Analysis of Function of Rectified Linear Unit Used in Deep learning

Kazuyuki Hara
College of Industrial Technology,
Nihon University
Narashino, Chiba 275–8575 Japan.
E-mail: hara.kazuyuki@nihon-u.ac.jp

Daisuke Saito
Graduate School of
Industrial Technology,
Nihon University
E-mail: cida14004@g.nihon-u.ac.jp

Hayaru Shouno
Graduate School of
Information and Engineering,
the University of Electro-communications
E-mail: shouno@uec.ac.jp

***Abstract*—Deep Learning is attracting much attention in object recognition and speech processing. A benefit of using the deep learning is that it provides automatic pre-training. Several proposed methods that include auto-encoder are being successfully used in various applications. Moreover, deep learning uses a multilayer network that consists of many layers, a huge number of units, and huge amount of data. Thus, executing deep learning requires heavy computation, so deep learning is usually utilized with parallel computation with many cores or many machines. Deep learning employs the gradient algorithm, however this traps the learning into the saddle point or local minima. To avoid this difficulty, a rectified linear unit (ReLU) is proposed to speed up the learning convergence. However, the reasons the convergence is speeded up are not well understood. In this paper, we analyze the ReLU by a using simpler network called the soft-committee machine and clarify the reason for the speedup. We also train the network in an on-line manner. The soft-committee machine provides a good test bed to analyze deep learning. The results provide some reasons for the speedup of the convergence of the deep learning.**

I. Introduction

Learning in neural networks can be formulated as the optimization of an objective function that quantifies the system's performance. An important property of feed-forward networks is their ability to learn a rule from examples. This property has been successfully studied by using statistical mechanics [1]-[4]. A compact description of learning dynamics can be obtained by using statistical mechanics, which feature a large input dimension N and provide an accurate model of mean behavior for a realistic N [1], [2].

In the field of neural network and its applications include object recognition and speech processing, deep learning [5] is attracting much attention. Key technology in deep learning is an automatic pre-training that extracts specifications of data while learning [5], [6]. Moreover, the deep learning uses a multilayer network with more than three layers with a huge numbers of units. Then, slow convergence is caused by a drop into the saddle point or local minima during the learning process. To avoid this problem, Zieler et al. proposed the rectified linear unit (ReLU) [16], [7]. The ReLU can be used in either pre-training or classification. The mainstream of study of deep learning is applying it for the object recognition and the speech processing, so there are few articles concerning the theoretical side [8].

Our motivation is to clarify why the ReLU in multilayer network speeds up the learning convergence. For this purpose, we analyze the dynamic behavior of learning using ReLU in multilayer networks learned by online learning. In multilayer networks, one problem is that slow convergence due to *plateaus* occurs in learning processes that use a gradient descent algorithm [9]. In the gradient descent algorithm, the weight vector is updated in the direction of the steepest descent of the objective function and the derivative of the output is taken into account. To avoid the above problem, Fahlman [10] proposed a learning method in which the derivative term is replaced with a constant and empirically showed that his method could speed up the convergence. We supplied the theoretical support for this learning method with a simple perceptron [13], [14]. The derivative of the ReLU is constant, so our previous studies [13], [14] give insight for analysis of the ReLU.

In this work, we analyze dynamics of the learning of a soft-committee machine using the ReLU as a simplified version of the deep learning through computer simulations. We formulate the learning settings similar to those of statistical mechanics because we will build a theory for the proposed method for a future study. Then, we employ a teacher-student formulation that is used in statistical mechanics of machine learning to analyze dynamic behavior of learning in detail. We analyzed dynamic behavior of the plateau of the soft-committee machine using the ReLU. These results clarify why using the ReLU improves the learning process.

II. Model

In this work, we train the network using supervised learning. In supervised learning, a desired output t is given for a input $\boldsymbol{\xi}$ by using a function $f(\boldsymbol{\xi})$. The relation between an input $\boldsymbol{\xi}$ and a desired output t is written by $t = f(\boldsymbol{\xi})$. The network to be trained outputs $t' = g(\boldsymbol{\xi})$. Usually, we don't know the function f, then to measure the performance of the network, the error between desired output t and the network output t' is calculated. As you can see, this error depend on an input data. To measure the generalization error, that is independent of an input data, we must know a network that generates the desired output from an input data.$\boldsymbol{\xi}$.

In this work, we employ a teacher-student formulation. In teacher-student formulation, the function $f(\boldsymbol{\xi})$ is realized by using a network. This network is called a teacher network (refer as teacher). The teacher gives the student network(refer

978-1-4799-1959-8/15/$31.00

as student) the desired output for the input. By introducing the teacher, we can directly measure the similarity of the student weight vector to the teacher weight vector. First we formulate a teacher and a student and then introduce the gradient descent algorithm. Note that teacher-student formulation is only for analysis. So, desired output is used to solve the real problems, and teacher network is no longer used.

A deep network is a full connected multilayer neural network used by the deep learning. The structure of a deep network is complex, so it is hard to analyze the dynamics of learning process. Thus, we employ a simpler multilayer network called the soft-committee machine. Soft-committee machine is a kind of multilayer neural network that holds similar behaviors to the deep network in the learning process. However, the soft-committee machine is much simple architecture that can analyze dynamics of learning behavior. Therefore, the soft-committee machine provides a good test bed to analyze deep learning. A soft-committee machine consists of input units, hidden units, and an output unit. The inputs-to-hidden weights are learnable by using some learning rules. The the hidden unit output function is non-linear function of $g(\cdot)$. All the hidden-to-output weights are fixed to $+1$ [2]. This network calculates the majority vote of hidden outputs. The soft-committee machine is a kind of multi-layer perceptron, so plateaus can appear in the learning process for any inputs [9].

The teacher and student are a soft-committee machine with N input units, hidden units, and an output, as shown in Fig. 1. The teacher and student consist of K and K' hidden units, respectively. Each hidden unit is considered as a perceptron. The kth hidden weight vector of the teacher is $\boldsymbol{B}_k = (B_{k1}, \ldots, B_{kN})$, and the k'th hidden weight vector of student is $\boldsymbol{J}_{k'}^{(m)} = (J_{k'1}^{(m)}, \ldots, J_{k'N}^{(m)})$, where m denotes learning iterations.

