NOISE BENEFITS IN CONVOLUTIONAL NEURAL NETWORKS

Kartik Audhkhasi, Osonde Osoba, Bart Kosko

Signal and Information Processing Institute
Electrical Engineering Department
University of Southern California, Los Angeles, CA
Email: kosko@sipi.usc.edu

ABSTRACT

We prove that noise speeds convergence in the back-propagation (BP) training of a convolutional neural network (CNN). CNNs are a popular model for large-scale image recognition. The proof builds on two recent theorems. The first result shows that the BP algorithm is a special case of the Expectation-Maximization (EM) algorithm. The second result states when adding noise to the data will speed convergence of the EM algorithm. Then this noisy EM algorithm (NEM) algorithm gives a simple geometrical condition for a noise speed up in the BP training of a CNN. Noise added to the output neurons speeds up CNN training if the noise lies above a hyperplane that passes through the origin. Simulations on the MNIST digit recognition data set show that the noisy BP algorithm reduces the mean per-iteration training-set cross entropy by 39% compared with the noiseless BP. The noisy BP algorithm also reduces the mean per-iteration training-set classification error by 47%. The noise benefit is more pronounced for small data sets.

Index Terms— Convolutional neural network, back-propagation algorithm, Expectation-Maximization (EM) algorithm, Noisy EM algorithm, noise benefit, stochastic resonance.

1. NOISE INJECTION IN CONVOLUTIONAL NEURAL NETWORKS

We prove that noise speeds up convergence in the popular back-propagation (BP) training algorithm [1] of a convolutional neural network (CNN) [2, 3]. CNNs are standard neural network systems for large-scale image recognition [4–9]. Figure 1 shows a 3-layer CNN with one hidden convolutional layer and 3 convolution masks. The proof builds on our previous result [10] that the backpropagation algorithm is a special case of the Expectation-Maximization (EM) algorithm [11]. We then use the recent noisy EM (NEM) algorithm [12, 13] that gives a sufficient condition to speed up convergence in the EM algorithm. Then the NEM algorithm's sufficient condition gives a simple geometrical sufficient condition for a noise speed up in the BP training of a CNN.

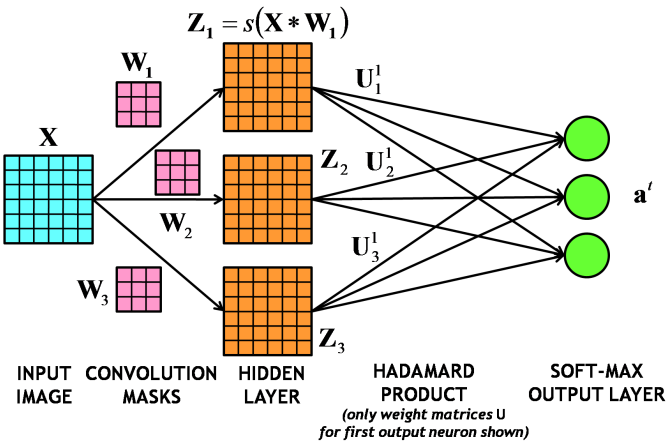

Fig. 1. Convolutional Neural Network (CNN): The figure shows a CNN with just one hidden layer. The input image $\mathbf{X}$ convolves with 3 masks $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3$. These masks act as receptive fields in the retina. The resulting images pass pixel-wise through logistic sigmoid functions s that give the hidden neuron activations. Then the CNN computes element-wise Hadamard products between the hidden neuron activation matrices $\mathbf{Z}_1, \mathbf{Z}_2$, and $\mathbf{Z}_3$ with weight matrices $\mathbf{U}_j^k$ where $j = 1, 2, 3$ and $k = 1, 2, 3$. The soft-max Gibbs signal function gives the activations of the output layer neurons.

Figure 2 shows the noise-benefit region for a CNN with three output neurons. Noise added to the output neurons speeds up the BP training of a CNN if the noise lies above a hyperplane that passes through the origin of the noise space. This is a simple linear condition that the noise must satisfy. The output layer activation vector $\mathbf{a}^t$ determines the normal to the hyperplane.

