of the layers. It maximizes the posterior probability of the network. It differs from other forms of neural Bayesian estimation because it uses the bidirectional likelihood of the network instead of the unidirectional likelihood. Bayesian bidirectional backpropagation outperformed classifiers trained with both unidirectional and bidirectional backpropagation. The networks trained on the CIFAR-10 and CIFAR-100 image test sets. A Laplacian or Lasso-like prior outperformed both Gaussian and uniform priors.**

***Index Terms*—Bidirectional backpropagation, regularization term, backpropagation invariance, Bayesian training**

I. Bayesian Bidirectional Backpropagation

We show how prior probabilities can improve the likelihood structure of the recent bidirectional backpropagation algorithm [1]–[4]. This greater control over the network's layer likelihood structure improves classification accuracy.

Bidirectional backpropagation allows deep networks to run and train in reverse through the same network structure. Bidirectional operation exploits information in the training data that forward-only operation simply ignores. It also takes full advantage of the likelihood structure of all neural layers. This allows the careful addition of layer priors to improve classification or regression.

Figure 1 shows the Bayesian bidirectional architecture when the input weights include a Laplacian prior on the multivariate normal likelihood. The identity input neurons define a regressor in the backward direction. The Laplacian prior acts like a Lasso or l^1 constraint on the backward regression. Figures 2 and 3 show image samples from the respective CIFAR-10 and CIFAR-100 datasets that trained the deep neural classifiers.

Ordinary backpropagation trains a neural network for forward inference from an input pattern $\mathbf{x}$ to an output vector $\mathbf{y}$ for classification or regression [5]–[7]. This backpropagation training is a form of the Expectation-Maximization algorithm [8], [9]. So it iteratively climbs the nearest hill of likelihood or log-likelihood in parameter space. It also maximizes only the *forward* likelihood $p_f(\mathbf{y}|\mathbf{x}, \mathbf{\Theta})$ for a given vector of network parameters Θ [8], [9].

Bidirectional backpropagation maximizes *both* the forward likelihood $p_f(\mathbf{y}|\mathbf{x}, \mathbf{\Theta})$ and the backward likelihood $p_b(\mathbf{x}|\mathbf{y}, \mathbf{\Theta})$. It does not overwrite the forward training because it maximizes the *joint* likelihood. Ordinary backpropagation training ignores the backward probability because it maximizes only the forward likelihood $p_f(\mathbf{y}|\mathbf{x}, \mathbf{\Theta})$.

The key idea is that the forward and backward training epochs maximize the likelihood product $p_f(\mathbf{y}|\mathbf{x}, \mathbf{\Theta})p_b(\mathbf{x}|\mathbf{y}, \mathbf{\Theta})$. So bidirectional backpropagation maximizes the sum L of the log-likelihoods [1], [2], [4]:

$$L = \log p_f(\mathbf{y}|\mathbf{x}, \Theta) + \log p_b(\mathbf{x}|\mathbf{y}, \mathbf{\Theta}). \tag{1}$$

The log-likelihood sum L corresponds to different error functions in general. Using the correct error function at the terminal layers prevents overwriting. It also allows the network to exploit more of the information in the input-output training data. Then running a bidirectionally trained classifier in reverse produces an estimate of a pattern-class centroid at the input because the input layer acts as a regressor with a squared-error error function. But running a forward-only trained classifier in reverse tends to produce only noise at the input.

Figure 4 shows these different effects in a trained deep classifier. The network had 7 hidden layers with 512 ReLU neurons in each hidden layer. The network had 10 output softmax neurons and it trained on the CIFAR-10 image dataset. Panel (b) shows that the network's backward pass produced only visual noise after it trained with ordinary or unidirectional backpropagation. Panel (c) shows that the bidirectionally trained network's backward pass produced good centroidal estimates of the pattern class given the same pattern-class labels or unit bit vectors as input stimuli.

The bidirectionally trained network in Figure 4 also had better classification accuracy with only slightly greater training cost. It had 56.07% accuracy while the unidirectionally trained classifier had only 54.36% accuracy. Bayesian bidirectional training with the Laplacian or Lasso-like prior in (52) further increased the classification accuracy to 57.01%. Training on the CIFAR-10 dataset also produced overwriting in the backward direction for unidirectional backpropagation but not for bidirectional backpropagation. The bidirectionally trained network had 28.02% classification accuracy on the CIFAR-100 dataset. The unidirectionally trained network had only 26.01% accuracy.

Matching the layer likelihood to the neurons at that layer preserves the backpropagation learning algorithm. We call this *backpropagation invariance* [9]. Backpropagation invariance ensures that the same backpropagation gradient learning laws hold at each layer and in each direction. BP invariance also allows direct noise injection in the layers to improve both convergence and classification accuracy. This holds because backpropagation is a special case of generalized Expectation-Maximization [8], [9] and because we can always noise-boost the EM algorithm with just that noise that makes the current signal more probable as the EM algorithm climbs the nearest hill of log-likelihood [10], [11].

We can extend the joint likelihood structure of bidirectional backpropagation to a joint posterior structure by adding a prior probability at the input and output layer. We call this *Bayesian* bidirectional backpropagation. It amounts to multiplying the layer likelihood by the prior. It gives back ordinary likelihood-only bidirectional backpropagation if the priors are uniform.