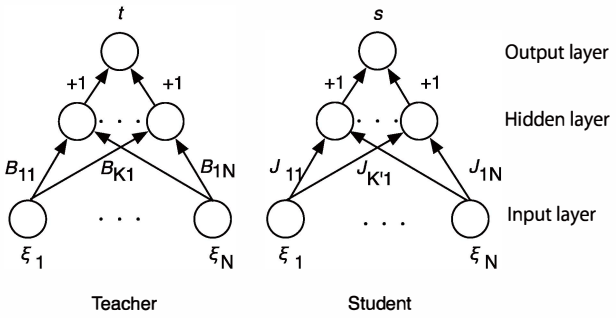


Fig. 1. Network structures of teacher and student.

We assume that both the teacher and the student receive N-dimensional input $\boldsymbol{\xi}^{(m)} = (\xi_1^m, \ldots, \xi_N^{(m)})$, that the teacher outputs $t^{(m)} = \sum_{k=1}^{K} t_k^{(m)} = \sum_{k=1}^{K} g(d_k^{(m)})$, and that the student outputs $s^{(m)} = \sum_{k'=1}^{K'} s_{k'}^{(m)} = \sum_{k'=1}^{K'} g(y_{k'}^{(m)})$. Here, $g(\cdot)$ is the output function of hidden unit, $d_k^{(m)}$ is the inner potential of the kth hidden unit of the teacher calculated using $d_k^{(m)} = \sum_{i=1}^{N} B_{ki}\xi_i^{(m)}$, and $y_{k'}^{(m)}$ is the inner potential of the k'th hidden unit of the student calculated using $y_{k'}^{(m)} = \sum_{i=1}^{N} J_{k'i}^{(m)}\xi_i^{(m)}$.

We assume that the elements $\xi_i^{(m)}$ of the independently drawn input $\boldsymbol{\xi}^{(m)}$ are uncorrelated random variables with zero mean and unit variance; that is, that the ith element of the input is drawn from a probability distribution $\mathrm{P}(\xi_i)$. The thermodynamic limit of $N \to \infty$ is also assumed. The statistics of the inputs in the thermodynamic limit are $\left\langle \xi_i^{(m)} \right\rangle = 0$, $\left\langle (\xi_i^{(m)})^2 \right\rangle \equiv \sigma_\xi^2 = 1$, and $\left\langle \|\boldsymbol{\xi}^{(m)}\| \right\rangle = \sqrt{N}$, where $\langle \cdots \rangle$ denotes the average and $\|\cdot\|$ denotes the norm of a vector. Each element $B_{ki},\ k = 1 \sim K$ is drawn from a probability distribution with zero mean and $1/N$ variance. With the assumption of the thermodynamic limit, the statistics of the teacher weight vector are $\langle B_{ki} \rangle = 0, \left\langle (B_{ki})^2 \right\rangle \equiv \sigma_B^2 = 1/N$, and $\langle \|\boldsymbol{B_k}\| \rangle = 1$. This means that any combination of $\boldsymbol{B}_l \cdot \boldsymbol{B}_{l'} = 0$. The distribution of inner potential $d^{(m)}$ follows a Gaussian distribution with zero mean and unit variance in the thermodynamic limit, then $\langle d_k \rangle = \sum_i \langle B_{ki} \rangle \langle \xi_i \rangle = 0$, and $\left\langle d_k^2 \right\rangle \equiv \sigma_d^2 = \left\langle (\sum_i B_{ki}\xi_i)^2 \right\rangle = N\sigma_B^2\sigma_\xi^2 = 1$.

For the sake of analysis, we assume that each element of $J_{k'i}^{(0)}$, which is the initial value of the student vector $\boldsymbol{J}_{k'}^{(0)}$, is drawn from a probability distribution with zero mean and $1/N$ variance. The statistics of the k'th hidden weight vector of the student are $\left\langle J_{k'i}^{(0)} \right\rangle = 0, \left\langle (J_{k'i}^{(0)})^2 \right\rangle \equiv \sigma_J^2 = 1/N$, and $\left\langle \|\boldsymbol{J}_{k'}^{(0)}\| \right\rangle = 1$ in the thermodynamic limit. This means that any combination of $\boldsymbol{J}_l^{(0)} \cdot \boldsymbol{J}_{l'}^{(0)} = 0$. The output function of the hidden units of the student $g(\cdot)$ is the same as that of the teacher. The statistics of the student weight vector at mth iteration are $\left\langle J_{k'i}^{(m)} \right\rangle = 0$, $\left\langle (J_{k'i}^{(m)})^2 \right\rangle = (Q_{k'k'}^{(m)})^2/N$, and $\left\langle \|\boldsymbol{J}_{k'}^{(m)}\| \right\rangle = Q_{k'k'}^{(m)}$. Here, $(Q_{k'k'}^{(m)})^2 = \boldsymbol{J}_{k'}^{m} \cdot \boldsymbol{J}_{k'}^{m}$. By a calculation similar to $\langle d_k \rangle$ and $\left\langle d_k^2 \right\rangle$, the distribution of the inner potential $y_{k'}^{(m)}$ follows a Gaussian distribution with zero mean and $(Q_{k'k'}^{(m)})^2$ variance in the thermodynamic limit. Note that these assumptions are required in theoretical analysis. However, these assumptions are almost satisfied for $N \sim 100$ or smaller N in the simulations in this paper.

