Supplementary: Training Better Deep Neural Networks with Uncertainty Mining Net

Yang Sun [‡2]
ys2899@columbia.edu
Abhishek Kolagunda[1]
abhishek.kolagunda@ibm.com
Steven Eliuk[1]
steven.eliuk@ibm.com
Xiaolong Wang[1]
visionxiaolong@gmail.com

[1] IBM
California, US
[2] ByteDance Ltd.
Beijing, China

1 Algorithm of UMN

Algorithm 1: UMN Algorithm

while *Traning()* **do**
  Sample a minibatch S^i from dataset;
  Draw z_a^i from $q_\phi(z_a^i|x^i)$ given $x^i \in S^i$;
  for *all data* $x^i \in S^i$*:* **do**
    $y_i \sim q_\phi(y_i|z_a^i)$
  end
  for *all labeled data* $x_L^i \in S^i$*:* **do**
    Compute $\hat{y}_i \sim p_\theta(\hat{y}_i|y, z_a^i)$ based on ε, the uncertainty estimated as the difference between predictions of Learner and Guider
  end
  Compute the remaining outputs of the generative process;
  Compute the loss $\mathcal{L}$ Eq.(21) ;
  $g_\theta = \frac{\partial \mathcal{L}}{\partial \theta}$; $g_\phi = \frac{\partial \mathcal{L}}{\partial \phi}$;
  $\theta = \theta$ - AdamUpdate(θ); $\phi = \phi$ - AdamUpdate(ϕ);
  For ϕ in classifier $y_i \sim q_\phi(y_i|x_i)$, update its moving average, the guider as $\phi_G(t) = \alpha * \phi_G(t-1) + (1-\alpha) * \phi_L(t)$
end

[‡]The author contributed to this work while employed in IBM, California, USA.

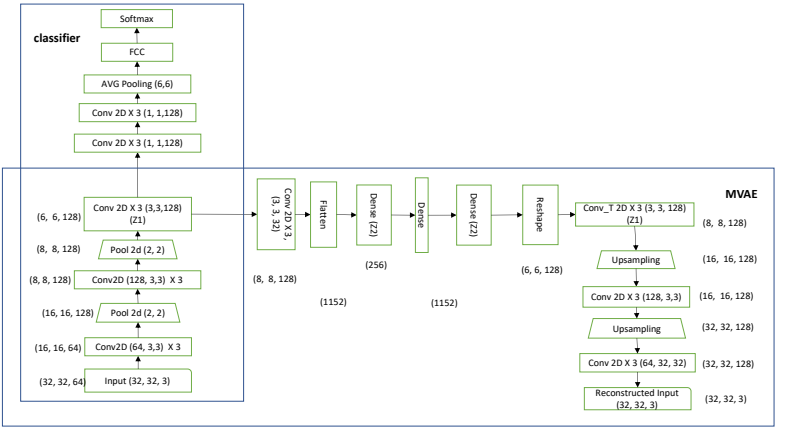

Figure 1: The network architecture used in UMN framework. It is used in experiments on SVHN and CIFAR10 datasets. The network includes two modules. The first one is the classifier which is composed of convolution, pooling, dense and softmax layers. The second module is **M1** + **M2** stacked variational autoencoder. $Z1$ and $Z2$ are the latent representation of **M1** and **M2** respectively. We use $Z1$ denote the output of the encoder from **M1**. It is used as the input to the classifier.

2 Model Architecture of UMN with Convolutional Neural Network

Details of the model architecture is listed in Table. 1. In addition, Figure. 1 demonstrates how the classifier and VAE models are setup. The VAE is a stacked autoencoder with **M1** + **M2** [2]. **M1** is a variational autoencoder built with convolution neural networks. **M2** is composed of only fully connected layers.

3 Uncertainty Estimation as Difference between the Guider and the Learner

We discuss the uncertainty estimation for a model near convergence in a convex domain [3].

Assumption 1 We assume that in the optimization problem, the stationary distribution is constrained to a convex region, where the loss function has a quadratic form:

$$\mathcal{L} = \frac{1}{2}\theta^{\top}\mathcal{H}\,\theta, \tag{1}$$

where $\mathcal{H}$ is the Hessian of the loss surface near minimum and the vector θ of the model parameters. Without loss of generality, optimal θ lies on the origin.