Figure 1 shows the posterior structure of Bayesian bidirectional backpropagation. The figure shows the important case of putting a Laplacian or Lasso-like prior on the input likelihood layer. The input layer of identity neurons acts as a regressor when neural signals pass backwards from the output or classification probability vector $\mathbf{y}$. So it has a multivariate normal likelihood. Putting a normal prior on this likelihood gives a type of ridge regressor [12], [13] in the backward direction. Simulations on the CIFAR-10 and CIFAR-100 image datasets found that the Laplacian prior produced better overall classification accuracy than did Gaussian or uniform priors. Table II also shows that the Laplacian prior did better with a Gaussian than a Laplacian backward likelihood.

II. Bayesian Bidirectional Backpropagation

Bayesian training of a neural network is a form of maximum $a\ posteriori$ (MAP) estimation. It maximizes the posterior $f(\Theta|x)$:

$$f(\Theta|x) = \frac{g(x|\Theta)h(\Theta)}{\int g(x|\Theta)h(\Theta)\ d\Theta} \propto g(x|\Theta)h(\Theta) \tag{2}$$

where $g(x|\Theta)$ is the likelihood and $h(\Theta)$ is the prior density of the now random vector of parameters Θ. The MAP estimate Θ^{MAP} equivalently maximizes the log-posterior and thus maximizes the sum of the log-likelihood and the log-prior:

$$\begin{aligned}\Theta^{MAP} &= \arg\max_{\Theta}\ f(\Theta|x) &(3)\\ &= \arg\max_{\Theta}\ g(x|\Theta)h(\Theta) &(4)\\ &= \arg\max_{\Theta}\ \log g(x|\Theta) + \log h(\Theta) &(5)\end{aligned}$$

because the unconditional total-probability denominator term $\int g(x|\Theta)h(\Theta)d\Theta$ is not a function of Θ.

The bidirectional backpropagation algorithm uses separate directional posteriors (and can extend to any finite number of directions). The *forward* posterior $p_f(\Theta|x)$ has the usual network form from Bayes theorem:

$$p_f(\Theta|x) = \frac{g_f(x|\Theta)h_f(\Theta)}{\int g_f(x|\Theta)h_f(\Theta)\ d\Theta} \propto g_f(x|\Theta)h_f(\Theta) \tag{6}$$

where $g_f(x|\Theta)$ is the forward likelihood and $h_f(\Theta)$ is the forward prior. The *backward* posterior $p_b(\Theta|y)$ has the like form

$$p_b(\Theta|y) = \frac{g_b(y|\Theta)h_b(\Theta)}{\int g_b(y|\Theta)h_b(\Theta)\ d\Theta} \propto g_b(y|\Theta)h_b(\Theta) \tag{7}$$

where $g_b(y|\Theta)$ is the backward likelihood and $h_b(\Theta)$ is the backward prior.

The bidirectional network's total or joint bidirectional posterior combines the forward posterior $p_f(\Theta|x)$ *and* the backward posterior $p_b(\Theta|y)$. The conjunctive "and" gives rise to the posterior product $p_f(\Theta|x)p_b(\Theta|y)$ because to first order the two directional passes are independent of each other. This gives the joint posterior as this product $p_f(\Theta|x)p_b(\Theta|y)$ (this generalizes to k directions as just the product of the k directional posteriors). Then the bidirectional MAP estimate Θ^{BMAP} maximizes the joint bidirectional posterior:

$$\begin{aligned}\Theta^{BMAP} &= \arg\max_{\Theta}\ p_f(\Theta|x)p_b(\Theta|y) &(8)\\ &= \arg\max_{\Theta}\ g_f(x|\Theta)h_f(\Theta)g_b(y|\Theta)h_b(\Theta). &(9)\end{aligned}$$

Maximizing the log-posterior gives the same MAP estimate:

$$\begin{aligned}\Theta^{BMAP} &= \arg\max_{\Theta}\ \log g_f(x|\Theta) + \log h_f(\Theta)\\ &\qquad + \log g_b(y|\Theta) + \log h_b(\Theta) &(10)\\ &= \arg\max_{\Theta}\ \log g_f(x|\Theta) + \log g_b(y|\Theta)\\ &\qquad + \log h_f(\Theta) + \log h_b(\Theta) &(11)\\ &= \arg\max_{\Theta}\ \log g_f(x|\Theta) + \log g_b(y|\Theta) + \log h(\Theta) &(12)\end{aligned}$$

where $h(\Theta) = h_f(\Theta)h_b(\Theta)$.