Next, we introduce the stochastic gradient descent algorithm for the soft-committee machine. Deep learning trained by using gradient descent algorithm with mini batch. However, theoretical analysis of on-line learning with mini batch is hard work, so we train the network in on-line manner. In stochastic gradient learning, for the possible inputs $\{\boldsymbol{\xi}\}$, we want to train the student to produce the desired outputs $t = s$. The generalization error is defined as the squared error averaged ε over some number of inputs:

$$\begin{aligned} \varepsilon_g^{(m)} &= \left\langle \varepsilon^{(m)} \right\rangle = \frac{1}{2} \left\langle (t^{(m)} - s^{(m)})^2 \right\rangle \\ &= \frac{1}{2} \left\langle \left(\sum_{k=1}^{K} g(d_k^{(m)}) - \sum_{k'=1}^{K'} g(y_{k'}^{(m)}) \right)^2 \right\rangle, \end{aligned} \tag{1}$$

At each learning step m, a new uncorrelated input $\boldsymbol{\xi}^{(m)}$ is presented, and the current hidden weight vector of student $\boldsymbol{J}_{k'}^{(m)}$ is updated using

$$\begin{aligned}\boldsymbol{J}_{k'}^{(m+1)} &= \boldsymbol{J}_{k'}^{(m)} - \frac{\eta}{N}\frac{\partial \varepsilon}{\partial \boldsymbol{J}_{k'}} \\ &= \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{K'} g(y_{l'}^{(m)})\right) \\ &\quad \times g'(y_{k'}^{(m)})\boldsymbol{\xi}^{(m)}, \\ &= \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\delta g'(y_{k'}^{(m)})\boldsymbol{\xi}^{(m)}, \end{aligned} \tag{2}$$

where η is the learning step size and $g'(x)$ is the derivative of the output function of the hidden unit $g(x)$.

III. Comparison of convergence properties using sigmoid function or rectified linear unit function

In this section, we discuss the convergence property of the soft-committee machine using different hidden output functions: one is the sigmoid function, and the other is the rectified linear unit function referred to as ReLU. The sigmoid function $g_S(\cdot)$ and its derivative $g'_S(\cdot)$ are defined as

$$g_S(y_{k'}) = \mathrm{erf}\left(\frac{y_{k'}}{\sqrt{2}}\right), \tag{3}$$

$$g'_S(y_{k'}) = \sqrt{\frac{2}{\pi}}\exp\left(-\frac{y_{k'}^2}{2}\right). \tag{4}$$

$\mathrm{erf}(x)$ is the error function (similar to a sigmoidal function) defined as $2/\sqrt{\pi}\int_0^x \exp(-t^2)dt$. ReLU $g_R(\cdot)$ and its derivative $g'_G(\cdot)$ are defined as

$$g_R(y_{k'}) = \max(0, y_{k'}), \tag{5}$$

$$g'_R(y_{k'}) = \begin{cases} 1 & y_{k'} > 0 \\ 0 & y_{k'} < 0. \end{cases} \tag{6}$$

Here, $g'(y_{k'})$ at $y_{k'} = 0$ is not defined. However, we can use the ReLU in practical use. There is a similar function called "softplus" [11] defined as $\ln(1+\exp(y_{k'}))$. Then, the learning equation Eq. (2) is rewritten by Eq.(7) for using the sigmoid function, and that of the ReLU is rewritten by Eq.(8).

$$\begin{aligned}\boldsymbol{J}_{k'}^{(m+1)} &= \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K}\mathrm{erf}\left(\frac{d_l^{(m)}}{\sqrt{2}}\right) - \sum_{l'=1}^{K'}\mathrm{erf}\left(\frac{y_{l'}^{(m)}}{\sqrt{2}}\right)\right) \\ &\quad \times \sqrt{\frac{2}{\pi}}\exp\left(y_{k'}^{(m)}\right)\boldsymbol{\xi}^{(m)}, \end{aligned} \tag{7}$$

$$\begin{aligned}\boldsymbol{J}_{k'}^{(m+1)} &= \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K} g_S(d_l^{(m)}) - \sum_{l'=1}^{K'} g_R(y_{l'}^{(m)})\right)\boldsymbol{\xi}^{(m)}, \\ &\quad \text{if } y_{k'} > 0. \end{aligned} \tag{8}$$

Eq. (8) is not updated when $y_{k'} \leq 0$.

Figure 2 shows the output and derivative of the sigmoid function and the ReLU. In the figures, solid lines show output of the sigmoid function $g_S(y_{k'})$ and the ReLU $g_R(y_{k'})$. Broken lines show derivative of $g'_S(y_{k'})$ and $g'_R(y_{k'})$. From the figure, the output of the sigmoid function is saturated beyond $|y_{k'}| \geq 2$. This means that the sigmoid function outputs the same value for larger input potential $y_{k'}$. Moreover, the ReLU outputs are proportional to $y_{k'}$ when $y_{k'} > 0$. Then the ReLU responds to larger inputs better than the sigmoid function. From eqs. 7 and 8, the student weight $\boldsymbol{J}_{k'}$ is updated when the derivative has a non-zero value. From Fig. 2, derivative of the sigmoid function $g'_S(y_{k'})$ has a non-zero value between $|y_{k'}| \leq 2$. Derivative of the ReLU $g'_R(y_{k'})$ has a non-zero value when $y_{k'} > 0$. Therefore, derivative of the ReLU has a wider non-zero region than the sigmoid function. This means that the ReLU responds to a wider range of input potentials.

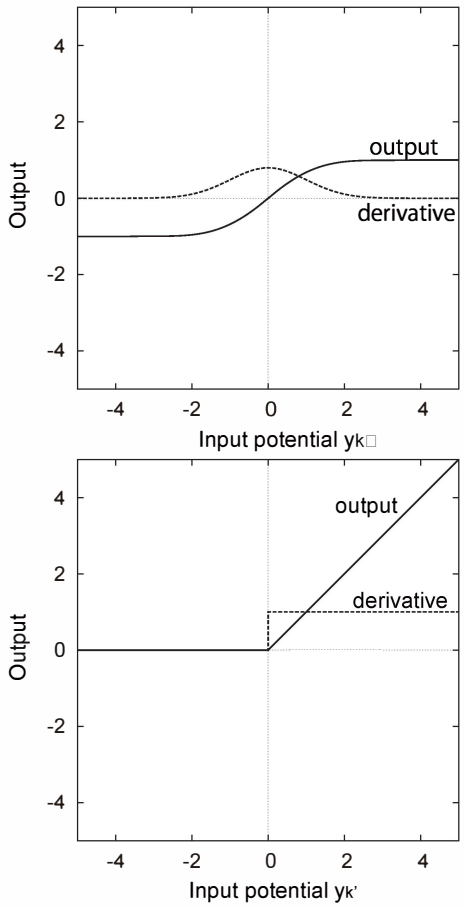


Fig. 2. Output and derivative of the sigmoid function (top) and ReLU (bottom).

