Submitted to *INFORMS Journal on Computing*

Online Appendices for "A New Likelihood Ratio Method for Training Artificial Neural Networks"

Yijie Peng
Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing, 100871 China, pengyijie@pku.edu.cn

Li Xiao
Key Laboratory of Intelligent Information Processing, Advanced Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, xiaoli@ict.ac.cn

Bernd Heidergott
Department of Operations Analytics, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands, b.f.heidergott@vu.nl

L. Jeff Hong
Department of Management Science, School of Management, Fudan University, Shanghai, 200433, China, hong_liu@fudan.edu.cn

Henry Lam
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA, henry.lam@columbia.edu

Appendix A: Supplements for Sections 3 and 4

Proof of Theorem 1. Note that for $t<l$,

$$\frac{\partial x_i^{(t)}(n)}{\partial \theta_{a,b}^{(l)}}=0, \qquad i=1,\dots,m_t,$$

and

$$\frac{\partial \theta_{i,j}^{(t)}}{\partial \theta_{a,b}^{(l)}}=\begin{cases}1 & \text{if } i=a,\ j=b,\ t=l,\\ 0 & \text{otherwise.}\end{cases}$$

Thus,

$$\frac{\partial x_i^{(t+1)}(n)}{\partial \theta_{a,b}^{(l)}}=\varphi'\left(v_i^{(t)}(n)\right)\left(\sum_{j=0}^{m_t}\theta_{i,j}^{(t)}\frac{\partial x_j^{(t)}(n)}{\partial \theta_{a,b}^{(l)}}\right), \qquad t>l,$$

$$\frac{\partial x_a^{(l+1)}(n)}{\partial \theta_{a,b}^{(l)}}=\varphi'\left(v_a^{(t)}(n)\right)x_b^{(t)}(n), \quad \frac{\partial x_i^{(l+1)}(n)}{\partial \theta_{a,b}^{(l)}}=0, \quad i\neq a.$$

Then, we can write the IPA estimator on the left hand side of (5) as the following nested summations:

$$\sum_{i_\tau=1}^{m_\tau} e_{i_\tau}^{(\tau)}(n)\varphi'\left(v_{i_\tau}^{(\tau-1)}(n)\right)$$

$$\times \left\{ \sum_{i_{\tau-1}=1}^{m_{\tau-1}} \theta_{i_\tau,i_{\tau-1}}^{(\tau-1)} \left[\times \cdots \times \sum_{i_{l+2}=1}^{m_{l+2}} \theta_{i_{l+3},i_{l+2}}^{(l+2)} \varphi' \left(v_{i_{l+2}}^{(l+1)}(n) \right) \theta_{i_{l+2},a}^{(l+1)} \varphi' \left(v_a^{(l)}(n) \right) x_b^{(l)}(n) \right] \right\} .$$

By reversing the order of summations, we obtain

$$x_b^{(l)}(n)\varphi' \left(v_a^{(l)}(n) \right) \sum_{i_{l+2}=1}^{m_{l+2}} \theta_{i_{l+2},a}^{(l+1)} \times \left\{ \varphi' \left(v_{i_{l+2}}^{(l+1)}(n) \right) \left[\sum_{i_{l+3}=1}^{m_{l+3}} \theta_{i_{l+3},i_{l+2}}^{(l+2)} \times \cdots \times \sum_{i_\tau=1}^{m_\tau} \theta_{i_\tau,i_{\tau-1}}^{(\tau-1)} e_{i_\tau}^{(\tau)}(n) \varphi' \left(v_{i_\tau}^{(\tau-1)}(n) \right) \right] \right\},$$

which leads to the right hand side of (5) with $\delta_i^{(t)}(n)$ defined by the backwardly recursive formula (6). □

Proof of Proposition 1. The dominated convergence theorem can be used to justify the interchange of derivative and expectation in (7) (Rudin, 1987). The uniform integrability condition can be relaxed as a Lipchitz condition (Glasserman, 1991). □

Proof of Proposition 2. For $l = r$, we can write the left hand side of (9) in a nested summations:

$$\sum_{j_\tau=1}^{m_\tau} \sum_{i_\tau=1}^{m_\tau} e_{i_\tau,j_\tau}^{(\tau)}(n) \varphi' \left(v_{i_\tau}^{(\tau-1)}(n) \right) \varphi' \left(v_{j_\tau}^{(\tau-1)}(n) \right) \times \left\{ \sum_{i_{\tau-1}=1}^{m_{\tau-1}} \sum_{j_{\tau-1}=1}^{m_{\tau-1}} \theta_{i_\tau,i_{\tau-1}}^{(t)} \theta_{j_\tau,j_{\tau-1}}^{(t)} \left[\cdots \sum_{i_{l+2}=1}^{m_{l+2}} \sum_{j_{l+2}=1}^{m_{l+2}} \theta_{i_{l+3},i_{l+2}}^{(l+2)} \theta_{j_{l+3},j_{l+2}}^{(l+2)} \varphi' \left(v_{i_{l+2}}^{(l+1)}(n) \right) \varphi' \left(v_{j_{l+2}}^{(l+1)}(n) \right) \right. \right.$$
$$\left. \left. \times \theta_{i_{l+2},a}^{(l+1)} \theta_{i_{l+2},c}^{(l+1)} \varphi' \left(v_a^{(l)}(n) \right) \varphi' \left(v_c^{(l)}(n) \right) x_b^{(l)}(n) x_d^{(l)}(n) \right] \right\} .$$

By reversing the order of summations, we obtain

$$x_b^{(l)}(n) x_d^{(l)}(n) \varphi' \left(v_a^{(l)}(n) \right) \varphi' \left(v_c^{(l)}(n) \right) \sum_{i_{l+2}=1}^{m_{l+2}} \sum_{j_{l+2}=1}^{m_{l+2}} \theta_{i_{l+2},a}^{(l+1)} \theta_{i_{l+2},c}^{(l+1)} \times \left\{ \varphi' \left(v_{i_{l+2}}^{(l+1)}(n) \right) \varphi' \left(v_{j_{l+2}}^{(l+1)}(n) \right) \left[\sum_{i_{l+3}=1}^{m_{l+3}} \sum_{j_{l+3}=1}^{m_{l+3}} \theta_{i_{l+3},i_{l+2}}^{(l+2)} \theta_{j_{l+3},j_{l+2}}^{(l+2)} \right. \right.$$
$$\left. \left. \times \cdots \times \sum_{j_\tau=1}^{m_\tau} \sum_{i_\tau=1}^{m_\tau} \theta_{i_\tau,i_{\tau-1}}^{(\tau-1)} \theta_{j_\tau,j_{\tau-1}}^{(\tau-1)} e_{i_\tau,j_\tau}^{(\tau)}(n) \varphi' \left(v_{i_\tau}^{(\tau-1)}(n) \right) \varphi' \left(v_{j_\tau}^{(\tau-1)}(n) \right) \right] \right\},$$

which leads to the right hand side of (9) with $\Delta_{i,j}^{(t)}(n)$ defined by the backwardly recursive formula (10). For $l > r$, we have

$$\sum_{j_\tau=1}^{m_\tau} \sum_{i_\tau=1}^{m_\tau} e_{i_\tau,j_\tau}^{(\tau)}(n) \varphi' \left(v_{i_\tau}^{(\tau-1)}(n) \right) \varphi' \left(v_{j_\tau}^{(\tau-1)}(n) \right) \times \left\{ \sum_{i_{\tau-1}=1}^{m_{\tau-1}} \sum_{j_{\tau-1}=1}^{m_{\tau-1}} \theta_{i_\tau,i_{\tau-1}}^{(\tau-1)} \theta_{j_\tau,j_{\tau-1}}^{(\tau-1)} \left[\cdots \sum_{i_{l+2}=1}^{m_{l+2}} \sum_{j_{l+2}=1}^{m_{l+2}} \theta_{i_{l+3},i_{l+2}}^{(l+2)} \theta_{j_{l+3},j_{l+2}}^{(l+2)} \varphi' \left(v_{i_{l+2}}^{(l+1)}(n) \right) \varphi' \left(v_{j_{l+2}}^{(l+1)}(n) \right) \right. \right.$$
$$\times \sum_{j_{l+1}=1}^{m_{l+1}} \theta_{i_{l+2},a}^{(l+1)} \theta_{j_{l+2},j_{l+1}}^{(l+1)} \varphi' \left(v_a^{(l)}(n) \right) \varphi' \left(v_{j_{l+1}}^{(l)}(n) \right) x_b^{(l)}(n)$$
$$\left. \left. \times \sum_{j_l=1}^{m_l} \theta_{j_{l+1},j_l}^{(l)} \left(\times \cdots \times \sum_{j_{r+2}=1}^{m_{r+2}} \theta_{j_{r+3},j_{r+2}}^{(r+2)} \varphi' \left(v_{j_{r+2}}^{(r+1)}(n) \right) \theta_{j_{r+2},c}^{(r+1)} \varphi' \left(v_c^{(r)}(n) \right) x_d^{(r)}(n) \right) \right] \right\} .$$

