Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Adaptive Normalized Risk-Averting Training for Deep Neural Networks

Zhiguang Wang, Tim Oates
Department of Computer Science and Electric Engineering
University of Maryland Baltimore County
zgwang813@gmail.com, oates@umbc.edu

James Lo
Department of Mathematics and Statistics
University of Maryland Baltimore County
jameslo@umbc.edu

Abstract

This paper proposes a set of new error criteria and a learning approach, called Adaptive Normalized Risk-Averting Training (ANRAT) to attack the non-convex optimization problem in training deep neural networks without pretraining. Theoretically, we demonstrate its effectiveness based on the expansion of the convexity region. By analyzing the gradient on the convexity index λ, we explain the reason why our learning method using gradient descent works. In practice, we show how this training method is successfully applied for improved training of deep neural networks to solve visual recognition tasks on the MNIST and CIFAR-10 datasets. Using simple experimental settings without pretraining and other tricks, we obtain results comparable or superior to those reported in recent literature on the same tasks using standard ConvNets + MSE/cross entropy. Performance on deep/shallow multilayer perceptron and Denoised Auto-encoder is also explored. ANRAT can be combined with other quasi-Newton training methods, innovative network variants, regularization techniques and other common tricks in DNNs. Other than unsupervised pretraining, it provides a new perspective to address the non-convex optimization strategy in training DNNs.

Introduction

Deep neural networks (DNNs) are attracting attention largely due to their impressive empirical performance in image and speech recognition tasks. While Convolutional Networks (ConvNets) are the de facto state-of-the-art for visual recognition, Deep Belief Networks (DBN), Deep Boltzmann Machines (DBM) and Stacked Auto-encoders (SA) provide insights as generative models to learn the full generating distribution of input data. Recently, researchers have investigated various techniques to improve the learning capacity of DNNs. Unsupervised pretraining using Restrict Boltzmann Machines (RBM), Denoised Autoencoders (DA) or Topographic ICA (TICA) has proved to be helpful for training DNNs with better weight initialization (Ngiam et al. 2010; Coates and Ng 2011). Rectified Linear Unit (ReLU) and variants are proposed as the optimal activation functions to better interpret hidden features Various regularization techniques such as dropout (Srivastava et al. 2014) with Maxout (Goodfellow et al. 2013b) are proposed to regulate the DNNs to be less prone to overfitting.

Neural network models always lead to a non-convex optimization problem. The optimization algorithm impacts the quality of the local minimum because it is hard to find a global minimum or estimate how far a particular local minimum is from the best possible solution. The most standard approach to optimize DNNs is Stochastic Gradient Descent (SGD). There are many variants of SGD and researchers and practitioners typically choose a particular variant empirically. While nearly all DNNs optimization algorithms in popular use are gradient-based, recent work has shown that more advanced second-order methods such as L-BFGS and Saddle-Free Newton (SFN) approaches can yield better results for DNN tasks (Ngiam et al. 2011; Dauphin et al. 2014). Second order derivatives can be addressed by hardware extensions (GPUs or clusters) or batch methods when dealing with massive data, SGD still provides a robust default choice for optimizing DNNs.

Instead of modifying the network structure or optimization techniques for DNNs, we focused on designing a new error function to convexify the error space. The convexification approach has been studied in the optimization community for decades, but has never been seriously applied within deep learning. Two well-known methods are the graduated nonconvexity method (Blake and Zisserman 1987) and the LiuFloudas convexification method (Liu and Floudas 1993). LiuFloudas convexification can be applied to optimization problems where the error criterion is twice continuously differentiable, although determining the weight α of the added quadratic function for convexifying the error criterion involves significant computation when dealing with massive data and parameters.