Next, we compare convergence property of the MSE of the sigmoid function and the ReLU used in the student. Usually, we have pairs of input data and its target in the supervised learning. Thus, we cannot have a teacher. However, in this paper, we use teacher-student formulation, then we can assume that a target is generated by a teacher. In this case, we consider two cases: the teacher and student use the same output function, or the teacher and student use different output functions. Therefore, we analyzed (1) teacher uses the sigmoid function, (2) teacher uses the ReLU, and (3) teacher uses the Gaussian function. If the teacher and student use the same output function, the student can mimic the teacher

perfectly. However, if the teacher and student use different output functions, the student can only approximate the teacher. We assume that the architectures of teacher and student are the same. (1) and (2) correspond to learnable data set, and (3) correspond to unlearnable data set.

(1) teacher using the sigmoid function

The teacher uses the sigmoid function, and the student uses the sigmoid function or the ReLU. Here, the student using the sigmoid function can be expected to achieve better results. Figure 3 shows time course of the mean square error (MSE) and the direction cosines R for soft-committee machine. Computer simulation results are shown in Fig. 3, where $N = 1000$ and $\eta = 0.1$. The number of hidden units is $K = K' = 2$. The elements ξ_i of independently drawn input $\boldsymbol{\xi}$ are uncorrelated random variables with $\mathcal{N}(0, 1)$ and the elements of teacher weight vectors and initial student weight vectors are sampled from $\mathcal{N}(0, 1/N)$. The MSE is calculated at every iteration for 1000 individual inputs.

Figure 3 (a) shows the time course of MSE of a soft-committee machine using the sigmoid function. The direction cosine of the teacher and student weight vector defined by $R_{kk'} = (\boldsymbol{B}_k \cdot \boldsymbol{J}_{k'})/(\|\boldsymbol{B}_k\| \cdot \|\boldsymbol{J}_{k'}\|)$ are shown in the same figure. As shown, the learning starts from the left of the figure with large MSE, and as learning proceeds, the error is relaxed and reaches the state in which the error stays at the same value for an extended period called the plateau. The period before the plateau is reached is called the relaxation time is this paper. After the plateau breaks, the error decreases to the minimum state called residual error. If the plateau cannot be broken in a learning process, it becomes a large residual error. Two direction cosines R_{11} and R_{12} were different values at the beginning of the learning (left end of the figure), however as learning proceeded, they became the same value (roughly at the middle of the learning). Finally, R_{11} and R_{12} again separated from each other and converged into $R_{11} = 1$ and $R_{12} = 0$. These mean that the angle between $\boldsymbol{B}_1$ and $\boldsymbol{J}_1$ is zero, and that of $\boldsymbol{B}_1$ and $\boldsymbol{J}_2$ is $\pi/2$, therefore, $\boldsymbol{B}_1$ and $\boldsymbol{J}_2$ are orthogonal to each other. These results show how the plateau was broken after a long time interval.

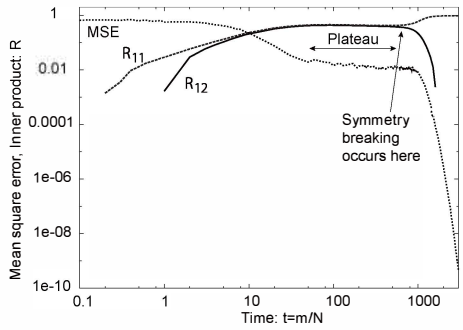
Fig. 3 (b) shows learning dynamics of a student using the sigmoid function in a plateau. $R_{kk'}$ denotes the direction cosine of $\boldsymbol{B}_k$ and $\boldsymbol{J}_{k'}$, then $R_{kk'} = 1$ means that the angle between the kth teacher and k'th student weight vector is zero, and $R_{kk'} = 0$ means that the kth teacher and k'th student weight vectors are orthogonal to each other. The norm of the student weight vector $\|\boldsymbol{J}_{k'}\| = Q_{k'k'}$ shows the length of the vector $\boldsymbol{J}_{k'}$. The norm of teacher weight vector and that of initial student weight vector are set to 1, so if $Q_{k'k'} = 1$ and $R_{kk'} = 1$, kth teacher weight vector and k'th student weight vector are the same. From the above facts, behavior of the student weight vectors in the plateau can be described by using $R_{kk'}$ and $Q_{k'k'}$. The horizontal axis is Q_{11} or Q_{22}, and the vertical axis is R_{11}, R_{12}, R_{21}, or R_{22}. Figure 3 (b) shows the dynamic behavior of the student using the sigmoid function. Initial conditions were $R_{11}^{(0)} = R_{12}^{(0)} = R_{21}^{(0)} = R_{22}^{(0)} = 0$, and $Q_{11} = Q_{22} = 1$. Learning starts from the bottom right (as shown by 'start') of the figure, and $R_{11} \sim R_{22}$ gradually become larger after the dog-leg (as shown by dashed circle), finally settling at $R_{11} \sim R_{22}$, which are about 0.4. At the same time, Q_{11} and Q_{22} become shorter until Q_{11} and Q_{22} are about 0.65. After that, R_{12} and R_{21} converge into $Q_{11} = 1.0$, and $R_{12} = R_{21} = 1.0$ (as shown by 'final'). R_{11} and R_{22} pass the other way and converge into $R_{11} = R_{22} = 0.0$, and $Q_{11} = Q_{22} = 1.0$ (as shown by 'final'). Therefore, the credit assignment of $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ is achieved. From the figure, $R_{11} \sim R_{22}$ are almost the same value when Q_{11} and Q_{22} are from 0.9 to 0.7. This interval constitutes the plateau. The plateau breaks around $R_{11} = R_{12} = R_{21} = R_{22} = 0.6$, and $Q_{11} = Q_{22} = 0.7$.

Figure 3 (c) shows the time course of MSE of the soft-committee machine using the ReLU. The direction cosine of the teacher and student weight vector R_{11} and R_{12} are also shown. Because of the derivative of the ReLU is constant , the learning equation using the ReLU relates only to common error δ–i.e., it does not relate to $y_{k'}$–so the plateau does not break. Therefore, the residual error becomes large as shown in Fig. 3 (c). We don't show here, however, learning dynamics of R_{11} and R_{12} are converged at the point $(Q_{11}, R_{11}) = (Q_{11}, R_{12}) = (0.4, 0.7)$. At the point, the angle between $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ was nearly zero and they are identical each other. Therefore, credit assignment of $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ is not achieved.