Figure 3 shows the training-set cross entropy of a CNN using standard noiseless BP, BP with blind noise (Blind-BP), and BP with NEM noise (NEM-BP). Noisy back-propagation reduces the average training-set cross entropy by 39.26% compared with noiseless back-propagation. Figure 4 plots the training-set classification error rates as the system trains. The testing-set classification error rate is essentially the same at

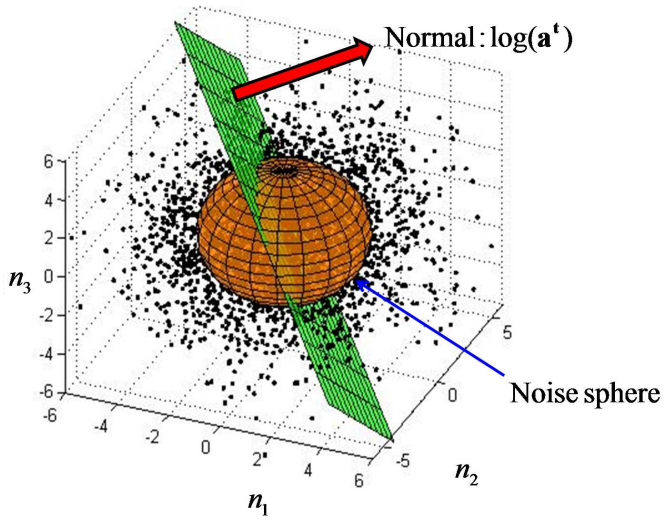

Fig. 2. Noise-benefit region for a CNN with soft-max output neurons: Noise speeds up maximum-likelihood parameter estimation of the CNN with soft-max output neurons if the noise lies above a CNN-based hyperplane that passes through the origin of the noise space. The activation signal $\mathbf{a}^t$ of the output layer controls the normal to the hyperplane. The hyperplane changes as learning proceeds because the parameters and hidden-layer neuron activations change. We used independent and identically distributed (i.i.d.) Gaussian noise with mean 0, variance 3, and $(3, 1, 1)$ as the normal to the hyperplane.

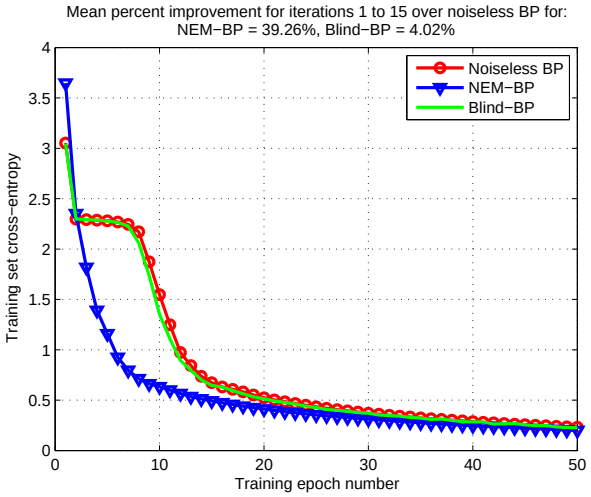

Fig. 3. NEM noise-benefit in BP training of a CNN using MNIST data: The figure shows that the NEM-BP training reduces the average training-set cross entropy of the MNIST data set compared with standard noiseless BP training. We obtain a 39.26% average reduction in cross entropy for the NEM-BP case when compared with the standard BP over the first 15 training iterations. Adding blind noise gives a minor average reduction of 4.02% in cross entropy. Training used 1000 images from the MNIST data for a CNN with one convolution hidden layer. The convolutional layer used three 3×3 masks or filters. Factor-2 downsampling followed the convolutional layer by removing all even index rows and columns of the hidden neuron images. The hidden layer fully connects to 10 output neurons that predict the class label of the input digit. We used uniform noise over $[-0.5/\sqrt{t^5}, 0.5/\sqrt{t^5}]$ where t is the training iteration number for both NEM and blind noise.

convergence. NEM-BP gives a 47.43% reduction in training-set error rate averaged over the first 15 iterations compared with noiseless BP. Adding blind noise only slightly improves cross entropy and classification accuracy. Figure 5 shows a noise-benefit inverted U-curve for NEM-BP training of a CNN on the MNIST data set. This inverted U-curve is the signature of a nonlinear noise benefit or so-called *stochastic resonance* [14–23]. The optimal uniform noise scale occurs at 1. NEM noise hurts CNN training when the noise scale increases beyond 2.6.

The next section presents an overview of CNNs. Section 3 presents the back-propagation algorithm for CNN training. Theorem 1 shows that the BP algorithm is a special case of the generalized EM (GEM) algorithm. Section 4 reviews the NEM algorithm. Section 5 presents the NEM-BP algorithm for CNN training. Section 6 summarizes the simulations on the MNIST data set.