Following the same name employed for deriving robust controllers and filters (Speyer, Deyst, and Jacobson 1974), a new type of Risk-Averting Error (RAE) is proposed theoretically for solving non-convex optimization problems (Lo 2010). Empirically, with the proposal of Normalized Risk-Averting Error (NRAE) and the Gradual Deconvexification method (GDC), this error criterion is proved to be competitive with the standard mean square error (MSE) in single layer and two-layer neural networks for solving data fit-

ting and classification problems (Gui, Lo, and Peng 2014; Lo, Gui, and Peng 2012). Interestingly, SimNets, a generalization of ConvNets that was recently proposed in (Cohen and Shashua 2014), uses the MEX operator (whose name stands for Maximum-minimum-Expectation Collapsing Smooth) as an activation function to generalize ReLU activation and max pooling. We notice that the MEX operator with L_2 units has exactly the *same* mathematical form with NRAE. However, NRAE is still hard to optimize in practice due to plateaus and the unstable error space caused by the fixed large convexity index. GDC alleviates these problems but its performance is limited and suffers from the slow learning speed. Instead of fixing the convexity index λ, Adaptive Normalized Risk-Averting Training (ANRAT) optimizes NRAE by tuning λ adaptively using gradient descent. We give theoretical proofs of its optimal properties against the standard L_p-norm error. Our experiments on MNIST and CIFAR-10 with different deep/shallow neural nets demonstrate the effectiveness empirically. Being an optimization algorithm, our approach are not supposed to deal specifically with the problem of over-fitting, however we show that this can be handled by the usual methods of regularization such as weight decay or dropout.

Convexification on Error Criterion

We begin with the definition of RAE for the L_p norm and the theoretical justifications on its convexity property. RAE is not suitable for real applications since it is not bounded. Instead, NRAE is bounded to overcome the register overflow in real implementations. We prove that NRAE is quasi-convex, and thus shares the same global and local optimum with RAE. Moreover, we show the lower-bound of its performance is as good as L_p-norm error when the convexity index satisfies a constraint, which theoretically supports the ANRAT method proposed in the next section.

Risk-averting Error Criterion

Given training samples $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, the function $f(\boldsymbol{x}_i, \boldsymbol{W})$ is the learning model with parameters W. The loss function of L_p-norm error is defined as:

$$l_p(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) = \frac{1}{m}\sum_{i=1}^{m} ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p \qquad (1)$$

When $p = 2$, Eqn. 1 denotes to the standard Mean Square Error (MSE). The Risk-Averting Error criterion (RAE) corresponding to the L_p-norm error is defined by

$$RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) = \frac{1}{m}\sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} \qquad (2)$$

λ is the convexity index. It controls the size of the convexity region.

Because RAE has the sum-exponential form, its Hessian matrix is tuned exactly by the convexity index λ^q. The following theorem indicates the relation between the convexity index and its convexity region.

Theorem 1 (Convexity)**.** *Given the Risk-Averting Error criterion $RAE_{p,q}$ ($p, q \in \mathcal{N}^+$), which is twice continuous differentiable. $J_{p,q}(\boldsymbol{W})$ and $H_{p,q}(\boldsymbol{W})$ are the corresponding Jacobian and Hessian matrix. As $\lambda \to \pm\infty$, the convexity region monotonically expands to the entire parameter space except for the subregion $S := \{W \in \mathcal{R}^n | rank(H_{p,q}(\boldsymbol{W})) < n, H_{p,q}(\boldsymbol{W} < 0)\}$.*

Please refer to the supplementary material for the proof. Intuitively, the use of the RAE was motivated by its emphasizing large individual deviations in approximating functions and optimizing parameters in an exponential manner, thereby avoiding such large individual deviations and achieving robust performances. Theoretically, Theorem 1 states that when the convexity index λ increases to infinity, the convexity region in the parameter space of RAE expands monotonically to the entire space except the intersection of a finite number of lower dimensional sets. The number of sets increases rapidly as the number m of training samples increases. Roughly speaking, larger λ and m cause the size of the convexity region to grow larger respectively in the error space of RAE.

When $\lambda \to \infty$, the error space can be perfectly stretched to be strictly convex, thus avoid the local optimum to guarantee a global optimum. Although RAE works well in theory, it is not bounded and suffers from the exponential magnitude and arithmetic overflow when using gradient descent in implementations .

Normalized Risk-Averting Error Criterion

RAE ensures the convexity of the error space to find the global optimum. By using NRAE, we relax the global optimum problem by finding a better local optimum to meet a theoretically and practically reasonable trade-off in real applications.

Given training samples $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, the function $f(\boldsymbol{x}_i, \boldsymbol{W})$ is the learning model with parameters W. The Normalized Risk-Averting Error Criterion (NRAE) corresponding to the L_p-norm error is defined as:

$$\begin{aligned} &NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\ &= \frac{1}{\lambda^q}\log RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\ &= \frac{1}{\lambda^q}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} \end{aligned} \qquad (3)$$

Theorem 2 (Bounded)**.** *$NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is bounded.*

The proof is provided in the supplemental materials. Briefly, NRAE is bounded by functions independent of λ and no overflow occurs for $\lambda \gg 1$. The following theorem states the quasi-convexity of NRAE.