Table 1: Configuration of the neural networks

Layers of Classifier	**Hyperparameters**
Input	32×32 RGB
Translation (shared with MVAE)	Randomly $[\delta x, \delta y] \sim [-2, 2]$
Gaussian noise (shared with MVAE)	$\sigma = 0.15$
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Pooling (shared with MVAE)	MaxPool, (2, 2)
Dropout(shared with MVAE)	$p = 0.5$
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Pooling (shared with MVAE)	MaxPool, (2, 2)
Dropout (shared with MVAE)	$p = 0.5$
Conv2D (shared with MVAE)	filter size: (3, 3, 512), valid padding
Conv2D	filter size: (1, 1, 256), valid padding
Conv2D	filter size: (1, 1, 128), valid padding
Pooling	AvgPool, (6,6)
Fully Connected + Softmax	$128 \rightarrow 10$
Layers of UMN	**Hyperparameters**
Conv2D	filter size: (3, 3, 32), same padding
Flatten	$(8,8,32) \rightarrow 1152$
Fully Connected	$1152 \rightarrow 256$
Fully Connected	$256 \rightarrow 1152$
Reshape	$1152 \rightarrow (6,6,32)$
Conv2D Transpose	$(6,6,32) \rightarrow (8,8,128)$
Up-sampling	
Conv2D	filter size: (3, 3, 128), same padding
Conv2D	filter size: (3, 3, 128), same padding
Conv2D	filter size: (3, 3, 128), same padding
Up-sampling	
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding
Conv2D (shared with MVAE)	filter size: (3, 3, 128), same padding

3.1 The Convergence Behavior as Stochastic Process

The convergence behavior of Stochastic Gradient Descent (SGD) can be described by a stochastic differential equation as

$$d\theta = -\lambda g(\theta)dt + \frac{\lambda}{\sqrt{S}}B(\theta)d\mathbf{W}(t), \tag{2}$$

where $g(\theta)dt \equiv \mathcal{H}\theta(t)$ is the gradient for the weights and S is the mini-batch size. $d\mathbf{W}(t)$ represents the Wienner process in the stochastic gradient descent. $B(\theta)$ is introduced due to the noise in the stochastic gradient descent process. The covariance of the SGD process $\Sigma = E(\theta\theta^\top)$ satisfies $\mathcal{H}\Sigma + \Sigma\mathcal{H} = \frac{\lambda}{S}BB^\top$. λ and t represent the step size and step index respectively.

Lemma 1 *The stochastic process for parameter optimization of deep learning problem using only piecewise linear activation function with and without noisy labels can be described by the following Wienner processes,*

$$d\theta(t) = -\lambda\hat{\mathcal{H}}\theta(t)dt + \frac{\lambda}{\sqrt{S}}\hat{B}(\theta)d\mathbf{W}(t); \tag{3}$$

$$d\theta(t) = -\lambda\mathcal{H}\theta(t)dt + \frac{\lambda}{\sqrt{S}}B(\theta)d\mathbf{W}(t), \tag{4}$$

where $\mathcal{H} = \hat{\mathcal{H}}$ and $B = \hat{B}$, assuming the learning rate and batch size are the same.

The proof follows directly from a theorem (Theorem 4) from [4], stating that the curvature of the loss surface (Hessian) is invariant with respect to the noise when the neural network only uses piecewise linear function for its activation functions.

3.2 Polyak's Average for the Weights Optimization

Polyak et al. [5] proved that the iterate average gives the optimal convergence rate and approximates the optimal by using the average of the iterates online as

$$\begin{aligned} \theta_L(t+1) &= \theta_L(t) - \lambda g_L(\theta_t); \\ \mu_{t+1} &= \frac{t}{t+1}\mu_t + \frac{1}{t+1}\theta_t \end{aligned} \tag{5}$$

and the iterate average after T step is

$$\theta_G(T) = \mu_T \equiv \frac{1}{T}\int_0^T \theta_L(t)dt, \tag{6}$$

where θ_L and θ_G indicate the weights of the leaner and guider respectively.

