Overcoming Catastrophic Forgetting by Neuron-Level Plasticity Control

Inyoung Paik, Sangjun Oh, Taeyeong Kwak
Deep Bio Inc., Seoul, Republic of Korea
{iypaik, tykwak}@deepbio.com, me@juneoh.net

Injung Kim
Handong Global University, Pohang, Republic of Korea
ijkim@handong.edu

Abstract

To address the issue of catastrophic forgetting in neural networks, we propose a novel, simple, and effective solution called *neuron-level plasticity control (NPC)*. While learning a new task, the proposed method preserves the existing knowledge from the previous tasks by controlling the plasticity of the network at the neuron level. NPC estimates the importance value of each neuron and consolidates important *neurons* by applying lower learning rates, rather than restricting individual *connection weights* to stay close to the values optimized for the previous tasks. The experimental results on the several datasets show that neuron-level consolidation is substantially more effective compared to connection-level consolidation approaches.

Introduction

In the path to realizing artificial general intelligence with deep neural networks, catastrophic forgetting remains one of the most fundamental challenges. Gradient descent, the most popular learning algorithm, is problematic when applied to train a neural network for multiple tasks in a sequential manner. When gradient descent optimizes the neural network for the task at hand, the knowledge for the previous tasks is catastrophically overwritten by new knowledge.

Since the initial discovery of the problem (McCloskey and Cohen 1989), various approaches have been proposed to alleviate catastrophic forgetting in artificial neural networks. One of these approaches is to include the data for multiple tasks in every mini-batch. Although such a method can be effective in retaining the performance of the previous tasks, it causes an overhead to keep the training data for the previous tasks. There have been several attempts to achieve a similar effect using only a limited portion of the previous data, (Gepperth and Karaoguz 2016; Lopez-Paz 2017; Nguyen et al. 2017) or none at all.(Li and Hoiem 2018; Shin et al. 2017; Kamra, Gupta, and Liu 2017; Zacarias and Alexandre 2018; Kim, Kim, and Lee 2018)

Another approach is to isolate the parts of the neural network containing previous knowledge, and learn the new task using other parts of the network. This includes designing dynamic architectures for neural networks,(Fernando et al. 2017; Aljundi, Chakravarty, and Tuytelaars 2017; Lee et al. 2017a; Serrà et al. 2018; Masse, Grant, and Freedman 2018; Siavash Golkar 2019) where the capacity to learn the new task is obtained by assigning different parts of the network to the new task. Note that these methods isolate a portion of *neurons*, rather than *parameters*, to effectively(sometimes perfectly) preserve the existing knowledge of the neural network.

Weight consolidation is a remarkable step made in this area. Elastic Weight Consolidation(EWC)(Kirkpatrick et al. 2017) uses the diagonals of the Fisher information matrix to identify and consolidate the parameters, which correspond to the connection weights in neural networks, that are important for the previous tasks. In this way, the network learns the new task using less important parameters while preserving previously learned knowledge. In Memory Aware Synapses(MAS)(Aljundi et al. 2018), the importance value of each weight is measured by the sample mean of the absolute value of the gradient. In Synaptic Intelligence(SI)(Zenke, Poole, and Ganguli 2017), the importance value of each weight is computed by integrating the contribution to the change in loss. Meanwhile, Selfless Sequential Learning(SSL)(Aljundi, Rohrbach, and Tuytelaars 2018) suggests imposing sparsity on neuron activation while learning the preceding tasks to avoid exhausting all capacities. Their work and ours both focus on neuron-level information. However, in contrast, while the purpose of SSL is to save network capacity for the subsequent tasks, we aim to preserve knowledge from the previous tasks while learning a new task. Weight consolidation algorithms have drawn much attention, and therefore, have been adopted in many studies(Lee et al. 2017b; Liu et al. 2018). Recent works(Kim, Kim, and Lee 2018; Lee et al. 2017a) show that this approach may be used in combination with other methods as a means of regularization.