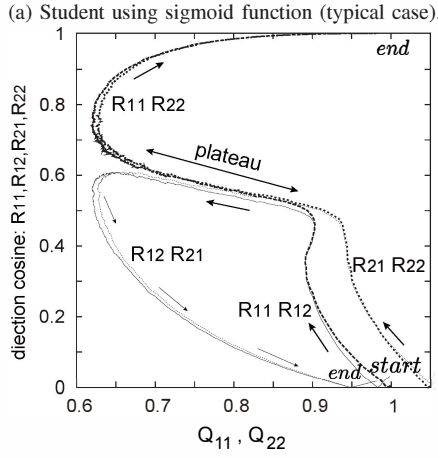
(2) teacher using the ReLU

The teacher uses the ReLU, and the student uses the sigmoid or ReLU. Simulation conditions are the same as those in Fig. 3. Figure 4 shows the results. Fig. 4(a) shows the time course of MSE of a soft-committee machine using the sigmoid function. The output function of the teacher differs from that of the student, so this is not a learnable case. Thus, the student cannot break the plateau and a large residual error remains. The learning using the sigmoid function tends to be trapped into the saddle point, so the direction cosines R_{11} and R_{12} stay the same.

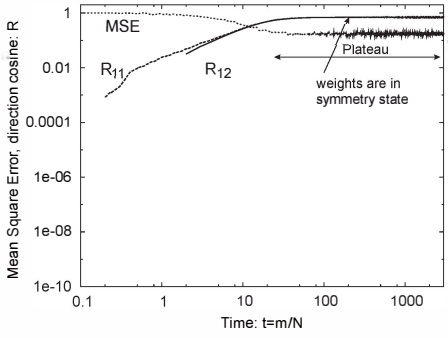
Figure 4(b) shows the time course of MSE of a soft-committee machine using the ReLU. The student using the ReLU is a learnable case and it can be expected to achieve better results. From the learning equation using the ReLU (Eq. 8), it relates only to common error δ–i.e., it does not relate to $y_{k'}$–so it supposes that the plateau does not break. However, as shown in Fig. 4(b), the plateau is broken and small residual error is achieved. Note that the plateau broke at $t = 200$, which is five times faster than that when the sigmoid function is used (refer to Fig. 3(a)). Fig. 4(c) clarifies the reason for the plateau breaking. Learning starts from the bottom right (as shown by 'start') of the figure, and R_{11} and R_{22} gradually become larger until $R_{11} = 0.3$ and $R_{22} = 0.35$. At the same time, the norm of $\boldsymbol{J}_1$, that is Q_{11}, and that of $\boldsymbol{J}_2$, that is Q_{22} become shorter (at the left end of the figure). This can be explained as follows: by shortening the norm of the student weight vector, the angle between the teacher and student weight vectors will be easily changed. Finally, R_{11} and R_{22} are converted at $(Q_{11}, R_{11}) = (Q_{22}, R_{22}) = (1, 1)$ (as shown by 'end' at the left top). Therefore, the credit assignment of $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ is achieved. R_{12} and R_{21} gradually become larger, however they become smaller and move back to the start point (as shown by 'end' at the right bottom). The dynamics in Fig. 4(c) differ from those in Fig. 3(b). In Fig. 3(c), R_{11}, R_{12}, R_{21}, and R_{22} are almost the same value in the plateau. However, in Fig. 4(c), couples of R_{11} and R_{22} and also R_{12} and R_{21} are not the same value at the point marked by the dashed circle in the figure. The student weight vectors are symmetrical for a short



(a) Student using sigmoid function (typical case).

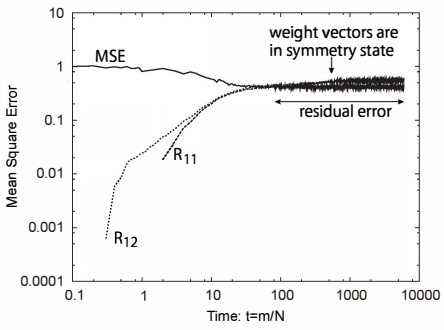


(b) Learning dynamics of student using sigmoid function.

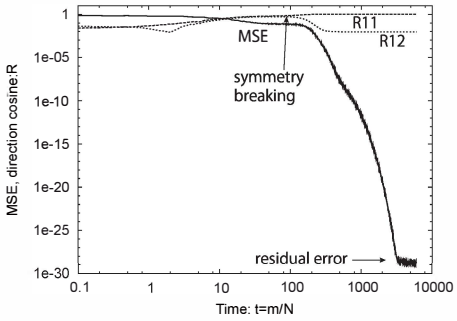


(c) Student using ReLU (typical case).

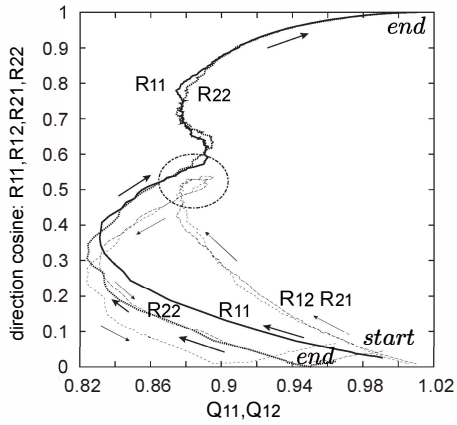
Fig. 3. Time course of mean square error for soft-committee machine using sigmoid function or rectified linear function. Learning step size is set to $\eta = 0.1$.



(a) Student using sigmoid function (Typical).



(b) Student using ReLU (Typical).



(c) Learning dynamics of student using ReLU.

Fig. 4. Time course of mean square error for soft-committee machine using sigmoid function or ReLU. Teacher using ReLU. Learning step size is set to $\eta = 0.1$.

time near the dashed circle in Fig. 4(c). Therefore, the student easily broke the plateau.

From our previous study[13], [14], the relaxation time can be improved by replacing the derivative with constant. For ReLU, from Eq. 6, the derivative of ReLU is constant when the inner potential of the hidden unit is $y_{k'} \leq 0$. Thus, this can be the reason the relaxation time of ReLU is improved.

