Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation

Binghui Chen[1], Weihong Deng[1], Junping Du[2]
[1]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications,
[2]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing China.
chenbinghui@bupt.edu.cn, whdeng@bupt.edu.cn, junpingd@bupt.edu.cn

Abstract

*Over the past few years, softmax and SGD have become a commonly used component and the default training strategy in CNN frameworks, respectively. However, when optimizing CNNs with SGD, the saturation behavior behind softmax always gives us an illusion of training well and then is omitted. In this paper, we first emphasize that the **early saturation** behavior of softmax will impede the exploration of SGD, which sometimes is a reason for model converging at a bad local-minima, then propose **Noisy Softmax** to mitigating this early saturation issue by injecting annealed noise in softmax during each iteration. This operation based on noise injection aims at postponing the early saturation and further bringing continuous gradients propagation so as to significantly encourage SGD solver to be more exploratory and help to find a better local-minima. This paper empirically verifies the superiority of the early softmax desaturation, and our method indeed improves the generalization ability of CNN model by regularization. We experimentally find that this early desaturation helps optimization in many tasks, yielding state-of-the-art or competitive results on several popular benchmark datasets.*

1. Introduction

Recently, deep convolutional neural networks (DCNNs) have taken the computer vision field by storm, significantly improving the state-of-the-art performances in many visual tasks, such as face recognition [43, 44, 33, 36], large-scale image classification [23, 39, 46, 11, 13], and fine-grained object classification [31, 18, 21, 48]. Meanwhile, softmax layer and the training strategy of SGD together with back-propagation (BP) become the default components, and are generally applied in most of the aforementioned works.

It is widely observed that when optimizing with SGD and BP, the smooth and free gradients propagation is crucial to improve the training of DCNNs. For example, replacing sigmoid activation function with the piecewise-linear activation functions such as ReLU and PReLU [12] handles the problem of gradients vanishing caused by sigmoid saturation and, allows the training of much deeper networks. While, it is interesting that softmax activation function (illustrated in Figure 1) is implicitly like sigmoid function due to their similar formulation (shown in Sec. 3), and has the saturation behavior as well when its input is large. However, many take the softmax activation for granted and the problem behind its saturation behavior is omitted as a result of illusion of performance improvements based on DCNNs.

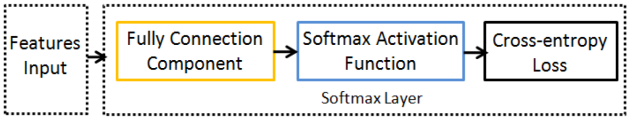

Figure 1. Decomposition of typical softmax layer in DCNN. It can be rewritten into three parts: fully connection component, softmax activation and cross-entropy loss.

In standard SGD, the saturation behavior of softmax turns up when its output is very close to the ground truth, certainly it is our goal of model training. However, in some ways, it is a barrier to improve the generalization ability of CNNs especially when it shows up early (inopportune). Concretely, for one instance input, it will early stop contributing gradients to BP when its softmax output is prematurely saturated, yielding short-lived gradients propagation in history which is not enough for robust learning. And in this case, the learning process with SGD and BP hardly explore more due to poor gradients propagation and parameters update. We define this saturation behavior as *individual saturation* and the corresponding individual as saturated one. As the training going, the number of non-saturated contributing training samples gradually decreases and the robust learning of network will be impeded. It sometimes is a reason for algorithm going to bad local-minima[1] and being difficult to escape. Furthermore, the problem of over-

[1] For simplicity, we use local or 'global' minima to represent a neighbouring region not a single point.

fitting turns up. To this end, we need to give SGD chances to explore more parts of parameter space and the early individual saturation is undesired.

In this paper, we propose Noisy Softmax, a novel technique of early softmax desaturation, to address the aforementioned issue. This is mainly achieved by injecting annealed noise directly into softmax activations during each iteration. In another word, Noisy Softmax allows SGD to escape from a bad local-minima and explore more by postponing the early individual saturation. Furthermore, it improves the generalization ability of system by reducing over-fitting as a direct consequence of more exploration. The main contributions of this work are summarized as follows:

- We provide an insight of softmax saturation, interpreted as *individual saturation*, that early individual saturation produces short-lived gradients propagation which is poor for robust exploration of SGD and further causes over-fitting unintentionally.
- We propose **Noisy Softmax** to aim at producing rich and continuous gradients propagation by injecting annealed noise into softmax activations. It allows the 'global' convergence of SGD solver and aids generalization by reducing over-fitting. To our knowledge, it is the first attempt to address the early saturation behavior of softmax by adding noise.
- Noisy Softamx can be easily performed as a drop-in replacement for standard softmax and optimized with standard SGD. It can be also applied in other performance-improving techniques, such as neural activation functions and network architectures.
- Extensive experiments have been performed on several datasets, including MNIST [26], CIFAR10/100 [22], LFW [17], FGLFW [54] and YTF [49]. The impressive results demonstrate the effectiveness of Noisy Softmax.

2. Related Work

Many promising techniques have been developed, such as novel network structures [30, 13, 41], non-linear activation functions [12, 7, 6, 38], pooling strategies [11, 8, 53] and objective loss functions [43, 36], *etc*.

These approaches are mostly optimized with SGD and back-propagation. In standard SGD, we use the chain rule to compute and propagate the gradients. Thus, any saturation behavior of neuron units or layer components[2] are undesired because the training of deeper framework attributes to smooth and free flow of gradients information. The early solution is to replace sigmoid function with non-linear piecewise activation function [23]. This is neuron desaturation. Skip connections between different layers exponentially expands the paths of propagation [41, 11, 13, 16, 15]. These belong to layer desaturation, since the forward and backward information can be directly propagated from one layer to any other layer without gradients vanishing. In contrast, only the early saturation behavior is harmful instead of all of them, we focus on the early desaturation of softmax, which hasn't been investigated, and we achieve this by injecting noise explicitly into softmax activations.