2. CONVOLUTIONAL NEURAL NETWORKS

A convolutional neural network (CNN) convolves the input data with a set of filters. This is a rough analogy to the use of receptive fields in the retina [24] as in the Neocognitron network [25]. Consider a CNN with one hidden layer for simplicity. The notation extends directly to allow multiple hidden layers. Let $\mathbf{X}$ denote the input 2-dimensional data of size $M_X \times N_X$ where M_X and N_X are positive integers. Consider 2D filters $\mathbf{W}_1, \ldots, \mathbf{W}_J$ each of size $M_W \times N_W$. The convolution of $\mathbf{X}$ with the filter $\mathbf{W}_j$ gives

$$\mathbf{C}_j = \mathbf{X} \circledast \mathbf{W}_j \tag{1}$$

where $\circledast$ denotes 2D convolution. The 2D data matrix $\mathbf{C}_j$ has size $(M_X + M_W - 1) \times (N_X + N_Y - 1)$ with (m, n)-th entry

$$\mathbf{C}_j(m, n) = \sum_{a=1}^{M_W} \sum_{b=1}^{N_W} \mathbf{X}(a - m, b - n)\mathbf{W}_j(a, b) \; . \tag{2}$$

Pad $\mathbf{X}$ with zeros to define it at all points in the above double sum. Then pass the J matrices $\mathbf{C}_1, \ldots, \mathbf{C}_J$ element-wise through logistic sigmoid function s to give the hidden-neuron

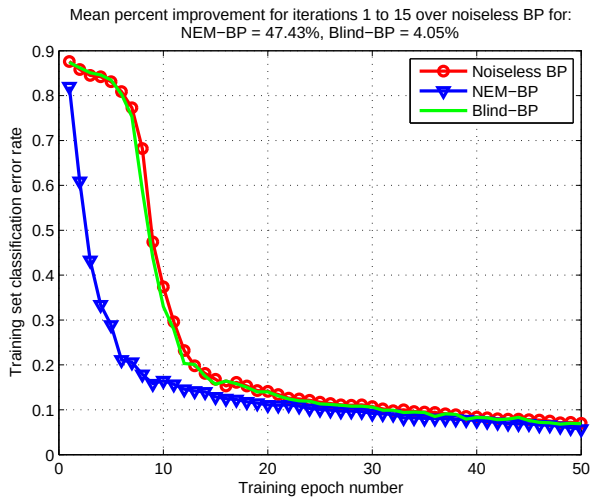

Fig. 4. NEM noise-benefit in BP training of a CNN using MNIST data: The figure shows that the NEM-BP training reduces the training-set classification error rate of the MNIST data set compared with standard noiseless BP training. We obtain a 47.43% average reduction in classification error rate for the NEM-BP case when compared with the standard BP over the first 15 training iterations. Adding blind noise gives a minor average reduction of 4.05% in classification error rate. Training used 1000 images from the MNIST data for a CNN with one convolution hidden layer. The convolutional layer used three 3×3 masks or filters. Factor-2 downsampling followed the convolutional layer by removing all even index rows and columns of the hidden neuron images. The hidden layer fully connects to 10 output neurons that predict the class label of the input digit. We used uniform noise over $[-0.5/\sqrt{t^5}, 0.5/\sqrt{t^5}]$ where t is the training iteration number for both NEM and blind noise.

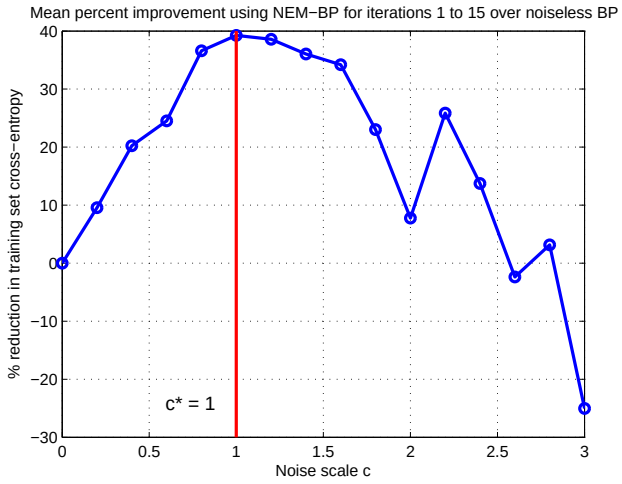

Fig. 5. NEM noise-benefit inverted U-curve for NEM-BP training of a CNN: The figure shows the mean percent reduction in per-iteration training-set cross entropy for NEM-BP training of a CNN with different uniform noise variances. We add zero mean uniform$(-0.5\sqrt{c/t^d}, 0.5\sqrt{c/t^d})$ noise where $c = 0, 0.2, \ldots, 2.8, 3$, t is the training epoch, and $d = 5$ is the noise annealing factor. The noise benefit increases when c increases from 0 to 1 and tends to decrease after 1. The optimal noise scale is $c^* = 1$. NEM noise addition hurts the training-set cross entropy when $c \geqslant 2.6$.