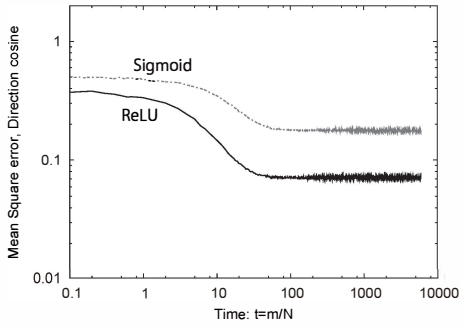
(3) Teacher is Gaussian function

The teacher uses a Gaussian function, and the student uses the sigmoid function or the ReLU. Here, neither the student using the sigmoid function nor that using ReLU can achieve good results.

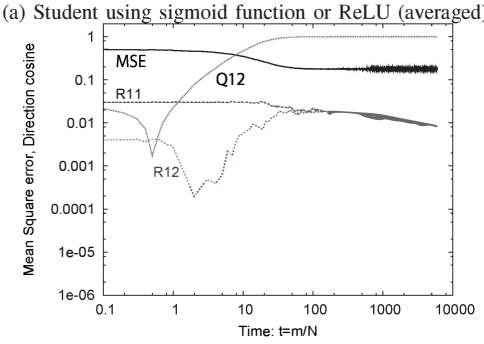
Simulation conditions in Fig. 5 are the same as those in Fig. 3. The Fig. 5(a) shows the time course of MSE of a soft-committee machine using the sigmoid function. Each plot is averaged over 10 trials. From the figure, the residual error of the student using ReLU is smaller than that of the student using sigmoid function.

The convergence property of the direction cosines R_{11} and R_{12} are also shown in Fig. 5(b) and (c). From Fig. 5(b), R_{11} and R_{12} converged to about 0.01. Therefore, the teacher weight vector $\boldsymbol{B}_1$ and both student weight vectors $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are not similar to each other. We also shows direction cosine between $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ referred by Q_{12}. From the figure, Q_{12} is converged into $Q_{12} = 1$, then, the angle between $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ is zero. Therefore, $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ act as the same for the inputs. From Fig. 5(c) R_{11} and R_{12} are oscillated beyond $t > 50$ around 0.02. Note that this oscillation not effect the MSE worth. From the

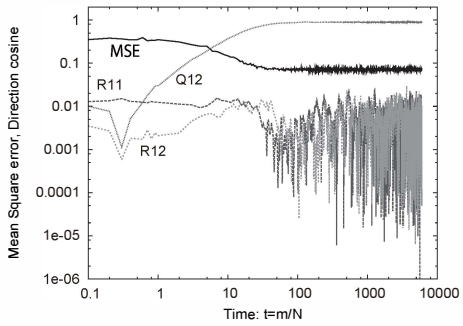
figure, Q_{12} is converged into $Q_{12} = 0.89$, then angle between $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are not zero. This may cause of smaller MSE of using ReLU compares to that of using the sigmoid function. From these results, the ReLU is robust for unlearnable cases compare to that of the sigmoid function. Neither network could brake the plateau, and both retained large residual error.



(a) Student using sigmoid function or ReLU (averaged).



(b) Student using sigmoid function (typical case).



(c) Student using ReLU (typical case).

Fig. 5. Time course of mean square error for soft-committee machine using sigmoid function or ReLU. Learning step size is set to $\eta = 0.1$.

A. *Effect of non-limiting derivative region*

In the ReLU, the derivative is non-zero during an interval of $y_{k'} > 0$. However, in the sigmoid function, the derivative is non-zero between $|y_{k'}| < 2$ (as shown in Fig. 2). Thus, we considered that the wider interval of non-zero of the derivative may cause the plateau to shorten. Therefore, we prove this hypothesis by using a limiting non-zero interval of derivative of the ReLU and analyze the learning results. The teacher and student use the ReLU. We modify the derivative $g'_R(y_{k'})$ as follows:

$$g'_R(y_{k'}) = \begin{cases} 1 & 0 < y_{k'} < a \\ 0 & otherwise. \end{cases} \tag{9}$$

In this equation, we assume that the derivative is zero during an interval $y_{k'} > a$. Then, the output of ReLU is need to be a constant during the interval. We use $a = 0.5, a = 1.0$, or $a = 2.0$. The results are shown in Fig. 6. As shown in Fig. 6(a), the plateau becomes significant when $a = 0.5$ and $a = 1$. This means that when a non-zero interval of derivative becomes narrower, the plateau becomes significant. On the other side, by using Eq. (9), the student does not learning the input that the inner potential is $y_{k'} > a$. Therefore, the input that $y_{k'}$ holds larger inner potential $y_{k'} > a$ are need to break the plateau earlier.

Next, we analyze the dynamics of the learning when $a = 1$ (Fig. 6(b)) and $a = 2$ (Fig. 6(c)). Figure 6(b) shows dynamics of the learning when $a = 1$. From the figure, the couple of R_{11} and R_{12} takes time to separate (as shown by dashed circle). The couple of R_{21} and R_{22} also behave similarly to R_{11} and R_{12}. This behavior is not the same as that in Fig. 3(c), however this phenomena is a cause of the slowdown of breaking the plateau. Comparing Fig. 6 with Fig. 3(a), even if a narrower non-zero interval is used, the ReLU can break the plateau. In Fig. 6(c), when $a = 2$, the plateau like that in Fig. 6(b) does not occur. Therefore, by limiting the non-zero interval of the ReLU, the plateau becomes difficult to break relative to the limit a in Eq. 9. However, behavior of breaking the plateau differs from that of using the sigmoid function.

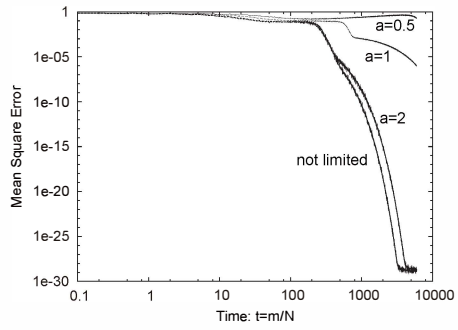
IV. Application

In this section, we show some results apply the ReLU to real problem to verify our analytical results. For this purpose, we use the three layer multilayer network.