There are some other works that are related to noise injection. Adding noise to ReLU has been developed to encourage components to explore more in Boltzmann machines and feed-forward networks [4, 1]. Adding noise to sigmoid provides possibilities of training with much wider family of activation functions than previous [10]. Adding weight noise [42], adaptive weight noise [9, 3] and gradients noise [32] also improve the learning. Adding annealed noise can help the solver escape from a bad local-minima and find a better one. We follow these inspiring ideas to address individual saturation and encourage SGD to explore more. The main differences are that we apply noise injection on CNN and impose noise on loss layer instead of previous layers. But different from adding noise on loss layer in DisturbLabel [51], a method that seems weird to disturb labels but indeed improves the performances of models, our work has a clear object of the delay of early softmax saturation by explicitly injecting noise into softmax activition.

Another noise injection way is randomly transforming the input data, which is commonly referred to data augmentation, such as randomly cropping, flipping [23, 50], rotating [25, 24] and jittering input data [34, 35]. And our work can also be interpreted as a way of data augmentation which will be discussed in the following discussion part.

3. Early Individual Saturation

In this section, we will give a toy example to describe the early individual saturation of softmax, which is always omitted, and analyse its impact on generalization. Define the i-th input data x_i with the corresponding label y_i, $y_i \in [1 \cdots C]$. Then processing training images with standard DCNN, we can obtain the cross-entropy loss and partial derivative as follows:

$$L = -\frac{1}{N}\sum_i \log P(y_i|x_i) = -\frac{1}{N}\sum_i \log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \quad (1)$$

$$\frac{\partial L}{\partial f_j} = P(y_i = j|x_i) - 1\{y_i = j\} = \frac{e^{f_j}}{\sum_k e^{f_k}} - 1\{y_i = j\} \quad (2)$$

where f_j refers to the j-th element of the softmax input vector $\mathbf{f}$, $j \in [1 \cdots C]$, N is the number of training images. $1\{condition\} = 1$ if *condition* is satisfied and $1\{condition\} = 0$ if not.

To simplify our analysis, we consider the problem of bi-

[2] The saturation of layer refers to gradients vanishing at a certain layer in back-propagation.

nary classification[3], where $y_i \in [1, 2]$. Under binary scenario, we plot the softmax activation for class 1 in Figure 2. Intuitively, the softmax activation is totally like sigmoid function. The standard softmax encourages $f_1 > f_2$ in order to classify class 1 correctly and can be regarded as a genius when its output $P(y_i = 1|x_i) = \frac{1}{1+e^{-(f_1-f_2)}}$ is very close to 1. In this case, the softmax output of data x_i is saturated and we define this as individual saturation. Of course, making its softmax output close to 1 is our ultimate goal of CNN training. However, we would like to achieve it at the end of SGD exploration not in the beginning or middle stage. Since, when optimizing CNN with gradients-based methods such as SGD, the prematurely saturated individual early stops contributing gradients to back-propagation due to negligible gradients, i.e. $P(y_i = 1|x_i) \approx 1, \frac{\partial L}{\partial f_{y_i}} \approx 0$ (see Eq. 2). And with the saturated individuals number rising, the amount of contributing data decreases and, SGD has few chances to move around and is more likely to converge at a local minima, therefore, it is easy to be over-fitting and it requires extra data to recover. In short, the early saturated ones introduce short-lived gradients propagation which is not enough to help system converge at a 'global minima' (i.e. a better local-minima), so the early individual saturation is undesired.

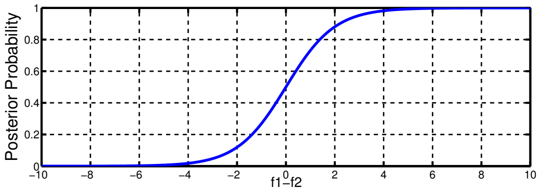

Figure 2. Softmax activation function: $\frac{1}{1+e^{-(f_1-f_2)}}$. X axis represents the difference between f_1 and f_2.

4. Noisy Softmax

Based on the fact analysed in Section 3, the short-lived gradients propagation caused by early individual saturation would not guide the robust learning. Thus the intuitive solution is to **set up 'barrier' along its way to satuation so as to postpone the early saturation behavior and produce rich and continuous gradients propagation**. Particularly, for training data point (x_i, y_i), a simple way to achieve this is to factitiously reduce its softmax input f_{y_i} (note that, it is theoretically same to enlarge $f_j, \forall j \neq y_i$, but it is so complex to operate). Moreover, many research works point out that adding noise gives the system chances to find 'global minima', such as injecting noise to sigmoid [10]. We follow this inspiring idea to address the problem of early individual saturation. Therefore, our technique of slowing down the early saturation is to inject appropriate noise in softmax input f_{y_i}, and the resulting noise-associated one is as follows:

$$f_{y_i}^{noise} = f_{y_i} - n \tag{3}$$

where $n = \mu + \sigma\xi$, $\xi \sim \mathcal{N}(0, 1)$, μ and σ are used to generate a wider family of noise from ξ. Intuitively, we would prefer $f_{y_i}^{noise}$ to be less than f_{y_i} (because $f_{y_i}^{noise} > f_{y_i}$ will speed up saturation). Thus, we simply require noise n to be always positive, and we have the following form:

$$f_{y_i}^{noise} = f_{y_i} - \sigma|\xi| \tag{4}$$

where noise n has mean 0 and standard variance σ.