Theorem 3 (Quasi-convexity)**.** *Given a parameter space $\{W \in \mathcal{R}^n\}$, Assume $\exists\ \psi(W)$, s.t. $H_{p,q}(\boldsymbol{W}) > 0$ when $|\lambda^q| > \psi(W)$ to guarantee the convexity of $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$. Then, $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is*

quasi-convex *and share the same local and global optimum with* $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$.

Proof. If $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is convex, it is quasi-convex. log function is monotonically increasing, so the composition $\log RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex. [1]

log is a strictly monotone function and $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex, so it shares the same local and global minimizer with $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$. □

The convexity region of NRAE is consistent with RAE. To interpret this statement in another perspective, the log function is a strictly monotone function. Even if RAE is not strictly convex, NRAE still shares the same local and global optimum with RAE. If we define the mapping function $f : RAE \rightarrow NRAE$, it is easy to see that f is bijective and continuous. Its inverse map f^{-1} is also continuous, so that f is an open mapping. Thus, it is easy to prove that the mapping function f is a homeomorphism to preserve all the topological properties of the given space.

The above theorems state the consistent relations among NRAE, RAE and MSE. It is proven that the greater the convexity index λ, the larger is the convex region is. Intuitively, increasing λ creates tunnels for a local-search minimization procedure to travel through to a good local optimum. However, we care about the justification on the advantage of NRAE against MSE. Theorem 4 provides the theoretical justification for the performance lower-bound of NRAE.

Theorem 4 (Lower-bound)**.** *Given training samples* $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$ *and the model* $f(\boldsymbol{x}_i, \boldsymbol{W})$ *with parameters* W*. If* $\lambda^q \geq 1$*,* $p, q \in \mathcal{N}^+$ *and* $p \geq 2$*, then both* $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *and* $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *always have the higher chance to find a better local optimum than the standard* L_p*-norm error due to the expansion of the convexity region.*

Proof. Let $h_p(\boldsymbol{W})$ denotes the Hessian matrix of standard L_p-norm error (Eqn. 1), note $\alpha_i(\boldsymbol{W}) = f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i$ we have

$$\begin{aligned} h_p(\boldsymbol{W}) &= \frac{p}{m}\sum_{i=1}^{m}\{(p-1)\alpha_i(\boldsymbol{W})^{p-2}\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})^2}{\partial \boldsymbol{W}} \\ &+ \alpha_i(\boldsymbol{W})^{p-1}\frac{\partial f^2(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \end{aligned} \tag{4}$$

Since $\lambda^q \geq 1$, let $diag_{eig}$ denotes the diagonal matrix of the eigenvalues from SVD decomposition. $\succeq$ here means 'element-wise greater'. When $A \succeq B$, each element in A is greater than B. Then we have

$$\begin{aligned} diag_{eig}[H_{p,q}(\boldsymbol{W})] &\succeq diag_{eig}[h_p(\boldsymbol{W}) + \\ &\frac{p^2}{m}\sum_{i=1}^{m}||\alpha_i(\boldsymbol{W})||^{2p-2}\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}}^2\}] \\ &\succeq diag_{eig}[h_p(\boldsymbol{W})] \end{aligned} \tag{5}$$

[1] Because the function f defined by $f(x) = g(U(x))$ is quasiconvex if the function U is quasiconvex and the function g is increasing.

This indicates that the $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ always has larger convexity regions than the standard L_p-norm error to better enable escape of local minima. Because $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex, sharing the same local and global optimum with $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$, the above conclusions are still valid. □

Roughly speaking, NRAE always has a larger convexity region than the standard L_p-norm error in terms of their Hessian matrix when $\lambda \geq 1$. This property guarantees the higher probability to escape poor local optima using NRAE. In the worst case, NRAE will perform as good as standard L_p-norm error if the convexity region shrinks as λ decreases or the local search deviates from the "tunnel" of convex regions.