By reversing the order of summations, we obtain

$$x_b^{(l)}(n)x_d^{(r)}(n)\varphi'\left(v_c^{(r)}(n)\right)\sum_{j_{r+2}=1}^{m_{r+2}}\theta_{j_{r+2},c}^{(r+1)}\varphi'\left(v_{j_{r+2}}^{(r+1)}(n)\right)\left\{\sum_{j_{r+3}=1}^{m_{r+3}}\theta_{j_{r+3},j_{r+2}}^{(r+2)}\times\cdots\times\sum_{j_{l+1}=1}^{m_{l+1}}\theta_{j_{l+1},j_l}^{(l)}\right.$$
$$\times\varphi'\left(v_a^{(l)}(n)\right)\varphi'\left(v_{j_{l+1}}^{(l)}(n)\right)\left[\sum_{j_{l+2}=1}^{m_{l+2}}\theta_{i_{l+2},a}^{(l+1)}\theta_{j_{l+2},j_{l+1}}^{(l+1)}\varphi'\left(v_{i_{l+2}}^{(l+1)}(n)\right)\varphi'\left(v_{j_{l+2}}^{(l+1)}(n)\right)\right.$$
$$\times\left(\sum_{i_{l+3}=1}^{m_{l+3}}\sum_{j_{l+3}=1}^{m_{l+3}}\theta_{i_{l+3},i_{l+2}}^{(l+2)}\theta_{j_{l+3},j_{l+2}}^{(l+2)}\times\cdots\times\sum_{j_\tau=1}^{m_\tau}\sum_{i_\tau=1}^{m_\tau}\theta_{i_\tau,i_{\tau-1}}^{(\tau-1)}\theta_{j_\tau,j_{\tau-1}}^{(\tau-1)}\right.$$
$$\left.\left.\left.\left.\times\varphi'\left(v_{i_\tau}^{(\tau-1)}(n)\right)\varphi'\left(v_{j_\tau}^{(\tau-1)}(n)\right)e_{i_\tau,j_\tau}^{(\tau)}(n)\right)\right]\right\},$$

which leads to the right hand side of (9) with $\Delta_{i,j}^{(t)}(n)$ defined by the backwardly recursive formula (10) and (11). □

Proof of Theorem 2. Notice that conditional on $X^{(l)}(n)$, $v_a^{(l)}(n)$ follows a conditional distribution with the density:

$$f_{a,l}\left(z-\mu_a^{(l)}\right)\ .$$

Then, we have

$$\frac{\partial\mathbb{E}\left[L(X^{(\tau)}(n),O(n))\right]}{\partial\theta_{a,b}^{(l)}}=\frac{\partial\,\mathbb{E}\left[\int_{\mathbb{R}}\mathbb{E}\left[L(X^{(\tau)}(n),O(n))\middle|\,z,X^{(l)}(n)\right]f_{a,l}\left(z-\mu_a^{(l)}\right)dz\right]}{\partial\theta_{a,b}^{(l)}}$$
$$=\mathbb{E}\left[-\int_{\mathbb{R}}\mathbb{E}\left[L(X^{(\tau)}(n),O(n))\middle|\,z,X^{(l)}(n)\right]x_b^{(l)}(n)\ f_{a,l}'\left(z-\mu_a^{(l)}\right)dz\right]$$
$$=\mathbb{E}\left[-\int_{\mathbb{R}}\mathbb{E}\left[L(X^{(\tau)}(n),O(n))x_b^{(l)}(n)f_{a,l}'(z)\middle|\,z+\mu_a^{(l)},X^{(l)}(n)\right]dz\right]$$
$$=\mathbb{E}\left[-\mathbb{E}\left[L(X^{(\tau)}(n),O(n))x_b^{(l)}(n)\frac{f_{a,l}'\left(z_a^{(l)}(n)\right)}{f_{a,l}\left(z_a^{(l)}(n)\right)}\middle|\,z_a^{(l)}(n),X^{(l)}(n)\right]\right]$$
$$=\mathbb{E}\left[L(X^{(\tau)}(n),O(n))\ \omega_{a,b}^{(l)}(n)\right]\ .$$

The third equality is justified by the dominated convergence theorem using the conditions in the proposition, the fourth equality holds by the change of variable, and the rest of the equalities come from definitions and iterative property of the conditional expectation. □

Proof of Proposition 3. The proof is essentially a repeat of that in Theorem 2. That is to say we first push the parameter $\theta_{a,b}^{(l)}$ into the density by a change of variable, then differentiate the density by changing the differentiation with the integration justified by the condition, and lastly change the variable back to the original one. □

Appendix B: Gradient Estimator for Noise Variance

The processing noise adds an extra level of generality to the ANN. Let $z_i^{(t)}(n)=\sigma_{i,t}\varepsilon_i^{(t)}(n)$, where $\varepsilon_i^{(t)}(n)$, $t=1,\ldots,\tau$, $i=1,\ldots,m_t$, are i.i.d. standard normal random variables. We can tune the standard deviation of the noises as extra parameters to find an optimal fit between the output of the ANN and the observation. The IPA estimator for the sensitivity with respect to $\sigma_{a,l}$ is

$$\sum_{i=1}^{m_\tau}\left.\frac{\partial L(x,O(n))}{\partial x_i}\right|_{x=X^{(\tau)}(n)}\frac{\partial x_i^{(\tau)}(n)}{\partial\sigma_{a,l}},\tag{A.1}$$

where

$$\frac{\partial x_i^{(t+1)}(n)}{\partial \sigma_{a,l}} = \varphi'\left(v_i^{(t)}(n)\right)\frac{\partial v_i^{(t)}(n)}{\partial \sigma_{a,l}}, \quad \frac{\partial v_i^{(t)}(n)}{\partial \sigma_{a,l}} = \sum_{j=1}^{m_t}\theta_{i,j}^{(t)}\frac{\partial x_j^{(t)}(n)}{\partial \sigma_{a,l}} + \frac{\partial \sigma_{i,t}}{\partial \sigma_{a,l}}\varepsilon_i^{(t)}(n) \ .$$

The IPA estimator for the derivative with respect to $\sigma_{i,t}$, $t=1,\ldots,\tau$, $i=1,\ldots,m_t$ can be calculated as a byproduct of the BP method in Theorem 1.

Proposition A. 1 *Assume the activation function φ and the loss function L are differentiable. We have*

$$\frac{\partial L(X_\theta^{(\tau)}(n), O(n))}{\partial \sigma_{a,l}} = \frac{1}{\sigma_{a,l}}\delta_a^{(l+1)}(n) z_a^{(l)}(n) \ .$$

Proof. Notice that

$$\frac{\partial v_i^{(t)}(n)}{\partial \sigma_{a,l}} = \varepsilon_a^{(l)}(n), \quad i=a,\ t=l,$$
$$\frac{\partial v_i^{(t)}(n)}{\partial \sigma_{a,l}} = \sum_{j=1}^{m_t}\theta_{i,j}^{(t)}\frac{\partial x_j^{(t)}(n)}{\partial \sigma_{a,l}}, \quad i\neq a \text{ or } t\neq l.$$

Then, we can write the IPA estimator on the left hand side of (A.1) as the following nested summations:

$$\sum_{i_\tau=1}^{m_\tau} e_{i_\tau}^{(\tau)}(n)\varphi'\left(v_{i_\tau}^{(\tau-1)}(n)\right)$$
$$\times\left\{\sum_{i_{\tau-1}=1}^{m_{\tau-1}}\theta_{i_\tau,i_{\tau-1}}^{(\tau-1)}\left[\times\cdots\times\left[\sum_{i_{l+1}=1}^{m_{l+1}}\theta_{i_{l+2},i_{l+1}}^{(l+1)}\varphi'\left(v_{i_{l+1}}^{(l+1)}(n)\right)\left(\sum_{i_l=1}^{m_l}\theta_{i_{l+1},i_l}^{(l)}\varphi'\left(v_{i_l}^{(l)}(n)\right)\varepsilon_{i_l}^{(l)}(n)\right)\right]\right]\right\} \ .$$