Moreover, we would like to make our noise annealed by controlling the parameter σ. Considering our initial thought, we intend to postpone the early saturation of x_i instead of to not allow its saturation, implying that the initially larger noise is required to boost the exploration ability and later the relatively smaller noise is required for model convergence.

In standard Softmax layer (Figure 1), f_{y_i} is also the output of the fully connected component and can be written as $f_{y_i} = W_{y_i}^T X_i + b_{y_i}$ where W_{y_i} is the y_i-th column of W, X_i is the input feature of this layer from training data x_i and b_{y_i} is the basis. Since b_{y_i} is a constant and f_{y_i} mostly depends on $W_{y_i}^T X_i$, we construct our annealed noise by making σ to be related to $W_{y_i}^T X_i$. In consideration of the fact that $W_{y_i}^T X_i = \|W_{y_i}\|\|X_i\| \cos\theta_{y_i}$, where θ_{y_i} is the angle between vector W_{y_i} and X_i, σ should be a joint function of $\|W_{y_i}\|\|X_i\|$ and θ_{y_i} which hold amplitude and angular information respectively. Parameter W_{y_i} followed by a loss function can be regarded as a linear classifier of class y_i. And this linear classifier uses cosine similarity to make angular decision boundary. As a result, with the converging of system, the angle θ_{y_i} between W_{y_i} and X_i will gradually decrease. Therefore, our annealed-noise-associated softmax input is formulated as:

$$f_{y_i}^{noise} = f_{y_i} - \alpha\|W_{y_i}\|\|X_i\|(1 - \cos\theta_{y_i})|\xi| \tag{5}$$

where $\alpha\|W_{y_i}\|\|X_i\|(1-\cos\theta_{y_i}) = \sigma$, and hyper-parameter α is used to adjust the scale of noise. In our annealed noise, we leverage $\|W_{y_i}\|\|X_i\|$ to make the magnitude of the noise and f_{y_i} to be comparable, and use $(1 - \cos\theta_{y_i})$ to adaptively anneal the noise. Notably, our early desaturation work implies that make softmax later saturated instead of non-saturated. We experimented with various function types of σ and empirically found that this surprising simple formulation performs better. Putting Eq. 5 into original softmax, the Noisy Softmax loss is defined as:

$$L = -\frac{1}{N}\sum_i \log \frac{e^{f_{y_i} - \alpha\|W_{y_i}\|\|X_i\|(1-\cos\theta_{y_i})|\xi|}}{\sum_{j \neq y_i} e^{f_j} + e^{f_{y_i} - \alpha\|W_{y_i}\|\|X_i\|(1-\cos\theta_{y_i})|\xi|}} \tag{6}$$

Optimization. We use Eq. 6 throughout our experiments and optimize our model with the commonly used SGD. Thus we need to compute the forward and backward propagation, and $\cos\theta_{y_i}$ is required to be replaced with

[3] Multi-classification complicates our analysis but has the same mechanism as binary scenario.

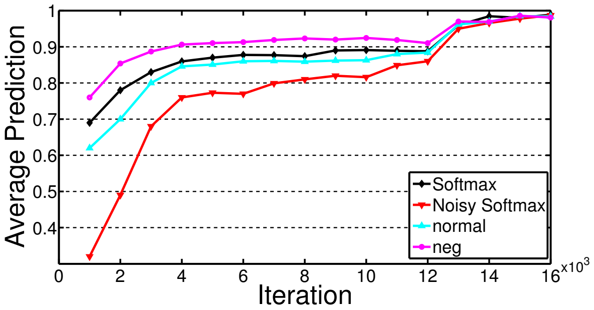

Figure 3. Saturation status vs. iteration with different formulations of noise. Normal and Neg represent *normal* noise and *negative* noise respectively. α^2 is set to 0.1 in our experiments.

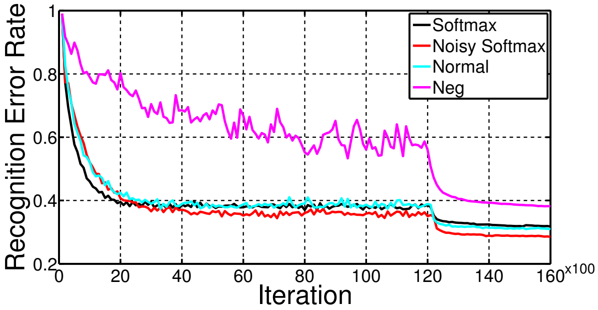

Figure 4. CIFAR100 testing error vs. iteration with different formulations of noise. Normal and Neg represent *normal* noise and *negative* noise respectively. α^2 is set to 0.1 in our experiments.

$\frac{W_{y_i}^T X_i}{\|W_{y_i}\|\|X_i\|}$. For forward and backward propagation, the only difference between Noisy Softmax loss and standard softmax loss exists in f_{y_i}. For example, in forward propagation, $\forall j \neq y_i$ f_j is computed as the same as original softmax while f_{y_i} is replaced with $f_{y_i}^{noise}$. In backward propagation, $\frac{\partial L}{\partial X_i} = \sum_j \frac{\partial L}{\partial f_j}\frac{\partial f_j}{\partial X_i}$ and $\frac{\partial L}{\partial W_{y_i}} = \sum_j \frac{\partial L}{\partial f_j}\frac{\partial f_j}{\partial W_{y_i}}$, only when $j = y_i$ the computations of $\frac{\partial f_j}{\partial X_i}$ and $\frac{\partial f_j}{\partial W_{y_i}}$ are not the same as original softmax which are listed as follows:

$$\frac{\partial f_{y_i}^{noise}}{\partial X_i} = W_{y_i} - \alpha|\xi|(\frac{X_i\|W_{y_i}\|}{\|X_i\|} - W_{y_i}) \tag{7}$$

$$\frac{\partial f_{y_i}^{noise}}{\partial W_{y_i}} = X_i - \alpha|\xi|(\frac{W_{y_i}\|X_i\|}{\|W_{y_i}\|} - X_i) \tag{8}$$

For simplicity, we leave out $\frac{\partial L}{\partial f_j}$ and $\frac{\partial f_j}{\partial X_i}$, $\frac{\partial f_j}{\partial W_{y_i}}(\forall j \neq y_i)$ since they are the same for both Noisy Softmax and original softmax. In short, except for when $j = y_i$, the overall computation of Noisy Softmax is similar with original softmax.