Theorem 1 *Suppose that θ^* is the optimal solution of the optimization problem. θ_L is the stochastic random variable for SGD (learner) and θ_G is iterate average (guider) of θ_L, $\theta_G(T) = \mu_T \equiv \frac{1}{T}\int_0^T \theta_L(t)dt$. The distance between θ_L and θ^* is consistent with the distance between θ_L and θ_G in the sense that*

$$E\left[\left(\frac{\partial}{\partial\theta_L}\left[\|\theta_L - \theta^*\|^2\right]\right)^\top(\theta_G - \theta_L)\right] > 0 \tag{7}$$

which holds when the training epoch T satisfies the following relation

$$T < \frac{1}{\lambda}\frac{\mathrm{Tr}\{\mathcal{H}^{-1}\Sigma\}}{\mathrm{Tr}\{\Sigma\}}, \tag{8}$$

where λ is the constant learning rate.

We prove that the theorem holds in the presence of the noisy labels. Assuming the invariance of the Hessian (Lemma 1), the distance between the learner and the guider suggests the correctness of the labels.

Starting with

$$\begin{aligned}
&E\Big[\big(\nabla_{\theta_L}\big[\|\theta_L-\theta^*\|^2\big]\big)^\top(\theta_G-\theta_L)\Big]\\
=&2E\Big[(\theta_L^\top-\theta^*)^\top(\theta_G-\theta_L)\Big]\\
\approx&E[\theta_L^\top\theta_G]-E[\theta_L^\top\theta_L]-\theta^{*\top}\hat{\theta}^*+\theta^{*\top}\hat{\theta}^*\\
=&E[\theta_L^\top\theta_G]-E[\theta_L^\top\theta_L]
\end{aligned} \tag{9}$$

to prove the theorem, we need to show that

$$E[\theta_L^\top\theta_L] < E[\theta_L^\top\theta_G] \tag{10}$$

holds when Eq 8 is true. We denote

$$\theta_L = \bar{\theta}_S + \hat{\theta}^*; \qquad \theta_G = \bar{\theta}_T + \hat{\theta}^*$$

These are two stochastic optimization processes described above for the stochastic gradient descent and the iterate average. $\bar{\theta}_S$ and $\bar{\theta}_T$ can be seen as the deviation from the optimal point. Hence we now need to show

$$E[\bar{\theta}_S^\top\bar{\theta}_S] < E[\bar{\theta}_S^\top\bar{\theta}_T] \tag{11}$$

holds when Eq. 8 is true. The left hand side of Eq. 11 gives

$$E[\bar{\theta}_S^\top\bar{\theta}_S] = \mathrm{Tr}(\Sigma) \tag{12}$$

In order to evaluate the right hand side of the equation, we recall the Green's function for the Ornstein-Uhlenbeck process,

$$E(\theta(t)\theta^\top(s)) = \begin{cases}\Sigma e^{-\lambda\mathcal{H}(s-t)}, & \text{if} \quad t<s\\ \Sigma e^{-\lambda\mathcal{H}(t-s)}\Sigma, & \text{if} \quad t\geq s\end{cases} \tag{13}$$

Then, the right hand side of Eq. 11 becomes:

$$\begin{aligned}
E[\bar{\theta}_S^\top\bar{\theta}_T] &= \mathrm{Tr}\Big[\frac{1}{T}\int_0^T E[\bar{\theta}_S^\top(t)\bar{\theta}_S(t')]dt'\Big]\\
&= \mathrm{Tr}\Big[\frac{1}{T}\int_0^T e^{-\lambda\mathcal{H}(t-t')}\Sigma dt'\Big]\\
&= \mathrm{Tr}\Big[\frac{1}{T}U\Lambda^{-1}(\mathbf{I}-e^{-\lambda T\Lambda})U^T\Sigma\Big]\\
&\approx \mathrm{Tr}\Big[\frac{1}{\lambda T}\mathcal{H}^{-1}\Sigma\Big]
\end{aligned} \tag{14}$$

where we assume that the Hessian is full rank and its inverse has an eigen decomposition

$$\mathcal{H}^{-1} = U\Lambda^{-1}U^T \tag{15}$$

From Eq. 12 and Eq. 14, when

$$T < \frac{1}{\lambda}\frac{\mathrm{Tr}\{\mathcal{H}^{-1}\Sigma\}}{\mathrm{Tr}\{\Sigma\}} \tag{16}$$

the Eq. 7 holds.
If we make an assumption as in [1], i.e., the covariance of the local minimal is approximated by the Hessian, namely, $\mathcal{H} = \Sigma$. We observe that the last equation in the proof above becomes

$$T < \frac{1}{\lambda}\frac{\mathrm{Tr}\{\mathcal{H}^{-1}\Sigma\}}{\mathrm{Tr}\{\Sigma\}} = \frac{1}{\lambda\mathcal{L}}, \tag{17}$$

where $\mathcal{L}$ is the Laplacian.