By reversing the order of summations, we obtain

$$\sum_{i_l=1}^{m_l}\varepsilon_{i_l}^{(l)}(n)\varphi'\left(v_{i_l}^{(l)}(n)\right)\theta_{i_{l+1},i_l}^{(l)}\left\{\sum_{i_{l+1}=1}^{m_{l+1}}\theta_{i_{l+2},i_{l+1}}^{(l+1)}\varphi'\left(v_{i_{l+1}}^{(l+1)}(n)\right)\left[\times\cdots\times\sum_{i_\tau=1}^{m_\tau}\theta_{i_\tau,i_{\tau-1}}^{(\tau-1)}e_{i_\tau}^{(\tau)}(n)\varphi'\left(v_{i_\tau}^{(\tau-1)}(n)\right)\right]\right\}$$

which leads to the right hand side of (A.1) with $\delta_i^{(t)}(n)$ defined by the backwardly recursive formula (6). □

The LR estimator for the derivative with respect to $\sigma_{a,l}$ is

$$L(X^{(\tau)}(n), O(n))\frac{1}{\sigma_{a,l}}\left[\left(\frac{z_a^{(l)}(n)}{\sigma_{a,l}}\right)^2 - 1\right] \ .$$

Suppose the noises have a common standard deviation σ. Then, the IPA estimator and and LR estimator are given respectively by

$$\sum_{t=1}^{\tau-1}\sum_{i=1}^{m_t}\frac{1}{\sigma}\delta_i^{(t)}(n)z_i^{(t)}(n),$$

and

$$L(X^{(\tau)}(n), O(n))\sum_{t=1}^{\tau-1}\sum_{i=1}^{m_t}\frac{1}{\sigma}\left[\left(\frac{z_i^{(t)}(n)}{\sigma}\right)^2 - 1\right] \ .$$

Appendix C: Variants of LR

We discuss the implementation of MVD. For the normally distributed noise, we have

$$f'_{i,t}(z) = C(f^+_{i,t}(z) - f^-_{i,t}(z)),$$

where $C = 1/\sqrt{2\pi\sigma^2_{i,t}}$, and

$$f^+_{i,t}(z) = \frac{2}{\sqrt{2}\sigma_{i,t}} \frac{z}{\sqrt{2}\sigma_{i,t}} \exp\left(-\frac{z^2}{2\sigma^2_{i,t}}\right) \mathbf{1}\{z \geq 0\}, \quad f^-_{i,t}(z) = -f^+_{i,t}(-z).$$

Then, an unbiased MDV estimator for the first order derivative with respect to $\theta^{(l)}_{a,b}$ can be given by

$$Cx^{(l)}_b(n)\left(L(X^{(\tau+)}_{[a,l]}(n), O(n)) - L(X^{(\tau-)}_{[a,l]}(n), O(n))\right),$$

where $X^{(\tau+)}_{[a,l]}(n)$ can be simulated by (1) except that $z^{(l)}_a(n)$ follows a Weibull distribution with the density given by $f^+_{a,l}(z)$, and $X^{(\tau-)}_{[a,l]}(n)$ can be simulated by (1) except that $-z^{(l)}_a(n)$ follows a Weibull distribution with the density given by $f^+_{a,l}(z)$. In general, the MVD estimator requires simulating two sampling paths for estimating the derivative with respect to one scalar parameter. Fortunately, MVD requires simulating only one sample path in the case of normally distributed noise, because $X^{(\tau+)}_{[a,l]}(n)$ and $X^{(\tau-)}_{[a,l]}(n)$ can share the common Weibull distributed noise $z^{(l)}_a(n)$. This advantage is, however, counterbalanced by the fact that the MVD requires simulating a sample path for estimating derivatives with respect to parameters in each one of the neurons.

Appendix D: A Hybrid between Push-out LR and BP

Now we consider the case where the noise is added only to the input data. Suppose that we add noises $Z(n) = (z_1(n), \ldots, z_{m_1}(n))$ to each component of the input data $X^{(1)}(n) = (x^{(1)}_1(n), \ldots, x^{(1)}_{m_1}(n))$. If the activation function φ is twice differentiable, then we can use the GLR estimator. Suppose $m_\tau \leq m_1$, which is true in almost all practical problems, and f is the joint density of $X^{(1)}(n) + Z(n)$. Notice that $(x^{(t)}_1(n), \ldots, x^{(t)}_{m_t}(n))$, $t = 1, \ldots, \tau$, are functions of the input random variables. Then, we have

$$\frac{\partial x^{(t+1)}_i(n)}{\partial x^{(1)}_a(n)} = \varphi'\left(v^{(t)}_i(n)\right) \frac{\partial v^{(t)}_i(n)}{\partial x^{(1)}_a(n)}, \qquad \frac{\partial v^{(t)}_i(n)}{\partial x^{(1)}_a(n)} = \sum_{j=0}^{m_t} \theta^{(t)}_{i,j} \frac{\partial x^{(t)}_j(n)}{\partial x^{(1)}_a(n)},$$

$$\frac{\partial^2 x^{(t+1)}_i(n)}{\partial x^{(1)}_a(n) \partial x^{(1)}_b(n)} = \varphi'\left(v^{(t)}_i(n)\right) \frac{\partial^2 v^{(t)}_i(n)}{\partial x^{(1)}_a(n) \partial x^{(1)}_b(n)} + \varphi''\left(v^{(t)}_i(n)\right) \frac{\partial v^{(t)}_i(n)}{\partial x^{(1)}_a(n)} \frac{\partial v^{(t)}_i(n)}{\partial \partial x^{(1)}_b(n)},$$
$$\frac{\partial^2 v^{(t)}_i(n)}{\partial x^{(1)}_a(n) \partial x^{(1)}_b(n)} = \sum_{j=0}^{m_t} \theta^{(t)}_{i,j} \frac{\partial^2 x^{(t)}_j(n)}{\partial x^{(1)}_a(n) \partial x^{(1)}_b(n)},$$

and

$$\frac{\partial^2 x^{(t+1)}_i(n)}{\partial x^{(1)}_a(n) \partial \theta^{(r)}_{c,d}} = \varphi'\left(v^{(t)}_i(n)\right) \frac{\partial^2 v^{(t)}_i(n)}{\partial x^{(1)}_a(n) \partial \theta^{(r)}_{c,d}} + \varphi''\left(v^{(t)}_i(n)\right) \frac{\partial v^{(t)}_i(n)}{\partial x^{(1)}_a(n)} \frac{\partial v^{(t)}_i(n)}{\partial \theta^{(r)}_{c,d}},$$
$$\frac{\partial^2 v^{(t)}_i(n)}{\partial x^{(1)}_a(n) \partial x^{(1)}_b(n)} = \sum_{j=0}^{m_t} \left(\frac{\partial \theta^{(t)}_{i,j}}{\partial \theta^{(r)}_{c,d}} \frac{\partial x^{(t)}_j(n)}{\partial x^{(1)}_a(n)} + \theta^{(t)}_{i,j} \frac{\partial^2 x^{(t)}_j(n)}{\partial x^{(1)}_a(n) \partial \theta^{(r)}_{c,d}}\right).$$

Define

$$J(n) := \begin{pmatrix} \frac{\partial x_1^{(\tau)}(n)}{\partial x_1^{(1)}(n)} & \frac{\partial x_2^{(\tau)}(n)}{\partial x_1^{(1)}(n)} & \cdots & \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial x_1^{(\tau)}(n)} \\ \frac{\partial x_1^{(\tau)}(n)}{\partial x_2^{(1)}(n)} & \frac{\partial x_2^{(\tau)}(n)}{\partial x_2^{(1)}(n)} & \cdots & \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial x_2^{(1)}(n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_1^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)} & \frac{\partial x_2^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)} & \cdots & \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)} \end{pmatrix},$$

$$D_i(n) := \begin{pmatrix} \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial x_i^{(1)}(n)} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial x_i^{(1)}(n)} & \cdots & \frac{\partial^2 x_{m_\tau}^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial x_i^{(1)}(n)} \\ \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial x_i^{(1)}(n)} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial x_i^{(1)}(n)} & \cdots & \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial x_i^{(1)}(n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial x_i^{(1)}(n)} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial x_i^{(1)}(n)} & \cdots & \frac{\partial^2 x_{m_\tau}^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial x_i^{(1)}(n)} \end{pmatrix},$$

$$D_{c,d}^{(r)}(n) := \begin{pmatrix} \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \cdots & \frac{\partial^2 x_{m_\tau}^{(\tau)}(n)}{\partial x_1^{(1)}(n)\partial \theta_{c,d}^{(r)}} \\ \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \cdots & \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial x_2^{(1)}(n)\partial \theta_{c,d}^{(r)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 x_1^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \frac{\partial^2 x_2^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial \theta_{c,d}^{(r)}} & \cdots & \frac{\partial^2 x_{m_\tau}^{(\tau)}(n)}{\partial x_{m_\tau}^{(1)}(n)\partial \theta_{c,d}^{(r)}} \end{pmatrix},$$