More specifically, $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$

- approaches the standard L_p-norm error as $\lambda^q \rightarrow 0$.
- approaches the minimax error criterion $inf_W \alpha_{max}(\boldsymbol{W})$ as $\lambda^q \rightarrow \infty$.

Please refer to the supplemental materials for the proofs. More rigid proofs that can be generalized to L_p-norm error are also given in (Lo 2010). In SimNets, the authors also include quite similar discussions about the robustness with respect to L_p-norm error (Cohen and Shashua 2014).

Learning Methods

We propose a novel learning method to training DNNs with NRAE, called the Adaptive Normalized Risk-Avering Training (ANRAT) approach. Instead of manually tuning λ like GDC (Lo, Gui, and Peng 2012), we learn λ adaptively in error backpropagation by considering λ as a parameter instead of a hyperparameter. The learning procedure is standard batch SGD. We show it works quite well in theory and practice.

The loss function of ANRAT is

$$l(W, \lambda) = \frac{1}{\lambda^q}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda^q||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} + a||\lambda||^{-r} \tag{6}$$

Together with NRAE, we also use a penalty term $a||\lambda||^{-r}$ to control the changing rate of λ. While minimize the NRAE score, small λ is penalized to regulate the convexity region. a is a hyperparameter to control the penalty index. The first-order derivatives on weight and λ are

$$\frac{dl(W, \lambda)}{dW} = \frac{p\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^{p-1}}\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}}}{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^{p-1}}} \tag{7}$$

$$\frac{dl(W, \lambda)}{d\lambda} = \frac{-q}{\lambda^{q+1}}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p} \tag{8}$$

$$+ \frac{q}{\lambda}\frac{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p}\alpha_i(W)^p}{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p}} \tag{9}$$

$$- ar\lambda^{-r-1} \tag{10}$$

We make a transformation on Eqn. 10 to better understand the gradient with respect to λ. Note that $k_i =$

$\frac{e^{\lambda^q \alpha_i(W)^p}}{\sum_{i=1}^m e^{\lambda^q \alpha_i(W)^p}}$ is actually performing like a probability ($\sum_{i=1}^m k_i = 1$). Ignoring the penalty term, Eqn. 10 can be formulated as follows:

$$\begin{aligned} \frac{dl(W,\lambda)}{d\lambda} &= \frac{q}{\lambda}(\sum_{i=1}^{m} k_i \alpha_i(\boldsymbol{W})^p - NRAE) \\ &= \frac{q}{\lambda}(\mathbb{E}(\alpha(\boldsymbol{W})^p) - NRAE) \\ &\approx \frac{q}{\lambda}(L_P\text{-norm error} - NRAE) \quad (11) \end{aligned}$$

Note that as $\alpha_i(\boldsymbol{W})^p$ becomes smaller, the expectation on $\alpha_i(\boldsymbol{W})^p$ approaches the standard L_p-norm error. Thus, the gradient on λ is approximately the difference between NRAE and the standard L_p-norm error. Because large λ can incur plateaus to prevent NRAE from finding better optima using batch SGD (Lo, Gui, and Peng 2012), they need GDC to gradually deconvexify the NRAE to make the error space well shaped and stable. Through Eqn. 11, ANRAT solve this problem in a more flexible and adaptive manner. When NRAE is larger, Eqn. 11 remains negative and makes λ increase to enlarge the convexity region, facilitating the search in the error space for better optima. When NRAE is smaller, the learned parameters are seemingly going through the optimal "tunnel" for better optima. Eqn. 11 becomes positive to decrease λ and helps NRAE not deviate far from the manifold of the standard L_p-norm error to make the error space stable without large plateaus. Thus, ANRAT adaptively adjusts the convexity index to find an optimal trade-off between better solutions and stability.

This training approach has more flexibility. The gradient on λ as the weighted difference between NRAE and the standard L_P-norm error, enables NRAE to approach the L_P-norm error by adjusting λ gradually. Intuitively, it keeps searching the error space near the manifold of the L_p-norm error to find better optima in a way of competing with and at the same time relying on the standard L_p-norm error space.

In Eqn. 6, the penalty weight a and index r control the convergence speed by penalizing small λ. Smaller a emphasizes tuning λ to allow faster convergence speed between NRAE and L_p-norm error. Larger a forces larger λ for a better chance to find a better local optimum but runs the risk of plateaus and deviating far from the stable error space. r regulates the magnitude of λ and its derivatives in gradient descent.