activations $\mathbf{Z}_j$:

$$\mathbf{Z}_j(m,n) = s(\mathbf{C}_j(m,n)) \tag{3}$$

$$= \frac{1}{1 + \exp(-\mathbf{C}_j(m,n))} \,. \tag{4}$$

Suppose the network has K output neurons. A $(M_X + M_W - 1) \times (N_X + N_Y - 1)$ weight matrix $\mathbf{U}_j^k$ multiplies the j-th hidden neuron matrix $\mathbf{Z}_j$ element-wise. The soft-max or Gibbs activation of the k-th output neuron is

$$a_k^t = \frac{\exp\left(\sum_{j=1}^{J} \mathbf{e}^T \mathbf{Z}_j \odot \mathbf{U}_j^k \mathbf{e}\right)}{\sum_{k_1=1}^{K} \exp\left(\sum_{j=1}^{J} \mathbf{e}^T \mathbf{Z}_j \odot \mathbf{U}_j^{k_1} \mathbf{e}\right)} \tag{5}$$

where $\odot$ denotes the element-wise Hadamard product between two matrices. $\mathbf{e}$ is a vector of all 1s of length $(M_X + M_W - 1)(N_X + N_W - 1)$. The JK matrices $\mathbf{U}_j^k$ ($j = 1, \ldots, J$ and $k = 1, \ldots, K$) are the weights of the connections between the hidden and output neurons. The next section presents the back-propagation training algorithm for a CNN.

3. BACK-PROPAGATION FOR CNN TRAINING

The back-propagation (BP) algorithm performs maximum likelihood (ML) estimation of the J convolution matrices $\mathbf{W}_1, \ldots, \mathbf{W}_J$ and the JK hidden-output weight matrices $\mathbf{U}_j^k$. Let $\mathbf{y}$ denote the 1-in-K encoding vector of the target label for a given input image $\mathbf{X}$. This means $y_k = 1$ when k corresponds to the correct class and 0 otherwise. BP computes the cross entropy between the soft-max activations of the output neurons and the target vector $\mathbf{y}$:

$$E(\Theta) = -\sum_{k_1=1}^{K} y_{k_1} \log(a_{k_1}^t) \tag{6}$$

where Θ denotes all the parameters of the CNN – the J convolution matrices $\mathbf{W}_1, \ldots, \mathbf{W}_J$ and the weight matrix $\mathbf{U}$. Note that $-E(\Theta)$ is the log-likelihood

$$L(\Theta) = \log(a_k^t) = -E(\Theta) \tag{7}$$

of the correct class label for the given input image. Hence the ML estimate of Θ is

$$\Theta^* = \arg\max_{\Theta} L(\Theta) \,. \tag{8}$$

BP performs gradient ascent on the log-likelihood surface $L(\Theta)$ to iteratively find the ML estimate of Θ. This also

holds when minimizing squared-error because BP is equivalent to ML estimation with a conditional Gaussian distribution [10, 26]. The estimate of Θ at the $(n+1)$-th iteration is

$$\Theta^{(n+1)} = \Theta^{(n)} - \eta \nabla_\Theta E(\Theta)\Big|_{\Theta=\Theta^{(n)}} \quad (9)$$

where η is a positive learning rate. A forward pass in BP computes the activations of all hidden and output neurons in the CNN. Back-propagating the output neuron activation errors through the network gives the gradient of the data log-likelihood function with respect to the CNN parameters. The gradient ascent in (9) updates these parameters.

The hidden neuron activations in a CNN are "latent" or unseen variables for the purposes of the EM algorithm. BP here performs ML estimation of a CNN's parameters. The EM algorithm is a popular iterative method for such ML estimation [11]. The EM algorithm uses the lower-bound Q of the log-likelihood function $L(\Theta)$:

$$Q(\Theta|\Theta^{(n)}) = \mathbb{E}_{p(\mathbf{Z}_1,\ldots,\mathbf{Z}_J|\mathbf{X},\mathbf{y},\Theta^{(n)})}\{\log p(\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\mathbf{y}|\mathbf{X},\Theta)\} \quad (10)$$

The J matrices $\mathbf{Z}_1,\ldots,\mathbf{Z}_J$ are the latent variables in the algorithm's expectation (E) step. Then the Maximization (M) step maximizes the Q-function to find the next parameter estimate

$$\Theta^{(n+1)} = \arg\max_\Theta Q(\Theta|\Theta^{(n)}) \; . \quad (11)$$

The generalized EM (GEM) algorithm performs this optimization by stochastic gradient ascent. Theorem 1 from [10] shows that BP is a special case of the GEM algorithm.

Theorem 1. Backpropagation is a special case of the GEM Algorithm [10]
The backpropagation update equation for a differentiable likelihood function $p(y|\mathbf{x},\Theta)$ at epoch n

$$\Theta^{(n+1)} = \Theta^{(n)} + \eta \nabla_\Theta \log p(y|\mathbf{x},\Theta)\Big|_{\Theta=\Theta^{(n)}} \quad (12)$$

equals the GEM update equation at epoch n

$$\Theta^{(n+1)} = \Theta^{(n)} + \eta \nabla_\Theta Q(\Theta|\Theta^{(n)})\Big|_{\Theta=\Theta^{(n)}} \quad (13)$$

where GEM uses the differentiable Q-function $Q(\Theta|\Theta^{(n)})$ in (10).