and

$$\Lambda_{c,d}^{(r)}(n) := \left(\frac{\partial x_1^{(\tau)}(n)}{\partial \theta_{c,d}^{(r)}}, \ldots, \frac{\partial x_{m_\tau}^{(\tau)}(n)}{\partial \theta_{c,d}^{(r)}} \right),$$
$$\Gamma(n) := \left(\frac{\partial \log f(x)}{\partial x_1}, \ldots, \frac{\partial \log f(x)}{\partial x_{m_1}} \right)\Big|_{x=X^{(1)}(n)+Z(n)},$$

where $x := (x_1, \ldots, x_{m_1})$. Under appropriate regularity conditions in Peng et al. (2018), we have

$$\frac{\partial \mathcal{E}(\theta)}{\partial \theta_{c,d}^{(r)}} = \mathbb{E}\left[L(X^{(\tau)}(n), O(n))\ \tilde{\omega}_{c,d}^{(r)}(n) \right],$$

where

$$\tilde{\omega}_{c,d}^{(r)}(n) := \sum_{i=1}^{m_1} \left(J^{-1}(n)\ D_i(n)\ J^{-1}(n)\ \mathbf{1}_i \right)^T \Lambda_{c,d}^{(r)}(n) - trace\left(J^{-1}(n)\ D_{c,d}^{(r)}(n) \right) - \left(\Lambda_{c,d}^{(r)}(n) \right)^T J^{-1}(n)\ \Gamma(n),$$

and $\mathbf{1}_i$ is the ith unit vector of dimension m_τ. We call $L(X^{(\tau)}(n), O(n))\ \tilde{\omega}_{c,d}^{(r)}(n)$ the GLR estimator. Proposition 2 presents the derivatives needed in GLR in recursive algorithms so that the computational complexities for calculating the derivatives can be analyzed.

Proposition A. 2 *For $t = 1, \ldots, \tau - 1$, we have*

$$\frac{\partial v_i^{(t)}(n)}{\partial x_a^{(1)}(n)} = \tilde{\eta}_{i,a}^{(t)}(n),$$

where

$$\tilde{\eta}_{j,a}^{(\ell)}(n) := \begin{cases} \theta_{j,a}^{(1)}, & \text{if } \ell = 1, \\ \sum_{j'=1}^{m_\ell} \theta_{j,j'}^{(\ell)} \varphi'\left(v_j^{(\ell-1)}(n) \right) \tilde{\eta}_{j',a}^{(\ell-1)}(n), & \text{if } 1 < \ell < t, \end{cases}$$

and

$$\frac{\partial v_i^{(t)}(n)}{\partial \theta_{c,d}^{(r)}} = \begin{cases} \tilde{\delta}_{i,c}^{(t,r+1)}(n)\varphi'\left(v_c^{(r)}(n)\right) x_d^{(r)}(n), & \text{if } t > r, \\ x_d^{(r)}(n), & \text{if } t = r,\ i = c, \\ 0, & \text{otherwise}, \end{cases}$$

where

$$\tilde{\delta}_{i,j}^{(t,\ell)}(n) := \begin{cases} \theta_{i,j}^{(t)}, & \text{if } \ell = t, \\ \sum_{j'=1}^{m_\ell} \theta_{j',j}^{(\ell)}\varphi'\left(v_{j'}^{(\ell)}(n)\right)\tilde{\delta}_{i,j'}^{(t,\ell+1)}(n), & \text{if } \ell < t; \end{cases}$$

in addition,

$$\frac{\partial^2 x_i^{(\tau)}(n)}{\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)} = \sum_{t=1}^{\tau-1}\sum_{j=1}^{m_{t+1}} \widetilde{\Delta}_{i,j}^{(t)}(n)\varphi''\left(v_j^{(t)}(n)\right)\tilde{\eta}_{j,a}^{(t)}(n)\tilde{\eta}_{j,b}^{(t)}(n),$$

where

$$\widetilde{\Delta}_{i,j}^{(t)}(n) := \begin{cases} 1, & \text{if } t = \tau-1,\ j = i\ , \\ 0, & \text{if } t = \tau-1,\ j \neq i\ , \\ \sum_{j'=1}^{m_{t+1}} \theta_{j',j}^{(t)}\varphi'\left(v_{j'}^{(t)}(n)\right)\widetilde{\Delta}_{i,j'}^{(t+1)}(n), & \text{if } t < \tau-1, \end{cases}$$

and

$$\begin{aligned}\frac{\partial^2 x_i^{(\tau)}(n)}{\partial x_a^{(1)}(n)\partial \theta_{c,d}^{(r)}} =& \sum_{t=r}^{\tau-1}\sum_{j=1}^{m_{t+1}} \widetilde{\Delta}_{i,j}^{(t)}(n)\varphi''\left(v_j^{(t)}(n)\right)\tilde{\eta}_{j,a}^{(t)}(n)\frac{\partial v_j^{(t)}(n)}{\partial \theta_{c,d}^{(r)}} \\ &+ \widetilde{\Delta}_{i,c}^{(r+1)}(n)\varphi'\left(v_c^{(r)}(n)\right)\varphi'\left(v_d^{(r-1)}(n)\right)\tilde{\eta}_{d,a}^{(r-1)}(n)\ .\end{aligned}$$

Proof of Proposition A. 2. For $t = 1, \ldots, \tau-1$, we have

$$\frac{\partial v_i^{(t)}(n)}{\partial x_a^{(1)}(n)} = \sum_{j_t=1}^{m_t} \theta_{i,j_t}^{(t)}\varphi'\left(v_{j_t}^{(t-1)}(n)\right)\cdots\sum_{j_2=1}^{m_2}\theta_{j_3,j_2}^{(2)}\varphi'\left(v_{j_2}^{(1)}(n)\right)\theta_{j_2,a}^{(1)}\ .$$

and

$$\frac{\partial v_i^{(t)}(n)}{\partial \theta_{c,d}^{(r)}} = \sum_{j_t=1}^{m_t} \theta_{i,j_t}^{(t)}\varphi'\left(v_{j_t}^{(t-1)}(n)\right)\cdots\sum_{j_{r+2}=1}^{m_{r+2}}\theta_{j_{r+3},j_{r+2}}^{(r+2)}\varphi'\left(v_{j_{r+2}}^{(r+1)}(n)\right)\theta_{j_{r+2},c}^{(r+1)}\varphi'\left(v_c^{(r)}(n)\right)x_d^{(r)}(n).$$

In addition,

$$\begin{aligned}&\frac{\partial^2 x_i^{(\tau)}(n)}{\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)} = \varphi'\left(v_i^{(\tau-1)}(n)\right)\frac{\partial^2 v_i^{(\tau-1)}(n)}{\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)} + \varphi''\left(v_i^{(\tau-1)}(n)\right)\frac{\partial v_i^{(\tau-1)}(n)}{\partial x_a^{(1)}(n)}\frac{\partial v_i^{(\tau-1)}(n)}{\partial x_b^{(1)}(n)} \\ =&\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_\tau=1}^{m_\tau}\theta_{i,j_{\tau-1}}^{(\tau-1)}\frac{\partial^2 x_{j_t}^{(\tau-1)}(n)}{\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)} + \varphi''\left(v_i^{(\tau-1)}(n)\right)\tilde{\eta}_{i,a}^{(\tau-1)}(n)\tilde{\eta}_{i,b}^{(\tau-1)}(n) \\ =&\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=1}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-1)}\left\{\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\sum_{j_{\tau-2}=1}^{m_{\tau-2}}\theta_{i,j_{\tau-2}}^{(\tau-2)}\frac{\partial^2 x_{j_{\tau-2}}^{(\tau-2)}(n)}{\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)}\right. \\ &\left.+\varphi''\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\tilde{\eta}_{i,a}^{(\tau-2)}(n)\tilde{\eta}_{i,b}^{(\tau-2)}(n)\right\} + \varphi''\left(v_i^{(\tau-1)}(n)\right)\tilde{\eta}_{i,a}^{(\tau-1)}(n)\tilde{\eta}_{i,b}^{(\tau-1)}(n) \\ =&\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=1}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-1)}\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\cdots\sum_{j_3=1}^{m_3}\theta_{j_4,j_3}^{(3)}\varphi'\left(v_{j_3}^{(2)}(n)\right) \\ &\times\sum_{j_2=1}^{m_2}\theta_{j_3,j_2}^{(2)}\varphi''\left(v_{j_2}^{(1)}(n)\right)\tilde{\eta}_{j_2,a}^{(1)}(n)\tilde{\eta}_{j_2,b}^{(1)}(n) \\ &+\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=1}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-1)}\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\cdots\sum_{j_4=1}^{m_4}\theta_{j_5,j_4}^{(4)}\varphi'\left(v_{j_4}^{(3)}(n)\right) \\ &\times\sum_{j_3=1}^{m_3}\theta_{j_4,j_3}^{(3)}\varphi''\left(v_{j_3}^{(2)}(n)\right)\tilde{\eta}_{j_3,a}^{(2)}(n)\tilde{\eta}_{j_3,b}^{(2)}(n) + \cdots + \varphi''\left(v_i^{(\tau-1)}(n)\right)\tilde{\eta}_{i,a}^{(\tau-1)}(n)\tilde{\eta}_{i,b}^{(\tau-1)}(n),\end{aligned}$$