The original bidirectional backpropagation (B-BP) algorithm [1], [3], [4], [14] is a form of maximum likelihood estimation. B-BP training endows a multilayer neural network with a form of bidirectional inference and proceeds as the joint gradient optimization of the forward likelihood $p_f(\mathbf{y}|\mathbf{x},\Theta)$ and the backward likelihood $p_b(\mathbf{x}|\mathbf{y},\Theta)$. So the B-BP maximum-likelihood estimate Θ^{BBP} has the form

$$\begin{aligned}\Theta^{BBP} &= \arg\max_{\Theta}\ p_f(\mathbf{y}|\mathbf{x},\Theta)\ p_b(\mathbf{x}|\mathbf{y},\Theta) &(13)\\ &= \arg\max_{\Theta}\ \log p_f(\mathbf{y}|\mathbf{x},\Theta) + \log p_b(\mathbf{x}|\mathbf{y},\Theta). &(14)\end{aligned}$$

Suppose the default case that both the forward prior $h_f(\Theta)$ and the backward prior $h_b(\Theta)$ are uniform probability densities. Then the Bayesian B-BP estimate Θ^{BMAP} reduces to the original maximum-likelihood B-BP estimate Θ^{BBP}:

$$\Theta^{BMAP} = \Theta^{BBP}. \tag{15}$$

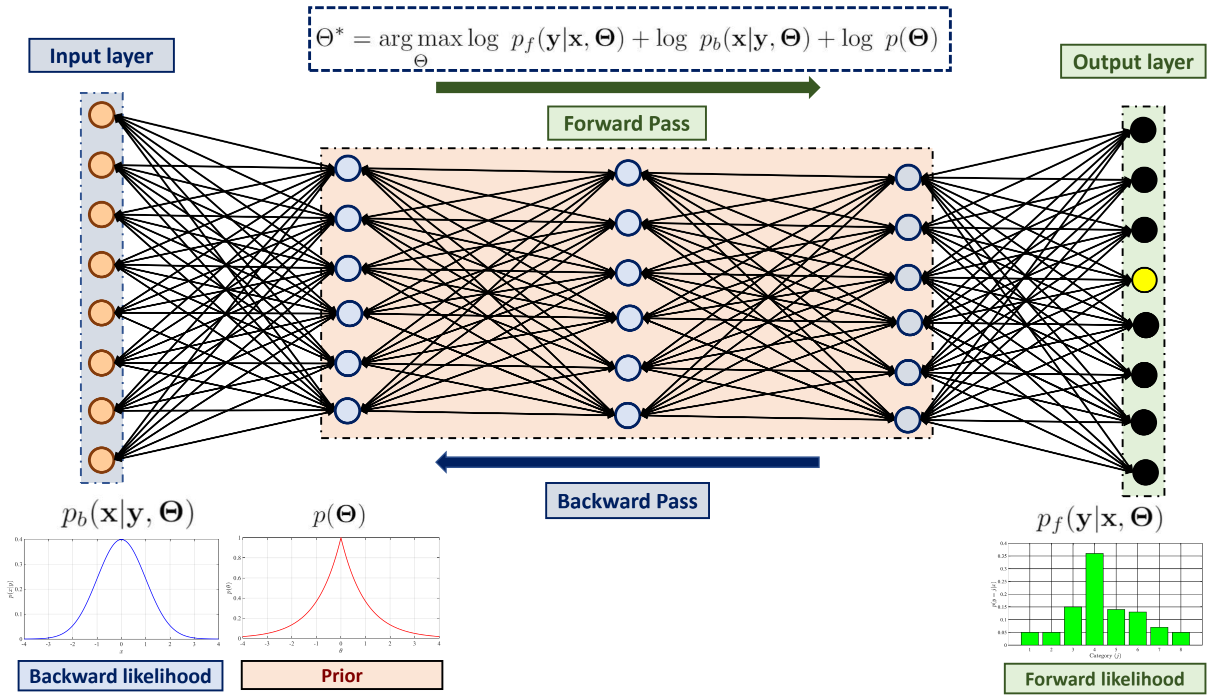

$\Theta^* = \arg\max_{\Theta} \log\ p_f(\mathbf{y}|\mathbf{x},\mathbf{\Theta}) + \log\ p_b(\mathbf{x}|\mathbf{y},\mathbf{\Theta}) + \log\ p(\mathbf{\Theta})$
Input layer
Output layer
Forward Pass
Backward Pass
$p_b(\mathbf{x}|\mathbf{y},\mathbf{\Theta})$
$p(\mathbf{\Theta})$
$p_f(\mathbf{y}|\mathbf{x},\mathbf{\Theta})$
Backward likelihood
Prior
Forward likelihood

Fig. 1: Bayesian Bidirectional Backpropagation: The network maximizes the *joint* forward and backward posterior probability. The network diagram shows the simple but practical case of a Laplacian or Lasso-like prior on the input weights. The identity neurons at the input field give rise to a vector normal likelihood and thus a squared-error error function in the backward direction. The softmax neurons at the output field give rise to a multinomial likelihood and thus a cross-entropy error function in the forward direction. So the network acts as classifier in the forward direction and a regressor in the backward direction.

A. Bidirectional Mapping Structure

A bidirectional neural network $\mathcal{N}$ is a feedback dynamical system [4], [15]. It passes neural information forward and backward through the same web of synapses [1], [4] and thus weight matrices W and their transposes W^T.

The forward pass propagates the input $\mathbf{x} \in \mathcal{X}$ from the input layer through the hidden layers to the output layer of the network. The forward pass has the form

$$\mathbf{a}^t = \mathcal{N}(\mathbf{x}) \tag{16}$$

where $\mathbf{a}^t$ represents the activation at the output layer. The backward pass propagates the target vector $\mathbf{t} \in \mathcal{Y}$ from the output layer of the network through the hidden layers to its input layer. This requires propagating $\mathbf{t}$ through the transpose W^T of the weight matrices. The backward pass representation has the form

$$\mathbf{a}^x = \mathcal{N}^T(\mathbf{t}) \tag{17}$$

where $\mathbf{a}^x$ represents the activation at the input layer.