This result lets us use the noisy EM algorithm to speed up BP training of a CNN. The next section details the noisy EM algorithm.

4. NOISY EXPECTATION-MAXIMIZATION (NEM) ALGORITHM

The Noisy Expectation-Maximization (NEM) algorithm [12, 13] provably speeds up the EM algorithm. It adds noise to the data at each EM iteration. The noise decays with the iteration count to ensure convergence to the optimal parameters of the original data model. The additive noise must also satisfy the NEM condition below that ensures that the NEM parameter estimates will climb faster up the likelihood surface on average.

4.1. NEM Theorem

The NEM Theorem [12, 13] states when noise speeds up the EM algorithm's convergence to a local optimum of the likelihood surface. The NEM Theorem uses the following notation. The noise random variable $\mathbf{N}$ has pdf $p(\mathbf{n}|\mathbf{x})$. So the noise $\mathbf{N}$ can depend on the data $\mathbf{x}$. Vector $\mathbf{h}$ denotes the latent variables in the model. $\{\Theta^{(n)}\}$ is a sequence of EM estimates for Θ. $\Theta_* = \lim_{n\to\infty} \Theta^{(n)}$ is the converged EM estimate for Θ. Define the noisy Q function $Q_{\mathbf{N}}(\Theta|\Theta^{(n)}) = \mathbb{E}_{\mathbf{h}|\mathbf{x},\Theta_k}[\ln p(\mathbf{x}+\mathbf{N},\mathbf{h}|\theta)]$. Assume that the differential entropy of all random variables is finite and that the additive noise keeps the data in the support of the likelihood function. Then we can state the general NEM theorem [12, 13].

Theorem 2. Noisy Expectation Maximization (NEM)
The EM estimation iteration noise benefit

$$Q(\Theta_*|\Theta_*) - Q(\Theta^{(n)}|\Theta_*) \geqslant Q(\Theta_*|\Theta_*) - Q_N(\Theta^{(n)}|\Theta_*) \quad (14)$$

or equivalently

$$Q_N(\Theta^{(n)}|\Theta_*) \geqslant Q(\Theta^{(n)}|\Theta_*) \quad (15)$$

holds on average if the following positivity *condition holds:*

$$\mathbb{E}_{\mathbf{x},\mathbf{h},\mathbf{N}|\Theta*}\left[\ln\left(\frac{p(\mathbf{x}+\mathbf{N},\mathbf{h}|\Theta_k)}{p(\mathbf{x},\mathbf{h}|\Theta_k)}\right)\right] \geqslant 0 \; . \quad (16)$$

The NEM Theorem states that each iteration of a properly noisy EM algorithm gives higher likelihood estimates on average than do the regular EM's estimates. So the NEM algorithm converges faster than EM for a given data model. The faster NEM convergence occurs both because the likelihood function has an upper bound and because the NEM algorithm takes larger average steps up the likelihood surface. NEM also speeds up the training of hidden Markov models [27] and the K-means clustering algorithm [28] used in big-data processing [29].

5. NOISY BACKPROPAGATION FOR CNN TRAINING

We add noise to the 1-in-K encoding vector $\mathbf{y}$ of the target class label. The next theorem states the noise-benefit sufficient condition for Gibbs activation output neurons used in CNN K-class classification.

Theorem 3. Forbidden Hyperplane Noise-Benefit Condition for CNN

The NEM positivity condition holds for ML training of a CNN with Gibbs activation output neurons if

$$\mathbb{E}_{\mathbf{y},\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\mathbf{n}|\mathbf{X},\Theta*}\left\{\mathbf{n}^T \log(\mathbf{a}^t)\right\} \geqslant 0 \tag{17}$$

where the activation of the k-th output neuron is

$$a_k^t = \frac{\exp\left(\sum_{j=1}^J \mathbf{e}^T \mathbf{Z}_j \odot \mathbf{U}_j^k \mathbf{e}\right)}{\sum_{k_1=1}^K \exp\left(\sum_{j=1}^J \mathbf{e}^T \mathbf{Z}_j \odot \mathbf{U}_j^{k_1} \mathbf{e}\right)} \tag{18}$$

where $\odot$ denotes the element-wise Hadamard product between two matrices. $\mathbf{e}$ is a vector of all 1s of length $(M_X + M_W - 1)(N_X + N_W - 1)$.