which yields the third conclusion of the proposition by reversing the order of summations. Similarly, we have

$$\begin{aligned}
&\frac{\partial^2 x_i^{(\tau)}(n)}{\partial x_a^{(1)}(n)\partial\theta_{c,d}^{(r)}} = \varphi'\left(v_i^{(\tau-1)}(n)\right)\frac{\partial^2 v_i^{(\tau-1)}(n)}{\partial x_a^{(1)}(n)\partial\theta_{c,d}^{(r)}} + \varphi''\left(v_i^{(\tau-1)}(n)\right)\frac{\partial v_i^{(\tau-1)}(n)}{\partial x_a^{(1)}(n)}\frac{\partial v_i^{(\tau-1)}(n)}{\partial\theta_{c,d}^{(r)}}\\
&= \varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=1}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-1)}\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\\
&\quad\times\cdots\times\sum_{j_{r+2}=1}^{m_{r_2}}\theta_{j_{r+3},j_{r+2}}^{(r+2)}\varphi'\left(v_{j_{r+2}}^{(r+1)}(n)\right)\theta_{j_{r+2},c}^{(r+1)}\varphi'\left(v_c^{(r)}(n)\right)\varphi'\left(v_d^{(r-1)}(n)\right)\tilde{\eta}_{d,a}^{(r-1)}(n)\\
&\quad+\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=0}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-1)}\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\cdots\sum_{j_{r+2}=1}^{m_{r_2}}\theta_{j_{r+3},j_{r+2}}^{(r+2)}\varphi'\left(v_{j_{r+2}}^{(r+1)}(n)\right)\\
&\quad\times\sum_{j_{r+1}=1}^{m_{r+1}}\theta_{j_{r+2},j_{r+1}}^{(r+1)}\varphi''\left(v_{j_{r+1}}^{(r)}(n)\right)\tilde{\eta}_{j_{r+1},a}^{(r)}(n)\frac{\partial v_{j_{r+1}}^{(r)}(n)}{\partial\theta_{c,d}^{(r)}}\\
&\quad+\varphi'\left(v_i^{(\tau-1)}(n)\right)\sum_{j_{\tau-1}=1}^{m_{\tau-1}}\theta_{i,j_{\tau-1}}^{(\tau-2)}\varphi'\left(v_{j_{\tau-1}}^{(\tau-2)}(n)\right)\cdots\sum_{j_{r+3}=1}^{m_{r+3}}\theta_{j_{r+4},j_{r+3}}^{(r+3)}\varphi'\left(v_{j_{r+3}}^{(r+2)}(n)\right)\\
&\quad\times\sum_{j_{r+2}=1}^{m_{r+2}}\theta_{j_{r+3},j_{r+2}}^{(r+2)}\varphi''\left(v_{j_{r+2}}^{(r+1)}(n)\right)\tilde{\eta}_{j_{r+2},a}^{(r+1)}(n)\frac{\partial v_{j_{r+2}}^{(r+1)}(n)}{\partial\theta_{c,d}^{(r)}}\\
&\quad+\cdots+\varphi''\left(v_i^{(\tau-1)}(n)\right)\tilde{\eta}_{i,a}^{(\tau-1)}(n)\frac{\partial v_i^{(\tau-1)}(n)}{\partial\theta_{c,d}^{(r)}},
\end{aligned}$$

which yields the third conclusion of the proposition by reversing the order of summations. □

If the number of neurons at each level is of order $O(m)$, the computational complexity for calculating all $\tilde{\eta}_{i,a}^{(t)}(n)$, $a=1,\ldots,m_1$, $i=1,\ldots,m_t$, $t=1,\ldots,\tau-1$, is of order $O(m^3\tau)$, the computational complexity for calculating all $\tilde{\delta}_{i,j}^{(t,\ell)}(n)$, $i=1,\ldots,m_t$, $j=1,\ldots,m_{\ell-1}$, $\ell=1,\ldots,t$, $t=1,\ldots,\tau-1$, is of order $O(m^4\tau^2)$, the computational complexity for calculating all $\widetilde{\Delta}_{i,j}^{(t)}(n)$, $i=1,\ldots,m_\tau$, $j=1,\ldots,m_{t+1}$, $t=1,\ldots,\tau-1$, is of order $O(m^3\tau)$, the computational complexity for calculating all $\partial^2 x_i^{(\tau)}(n)/\partial x_a^{(1)}(n)\partial x_b^{(1)}(n)$, $i=1,\ldots,m_\tau$, $a,b=1,\ldots,m_1$, is of order $O(m^4\tau^2)$, and the computational complexity for calculating all $\partial^2 x_i^{(\tau)}(n)/\partial x_a^{(1)}(n)\partial\theta_{c,d}^{(r)}$, $i=1,\ldots,m_\tau$, $a=1,\ldots,m_1$, $c=1,\ldots,m_{r+1}$, $d=1,\ldots,m_r$, $r=1,\ldots,\tau-1$, is of order $O(m^5\tau^3)$.

By the same token, for $l\geq r$,

$$\frac{\partial^2 x_i^{(\tau)}(n)}{\partial\theta_{a,b}^{(l)}\partial\theta_{c,d}^{(r)}} = \sum_{t=l}^{\tau-1}\sum_{j=1}^{m_{t+1}}\widetilde{\Delta}_{i,j}^{(t)}(n)\varphi''\left(v_j^{(t)}(n)\right)\frac{\partial v_j^{(t)}(n)}{\partial\theta_{a,b}^{(l)}}\frac{\partial v_j^{(t)}(n)}{\partial\theta_{c,d}^{(r)}} + \widetilde{\Delta}_{i,a}^{(l+1)}(n)\varphi'\left(v_a^{(l)}(n)\right)\frac{\partial x_b^{(l)}(n)}{\partial\theta_{c,d}^{(r)}}.$$

With the recursive formula above, we can estimate the second term of (9). If the number of neurons in the each level is of order $O(m)$, the computational complexity for calculating all $\partial^2 x_i^{(\tau)}(n)/\partial\theta_{a,b}^{(l)}\partial\theta_{c,d}^{(r)}$, $i=1,\ldots,m_\tau$, $a=1,\ldots,m_{l+1}$, $b=1,\ldots,m_l$, $c=1,\ldots,m_{r+1}$, $d=1,\ldots,m_r$, $l,r=1,\ldots,\tau-1$, is of order $O(m^5\tau^3)$.

Appendix E: Additional Numerical Results

E.1. Supplements of Numerical Experiments in Section 4

We test the performance of Adagrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2015) with BP as the gradient estimation technique for training ANNs under adversarial attacks in MINST dataset. In Table A1, we can see that although the decline in accuracy of Adagrad and Adam is less than that of SGD under

	Orig	**Adv_L_BFGS**	**Adv_FGSM**
SGD	96.3%	57.3%	28.4%
Adagrad	93.1%	66.4%	25.0%
Adam	95.6%	67.1%	33.7%

Table A1 Adversarial tests for ANNs with Sigmoid activation function and cross-entropy loss function trained by BP using SGD, Adagrad, and Adam in identifying handwritten number images of MNIST.

	Orig	**Adv_L_BFGS**	**Adv_FGSM**
BP	96.3%	57.3%	28.4%
meProp-1	91.5%	74.7%	11.1%
meProp-5	95.1%	76.1%	10.8%
meProp-10	95.7%	76.1%	10.1%

Table A2 Adversarial tests for ANNs with Sigmoid activation function and cross-entropy loss function trained by BP and meProp in identifying handwritten number images of MNIST.

Activations + Entropy	**Orig**	**Adv_L_BFGS**	**Adv_FGSM**
BP benchmark	96.3%	57.3%	28.4%
Sigmoid	95.1%	77.3%	37.7%
Threshold	96.3%	77.9%	39.8%

Table A3 Adversarial tests for ANNs with Laplace noises trained by LR including BP benchmark in identifying handwritten number images of MNIST.

adversarial attacks, the performance of Adagrad and Adam with BP is still much worse than SGD with LR.