Experiments

We present the results from a series of experiments designed on the MNIST and CIFAR-10 datasets to test the effectiveness of ANRAT for visual recognition with DNNs. We did not explore the full hyperparameters in Eqn. 6. Instead we fix the hyperparameters at $p = 2$, $q = 2$ and $r = 1$ to mainly compare with MSE. So the final loss function of ANRAT we optimized is

$$l(W,\lambda) = \frac{1}{\lambda^2} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda^2 ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^2} + a|\lambda|^{-1} \quad (12)$$

Method[2]	Error %
Convolutional Kernel Networks + L-BFGS-B(1)	**0.39**
Deeply Supervised Nets + dropout(2)	**0.39**
ConvNets (Lenet-5)(3)	0.95
ConvNets + MSE/CE (this paper)	0.93
large ConvNets, random feature(4)	0.89
ConvNets + L-BFGS(5)	0.69
large ConvNets, unsup pretraining(6)	0.62
ConvNets, unsup pretraining(7)	0.6
ConvNets + dropout(8)	0.55
large ConvNets, unsup pretraining(9)	0.53
ConvNets + ANRAT (This paper)	0.52
ConvNets + ANRAT + dropout (This paper)	**0.39**

Table 1: Test set misclassification rates of the best methods that utilized convolutional networks on the original MNIST dataset using single model.

This loss function is minimized by batch SGD without complex methods, such as momentum, adaptive/hand tuned learning rates or tangent prop. The learning rate and penalty weight a are selected in $\{1, 0.5, 0.1\}$ and $\{1, 0.1, 0.001\}$ on validation sets respectively. The initial λ is fixed at 10. We use the hold-out validation set to select the best model, which is used to make predictions on the test set. All experiments are implemented quite easily in Python and Theano to obtain GPU acceleration (Bastien et al. 2012).

The MNIST dataset (LeCun et al. 1998) consists of hand written digits 0-9 which are 28x28 in size. There are 60,000 training images and 10,000 testing images in total. We use 10000 images in training set for validation to select the hyperparameters and report the performance on the test set. We test our method on this dataset without data augmentation.

The CIFAR-10 dataset (Krizhevsky and Hinton 2009) is composed of 10 classes of natural images. There are 50,000 training images in total and 10,000 testing images. Each image is an RGB image of size 32x32. For this dataset, we adapt pylearn2 (Goodfellow et al. 2013a) to apply the same global contrast normalization and ZCA whitening as was used by Goodfellow et. al (Goodfellow et al. 2013b). We use the last 10,000 images of the training set as validation data for hyperparameter selection and report the test accuracy.

Results and Discussion

Results on ConvNets

On the MNIST dataset we use the same structure of LeNet5 with two convolutional max-pooling layers but followed by only one fully connected layer and a densely connected softmax layer. The first convolutional layer has 20 feature maps of size 5×5 and max-pooled by 2×2 non-overlapping windows. The second convolutional layer has 50 feature maps

[2](1)(Mairal et al. 2014);(2)(Lee et al. 2014);(3)(LeCun et al. 1998);(4)(Ranzato et al. 2007);(5)(Ngiam et al. 2011);(6)(Ranzato et al. 2007);(7)(Poultney et al. 2006);(8)(Zeiler and Fergus 2013);(9)(Jarrett et al. 2009)

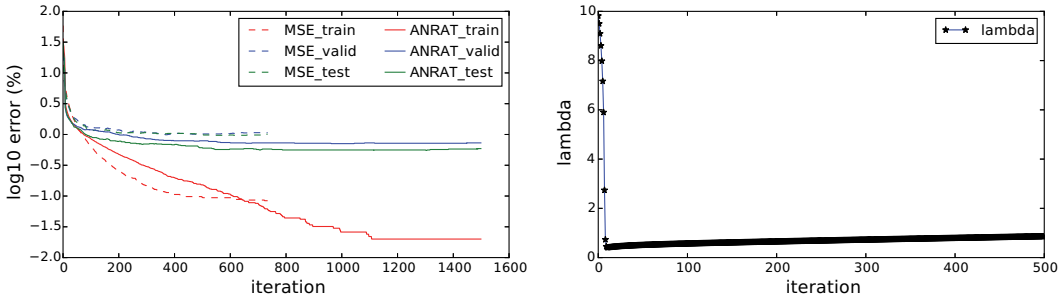

Figure 1: (a). MNIST train, validation and test error rates throughout training with Batch SGD for MSE and ANRAT with l_2 priors (left). (b). The curve of λ throughout ANRAT training (right).