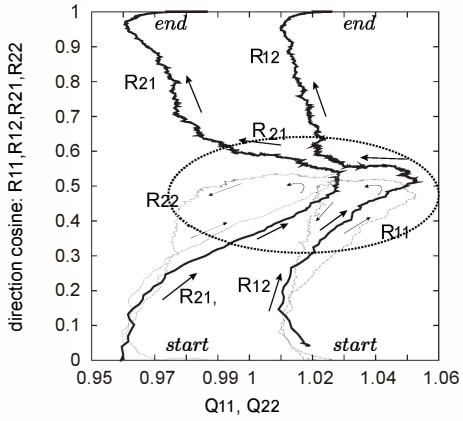
We use a stochastic gradient descent algorithm, which is the online learning scheme. The network structure is 785 input units, 100 hidden units, and 10 output units. We used the sigmoid function or ReLU in the hidden and output layers. We use the MNIST database[17], which is handwritten digits (0 to 9) and contains a training set of 60,000 examples and a test set of 10,000 examples .

We randomly selected one example from the pool of 60,000 training sets for learning. We randomly selected 1000 examples from a pool of 10,000 test sets to calculated MSE for testing the generalization performance of the network. MNIST is a classification problem. There are 10 outputs and each output corresponds to one digit. After the learning, we measured classification score by calculated in the following way: the kth output is the maximum value and the input belongs to kth class; the error is zero, otherwise, it is 1. To calculating classification rate, the classification score for all the examples are summed up, and then it is divided by the number of examples.

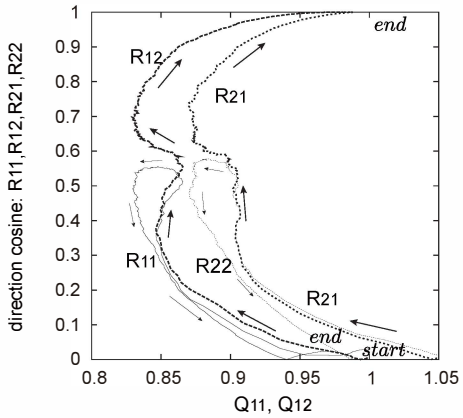
Figure 7 shows the results. The horizontal axis shows the learning time $t = m/N$ where m is learning iteration and N is the number of examples for testing ($N = 1000$). The vertical axis shows the MSE. Each element of the weight vector is initialized by the gaussian distribution of zero mean and unit variance when the sigmoid function is used. The learning rate is set to $\eta = 0.01$ for the sigmoid function. Each element of the weight vector is initialized by uniform distribution on the interval [0,1] when the ReLU is used. The learning rate is set to $\eta = 0.001$ for the ReLU. The MSE for the network using



(a) Time course of MSE when $a = 0.5, 1.0$, or 2.0



(b) Dynamics of learning when $a = 1$



(c) Dynamics of learning when $a = 2$

Fig. 6. Effect of limiting derivative of output function. Teacher and student use ReLU.

the sigmoid function is depicted by a solid line and is labeled "Sigmoid". The MSE for using the ReLU is depicted by a dashed line and is labeled "ReLU".

As shown in Fig. 7, the network using the ReLU shortens the relaxation time much faster than the network using sigmoid function. These results agreed with the analysis in Sec. III. Note that the learning step size of the network using the ReLU is much small than that of using the sigmoid function. Table I shows classification rates of 10 digits using the sigmoid and ReLU. As shown, the ReLU is slightly more accurate than the sigmoid function.

Table II shows the network output of using the ReLU. Row of the table shows output of the network, and column shows

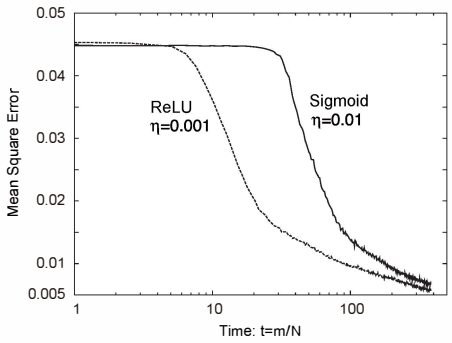


Fig. 7. Time course of mean square error with sigmoid function and ReLU. The learning rate is 0.01 for sigmoid and 0.001 for ReLU.

TABLE I. COMPARISON OF CLASSIFICATION RATE

digit	classification rate	
	Sigmoid	ReLU
0	0.97	0.98
1	0.97	0.98
2	0.82	0.87
3	0.92	0.95
4	0.95	0.96
5	0.87	0.93
6	0.96	0.96
7	0.93	0.95
8	0.93	0.95
9	0.85	0.88
average	0.92	0.94

digit of the input. Two results for each digit are shown. As shown, there are many zero outputs caused by using ReLU. Table III shows the network output of using the sigmoid function. There are also many zero outputs, however most are results of rounding off. Therefore, as many articles have pointed out, using the ReLU makes the output sparser than using the sigmoid function.

TABLE II. OUTPUT OF NETWORK USING THE RELU.

digit	output of network									
	0	1	2	3	4	5	6	7	8	9
0	0.82	0	0	0	0	0	0	0	0	0
	0.72	0	0	0	0	0	0	0	0	0
1	0	0.92	0	0	0	0	0	0	0	0
	0	1.03	0	0	0	0	0	0	0	0
2	0	0	0.76	0	0	0	0	0	0	0
	0	0	0.87	0	0	0.01	0	0	0	0
3	0	0	0	0.38	0	0.21	0	0	0.29	0
	0	0	0	1.15	0	0	0	0	0	0
4	0	0	0	0	0.93	0	0	0	0	0
	0	0	0	0	0.95	0	0	0	0	0.02
5	0	0	0	0	0	1.11	0	0	0	0
	0	0	0	0	0	1.00	0	0	0	0
6	0	0	0	0	0	0	0.56	0	0.07	0
	0	0	0	0	0	0.02	0.84	0	0	0
7	0	0	0	0	0	0	0	1.30	0	0
	0	0	0	0	0	0	0	0.66	0	0
8	0	0	0	0	0	0	0	0	1.27	0
	0	0	0	0.27	0	0	0	0	1.09	0
9	0	0	0	0	0	0	0	0	0	0.76
	0	0	0	0	0.12	0	0	0	0	0.76