We then compare the performance of BP and meProp in Sun et al. (2017) using the same ANN structure in the main body of the paper. meProp-n stands for the meProp algorithm with n neurons out of 50 neurons in the hidden layer, which backpropagates the errors. In Figure A2, we can see that meProp leads to worse performance than BP under FGSM adversarial attacks and achieves a comparable performance to LR under L_BFGS adversarial attacks.

From Tables A3, the robustness improvement of ANNs with Laplace noises trained by LR is consistent with that observed in Table 1 of the main body of the paper.

We construct a 5 hidden-layers ANN with output layer, and the number of neurons in each layer are $100, 50, 40, 30, 20, 10$, respectively. The weights in ANN are initialized using Xavier method. The activation functions in the hidden layers are Sigmoid. We use the SGD method to train the model in 100 epochs. The initial learning rate is 0.1, and the learning rate will decay by multiplying 0.8 every 20 epochs.

From Figures A1, A2, A3, A4, and Table A4, we can see that the ANN trained by the LR method performs better as the number of the replications increases at a cost of longer computational time, and can achieve a comparable performance of BP when the number of replications reaches 300.

Hessian matrix is needed to apply the Newton method to train the ANNs. Let $\Sigma^{(t)}$ be a matrix with the elements being the second-order derivatives estimated by LR at the t-th iteration, and the Hessian matrix

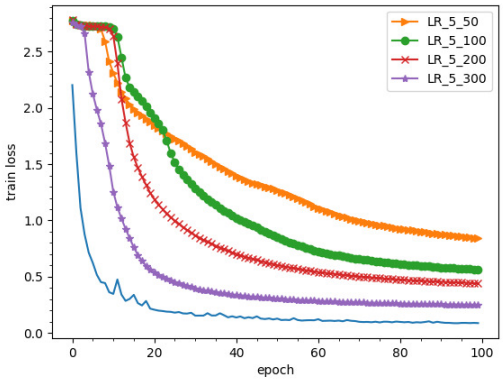

Figure A1 **Losses in training dataset**

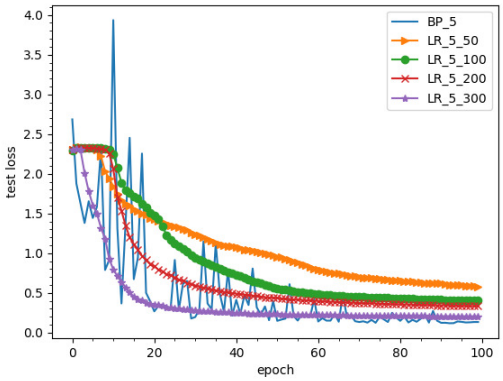

Figure A2 **Losses in test dataset**

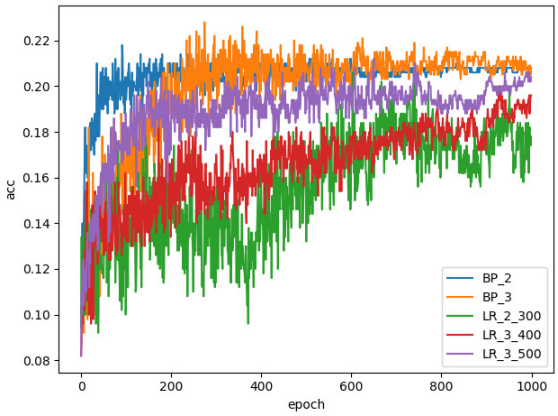

Figure A3 **Accuracy results of different methods**

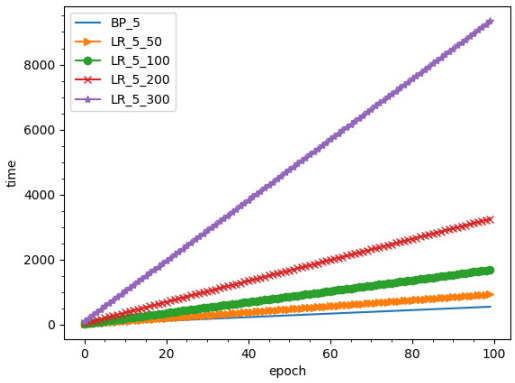

Figure A4 **Running time of different methods**

	BP_5	LR_5_50	LR_5_100	LR_5_200	LR_5_300
Acc	95.3%	82.4%	90.9%	92.5%	95.0%

Table A4 **The accuracy results for image identification using ANNs with five hidden layers trained by different methods in** 100 **epochs.**

is estimated by the following moving average:

$$H^{(t)} = \frac{1-\gamma^t}{1-\gamma}\left(\Sigma^{(t)} + \frac{\gamma(1-\gamma)}{1-\gamma^{t-1}}H^{(t-1)}\right),$$

where $\gamma \in (0,1)$ and $H^{(0)} = 0$. To test the performance of the Newton method (Rardin, 1998), we construct an ANN with one hidden layer containing 50 neurons and ReLu as the activation function, and set $\gamma = 0.8$.

Figures A5, A6, and A7 present experiment results about losses in the training dataset, losses in the testing dataset, and accuracy results of the different methods, respectively. In the legend, BP stands for the result of ANN trained by SGD with the BP method as gradient estimate, and LR stands for the result of ANN trained by SGD with the LR method as gradient estimate. LR-1, LR-3, and LR-5 stand for the results of ANN trained by SGD at the beginning and then the Newton method starting at the first, third, and filth epoch, respectively, where the first-and-second order derivatives are estimated by LR. Because inverting the

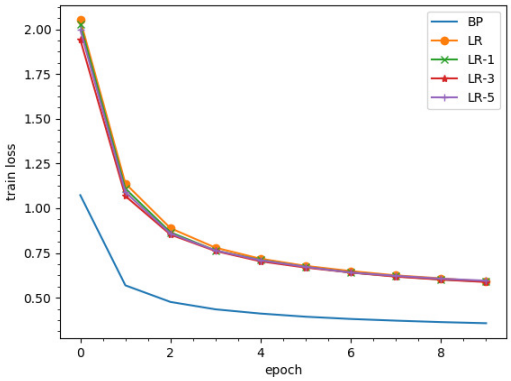

Figure A5 **Losses in training dataset**

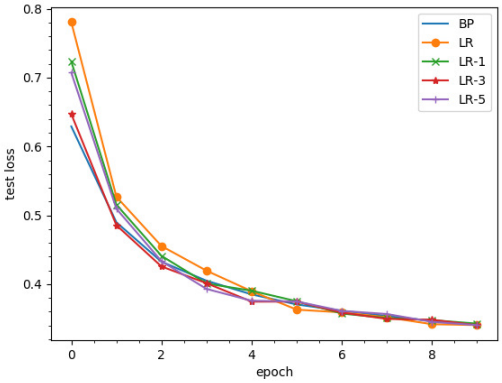

Figure A6 **Losses in testing dataset**

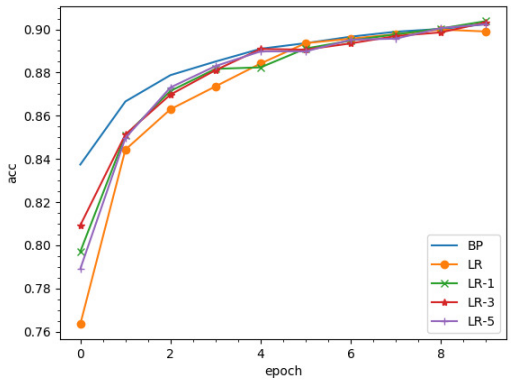

Figure A7 **Accuracy results of different methods**

	BP	LR	LR-1	LR-3	LR-5
Acc	90.2%	89.9%	90.4%	90.3%	90.2%

Table A5 **The accuracy results for image identification using ANNs with one hidden layer trained by SGD and the Newton method in** 10 **epochs.**

Hessian matrix is computationally expensive, the inversion of the Hessian matrix is only updated once in every 50 iterations in the Newton searching procedure. The number replications for LR is set as 5 and the the number of total epochs is 10 in training. The learning rate is initialized to 10^{-2}, and it will decay by multiplying 0.8 every 3 epochs. From Table A5, we can see that the Newton method can lead to a slightly better performance in 10 epochs but at a cost of higher computational burden.

E.2. Supplements of Numerical Experiments in Fashion-MNIST dataset

We then test the performance of the LR method for training ANNs under adversarial attacks in the Fashion-MNIST dataset (see Figure A8), consisting of a training set of $60,000$ images and a test set of $10,000$ images. Each image is a 28×28 grayscale image.