The inverse mapping $\mathcal{N}^{-1}$ does not exist as a point mapping in general. But it always exists as *set*-valued mapping or inverse or pullback mapping from the power set of the network's range set back to the power set of its pattern domain set. Neural classifiers use 1-in-K encoding of their K pattern classes in the forward direction. The K unit basis vectors $\mathbf{b_1}, \ldots, \mathbf{b_K}$ code for the classes and define the K target vectors in forward training. Then the K pullbacks $\mathcal{N}^{-1}(\mathbf{b_k})$ partition the input pattern space $\mathbb{R}^I$: $\mathbb{R}^I = \mathcal{N}^{-1}(\mathbf{b_1}) \cup \cdots \cup \mathcal{N}^{-1}(\mathbf{b_K})$. The whole point of training a classifier is to achieve such a partition that carves up the pattern space into exactly the right K pattern classes $C_1, \ldots, C_K$. Bidirectional processing assists with this task by using rather than ignoring the associative information inherent in the backward pass while training.

We next show how to structure this B-BP training for a bidirectional network that classifies in the forward direction and regresses in the backward direction. Then the backward-pass vector $\mathcal{N}^T(\mathbf{b_k})$ of the output target unit bit vector $\mathbf{b_k}$ should approximate the sample class centroid of pattern class C_k because the centroid minimizes the squared error of the input regression layer. Figure 4 confirms that both likelihood-only B-BP and Bayesian B-BP estimate these sample class centroids while ordinary unidirectional BP does not.

1) Forward Likelihood: Supervised training uses labeled input patterns from the K pattern classes $C_1, \ldots, C_K$. The forward pass maps the labeled input vector $\mathbf{x}$ to its corresponding target $\mathbf{t}$ or unit basis vector $\mathbf{b_k}$. The K output softmax neurons in the classifier define a Bayesian K-class classifier [16], [17]. Their softmax ratio structure dictates that this terminal layer has the forward likelihood $p_f(\mathbf{t}|\mathbf{x}, \Theta)$ of a one-shot multinomial. So a vector passing through the network corresponds to one roll of a K-sided die where the K sides have different probabilities in general. We encode the K pattern classes with the K unit bit vectors of the $K-$dimensional unit hypercube $\{0, 1\}^K$. An alternative uses independent Bernoulli probabilities as the terminal forward likelihood [18]. The output activation in that case is binary

Algorithm 1 How to train a neural classifier with Bayesian bidirectional backpropagation.

```
Require: Dataset $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, batch size $L$, learning rate $\eta$, backward training start $M^*$, number $M$ of epochs, number $B$ of iterations per epoch, penalty parameter $\lambda$, and backward loss coefficient $\alpha$.
Require: Initialize the network parameter $\Theta^{(0)}$.
1:  Compute the class centroids for all the $K$ classes:
        $\mathbf{c}_k = \frac{1}{N_k} \sum_{j=1}^{N_k} \mathbf{x}^{k(j)}.$
2:  Compute the projection $\tilde{\mathbf{x}}_k$ of class centroid $\mathbf{c}_k$ on the class sample set $\mathcal{S}_k$ for all the $K$ classes:
        $\tilde{\mathbf{x}}_k = \underset{\mathbf{x}^{(i)} \in \mathcal{S}_k}{\arg\min} ||\mathbf{x}^{(i)} - \mathbf{c}_k||^2.$
3:  for $m = 1$ to $M$ do
4:      Pick $L$ random samples from $\mathcal{D}$.
5:      Compute the forward pass of the data batch:
            $\mathbf{a}^{t(l)} = \mathcal{N}(\mathbf{x}^{(l)}).$
6:      Compute the forward error $E_f$:
            $E_f = -\frac{1}{L} \sum_{l=1}^{L} \mathbf{t}^{(l)^T} \log \mathbf{a}^{t(l)}.$
7:      if $m < M^*$ then
8:          Compute the backward pass of the data batch:
                $\mathbf{a}^{x(l)} = \mathcal{N}^T(\mathbf{t}^{(l)}).$
9:          Compute the backward error $E_b$:
                $E_b = -\frac{\alpha}{2L} \sum_{l=1}^{L} ||\mathbf{a}^{x(l)} - \tilde{\mathbf{x}}^{(l)}||_2^2.$
10:         Compute the penalty term:
                $R_\Theta = \lambda ||\Theta||_1 \ .$
11:         Bidirectional update:
                $\Theta^{(m+1)} = \Theta^{(m)} - \eta \nabla_\Theta \big(E_f + E_b + R_\Theta\big)\Big|_{\Theta=\Theta^{(m)}}$
12:     else
13:
                $\Theta^{(m+1)} = \Theta^{(t)} - \eta \nabla_\Theta \big(E_f + R_\Theta\big)\Big|_{\Theta=\Theta^{(m)}}$
14:     end if
15: end for
```

logistic or bipolar logistic.