From the theorem, the *Learner* model (θ_L) is closer to the optimal than the *Guider* model (θ_G) during the early part of training (Eq. 8). During the later stages of training, θ_L oscillates about θ_G which is converging to the optimal (θ^*). That is $E\left[\theta_L\right] = \theta_G$ and $\theta_G \approx \theta^*$. Given that the *minima* of the loss for the set of clean samples (X_{clean}) are more tightly clustered than that of the set of noisily labeled samples ($X_{corrupt}$), we can say that θ_G converges closer to the center of the *minima* of the set of clean samples (θ^*_{clean}) than that of the set of corrupt samples ($\theta^*_{corrupt}$) when training on the dataset $X_{clean} \cup X_{corrupt}$. Which indicates that in most cases the magnitude of the gradient of the loss for a sample with respect to *Guider* model is larger for noisily labeled samples than that of clean samples. That is $|\nabla_{\theta_G}\mathcal{L}_{corrupt}| > |\nabla_{\theta_G}\mathcal{L}_{clean}|$. Then, for a given sample x, the absolute difference ε between the predictions of the *Learner* and *Guider* can be an approximation of the gradient norm of the loss. That is $\varepsilon(x) \sim |\nabla_{\theta_G}\mathcal{L}_x|$. Larger the difference higher the likelihood of the label being corrupted. This is different from filtering out noisy labeled samples by applying a threshold on the confidence of the predictions, which does not take into account that the *minima* of some noisily labeled samples might be lower than that of the clean samples. Our approach builds an outlier rejection mechanism into the training process based on the gradient norm of the loss.

4 ELBO for VAE of Uncertainty Minining Net (UMN)

Let's start from the Jensen's inequality for the log probability of the observations,

$$\log P(X) \geq E_{q(z|x)} \log P(X|Z) - q(z|X)p(z) \tag{18}$$

where the observed variables are $X = \{\hat{y}, X\}$ and z is the latent variable in the VAE structure which we use in the stacked semi-supervised learning architecture. The posterior and likelihood can be factorized as

$$q(z_a, z_b, y|x, \hat{y}) = q(z_a|x)\, q(y|z_a)\, q(z_b|z_a, y)$$
$$p(x, \hat{y}|z_a, z_b, y) = p(x|z_a)\, p(\hat{y}|y, z_a).$$

The reconstruction loss on r.h.s of Eq. 18 then gives

$$\begin{aligned}
&\sum_{y\in C} E_{q_\theta(z_a|x)} p_\phi(y|z_a)\left[\log P(x|z_a) + \log P(\hat{y}|z_a, y)\right] \\
&= E_{q_\phi(z_a|x)} \log P(x|z_a) + E_{q_\phi(z_a|x)} \sum_{y_k\in C} q_\phi(y_k|z_a) \log P_\theta(\hat{y}|z_a, y_k) \\
&= E_{q_\phi(z_a|x)} \log P(x|z_a) + E_{q_\phi(z_a|x)} \sum_{y_k\in C} q_\phi(y_k|z_a) \log P_\theta(\hat{y}|y_k, z_a),
\end{aligned} \tag{19}$$

where q_ϕ and p_θ represent the encoder and decoder respectively in the variational autoencoder. These functions are modeled using deep neural networks. Note that the second term in Eq. 19 is summed over labeled samples in semi-supervised learning. The KL divergence in Eq. 18 can be represented as