Proof. Add noise to the target 1-in-K encoding vector $\mathbf{y}$ at the output neurons. Then the likelihood ratio in the NEM sufficient condition becomes

$$\frac{p(\mathbf{y}+\mathbf{n},\mathbf{Z}_1,\ldots,\mathbf{Z}_J|\mathbf{X},\Theta)}{p(\mathbf{y},\mathbf{Z}_1,\ldots,\mathbf{Z}_J|\mathbf{X},\Theta)} = \frac{p(\mathbf{y}+\mathbf{n}|\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\Theta)}{p(\mathbf{y}|\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\Theta)} \,. \tag{19}$$

The output soft-max activations $\mathbf{a}_k^t$ from (5) simplify the ratio on the right-hand side of the above equation. This gives

$$\frac{p(\mathbf{y}+\mathbf{n}|\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\Theta)}{p(\mathbf{y}|\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\Theta)} = \prod_{k=1}^K \frac{(a_k^t)^{t_k+n_k}}{(a_k^t)^{t_k}} = \prod_{k=1}^K (a_k^t)^{n_k} \,. \tag{20}$$

Substituting the above equation in (16) gives

$$\mathbb{E}_{\mathbf{y},\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\mathbf{n}|\mathbf{X},\Theta*}\left\{\log\left(\prod_{k=1}^K (a_k^t)^{n_k}\right)\right\} \geqslant 0 \tag{21}$$

or

$$\mathbb{E}_{\mathbf{y},\mathbf{Z}_1,\ldots,\mathbf{Z}_J,\mathbf{n}|\mathbf{X},\Theta*}\left\{\sum_{k=1}^K n_k \log(a_k^t)\right\} \geqslant 0 \,. \tag{22}$$

□

Figure 2 illustrates the sufficient condition in (17) for a CNN with three output neurons. All noise $\mathbf{n}$ above the hyperplane $\{\mathbf{n} : \mathbf{n}^T \log(\mathbf{a}^t) = 0\}$ speeds CNN training on average.

6. SIMULATION RESULTS

All simulations used the MNIST data set of handwritten digits. The MNIST data set contains 28×28 gray-scale pixel images with pixel intensities between 0 and 1. Figure 6 shows 20 sample images from this data set. Figure 7 shows a schematic diagram of the BP training of a CNN using images from the MNIST data set. The simulations used at least 1000 images from the MNIST training-set. We modified an open-source Matlab toolbox [30] to add noise during CNN training. The CNN contained one convolution layer with three 3×3 pixel masks each. We followed the convolution layer with factor-2 down-sampling to increase system robustness and to reduce the number of CNN parameters [2].

Data: T input images $\{\mathbf{X}_1, \ldots, \mathbf{X}_T\}$, T target label 1-in-K vectors $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, number J of convolution masks, size $M_W \times N_W$ of each convolution mask, number of BP epochs R
Result: Trained CNN weight matrices
while *epoch* $r : 1 \to R$ **do**
  while *training image number* $t : 1 \to T$ **do**
    • Compute the J hidden activation matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_J$ using (2) and (3);
    • Downsample the J hidden activation matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_J$ by a factor of 2. • Compute the K-D output soft-max activation vector $\mathbf{a}$ using (5);
    • Generate noise vector $\mathbf{n}$;
    if $\mathbf{n}^T \log(\mathbf{a}) \geqslant 0$ **then**
      • Add NEM noise: $\mathbf{y}_t \leftarrow \mathbf{y}_t + \mathbf{n}$;
    else
      • Do nothing
    end
    • Compute error $\mathbf{y}_t - \mathbf{a}$;
    • Back-propagate error to compute cross entropy gradient $\nabla_\Theta E(\Theta)$;
    • Update network parameters Θ using gradient descent in (9);
    .
  end
end

Algorithm 1: The NEM-BP Algorithm for a CNN.

The output layer neurons used the soft-max or Gibbs activation function for 10-way classification. All hidden neurons used the logistic sigmoid function. We used uniform noise over $(-0.5\sqrt{c/t^d}, 0.5\sqrt{c/t^d})$ where $c = 0, 0.2, \ldots, 3$, $d = 1, 2, \ldots, 5$, and t is the training epoch. The noise variance thus decreased to 0 as training epochs proceed. Figure 3 shows the training-set cross entropy of a CNN for three algorithms: standard noiseless BP, BP with blind noise (Blind-BP), and BP with NEM noise (NEM-BP). We obtain a 39.26% average reduction in training-set cross entropy over the first 15 iterations using NEM-BP compared with the noiseless BP. Figure 4 plots the training-set classification error rates as the CNN learns. NEM-BP gives a 47.43% reduction in training-set error rate averaged over the first 15 iterations as compared with noiseless BP. Adding blind noise (Blind-BP) gave only a minor improvement of 4.05%.