The multinomial or categorical forward likelihood has the form

$$p_f(\mathbf{t}|\mathbf{x}, \Theta) = \prod_{k=1}^{K} \left(a_k^t\right)^{t_k} \tag{18}$$

$$\log \ p_f(\mathbf{t}|\mathbf{x}, \Theta) = \sum_{k=1}^{K} t_k \log \ a_k^t \tag{19}$$

where a_k^t is the activation of the k^{th} output neuron and $\mathbf{t}^k$ is the target of the k^{th} output neuron with $0 \leq a_k^t \leq 1$ and $\sum_{k=1}^{K} a_k^t = 1$. Then the forward error $E_f(\Theta)$ at the output softmax layer is just the cross-entropy:

$$E_f(\Theta) = -\log \ p_f(\mathbf{t}|\mathbf{x}, \Theta) = -\sum_{k=1}^{K} t_k \log \ a_k^t. \tag{20}$$

2) Backward Likelihood: The backward pass maps the target vector $\mathbf{t}$ back through the network and its transposed weight matrices W^T to the input pattern space. We modeled the backward likelihood as a bell curve because the input neurons have identity activations. We chose the usual vector Gaussian density function or the vector Laplacian probability density. These densities give the respective loss functions as the squared error (SE) or the absolute error (AE). Assume there are J input neurons.

The backward likelihood $p_b(\mathbf{x}|\mathbf{t}, \Theta)$ in the Gaussian or multivariate-normal case is

$$p_b(\mathbf{x}|\mathbf{t}, \Theta) = \frac{1}{2\pi^{\frac{J}{2}}} \exp^{-\frac{||\mathbf{a}^x - \mathbf{x}||_2^2}{2}} \tag{21}$$

$$\log \ p_b(\mathbf{x}|\mathbf{t}, \Theta) = -\frac{J}{2} \log 2\pi - \frac{1}{2} ||\mathbf{a}^x - \mathbf{x}||_2^2 \tag{22}$$

because $p_b(\mathbf{x}|\mathbf{t}, \Theta) \sim \text{Gaussian}(\mathbf{x}|\mathbf{a}^x, \mathbf{I})$. Then the backward error $E_b(\Theta)$ is the negative log-likelihood and thus the squared error:

$$\begin{aligned} E_b(\Theta) &= -\log \ p_b(\mathbf{x}|\mathbf{t}, \Theta) - \log(2\pi)^{\frac{J}{2}} && (23)\\ &= -\log \ p_b(\mathbf{x}|\mathbf{t}, \Theta) - \frac{J}{2} \log 2\pi && (24)\\ &= \frac{1}{2} ||\mathbf{a}^x - \mathbf{x}||_2^2 && (25) \end{aligned}$$

where $u = \frac{J}{2} \log 2\pi$ does not depend on Θ.

The backward Laplacian likelihood has the form

$$\begin{aligned} p_b(\mathbf{x}|\mathbf{t}, \Theta) &= \prod_{j=1}^{J} \frac{1}{2} \exp^{-|a_j^x - x_j|} && (26)\\ &= \frac{1}{2^J} \exp^{-\sum_{j=1}^{J} |a_j^x - x_j|} && (27)\\ &= \frac{1}{2^J} \exp^{-||\mathbf{a}^x - \mathbf{x}||_1}. && (28) \end{aligned}$$

$$\log \ p_b(\mathbf{x}|\mathbf{t}, \Theta) = -J \log 2 - ||\mathbf{a}^x - \mathbf{x}||_1 \tag{29}$$

because $\mathbf{x} = (x_1, x_2, ..., x_J)$ consists of J *i.i.d.* random variables such that $p(x_i|\mathbf{t}, \Theta) \sim \text{Laplace}(x_i|\mathbf{a}^x, 1)$. This gives the absolute error (AE) as the backward error $E_b(\Theta)$:

$$\begin{aligned} E_b(\Theta) &= -\log\ p_b(\mathbf{x}|\mathbf{t}, \Theta) - \log 2^J && (30)\\ &= -\log\ p_b(\mathbf{x}|\mathbf{t}, \Theta) - J\log 2 && (31)\\ &= ||\mathbf{a}^x - \mathbf{x}||_1 && (32)\end{aligned}$$

where $u = J\log 2$ does not depend on Θ. Then (23)-(25) and (30)-(32) show that the negative of $\log p_b(\mathbf{x}|\mathbf{t}, \Theta)$ for the Laplacian and Gaussian densities equal the sum of $E_b(\Theta)$ and a constant with respect to Θ. So minimizing $E_b(\Theta)$ maximizes the log-likelihood $\log p_b(\mathbf{x}|\mathbf{t}, \Theta)$ with respect to Θ.

3) Parameter Prior: The prior $h(\Theta)$ is the unconditional distribution of Θ although hierarchical priors can also apply. The default priors is the uniform prior and reduces Bayesian B-BP to likelihood-only B-BP as in (15). We also tested normal and Laplacian (ridge-like and Lasso-like) priors on the backward weights at the input layer.