5. Discussion

5.1. The Effect of Noise Scale α

In Noisy Softmax, the scale of annealed noise is largely determined by the hyper-parameter α. Here we can imagine that, when $\alpha = 0$ in the 0 noise limit, Noisy Softmax is the same with ordinary softmax. Then individual saturation will turn up and SGD solver has a high chance to converge at a local-minima. Without extra data for training, the model will be easily over-fitting. However, when α is large enough, large gradients are obtained since backpropagating through $f_{y_i}^{noise}$ gives rise to large derivatives. So the algorithm just see the noise instead of real signal and move around anywhere blindly. Hence, a relatively small α is required to aid the generalization of model.

We evaluate the performances of Noisy Softmax with different α on several datasets. Note that, the value of α is not carefully adjusted and we make $\alpha = 0$ (i.e. softmax) as our baseline. And these comparison results are listed in Table 3, 4and 5. One can observe that Noisy Softmax with a relatively appropriate α (e.g. $\alpha^2 = 0.1$) obtains better recognition accuracy than ordinary softmax on all of the datasets. To be intuitional, we summarize the results on CIFAR100 in Figure 6. When $\alpha^2 = 0.1$, our method outperforms the original softmax. This demonstrates that our Noisy Softmax does improve the generalization ability of CNN by encouraging the SGD solver to be more exploratory and to converge at a 'global-minima'. When α rises to 1, the large noise causes the network to converge slower and produces worse performance than baseline as well. Since the large noise drowns the helpful signal and solver just sees the noise.

5.2. Saturation Study

To illuminate the significance of early softmax desaturation based on non-negative noise injection, we investigate the impacts of different formulations of noise, such as *normal* noise $n = \sigma\xi$ and *negative* noise $n = -\sigma|\xi|$ (σ is the same as in Eq. 5), on individual saturation. From the formulations of noise, we can imagine that there will be more saturated instances when training with *negative* noise (which is a counterexample). To intuitively analyse the saturation state, we compute the average possibility prediction over the entire training set as follows:

$$\overline{P} = \frac{1}{N}\sum_{j=1}^{C}\sum_{i=1}^{N_j} P(y_i|x_i) \tag{9}$$

where C is the number of classes and N_j is the number of images within the j-th class. Figure 3 and 4 show the saturation status of different noise and testing error rates on CIFAR100 respectively.

From the results in Figure 3, one can observe that when training with original softmax or *negative* noise, the average prediction rises quickly to a relatively high level, almost 0.9, implying that the early individual saturation is serious, and finally goes up to nearly 1. Moreover, the average prediction of *negative* noise is higher than that of softmax, implying that the early individual saturation is deteriorated since many instances are factitiously mapped to

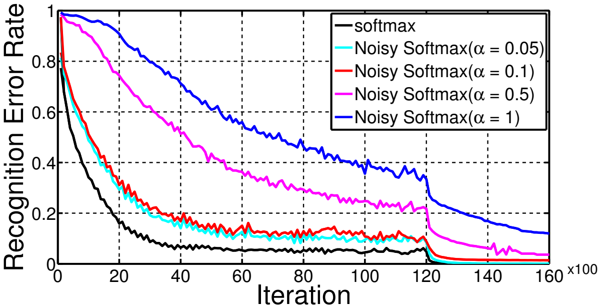

Figure 5. CIFAR100 training error vs. iteration with different α.

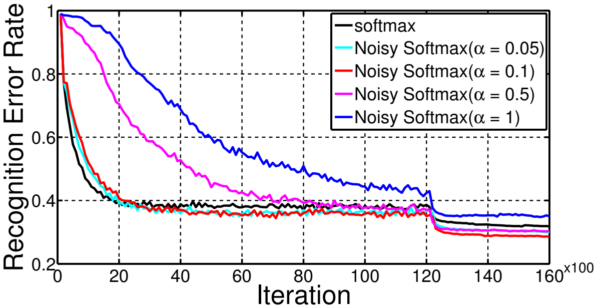

Figure 6. CIFAR100 testing error vs. iteration with different α.

saturated ones. From the results in Figure 4, one can observe that the testing error of $negative$ noise drops slowly and finally achieves a relatively high level, nearly 37%, verifying that the exploration of SGD is seriously impeded by early individual saturation. In $normal$ noise case, the testing error and the trend of average prediction rise are similar with original softmax, shown in Figure 4 and 3 respectively, since the expectation $E(n)$ is close to zero.

In contrast, when training with Noisy Softmax, the average prediction rises slowly and is much lower than original softmax at the early training stage, shown in Figure 3, verifying that the early individual saturation behavior is significantly avoided. And from the results in Figure 4, one can observe that Noisy Softmax outperforms the baseline and significantly improves the performance to 28.48% testing error rate. Note that, after 3,000 iterations, our method achieves better testing error result but lower average-prediction, demonstrating that the early desaturation of softmax gives SGD solver chances to traverse more portions of parameter space for optimal solution. As the noise level is decreased, it will prefer a better local-minima where signal gives strong response to SGD. Then the solver will spend more time to explore this region and converge, which can be regarded as 'global-minima', in a finite number of steps.