$$\begin{aligned}
&-q(z_a, z_b, y|X, \hat{y}) p(z_a, z_b, y) \\
&= \sum_{y\in C} p_\theta(y|z_a) E_{q_\phi(z_a|x)} E_{q_\phi(z_b|z_a,y)} \Big[\log q_\phi(z_a|x) + \log q_\phi(y|z_a) + \log q_\phi(z_b|z_a, y) \\
&\quad - \log p(z_a|z_b, y) - \log p(z_b) - \log p(y)\Big] \\
&= -E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a)\big(q(z_b|z_a, y) p(z_b)\big) \\
&\quad - E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a)\big(q_\phi(y|z_a) p(y)\big) \\
&\quad - E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a) E_{q_\phi(z_b|z_a,y)}\big(\log p_\theta(z_a|z_b, y) - \log q_\phi(z_a|x)\big).
\end{aligned} \tag{20}$$

To summarize, the ELBO for VAE in UMN can be calculated as

$$\begin{aligned}
&-E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a)\big(q(z_b|z_a, y) p(z_b)\big) \\
&-E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a)\big(q_\phi(y|z_a) p(y)\big) \\
&-E_{q_\phi(z_a|x)} \sum_{y\in C} q_\phi(y|z_a) E_{q_\phi(z_b|z_a,y)}\big(\log p_\theta(z_a|z_b, y) - \log q_\phi(z_a|x)\big) \\
&+E_{q_\phi(z_a|x)} \log P(x|z_a) + E_{q_\phi(z_a|x)} \sum_{y_k\in C} q_\phi(y_k|z_a) \log P_\theta(\hat{y}|y_k, z_a),
\end{aligned} \tag{21}$$

where in the last term, estimations for probabilities of the observed label is calculated via the true labels and the corresponding encoded sample z_a. The loss given by ELBO is calculated by summing over all samples except the last term which is summed over the samples that have the available observed variable $\hat{y}$ (labeled samples).

5 Evolution of uncertainty estimate during training

Fig. 2 shows the progression of the uncertainty estimate ε while training on CIFAR-10 dataset with a label corruption rate of 20%. We can see that the distribution is bimodal (has 2

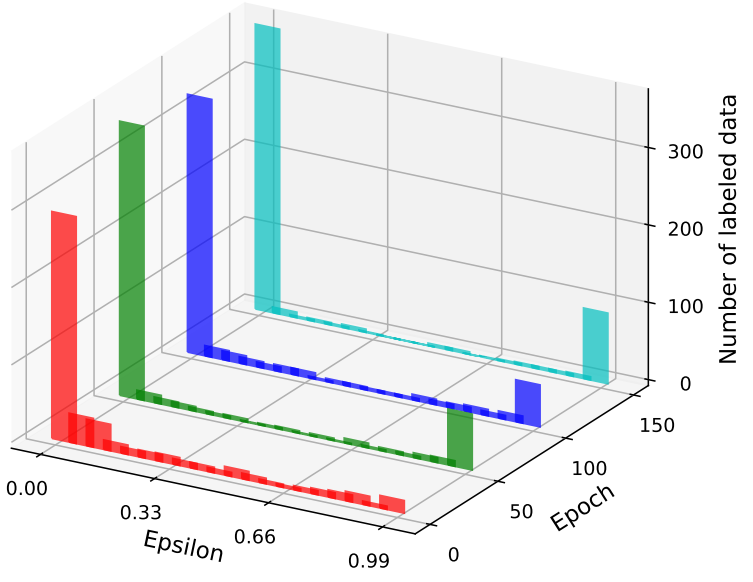

Figure 2: ε calculated at different epochs for corruption ratio = 0.2 on CIFAR-10 dataset. The plot shows the histogram of ε distribution for different training epochs.

peaks) and that the bimodal separation increases as the training progresses. i.e., our approach becomes more confident in differentiating samples with corrupt labels from samples with correct labels.

References

[1] Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in SGD. *CoRR*, abs/1711.04623, 2017. URL http://arxiv.org/abs/1711.04623.

[2] Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014. URL http://arxiv.org/abs/1406.5298.

[3] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18:134:1–134:35, 2017. URL http://jmlr.org/papers/v18/17-214.html.

[4] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, July 2017.

[5] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging.

SIAM J. Control Optim., 30(4):838–855, July 1992. ISSN 0363-0129. doi: 10.1137/0330046. URL http://dx.doi.org/10.1137/0330046.