The log-prior of the normal prior $\Theta \sim \text{Gaussian}(\Theta|\mathbf{0}, \tau^2\mathbf{I})$ involves a squared error:

$$\begin{aligned} \log h(\Theta) &= \log{(2\pi\tau^2)}^{-\frac{K}{2}} \exp^{-\frac{||\Theta||_2^2}{2\tau^2}} && (33)\\ &= -\frac{K}{2}\log{(2\pi\tau^2)} - \frac{||\Theta||_2^2}{2\tau^2} && (34)\\ &= -\frac{K}{2}\log{(2\pi\tau^2)} - \lambda||\Theta||_2^2 && (35)\\ &= w - \lambda||\Theta||_2^2 && (36)\end{aligned}$$

with $\lambda = \frac{1}{2\tau^2}$ and $w = -\frac{K}{2}\log{(2\pi\tau^2)}$.

The log-prior of the Laplace prior $\Theta \sim \text{Laplace}(\Theta|\mathbf{0}, \tau\mathbf{I})$ likewise involves the absolute error:

$$\begin{aligned} \log h(\Theta) &= \log{(2\tau)}^{-K} \exp^{-\frac{||\Theta||_1}{\tau}} && (37)\\ &= -K\log{(2\tau)} - \frac{||\Theta||_1}{\tau} && (38)\\ &= -K\log{(2\tau)} - \lambda||\Theta||_1 && (39)\\ &= w - \lambda||\Theta||_1 && (40)\end{aligned}$$

with $\lambda = \frac{1}{\tau}$ and $w = -K\log{(2\tau)}$.

B. Neural Classifier and Bayesian Bidirectional Backpropagation Algorithm

We next define the classifier-regressor probability structure that we used to train the Bayesian B-BP classifiers. The algorithms find the maximizing value Θ^* with gradient ascent on the given bidirectional posterior. .

The classifier's forward likelihood $p_f(\mathbf{t}|\mathbf{x}, \Theta)$ is multinomial for output target vector $\mathbf{t}$. Its backward likelihood $p_b(\mathbf{x}|\mathbf{t}, \Theta)$ is vector normal or Gaussian. The backward prior $h(\Theta)$ is Laplacian or a Lasso-like l^1 penalty. Then

$$\begin{aligned} \Theta^* &= \arg\max_{\Theta}\ p_f(\Theta|\mathbf{t}, \mathbf{x})\ p_b(\Theta|\mathbf{x}, \mathbf{t}) && (41)\\ &= \arg\max_{\Theta}\ \log p_f(\mathbf{t}|\mathbf{x}, \Theta) + \log p_b(\mathbf{x}|\mathbf{t}, \Theta)\\ &\qquad + \log h(\Theta) && (42)\\ &= \arg\min_{\Theta}\ -\log p_f(\mathbf{t}|\mathbf{x}, \Theta) - \log p_b(\mathbf{x}|\mathbf{t}, \Theta)\\ &\qquad - \log h(\Theta) && (43)\\ &= \arg\min_{\Theta}\ E_f(\Theta) + E_b(\Theta) + \lambda||\Theta||_1 - w && (44)\\ &= \arg\min_{\Theta}\ E_f(\Theta) + E_b(\Theta) + \lambda||\Theta||_1 && (45)\end{aligned}$$

because (20) and (23) - (25) show that

$$\begin{aligned} -\log p_f(\mathbf{t}|\mathbf{x}, \Theta) &= E_f(\Theta) && (46)\\ -\log p_b(\mathbf{x}|\mathbf{t}, \Theta) &= E_b(\Theta) + u && (47)\\ -\log h(\Theta) &= \lambda||\Theta||_1 - w && (48)\end{aligned}$$

where u and w do not depend on Θ. This simplifies to

$$\begin{aligned} \Theta^* &= \arg\min_{\Theta}\ E_f(\Theta) + E_b(\Theta) + \lambda||\Theta||_1 && (49)\\ &= \arg\min_{\Theta}\Big(- \mathbf{t}^T \log\ \mathbf{a}^t + \frac{1}{2}||\mathbf{a}^x - \mathbf{x}||_2^2 + \lambda||\Theta||_1\Big) && (50)\\ &= \arg\min_{\Theta}\Big(- \sum_{k=1}^{K} t_k \log\ a_k^t + \frac{1}{2}\sum_{j=1}^{J}|a_j^x - x_j|^2 + \lambda||\Theta||_1\Big). && (51)\end{aligned}$$

Algorithm 1 lists the pseudocode for Bayesian bidirectional backpropagation with a Laplacian or Lasso-like prior. The pseudocode describes the Bayesian error function

$$E(\Theta) = -\mathbf{t}^T \log\ \mathbf{a}^t + \alpha||\mathbf{a}^x - \mathbf{x}||^2 + \lambda||\Theta||_1 \qquad (52)$$

for a classifier-regressor bidirectional network. The forward pass uses the one-shot multinomial or categorical distribution. So its negative logarithm gives the error as a cross-entropy. The backward pass uses the multivariate Gaussian probability. So its error function is just the squared error. The forward pass includes the regularizing Laplacian prior and its corresponding l^1 or absolute error.

III. Simulation Experiments

A. Datasets

This classification simulations trained and tested on the CIFAR-10 and CIFAR-100 image datasets.