In summary, injecting non-negative noise $n = \sigma|\xi|$ in softmax does prevent the early individual saturation and further improve the generalization ability of CNN when optimized by standard SGD.

5.3. Annealed Noise Study

When addressing the early individual saturation, the key idea is to add annealed noise. In order to highlight the superiority of our annealed noise described in Sec. 4, we compare it to $free$ noise $n = \alpha|\xi|$ and $amplitude$ noise $n = \alpha\|W_{y_i}\|\|X_i\||\xi|$. We evaluate them on CIFAR100 and the results are listed in Table 1. From the results, it can be observed that our Noisy Softmax outperforms the other two formulations of noise. In $free$ noise case, where σ (described in Sec. 4) is set to a fixed value α and the noise is totally independent, although adding this noise is a desaturation operation the accuracy gain over the baseline is small, since it desaturates softmax worse not according to the magnitude of softmax input, in another word it cannot suit the remedy to the case. In $amplitude$ noise case, where σ is set to $\alpha\|W_{y_i}\|\|X_i\|$, the subtractive noise is prudent due to considering the level of softmax input, thus yielding better accuracy gain than $free$ noise. While it still is worse than Noisy Softmax. Because, in both Noisy Softmax and $amplitude$ noise cases, as the exploration going, the 'globally better' region of parameter space has been seen by SGD and it is time to be patient in exploring this region, in another word smaller noise is required. Noisy Softmax holds this idea by annealing the noise but, in $amplitude$ noise case, the level-unchanging noise seems a little large at this time and further causes difficulty in detailed learning. Reviewing the formulation of our annealed noise, one can observe that our annealed noise is constructed by combining θ_{y_i}, a time identifier, into $amplitude$ noise. With time function $1 - \cos\theta_{y_i}$ injected, the noise will be adaptively decreased.

α^2	Noisy Softmax	$free$	$amplitude$
0	31.77	31.77	31.77
0.05	29.99	31.43	30.96
0.1	**28.48**	31.04	29.97
0.5	30.22	30.88	fail
1	35.23	31.20	fail

Table 1. Testing error rates(%) vs. different noise on CIFAR100.

5.4. Regularization Ability

We find experimentally that Noisy Softmax can regularize the CNN model by preventing over-fitting. Figure 5 and 6 show the recognition accuracy results on CIFAR100 dataset with different α. One can observe that without noise injection (i.e. $\alpha = 0$), the training recognition error drops fast to quite a low level, almost 0%, however the testing recognition error stops falling at a relatively high level, nearly 31.77%. Conversely, when α^2 is set to an appropriate value such as 0.1, the training error drops slower and is much higher than the baseline. But the testing error reaches a lower level, nearly 28.48%, and still has a trend of decreasing. Even when $\alpha^2 = 0.5$, the training error is higher but the testing error becomes lower as well, nearly 30.22%. This demonstrates that encouraging SGD to converge at a better local-minima indeed prevent over-fitting and Noisy

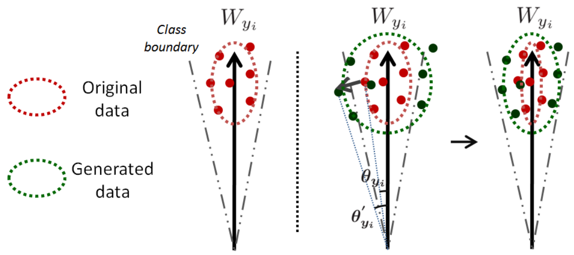

Figure 7. Geometric Interpretation of data augmentation.

Softmax has a strong regularization ability.

As analysed above, our Noisy Softmax can be regarded as a kind of regularization technique for preventing overfitting by making SGD to be more exploratory. Here we will analyse this regularization ability from another data augmentation perspective, which has a profound physical interpretation. Under the original case, the softmax input coming from data point (x_i, y_i) is $f_{y_i} = \|W_{y_i}\|\|X_i\|\cos\theta_{y_i}$ (we omit the constant b_{y_i} for simplicity). Now we consider a new input $(x_i^{'}, y_i)$, where $\|X_i^{'}\| = \|X_i\|$ and the angle $\theta_{y_i}^{'}$ between vector W_{y_i} and $X_i^{'}$ is $\arccos((1+\alpha|\xi|)\cos\theta_{y_i} - \alpha|\xi|)$. Thus we have $f_{y_i}^{'} = \|W_{y_i}\|\|X_i^{'}\|\cos\theta_{y_i}^{'} = W_{y_i}^T X_i - \alpha\|W_{y_i}\|\|X_i\|(1 - \cos\theta_{y_i})|\xi| = f_{y_i}^{noise}$, implying that $f_{y_i}^{noise}$ can be regarded as coming from a new data point $(x_i^{'}, y_i)$. Notably, since $\theta_{y_i}^{'} > \theta_{y_i}$, these generated data have many boundary examples which are much helpful for discriminative feature learning, illustrated in Figure 7. In summary, generating the noisy input $f_{y_i}^{noise}$ is equivalent to generating new training data, which is an efficient way of data augmentation.

To verify our discussion above, we evaluate Noisy Softmax on two subsets of the MNIST dataset, which have only $600(1\%)$ and $6000(10\%)$ training instances respectively. Our CNN configuration is shown in Table 2. With the same training strategy in Section 6.1, we achieve 3.82% and 1.30% testing error rates on the original testing set, respectively. Meanwhile, in both cases, the training error rates quickly drop to nearly 0% which show that the overfitting turns up. However, when training with Noisy Softmax ($\alpha^2 = 0.5$), we obtain 2.46% and 0.93% testing error rates, respectively. This demonstrates that Noisy Softmax improves the generalization ability of CNN with implicit data augmentation. And from the accuracy improvements on these two subsets and CIFAR100 (which has 500 instances per class), it acts as an effective algorithm especially in the case that the amount of training data is limited.