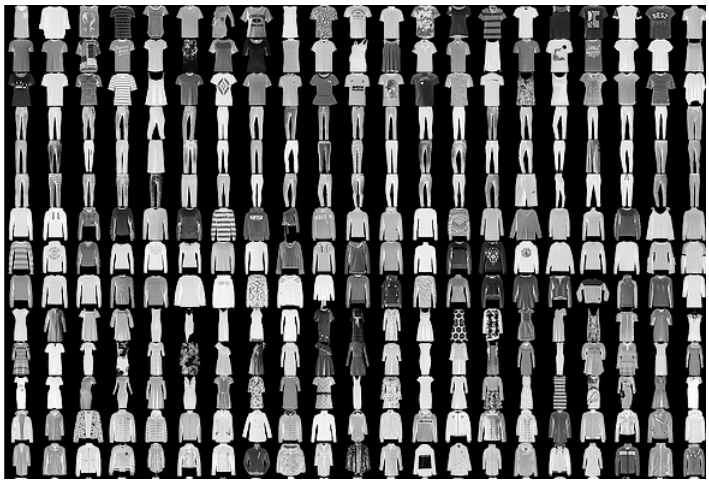

Figure A8 Images of Fashion-MNIST.

Activations + Entropy	**Orig**	**Adv_L_BFGS**	**Adv_FGSM**
BP benchmark	86.5%	49.6%	17.9%
Sigmoid	83.9%	53.5%	52.9%
Threshold	85.5%	52.9%	42.6%
Activations + 0-1 loss	**Orig**	**Adv_L_BFGS**	**Adv_FGSM**
Sigmoid	58.7%	51.0%	46.4%
Threshold	62.1%	60.2%	57.2%

Table A6 Adversarial tests for ANNs in identifying images of Fashion-MNIST with different activation and loss functions trained by LR including BP benchmark. Orig means the accuracy tested on original samples; Adv_L_BFGS means the accuracy tested on samples generated by the L-BFGS method; Adv_FGSM means the accuracy tested on samples generated by FGSM.

The ANN structure and other experiment configurations are set the same as that in Section 4 of the main body the paper. The accuracies of the ANN trained by the BP and LR methods reach around 85% in predicting the original samples. The accuracies of the ANN trained by the BP method reduces dramatically to about 50% and 18% in predicting adversarial samples generated by two different algorithms, respectively, whereas the ANNs trained by the LR method achieve $10\% - 30\%$ higher accuracies in predicting the adversarial samples.

Different from adversarial attacks where the input images are affected by small, additive, classifier-tailored perturbations, we can add small, general, classifier-agnostic perturbations to corrupt the input images. In Hendrycks and Dietterich (2018), an IMAGENET-C benchmark offers various corruption types with five severity levels for each type. In this work, we apply four algorithms to generate the corrupted samples for the MINST dataset. Assume the accuracy of the corruption type i, $i = 1, \ldots, 4$, at the severity level j, $j = 1, \ldots, 5$, is defined as $Acc_{j,i}$. Then the average accuracy is defined as the evaluation metrics:

$$Acc_i = \frac{1}{5} \sum_{j=1}^{5} Acc_{j,i}, \quad i = 1, \ldots, 4.$$

We then test the performance of the LR method for training ANNs with cross-entropy as the loss function under natural corruptions in the Fashion-MNIST dataset. From Table A6, we can see that ANNs trained

	BP	LR	
	Sigmoid	Sigmoid	Threshold
Original	86.5%	83.9%	85.5%
Gaussian Noise	81.1%	82.5%	82.8%
Impulse Noise	65.4%	80.2%	82.7%
Glass Blur	55.0%	79.1%	77.2%
Contrast	41.1%	50.0%	45.8%

Table A7 Test robustness to natural noises for ANNs in identifying images of Fashion-MNIST with different activation functions trained by LR and BP. Four types of corruption noises are adopted and the average accuracy is computed.

by the LR method perform more robustly than ANN trained by BP under four natural corruptions. Under Impulse noise and Glass Blur corruptions, the accuracies of ANN trained by the BP method drop $20\% - 30\%$, whereas the accuracies of the ANNs trained by the LR method are nearly intact.

E.3. Mean of Squared Difference Between LR and BP+

We construct a 3-layers ANN and a 4-layers ANN. In the process of training, we calculate the mean of the squared difference between LR and BP+ in the testing dataset for the same ANN. Specifically, we update parameters in ANN by SGD using LR in 30 epochs, and at the end of each epoch, the gradient of the loss of ANN is estimated by LR and BP+ in the testing dataset of MNIST and we calculate the following mean of squared difference for each parameter, say $\theta_{a,b}^{(l)}$:

$$\frac{1}{N}\sum_{n=1}^{N}\left(\delta_a^{(l+1)}(n)x_b^{(l)}(n) - L(X^{(\tau)}(n), O(n))\ \omega_{a,b}^{(l)}(n)\right)^2,$$

where N is the total number of samples in the testing dataset. By using BP+ as a benchmark, the mean of squared difference can reflect the average error and variance of LR. Then we report the max, mean, and min of the mean squared difference for all parameters in each layer of the ANN.

Figures A9, A10 and A11 present the losses in the training dataset, testing dataset, and accuracy results for a 3-layers ANN trained by LR with $1, 5, 10, 15$ replications, respectively, and Figures A9, A10 and A11 present those results for a 4-layers ANN trained by LR with $1, 15, 30$ replications. We can see the LR method with more replications lead to smaller losses and better accuracy results. The other figures report the max, mean, min of the mean of the squared difference between LR and BP+ for all parameters in each layers of ANN. In the legend of the figures, LR_ℓ/l_k stands for the gradient estimation results by LR for parameters in the ℓth layer of a l-layer ANN. We can see that in the training process, the mean of the squared difference between LR and BP+ in the test dataset approaches to zero for all parameters. Particularly, the min of the mean of the squared difference between LR and BP+ is close to zero for parameters in all layers during the entire training procedure.

References

Duchi, John, Elad Hazan, Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* **12**(7).

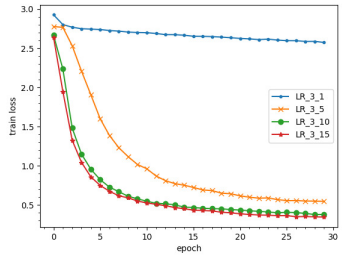

Figure A9 **Losses in training of 3-layers ANN trained by LR**

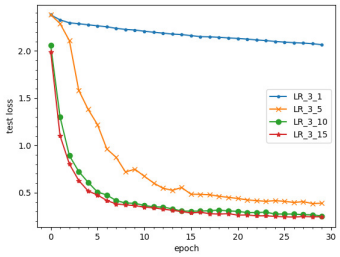

Figure A10 **Losses in testing of 3-layers ANN trained by LR**

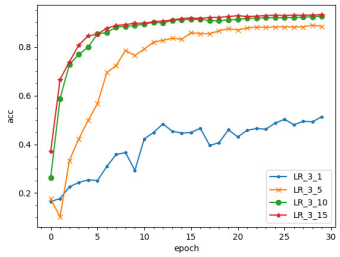

Figure A11 **Accuracy results of LR with different replications**

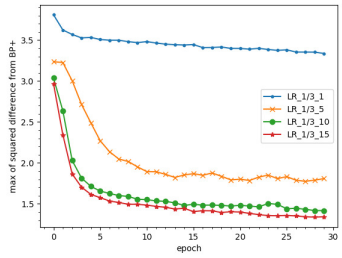

Figure A12 **Max in the first layer of 3-layers ANN**

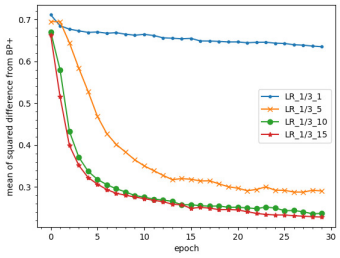

Figure A13 **Mean in the first layer of 3-layers ANN**

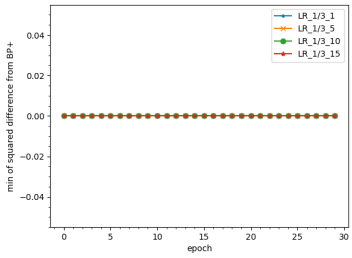

Figure A14 **Min in the first layer of 3-layers ANN**

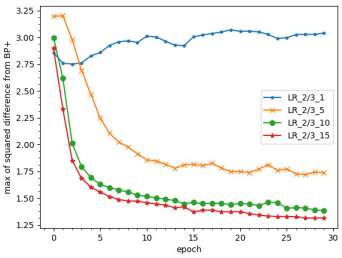

Figure A15 **Max in the second layer of 3-layers ANN**

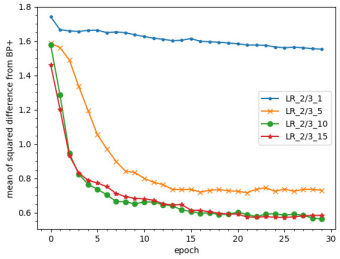

Figure A16 **Mean in the second layer of 3-layers ANN**

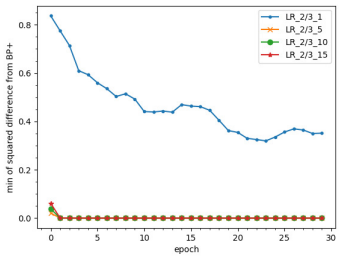

Figure A17 **Min in the second layer of 3-layers ANN**

Glasserman, Paul. 1991. *Gradient Estimation via Perturbation Analysis*. Kluwer Academic Publishers, Boston.