with the same convolutional and max-pooling size. The fully connected layer has 500 hidden units. An l_2 prior was used with the strength 0.05 in the Softmax layer. Trained by ANRAT, we can obtain a test set error of 0.52%, which is the best result we are aware of that does not use dropout on the pure ConvNets. We summarize the best published results on the standard MNIST dataset in Table 1.

The best performing neural networks for pure ConvNets that does not use dropout or unsupervised pretraining achieve an error of about 0.69% (Ngiam et al. 2011). They demonstrated this performance with L-BFGS. Using dropout, ReLU and a response normalization layer, the error reduces to 0.55% (Zeiler and Fergus 2013). Prior to that, Jarrett et. al showed by increasing the size of the network and using unsupervised pretraining, they can obtain a better result at 0.53% (Jarrett et al. 2009). Previous state of the art is 0.39% (Mairal et al. 2014; Lee et al. 2014) for a single model on the original MNIST dataset. Using batch SGD to optimize either CE or MSE on the ConvNets descried above, we can get an error rate at 0.93%. Replacing the training methods with ANRAT using batch GD leads to a sharply decreased validation error of 0.66% with a test error at 0.52%. With dropout and ReLU the test error rate drops to 0.39%, which is the same with the best results without averaging or data augmentation (Table 1) but we only use standard Convnets and simple experimental settings.

Fig. 1 (a) shows the progression of training, validation and test errors over 160 training epochs. The errors trained on MSE plateau as it can not train the ConvNets sufficiently and seems like underfit. Using ANRAT, the validation and test errors remain decreasing along with the training error. During training, λ sharply decrease, regulating the tunnel of NRAE to approach the manifold of MSE. Afterward the penalty term becomes significant, force λ to grow gradually while expanding the convex region for higher chance to find the better optimum (Figure 1 (b)).

Our next experiment is performed on the CIFAR-10 dataset. We observed significant overfitting using both MSE and ANRAT with the fixed learning rate and batch SGD, so dropout is applied to prevent the co-adaption of weights and improve generalization. We use a similar network layout as in (Srivastava et al. 2014) but with only two convolutional max-pooling layers. The first convolutional layer has 96 feature maps of size 5×5 and max-pooled by 2×2 non-overlapping windows. The second convolutional layer has 128 feature maps with the same convolutional and max-pooling size. The fully connected layer has 500 hidden units. Dropout was applied to all the layers of the network with the probability of retaining a hidden unit being $p = (0.9, 0.75, 0.5, 0.5, 0.5)$ for the different layers of the network. Using batch SGD to optimize CE on the simple configuration of ConvNets + dropout, a test accuracy of 80.6 % is achieved (Krizhevsky, Sutskever, and Hinton 2012). We also reported the performance at 80.58% with MSE instead of CE with the similar network layout. Replacing the training methods with ANRAT using batch SGD gives a test accuracy of 85.15%. This is superior to the results obtained by MSE/CE and unsupervised pretraining. In Table. 2, our result with simple setting is shown to be competitive to those achieved by different ConvNet variants.

Method[3]	Acc %
ConvNets + Stochastic pooling + dropout$^{(1)}$	84.87
ConvNets + dropout +Bayesian hyperopt$^{(2)}$	87.39
ConvNets + Maxout + dropout$^{(3)}$	88.32
Convolutional NIN + dropout$^{(6)}$	89.6
Deeply Supervised Nets + dropout$^{(7)}$	**90.31**
ConvNets + MSE + dropout (this paper)	80.58
ConvNets + CE + dropout$^{(4)}$	80.6
ConvNets + VQ unsup pretraining$^{(5)}$	82
ConvNets + ANRAT + dropout (This paper)	**85.15**

Table 2: Test accuracy of the best methods that utilized convolutional framework on CIFAR-10 dataset without data augmentation.