5.5. Relationship to Other Methods

Multi-task learning: combining several tasks to one system does improve the generalization ability [37]. Considering a multi-task learning system with input data x_i, the overall object loss function is a combination of several subobject loss functions, written as $L = \sum_j L_j(\vartheta_0, \vartheta_j, x_i)$, where ϑ_0 and $\vartheta_j, j \in [1, 2, \cdots]$ are generic parameters and task-specific parameters respectively. Optimizing with standard SGD, generic parameters ϑ_0 are updated as $\vartheta_0 = \vartheta_0 - \gamma(\sum_j \frac{\partial L_j}{\partial \vartheta_0})$, γ is the learning rate. While in Noisy Softmax, from an overall training perspective, our loss function can also be regarded as a combination of many noise-dependent changing losses $L_{noise_k} = -\log \frac{e^{f_{y_i}}}{\sum_{c \neq y_i} e^{f_c} + e^{f_{y_i}^{noise}}} + \alpha|\xi|_k\|W_{y_i}\|\|X_i\|(1 - \cos\theta_{y_i}), k \in [1, m]$, i.e. $L = \sum_{k=1}^{m} L_{noise_k}$ where m is an uncertain number and is related to noise scale and iteration number. Thus, the overall contribution to system can be regarded as $\vartheta = \vartheta - \gamma(\sum_{k=1}^{m} \frac{\partial L_{noise_k}}{\partial \vartheta})$. So our method can be regarded as a special case of multi-task learning, where the task-specific parameters are shared across tasks.

However, in multi-task learning system, factitiously designing task-specific losses is prohibitively expensive and the number of tasks is limited and small. While in Noisy Softmax training procedure, the model is constrained by many randomly generated tasks (quantified by L_{noise_k}). Thus, training a model with Noisy Softmax can be regarded as training with massive tasks that are very costly and often infeasible to design in original multi-task learning system.

Noise injection: some research works inject noise in previous layers of neural networks such as in neuron activation functions ReLU [4, 1] and sigmoid [10], in weights [9, 3] and gradients [32]. We emphasize that Noisy Softmax adds noise on a single loss layer instead of on many previous layers, which is more convenient and efficient for implementation and model training, and is applied in DCNNs. Intrinsically different from DisturbLabel [51], where noise is produced by disturbing labels and also exerts effect on loss layer, Noisy Softmax starts from a clear object of early softmax desaturation and the noise is adaptively annealed and injected in a explicit manner.

Desaturation: many other desaturation works, such as replacing sigmoid with ReLU [6] and building skip connections between layers [41, 11, 13, 15], solve the problems of gradients vanishing which happen in bottom layers. While our Noisy Softmax solves the problem of early gradients vanishing in the top layer (i.e. loss layer) which is caused by the early individual saturation. We emphasize that solving the early gradients vanishing in top layer is crucial to parameters update and model optimization, since top layer is the source of gradients propagation. In summary, by postponing the early individual saturation we can obtain continuous gradients propagation from the top layer and further encourage SGD to be more exploratory.

6. Experiments and Results

We evaluate our proposed Noisy Softmax algorithm on several benchmark datasets, including MNIST [26], CIFAR10/100 [22], LFW [17], FGLFW [54] and YTF [49]. Note that, in all of our experiments, we only use a single

Layer	MNIST(for Sec. 5.4)	MNIST	CIFAR10/10+	CIFAR100	LFW/FGLFW/YTF
Block1	[3x3,40]x2	[3x3,64]x3	[3x3,64]x4	[3x3,96]x4	[3x3,64]x1
Pool1	Max [2x2], stride 2				
Block2	[3x3,60]x1	[3x3,64]x3	[3x3,128]x4	[3x3,192]x4	[3x3,128]x1
Pool2	Max [2x2], stride 2				
Block3	[3x3,60]x1	[3x3,64]x3	[3x3,256]x4	[3x3,384]x4	[3x3,256]x2
Pool3	Max [2x2], stride 2				
Block4	-	-	-	-	[3x3,512]x3,padding 0
Fully Connected	100	256	512	512	3000

Table 2. CNN architectures for different benchmark datasets. Blockx denotes a container of several convolution components with the same configuration. E.g. [3x3, 64]x5 denotes 5 cascaded convolution layers with 64 filters of size 3x3.

model for the evaluation of Noisy Softmax, and both softmax and Noisy Softmax in our experiments use the same CNN architecture shown in Table 2.

6.1. Architecture Settings and Implementation

As VGG [39] becomes a commonly used CNN architecture, the cascaded layers with small size filters gradually take the place of single layer with large size filters. Since these cascaded layers have less parameters, lower computational complexity and stronger representation ability compared to the single layer. E.g. a single 5x5 convolution layer is replaced with 2 cascaded 3x3 convolution layers. Inspired by this, we design our architectures as shown in Table 2. In convolution layers, both the stride and padding are set to 1 if not specified. In pooling layers, we use 2x2 max-pooling filter with stride 2. We adopt the piece-wise linear functions PReLU [12] as our neuron activation functions. Then we use the weight initialization [12] and batch normalization [19] in our networks. All of our experiments are implemented by Caffe library [47] with our own modifications. We use standard SGD to optimize our CNNs and the batch sizes are 256 and 200 for object experiments and face experiments, respectively. For data preprocessing, we just perform the mean substraction.