Hendrycks, Dan, Thomas G. Dietterich. 2018. Benchmarking neural network robustness to common corruptions and perturbations. *CoRR* **abs/1807.01697**. URL `http://arxiv.org/abs/1807.01697`.

Kingma, Diederik P, Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* 1–13.

Peng, Yijie, Michael C Fu, Jian-Qiang Hu, Bernd Heidergott. 2018. A new unbiased stochastic derivative estimator for discontinuous sample performances with structural parameters. *Operations Research* **66**(2)

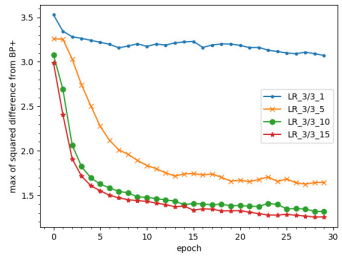

Figure A18 **Max in the third layer of 3-layers ANN**

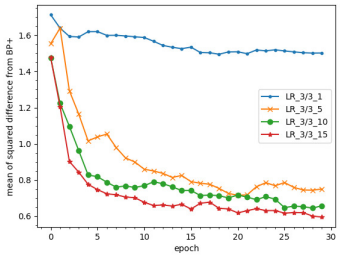

Figure A19 **Mean in the third layer of 3-layers ANN**

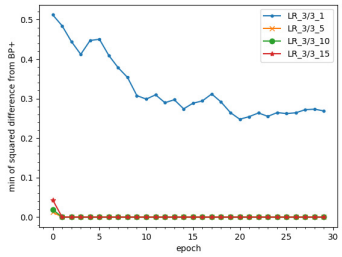

Figure A20 **Min in the third layer of 3-layers ANN**

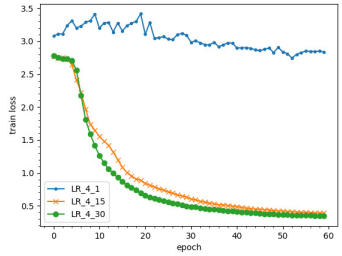

Figure A21 **Losses in training of 4-layers ANN trained by LR**

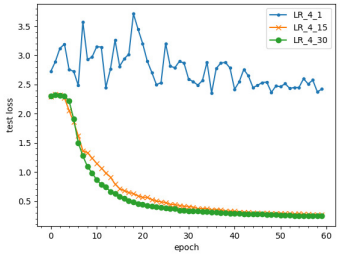

Figure A22 **Losses in testing of 3-layers ANN trained by LR**

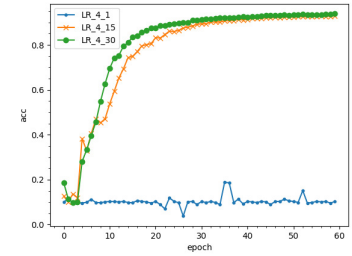

Figure A23 **Accuracy results of LR with different replications**

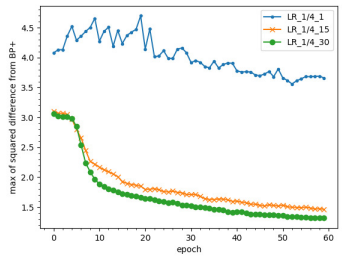

Figure A24 **Max in the first layer of 4-layers ANN**

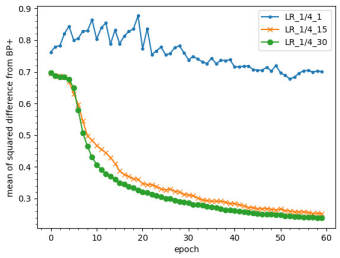

Figure A25 **Mean in the first layer of 4-layers ANN**

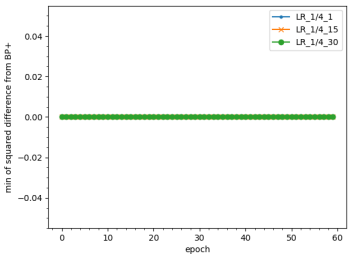

Figure A26 **Min in the first layer of 4-layers ANN**

487–499.

Rardin, Ronald L. 1998. *Optimization in operations research*, vol. 166. Prentice Hall Upper Saddle River, NJ.

Rudin, Walter. 1987. *Real and Complex Analysis*. McGraw-Hill Education, New York.

Sun, Xu, Xuancheng Ren, Shuming Ma, Houfeng Wang. 2017. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. *Proceedings of International Conference on Machine Learning* .

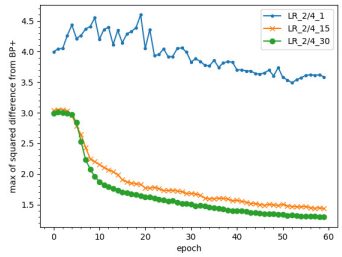

Figure A27 Max in the second layer of 4-layers ANN

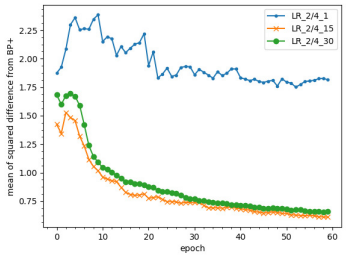

Figure A28 Mean in the second layer of 4-layers ANN

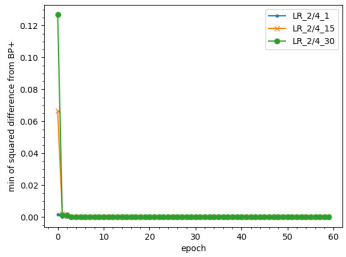

Figure A29 Min in the second layer of 4-layers ANN

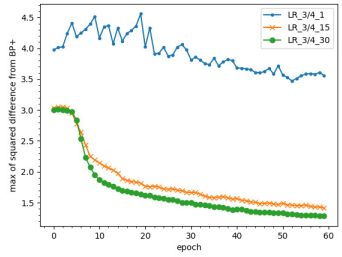

Figure A30 Max in the third layer of 4-layers ANN

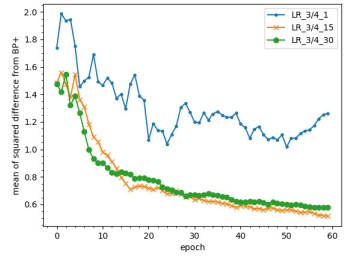

Figure A31 Mean in the third layer of 4-layers ANN

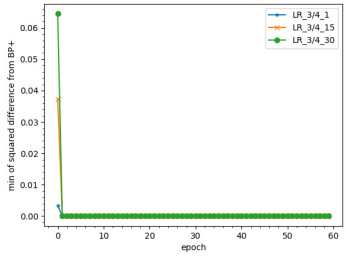

Figure A32 Min in the third layer of 4-layers ANN

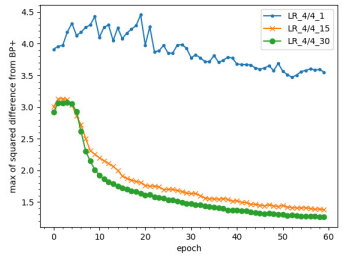

Figure A33 Max in the fourth layer of 4-layers ANN

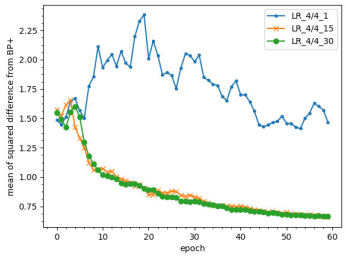

Figure A34 Mean in the fourth layer of 4-layers ANN

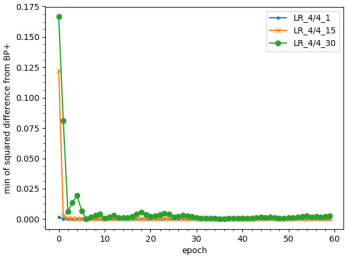

Figure A35 Min in the fourth layer of 4-layers ANN