1) CIFAR-10: CIFAR-10 is a set of 60,000 color images from 10 classes. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck [19]. Each class consists of 5,000 training samples and 1,000 testing samples. Each image has dimension of $32 \times 32 \times 3$.

2) CIFAR-100: CIFAR-100 is a set of 60,000 color images from 100 pattern classes with 600 images per class. The 100 classes divide into 20 super-classes. Each super-class consists of 5 classes [19]. Each image has dimension $32{\times}32{\times}3$.

Fig. 2: CIFAR-10 sample images: The figure shows 10 samples from the CIFAR-10 dataset that contains 10 pattern classes and a total of 60,000 sample images.

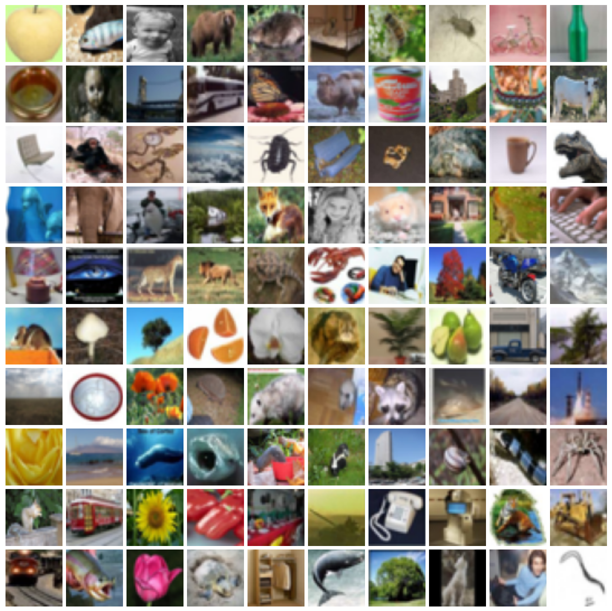


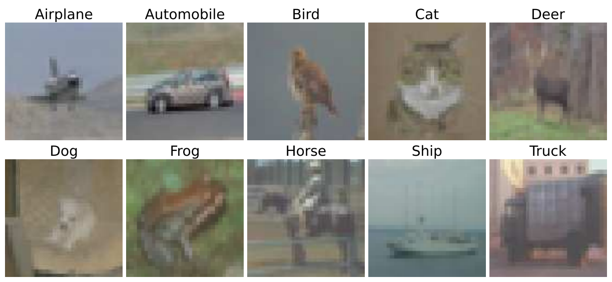
Fig. 3: CIFAR-100 sample images: This figure shows 100 samles from the CIFAR-100 dataset that contains 100 pattern classes with 600 images per class. CIFAR-100 consists of 20 super-classes with 5 classes per super-class.

B. Network Description

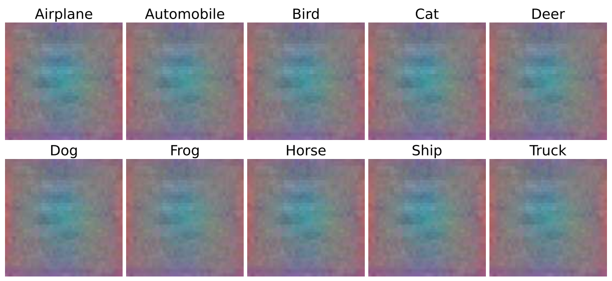
We trained deep neural classifiers on the CIFAR-10 and CIFAR-100 datasets. All the classifier networks used identity neurons in the input layer and 512 ReLU neurons in each hidden layer. They used either 10 or 100 softmax neurons in the output layer. We trained some of the classifiers with ordinary unidirectional BP as a baseline. We trained the other classifiers with B-BP either without or with a Bayesian prior. A dropout value of 0.2 for the hidden-layers reduced overfitting in the models.

C. Results and Discussion

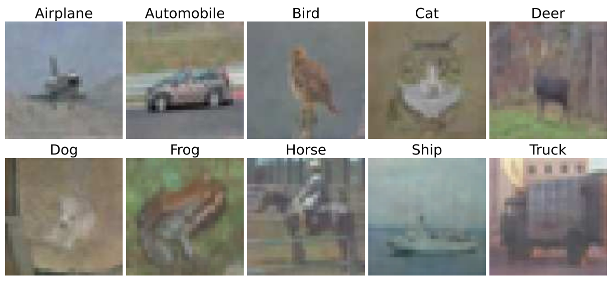
Table I shows the benefits of using Bayesian bidirectional backpropagation to train a deep neural classifier. The baseline trained with unidirectional BP. The backward prior or penalty/regularizer term was a Laplacian or Lasso-like prior for B-BP. The table data show that Bayesian B-BP outperformed both unidirectional BP and B-BP.



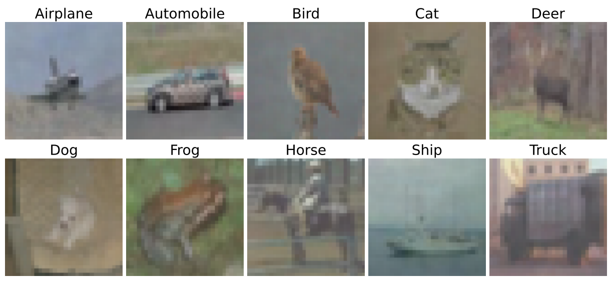
(a) Target images from the CIFAR-10 dataset for backward pass through a deep classifier.