Training. In object recognition tasks, the initial learning rate is 0.1 and is divided by 10 at 12k. The total iteration is 16k. Note that, although we train our CNNs with coarsely adjusted learning rate, the results of all experiments are impressive and consistent, verifying the effectiveness of our method. For face recognition tasks, we start with a learning rate of 0.01, divide it by 5 when the training loss does not drop.

Testing. We use the original softmax to classify the testing data in object datasets. In face datasets, we evaluate it with the cosine distance rule after PCA reduction for face recognition.

6.2. Evaluation on MNIST Dataset

MNIST [26] contains 60,000 training samples and 10,000 testing samples. These samples are uniformly distributed over 10 classes. And all samples are 28x28 gray images. Our CNN network architecture is shown in Table 2 and we use a 0.001 weight decay. The results of the state-of-the-art methods and our proposed Noisy Softmax with different α are listed in Table 3.

From the results, our Noisy Softmax ($\alpha^2 = 0.1$) not only outperforms the original softmax over the same architecture, but also achieves the competitive performance compared to the state-of-the-art methods. It can also be observed that Noisy Softmax produces consistent accuracy gain with coarsely adjusted α^2, such as 0.05, 0.1 and 0.5, and our method achieves the same accuracy with DisturbLabel [51] which adds dropout to several layers, demonstrating the effectiveness of our technique.

6.3. Evaluation on CIFAR Datasets

CIFAR [22] has two evaluation protocols over 10 and 100 classes respectively. CIFAR10/100 has 50,000 training samples and 10,000 testing samples, all the samples are 32x32 RGB images. And these images are uniformly distributed over 10 or 100 classes. We use different CNN architectures in CIFAR10 and CIFAR100 experiments, and these network configurations are shown in Table 2.

We evaluate our method on CIFAR10 and CIFAR100, as these results are shown in Table 4. For data augmentation, we perform a simple method: randomly crop a 30*30 image. From our experimental results, one can observe that Noisy Softmax ($\alpha^2 = 0.1$) outperforms all of the other methods on these two datasets. And it improves nearly 1% and more than 3% accuracies over the baseline on CIFAR10 and CIFAR100 respectively.

6.4. Evaluation on Face Datasets

LFW[17] contains 13,233 images from 5749 celebrities. Under unrestricted conditions, it provides 6,000 face pairs for verification protocol and, closed-set and open-set for identification protocol adopted in [2].

FGLFW [54] is a derivative of LFW, implying that the images are all coming from LFW but the face pairs are difficult to classify whether they are from the same person. It is a light and sweet dataset for performance evaluation due to the simple verification protocol but challenging face pairs.

YTF [49] provides $5,000$ video pairs for face verification. We use the average representation of 100 randomly

Method	MNIST
CNN [20]	0.53
NiN [30]	0.47
Maxout [7]	0.45
DSN [28]	0.39
R-CNN [29]	0.31
GenPool [27]	0.31
DisturbLabel [51]	0.33
Softmax	0.43
Noisy Softmax ($\alpha^2 = 1$)	0.42
Noisy Softmax ($\alpha^2 = 0.5$)	**0.33**
Noisy Softmax ($\alpha^2 = 0.1$)	**0.33**
Noisy Softmax ($\alpha^2 = 0.05$)	0.37

Table 3. Recognition error rates (%) on MNIST.

Method	CIFAR10	CIFAR10+	CIFAR100
NiN [30]	10.47	8.81	35.68
Maxout [7]	11.68	9.38	38.57
DSN [28]	9.69	7.97	34.57
All-CNN [40]	9.08	7.25	33.71
R-CNN [29]	8.69	7.09	31.75
ResNet [13]	N/A	6.43	N/A
DisturbLabel [51]	9.45	6.98	32.99
Softmax	8.11	6.98	31.77
Noisy Softmax ($\alpha^2 = 1$)	9.09	8.77	35.23
Noisy Softmax ($\alpha^2 = 0.5$)	7.84	7.13	30.22
Noisy Softmax ($\alpha^2 = 0.1$)	**7.39**	**6.36**	**28.48**
Noisy Softmax ($\alpha^2 = 0.05$)	7.58	6.61	29.99

Table 4. Recognition error rates (%) on CIFAR datasets. + denotes data augmentation.

Method	Images	Models	LFW	Rank-1	DIR@FAR=1%	FGLFW	YTF
FaceNet [36]	200M*	1	99.65	-	-	-	95.18
DeepID2+ [44]	300k*	1	98.7	-	-	-	91.90
DeepID2+ [44]	300k*	25	99.47	95.00	80.70	-	93.20
Sparse [45]	300k*	1	99.30	-	-	-	92.70
VGG [33]	2.6M	1	97.27	74.10	52.01	88.13	92.80
WebFace [52]	WebFace	1	97.73	-	-	-	90.60
Robust FR [5]	WebFace	1	98.43	-	-	-	-
Lightened CNN [50]	WebFace	1	98.13	89.21	69.46	91.22	91.60
Softmax	WebFace$^+$	1	98.83	91.68	69.51	92.95	94.22
Noisy Softmax($\alpha^2 = 0.1$)	WebFace$^+$	1	**99.18**	**92.68**	**78.43**	**94.50**	**94.88**
Noisy Softmax($\alpha^2 = 0.05$)	WebFace$^+$	1	99.02	92.24	75.67	94.02	94.51

Table 5. Recognition accuracies (%) on LFW, FGLFW and YTF datasets. * denotes the images are not publicly available and $^+$ denotes data expansion. In LFW, closed-set and open-set accuracies are evaluated by Rank-1 and DIR@FAR=1 respectively.

selected samples from each video for evaluation.