We next plot the relative average reduction in cross entropy for NEM-BP as the noise scale c varies from 0 to 3 in

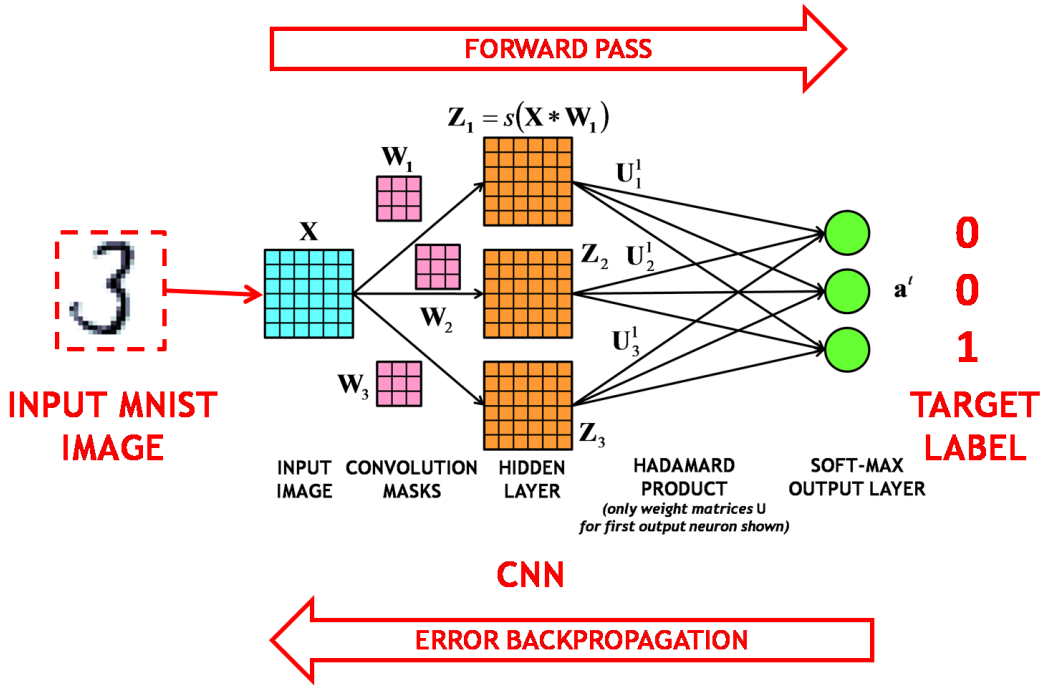

Fig. 7. CNN Training on MNIST: The figure shows a schematic diagram for training a CNN using an image from the MNIST data set. A forward pass through the CNN computes the hidden and output neuron activations. The error between the output activation vector $\mathbf{a}^t$ and the true target vector $(0, 0, 1)$ then propagates back through the CNN to compute the gradient of cross entropy. Then gradient descent updates the weights of the CNN using this gradient.

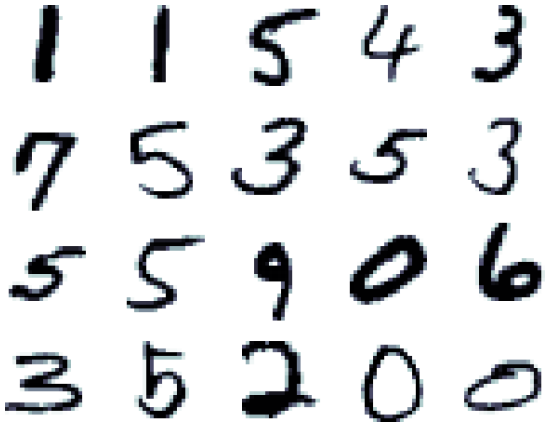

Fig. 6. MNIST Digits: The figure shows 20 sample images from the MNIST data set. Each digit is a 28×28 pixel grayscale image.

steps of 0.2. Figure 5 shows the resulting characteristic noise-benefit inverted U-curve. The optimal uniform noise scale occurs at $c^* = 1$ and NEM-BP gives a 39.26% improvement in average cross entropy. NEM noise hurts CNN training when the noise scale increases beyond 2.6.

We also explored how the training-data set size affects NEM performance. We varied the MNIST training-set size over $1000, 2000, \ldots, 5000$ and computed the relative average reduction in training cross entropy for NEM-BP using the optimal noise variance. Figure 8 shows the resulting decreasing bar chart: NEM-BP's performance falls as the number of training data samples increases. This shows that NEM-BP is especially useful when the number of training data samples is small relative to the number of estimated CNN parameters.