Results on Multilayer Perceptron

On the MNIST dataset, MLPs with unsupervised pretraining has been well studied in recent years, so we select

[3](1)(Zeiler and Fergus 2013);(2)(Srivastava et al. 2014);(3)(Goodfellow et al. 2013b);(4)(Zeiler and Fergus 2013);(5)(Coates and Ng 2011);(6)(Min Lin 2014);(7)(Lee et al. 2014)

this dataset to compare ANRAT in shallow and deep MLPs with MSE/CE and unsupervised pretraining. For the shallow MLPs, we follow the network layout as in (Gui, Lo, and Peng 2014; LeCun et al. 1998) that has only one hidden layer with 300 neurons. We build the stacked architecture and deep network using the same architecture as (Larochelle et al. 2009) with 500, 500 and 2000 hidden units in the first, second and third layers, respectively. The training approach is purely batch SGD with no momentum or adaptive learning rate. No weight decay or other regularization technique is applied in our experiments.

Experiment results in Table. 3 show that the deep MLP classifier trained by the ANRAT method has the lowest test error rate (1.45%) of benchmark MLP classifiers with MSE/CE under the same settings. It indicates that ANRAT has the ability to provide reasonable solutions with different initial weight vectors. This result is also better than deep MLP + supervised pretraining or Stacked Logistic Regression networks. We note that the deep MLP using unsupervised pretraining (auto-encoders or RBMs) remains to be the best with test error at 1.41% and 1.2%. Unsupervised pretraining is effective in initializing the weights to obtain a better local optimum. Compared with unsupervised pretraining + fine tuning, ANRAT sometimes still fall into the sightly worse local optima in this case. However, ANRAT is significantly better than MSE/CE without unsupervised pretraining.

Interestingly, we do not observe significant advantages with ANRAT in shallow MLPs. Although in early literature, the error rate on shallow MLPs were reported as 4.7% (LeCun et al. 1998) and 2.7% with GDC (Gui, Lo, and Peng 2014), both recent papers using CE (Larochelle et al. 2009) and our own experiments with MSE can achieve error rate of 1.93% and 2.02%, respectively. Trained by ANRAT, we can have a test rate at 1.94%. This performance is slightly better than MSE, but it is statistically identical to the performance obtained by CE. [4]. One possible reason is that in shallow networks which can be trained quite well by standard back propagation with normalized initializations, the local optimum achieved with MSE/CE is quite nearly a global optimum or good saddle point. Our result is also corresponding to the conclusion in (Dauphin et al. 2014), in which Dauphin et al. extend previous findings on networks with a single hidden layer to show theoretically and empirically that most badly suboptimal critical points are saddle points. Even with better convexity property, ANRAT is as good as MSE/CE in shallow MLPs. However, we find that the problem of poor local optimum becomes more manifest in deep networks. It is easier for ANRAT to find a way towards the better optimum near the manifold of MSE. For the sake of space, please refer to supplemental materials for the results on the shallow Denoised Auto-encoder. The conclusion is consistent that ANRAT performs better when attacking more difficult learning/fitting problems. While ANRAT is slightly better than CE/MSE + SGD on DA with uniform masking noise, it achieves a significant performance boost when Gaussian block masking noise is applied.

[4]in (Larochelle et al. 2009), the author do not report their network settings of the shallow MLP + CE, which may differ from 784-300-10.

Method[5]	Error %
Deep MLP + supervised pretraining$^{(1)}$	2.04
Stakced Logistic Regression Network$^{(1)}$	1.85
Stacked Auto-encoder Network$^{(1)}$	1.41
Stacked RBM Network$^{(1)}$	**1.2**
Shallow MLP + MSE$^{(2)}$	4.7
Shallow MLP + GDC$^{(3)}$	2.7 ± 0.03
Shallow MLP + MSE (this paper)	2.02
Shallow MLP + ANRAT (this paper)	1.94
Shallow MLP + CE$^{(1)}$	**1.93**
Deep MLP + CE$^{(1)}$	2.4
Deep MLP + MSE (this paper)	1.91
Deep MLP + ANRAT (this paper)	**1.45**

Table 3: Test error rate of deep/shallow MLP with different training techniques.