For data preprocessing, we align and crop images based on eyes and mouth centers, yielding 104×96 RGB images. Our CNN configuration is shown in Table 2, here we add element-wise maxout layer [50] after the 3,000-dimensional fully connected layer, yielding a 1,500-dimensional output, and contrastive loss is applied on this output as in DeepID2 [43]. Then we train a single CNN model with outside data from the publicly available CASIA-WebFace dataset [52] and our own collected data (about 400k from 14k identities). Extract the features for each image and its horizontally flipped one, then compute a mean feature vector as the representation. From the results shown in Table 5, one can observe that Noisy Softmax ($\alpha^2 = 0.1$) improves the performance over the baseline, and the result is also comparable to the current state-of-the-art methods with private data and even model ensemble. In addition, we further improve our results to $\mathbf{99.31\%, 94.43\%, 82.50\%, 94.88\%, 95.37\%}$ (listed in the same protocol order as in Table. 5) by two models ensemble.

7. Conclusion

In this paper, we propose Noisy Softmax to address the early individual saturation by injecting annealed noise to the softmax input. It is a way of early softmax desaturation by postponing the early individual saturation. We show that our method can be easily performed as a drop-in replacement for standard softmax and is easier to optimize. It significantly improves the performances of CNN models, since the early desaturation operation indeed exerts much effect on parameter update during back-propagation and furthermore improves the generalization ability of DCNNs. Empirical studies verify the superiority of softmax desaturation. Meanwhile, it achieves state-of-the-art or competitive results on several datasets.

8. Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Project 6157306861471048, 61375031 and 61532006), Beijing Nova Program under Grant No. Z161100004916088, the Fundamental Research Funds for the Central Universities under Grant No. 2014ZD03-01, and the Program for New Century Excellent Talents in University(NCET-13-0683).

References

[1] Y. Bengio. Estimating or propagating gradients through stochastic neurons. *Computer Science*, 2013.

[2] L. Best-Rowden, H. Han, C. Otto, B. F. Klare, and A. K. Jain. Unconstrained face recognition: Identifying a person of interest from a media collection. *IEEE Transactions on Information Forensics and Security*, 9(12):2144–2157, 2014.

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *Computer Science*, 2015.

[4] by V Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. *Proc Icml*, pages 807–814, 2015.

[5] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation for multimedia applications. *IEEE Transactions on Multimedia*, 17(11):2049–2058, 2015.

[6] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15, 2010.

[7] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *Computer Science*, pages 1319–1327, 2013.

[8] B. Graham. Fractional max-pooling. *Eprint Arxiv*, 2014.

[9] A. Graves. Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[10] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio. Noisy activation functions. 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Computer Science*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. pages 1026–1034, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. 2016.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *Computer Science*, 3(4):pgs. 212–223, 2012.

[15] G. Huang, Z. Liu, and K. Weinberger. Densely connected convolutional networks. 2016.

[16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. 2016.

[17] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. 2008.

[18] S. Huang, Z. Xu, D. Tao, and Y. Zhang. Part-stacked cnn for fine-grained visual categorization. *Computer Science*, 2015.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Computer Science*, 2015.

[20] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. Lecun. What is the best multi-stage architecture for object recognition? pages 2146–2153, 2009.

[21] J. Krause, H. Jin, J. Yang, and F. F. Li. Fine-grained recognition without part annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. 2012.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(2):2012, 2012.

[24] D. Laptev and J. M. Buhmann. Transformation-invariant convolutional jungles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3043–3051, 2015.

[25] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. Tipooling: transformation-invariant pooling for feature learning in convolutional neural networks. 2016.

[26] Y. Lecun and C. Cortes. The mnist database of handwritten digits.

[27] C. Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *Computer Science*, 2015.

[28] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 6, 2015.

[29] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

[30] M. Lin, Q. Chen, and S. Yan. Network in network. *Computer Science*, 2013.

[31] T. Y. Lin, A. Roychowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[32] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *Computer Science*, 2015.

[33] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[34] R. Reed, R. J. Marks, and S. Oh. An equivalence between sigmoidal gain scaling and training with noisy (jittered) input data. In *Neuroinformatics and Neurocomputers, 1992., RNNS/IEEE Symposium on*, pages 120 – 127, 1992.

[35] R. Reed, S. Oh, and R. J. Marks. Regularization using jittered training data. In *International Joint Conference on Neural Networks*, pages 147–152 vol.3, 1992.

[36] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. pages 815–823, 2015.

[37] M. L. Seltzer and J. Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. pages 6965–6969, 2013.

[38] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. 2016.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2015.

[40] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *Eprint Arxiv*, 2014.

[41] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *Computer Science*, 2015.

[42] M. Steijvers and P. Grunwald. A recurrent network that performs a context-sensitive prediction task. In *Conference of the Cognitive Science*, 2000.

[43] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. *Advances in Neural Information Processing Systems*, 27:1988–1996, 2014.

[44] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. *Computer Science*, pages 2892–2900, 2014.

[45] Y. Sun, X. Wang, and X. Tang. Sparsifying neural network connections for face recognition. *Computer Science*, 2015.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. pages 1–9, 2015.

[47] V. Turchenko and A. Luczak. Caffe: Convolutional architecture for fast feature embedding. *Eprint Arxiv*, pages 675–678, 2014.

[48] X. S. Wei, C. W. Xie, and J. Wu. Mask-cnn: Localizing parts and selecting descriptors for fine-grained image recognition. 2016.

[49] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. 42(7):529–534, 2011.

[50] X. Wu, R. He, and Z. Sun. A lightened cnn for deep face representation. *Computer Science*, 2015.

[51] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian. Disturblabel: Regularizing cnn on the loss layer. 2016.

[52] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *Computer Science*, 2014.

[53] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *Computer Science*, 2013.

[54] N. Zhang and W. Deng. Fine-grained lfw database. In *2016 International Conference on Biometrics (ICB)*, 2016.