(b) Backward pass: Unidirectional backpropagation produced only visual noise.



(c) Backward pass: Bidirectional backpropagation produced centroidal estimates of the correct pattern class.



(d) Backward pass: Bayesian bidirectional backpropagation with a Laplacian prior produced still better centroidal estimates.

Fig. 4: Backward-pass recall in deep classifiers trained the CIFAR-10 image dataset. The unidirectionally trained classifier produced only visual noise at the input layer on the backward pass while the bidirectionally trained classifier produced a good centroidal estimate of the pattern class. The classifiers used 7 hidden layers with 512 ReLU neurons in each hidden layer and 10 output softmax neurons. The 3072 input neurons had identity activations and so defined a regression layer on the backward pass. Unidirectional backpropagation caused overwriting in the backward direction during training while bidirectional backpropagation did not. Bidirectional and Bayesian training also had better classification accuracy.

Table II shows the classification accuracy for 4 different cases of Bayesian B-BP. The simulations compared two different backward likelihoods with two different priors. The two likelihoods were the usual squared error and the similar but more robust absolute error. The two priors were the Gaussian and the Laplacian. The best combination was a normal backward likelihood (squared error) and a backward Laplace prior for both the CIFAR-10 and CIFAR-100 datasets.

TABLE I: Bayesian B-BP training outperformed both unidirectional and likelihood-only B-BP training. Baseline training used unidirectional BP. B-BP outperformed the baseline and Lasso Bayesian B-BP outperformed likelihood-only B-BP for both the CIFAR-10 and CIFAR-100 datasets.

Dataset	Training Method	Penalty Term	Accuracy
CIFAR-10	Baseline		54.36%
	Bidirectional BP	No	55.16%
	Bidirectional BP	Yes	**57.01%**
CIFAR-100	Baseline		26.01%
	Bidirectional BP	No	27.78%
	Bidirectional BP	Yes	**28.02%**

TABLE II: 4 types of Bayesian B-BP training. Classifier simulations compared squared-error (SE) Gaussian and absolute-error (AE) Laplacian backwards likelihoods with Gaussian (SE) and Laplacian or Lasso-like (AE) priors or penalty terms. The best combination was a Gaussian likelihood and Laplace prior for both the CIFAR-10 and CIFAR-100 datasets.

Dataset	Backward Loss	Penalty Term	Accuracy
CIFAR-10	SE	SE	55.98%
	SE	AE	**57.01%**
	AE	SE	56.10%
	AE	AE	56.07%
CIFAR-100	SE	SE	27.74%
	SE	AE	**28.02%**
	AE	SE	27.86%
	AE	AE	27.80%

IV. Conclusion

The new bidirectional backpropagation algorithm maximizes the joint forward and backward likelihood structure of a given multilayer neural network. Matching the layer likelihood structure to the layer neuron structures ensures that backpropagation invariance holds. Then the same backpropagation gradient learning laws holds in each direction. Maximizing the bidirectional posterior probability combines a prior probability structure with the layer likelihood structure. The Lasso-like Laplace prior in (52) gave the best classifier performance compared with Gaussian and uniform priors.

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] ——, "Noise-boosted bidirectional backpropagation and adversarial learning," *Neural Networks*, vol. 120, pp. 9–31, 2019.

[3] ——, "Training generative adversarial networks with bidirectional backpropagation," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1178–1185.

[4] 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.

[5] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *Doctoral Dissertation, Applied Mathematics, Harvard University, MA*, 1974.

[6] D. Rumelhart, G. Hinton, and W. R., "Learning representationsby back-propagating errors." *Nature*, pp. 323–533, 1986.

[7] M. Jordan and T. Mitchell, "Machine learning: trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, 2015.

[8] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Networks*, vol. 78, pp. 15–23, 2016.

[9] B. Kosko, K. Audhkhasi, and O. Osoba, "Noise can speed backpropagation learning and deep bidirectional pretraining," *Neural Networks*, vol. 129, pp. 359–384, 2020.

[10] O. Osoba, S. Mitaim, and B. Kosko, "The noisy expectation–maximization algorithm," *Fluctuation and Noise Letters*, vol. 12, no. 3, pp. 1 350 012–1–1 350 012–30, 2013.

[11] O. Osoba and B. Kosko, "The noisy expectation-maximization algorithm for multiplicative noise injection," *Fluctuation and Noise Letters*, vol. 15, no. 01, p. 1650007, 2016.

[12] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[13] B. Efron and T. Hastie, *Computer age statistical inference*. Cambridge University Press, 2016, vol. 5.

[14] O. Adigun and B. Kosko, "Bidirectional representation and backpropagation learning," in *International Joint Conference on Advances in Big Data Analytics*, 2016, pp. 3–9.

[15] B. Kosko, "Adaptive bidirectional associative memories," *Applied optics*, vol. 26, no. 23, pp. 4947–4960, 1987.

[16] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] O. Adigun and B. Kosko, "High capacity neural block classifiers with logistic neurons and random coding," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

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