7. CONCLUSIONS

Proper noise injection speeds up the back-propagation (BP) training of a convolutional neural network (CNN). This follows because the BP algorithm is a special case of the EM algorithm and because the recent noisy EM (NEM) theorem gives a sufficient condition for speeding up the EM algorithm using noise. NEM noise injection experiments on the MNIST data set show substantial reduction in training-set cross entropy and classification error rate as compared with the noiseless BP algorithm. Blind noise gave at best a small noise ben-

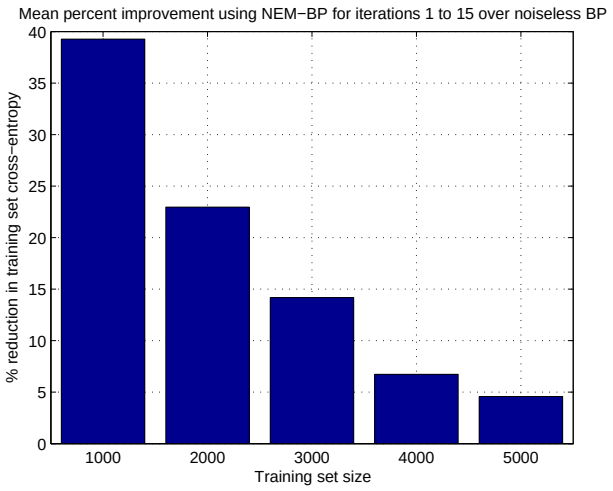

Fig. 8. Variation of NEM-BP performance benefit with increasing training-set size: The bar chart shows the relative average reduction in training-set cross entropy for NEM-BP as the training-set size increases. The noise benefit is greater for smaller training-data set sizes.

efit. Simulations show that the NEM noise benefit was largest for smaller data sets. Future work will explore adding noise to both the input data and hidden neurons.

8. REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. NIPS*, 1990.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, vol. 1, p. 4.

[5] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[6] P. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis.," in *ICDAR*, 2003, vol. 3, pp. 958–962.

[7] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata, "Pedestrian detection with convolutional neural networks," in *Proceedings of the IEEE Intelligent Vehicles Symposium*. IEEE, 2005, pp. 224–229.

[8] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. AAAI Press, 2011, vol. 2, pp. 1237–1242.

[9] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, vol. 16, no. 5, pp. 555–559, 2003.

[10] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise benefits in backpropagation and deep bidirectional pre-training," in *Proc. IJCNN*, 2013, pp. 1–8.

[11] Arthur P Dempster, Nan M Laird, and Donald B Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[12] O. Osoba, S. Mitaim, and B. Kosko, "Noise Benefits in the Expectation-Maximization Algorithm: NEM theorems and Models," in *The International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2011, pp. 3178–3183.

[13] O. Osoba, S. Mitaim, and B. Kosko, "The noisy expectation–maximization algorithm," *Fluctuation and Noise Letters*, vol. 12, no. 03, 2013.

[14] B. Kosko, *Noise*, Viking, 2006.

[15] A. Patel and B. Kosko, "Levy Noise Benefits in Neural Signal Detection," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, 2007, vol. 3, pp. III–1413 –III–1416.

[16] A. Patel and B. Kosko, "Stochastic Resonance in Continuous and Spiking Neurons with Levy Noise," *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 1993–2008, December 2008.

[17] M. Wilde and B. Kosko, "Quantum forbidden-interval theorems for stochastic resonance," *Journal of Physical A: Mathematical Theory*, vol. 42, no. 46, 2009.

[18] A. Patel and B. Kosko, "Error-probability noise benefits in threshold neural signal detection," *Neural Networks*, vol. 22, no. 5, pp. 697–706, 2009.

[19] A. Patel and B. Kosko, "Optimal Mean-Square Noise Benefits in Quantizer-Array Linear Estimation," *IEEE Signal Processing Letters*, vol. 17, no. 12, pp. 1005 – 1009, Dec. 2010.

[20] A. Patel and B. Kosko, "Noise Benefits in Quantizer-Array Correlation Detection and Watermark Decoding," *IEEE Transactions on Signal Processing*, vol. 59, no. 2, pp. 488 –505, Feb. 2011.

[21] B. Franzke and B. Kosko, "Noise Can Speed Convergence in Markov Chains," *Physical Review E*, vol. 84, no. 4, pp. 041112, 2011.

[22] A. R. Bulsara and L. Gammaitoni, "Tuning in to Noise," *Physics Today*, pp. 39–45, March 1996.

[23] L. Gammaitoni, "Stochastic Resonance in Multi-Threshold Systems," *Physics Letters A*, vol. 208, pp. 315–322, December 1995.

[24] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[25] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[26] C. M. Bishop, *Pattern recognition and machine learning*, vol. 1, Springer, 2006.

[27] K. Audhkhasi, O. Osoba, and B. Kosko, "Noisy hidden Markov models for speech recognition," in *Proc. IJCNN*, 2013, pp. 1–8.

[28] O. Osoba and B. Kosko, "Noise-Enhanced Clustering and Competitive Learning Algorithms," *Neural Networks*, Jan. 2013.

[29] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[30] Rasmus Berg Palm, "DeepLearn Toolbox," `https://github.com/rasmusbergpalm/DeepLearnToolbox`.