V. CONCLUSION

In this paper, we have conducted computer simulations to analyze the soft-committee machine using the ReLU. We assumed that the network is learned by an on-line stochastic gradient descent algorithm. A teacher-student formulation was used, enabling us to analyze details of dynamics of the learning. We used the soft-committee machine as a test bed

TABLE III. OUTPUT OF THE NETWORK USING THE SIGMOID FUNCTION

digit	output of network									
	0	1	2	3	4	5	6	7	8	9
0	**0.98**	0	0.01	0.00	0.00	0.02	0.01	0.01	0.02	0.00
	0.88	0.00	0.05	0.00	0.00	0.05	0.00	0.00	0.01	0.00
1	0.00	**0.96**	0.02	0.01	0.00	0.02	0.01	0.01	0.00	0.00
	0.00	**0.98**	0.01	0.02	0.00	0.01	0.00	0.05	0.01	0.00
2	0.01	0.00	**0.91**	0.02	0.00	0.02	0.13	0.00	0.01	0.00
	0.00	0.00	**0.99**	0.00	0.00	0.02	0.00	0.00	0.00	0.00
3	0.00	0.00	0.02	**0.34**	0.00	0.00	0.01	0.00	0.02	0.00
	0.00	0.00	0.00	**0.98**	0.00	0.06	0.00	0.03	0.00	0.02
4	0.00	0.00	0.01	0.00	**0.92**	0.00	0.01	0.02	0.01	0.04
	0.00	0.00	0.00	0.00	**0.96**	0.06	0.00	0.01	0.07	0.02
5	0.00	0.00	0.00	0.07	0.00	**0.82**	0.00	0.00	0.04	0.00
	0.01	0.00	0.00	0.00	0.02	**0.97**	0.02	0.00	0.00	0.00
6	0.00	0.00	0.05	0.00	0.00	0.00	**0.59**	0.00	0.07	0.00
	0.00	0.00	0.01	0.00	0.01	**0.09**	**0.98**	0.00	0.01	0.00
7	0.00	0.00	0.01	0.02	0.00	0.00	0.00	**0.99**	0.00	0.00
	0.02	0.00	0.01	**0.09**	0.00	0.00	0.00	**0.99**	0.00	0.00
8	0.01	0.00	0.35	0.00	0.00	0.01	0.01	0.00	**0.88**	0.02
	0.00	0.00	0.00	0.00	0.04	0.24	0.00	0.00	**0.94**	0.01
9	0.00	0.00	0.00	0.01	0.06	0.01	0.00	0.01	0.02	0.74
	0.00	0.00	0.00	0.00	**0.08**	0.00	0.00	0.07	0.07	**0.84**

of the deep learning, because soft-committee machine is easy to analyze. From the results, we clarified that the ReLU is not trapped into the plateau because of linear output for positive and wider non-zero interval of derivative arrow to learn input that need to break a plateau. Our future study will include analysis of the committee machine using the ReLU and its theory.

ACKNOWLEDGMENTS

The authors thank Masato Okada, Hideitsu Hino, and Ryo Karakida for their insightful discussions.

REFERENCES

[1] M. Biehl and H. Schwarze, "Learning by on-line gradient descent", Journal of Physics A: Mathematical and General Physics, **28**, 643–656, (1995).

[2] D. Saad and S. A. Solla, "On-line learning in soft-committee machines", Physical Review E, **52**, pp. 4225–4243 (1995).

[3] K. Fukumizu, "A Regularity Condition of the Information Matrix of a Multilayer Perceptron Network", Neural Networks, **9**, no. 5, pp. 871–879 1996 .

[4] M. Rattray and D. Saad, "Incorporating Curvature Information into On-line learning", in D. Saad (Ed) *On-line Learning in Neural Networks.*, Cambridge University Press, Cambridge UK, pp. 183–207, (1998).

[5] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets", Neural Computation, **18**, pp. 1527–1554 (2006).

[6] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layerwise training of deep networks," Advances in Neural Information Processing Systems 19, MIT Press, pp. 153–160 (2007).

[7] M. D. Zeiler, M. Ranzato, et al., On Rectified Linear Units For Speech Processing, Proceeding of ICASSP, pp. 3517–3521 (2013).

[8] A. M. Saxe, J. L. McClelland, S. Ganguli, "Exact solution to the nonlinear dynamics of learning in deep linear neural networks, arXiv:1312.6120v3 [cs.NE] 19 Feb (2014).

[9] S. Amari, "Natural gradient works efficiently in learning", Neural Computation, **10**, pp. 251–276 1998.

[10] S. E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks", CMU-CS-88-162, 1988.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks" , proceeding of AISTATS, pp. 315–323 (2011).

[12] T. Vatanen, H. Valpola, and Y. LeCun, "Deep Learning Made Easier by Linear Transformation in Perceptrons", AISTATS 2012, JMLR W&CP, vol. 22, pp. 924–932 (2012).

[13] K. Hara, K. Katahira, K. Okanoya, and M. Okada, "Theoretical Analysis of Function of Derivative Term in On-Line Gradient Descent Learning", A.E.P. Villa et al. (Eds.): ICANN 2012, Part II, LNCS 7553, Springer-Verlag Berlin Heidelberg, pp. 9–16, (2012).

[14] K. Hara and K. Katahira, "Theoretical Analysis of learning speed in Gradient Descent algorithm Replacing Derivative with Constant", Information Processing Society of Japan Transactions on Mathematical and Its Applications, vol. 6, no. 3, pp. 100–105 (2013).

[15] K. Hara and K. Katahira, "Improving the Convergence Property of soft-committeeMachines by Replacing Derivative with Truncated Gaussian Function," S. Wermter et al. (Eds.): ICANN 2014, LNCS 8681, pp. 499–506, (2014).

[16] Li Deng, et al., Recent Advances In Deep Learning For Speech Research At Microsoft, ICASSP pp. 8604–8608 (2013).

[17] Y. LeCun, C. Cortes, and C. J. C. Burges, The MNIST Database, http://yann.lecun.com/exdb/mnist/