Conclusions and Outlook

In this paper, we introduce a novel approach, Adaptive Normalized Risk-Averting Training (ANRAT), to help train deep neural networks. Theoretically, we prove the effectiveness of Normalized Risk-Averting Error on its arithmetic bound, global convexity and local convexity lower-bounded by standard L_p-norm error when convexity index $\lambda \geq 1$. By analyzing the gradient on λ, we explained the reason why using back propagation on λ works. The experiments on deep/shallow network layouts demonstrate comparable or better performance with the same experimental settings among pure ConvNets and MLP + batch SGD on MSE and CE (with or without dropout). Other than unsupervised pretraining, it provides a new perspective to address the non-convex optimization strategy in DNNs.

Finally, while these early results are very encouraging, clearly further research is warranted to address the questions that arise from non-convex optimization in deep neural networks. It is preliminarily showed that in order to generalize to a wide array of tasks, unsupervised and semi-supervised learning using unlabeled data is crucial. One interesting future work is to take advantage of unsupervised/semi-supervised pretraining with the non-convex optimization methods to train deep neural networks by finding the nearly global optimum. Another crucial question is to guarantee the generalization capability by preventing overfitting. Finally, we are quite interested in generalizing our approach to recurrent neural networks. We leave as future work any performance improvement on benchmark datasets by considering the cutting-edge approach to improve training and generalization performance.

[5](1)(Larochelle et al. 2009);(2)(LeCun et al. 1998);(3)(Gui, Lo, and Peng 2014)

References

Bastien, F.; Lamblin, P.; Pascanu, R.; Bergstra, J.; Goodfellow, I. J.; Bergeron, A.; Bouchard, N.; and Bengio, Y. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Blake, A., and Zisserman, A. 1987. *Visual reconstruction*, volume 2. MIT press Cambridge.

Coates, A., and Ng, A. Y. 2011. Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*, 2528–2536.

Cohen, N., and Shashua, A. 2014. Simnets: A generalization of convolutional networks. *arXiv preprint arXiv:1410.0781*.

Dauphin, Y. N.; Pascanu, R.; Gulcehre, C.; Cho, K.; Ganguli, S.; and Bengio, Y. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, 2933–2941.

Goodfellow, I. J.; Warde-Farley, D.; Lamblin, P.; Dumoulin, V.; Mirza, M.; Pascanu, R.; Bergstra, J.; Bastien, F.; and Bengio, Y. 2013a. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013b. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Gui, Y.; Lo, J. T.-H.; and Peng, Y. 2014. A pairwise algorithm for training multilayer perceptrons with the normalized risk-averting error criterion. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, 358–365. IEEE.

Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, 2146–2153. IEEE.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep* 1(4):7.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Larochelle, H.; Bengio, Y.; Louradour, J.; and Lamblin, P. 2009. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research* 10:1–40.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lee, C.-Y.; Xie, S.; Gallagher, P.; Zhang, Z.; and Tu, Z. 2014. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*.

Liu, W., and Floudas, C. A. 1993. A remark on the gop algorithm for global optimization. *Journal of Global Optimization* 3(4):519–521.

Lo, J. T.-H.; Gui, Y.; and Peng, Y. 2012. Overcoming the local-minimum problem in training multilayer perceptrons with the nrae training method. In *Advances in Neural Networks–ISNN 2012*. Springer. 440–447.

Lo, J. T.-H. 2010. Convexification for data fitting. *Journal of global optimization* 46(2):307–315.

Mairal, J.; Koniusz, P.; Harchaoui, Z.; and Schmid, C. 2014. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2627–2635.

Min Lin, Qiang Chen, S. Y. 2014. Network in network. *arXiv preprint arXiv:1312.4400v3*.

Ngiam, J.; Chen, Z.; Chia, D.; Koh, P. W.; Le, Q. V.; and Ng, A. Y. 2010. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1279–1287.

Ngiam, J.; Coates, A.; Lahiri, A.; Prochnow, B.; Le, Q. V.; and Ng, A. Y. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 265–272.

Poultney, C.; Chopra, S.; Cun, Y. L.; et al. 2006. Efficient learning of sparse reresentations with an energy-based model. In *Advances in neural information processing systems*, 1137–1144.

Ranzato, M.; Huang, F. J.; Boureau, Y.-L.; and LeCun, Y. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE.

Speyer, J. L.; Deyst, J.; and Jacobson, D. 1974. Optimization of stochastic linear systems with additive measurement and process noise using exponential performance criteria. *Automatic Control, IEEE Transactions on* 19(4):358–366.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Zeiler, M. D., and Fergus, R. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.