In this study, we present the limitation of weight consolidation (or, for purposes of comparison, 'connection-level consolidation') in deep neural networks, and propose a novel algorithm, called neuron-level plasticity control (NPC). As the name suggests, NPC retains existing knowledge by con-

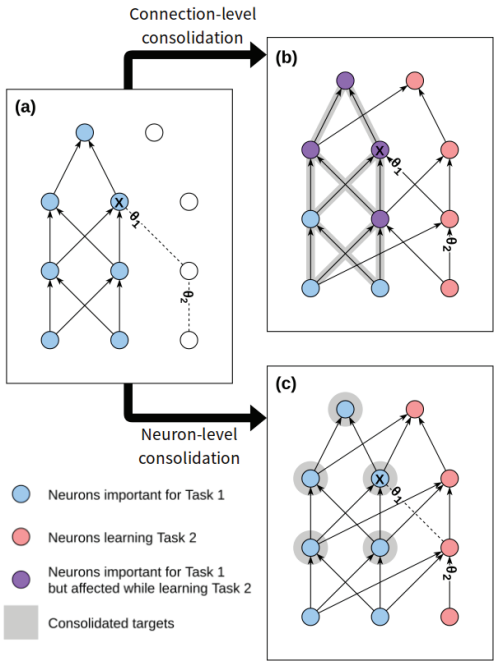

Figure 1: Comparison of connection-level and neuron-level consolidation. (a) The neurons and connections important for Task 1. (b) Connection-level consolidation. The important connections are consolidated, but the neurons can still be affected by other incoming connections that can change while learning Task 2. (c) Neuron-level consolidation. NPC consolidates all incoming connections of the important neurons, which is more effective in preserving the knowledge of the neurons. A similar intuition was pointed out by(Aljundi, Rohrbach, and Tuytelaars 2018; Lee et al. 2017a; Serrà et al. 2018; Siavash Golkar 2019).

trolling the plasticity of each *neuron* or each filter in a convolutional neural network (CNN). As a result, it is significantly more effective at preserving knowledge of deep neural networks. Moreover, NPC contains a memory-efficient consolidation algorithm. Most existing consolidation algorithms restrict individual connection weights to stay close to the values optimized for the previous tasks. Such algorithms need to save the weight and importance value of each connection for every task, and therefore, require memory and computation proportional to the number of tasks. On the other hand, NPC controls the plasticity of the neurons by simply adjusting their learning rates according to their importance value and eliminates the overhead to retain the task-specific information. As NPC stores only a single importance value per neuron instead of multiple sets of per-task parameter values, the memory requirement remains consistent regardless of the number of tasks.

Neuron-level Versus Connection-level Consolidation

While the connection-level consolidation algorithms(Kirkpatrick et al. 2017; Lee et al. 2017b; Liu et al. 2018; Zenke, Poole, and Ganguli 2017; Aljundi et al. 2018) focus on the idea that knowledge is stored in the parameters, which are the *connection weights* for neural networks, less emphasis is given to their *correlation*. The connection-level consolidation can be represented as the following loss function:

$$Loss = \lambda \sum_{k<n} \sum_{i} W_i(\theta_i - \theta_{i,k})^2 \quad (1)$$

Where θ_i and $\theta_{i,k}$ denote the i-th parameter and its value at the end of the learning of the k-th task, respectively. W_i denotes the importance value of θ_i and λ is a hyperparameter.

Note that eq. (1) is based on an implicit assumption that the weights in neural networks are roughly independent, and a neural network can be linearly approximated by its weights. However, the structure of deep neural networks is inherently hierarchical, and therefore, the parameters are highly correlated, i.e., modifying a parameter can affect the importance value of another.

We argue that the neuron, or CNN filter, is more appropriate than the individual connection for the basic unit of knowledge in the consolidation of the artificial neural network. Conventional connection-level algorithms do not guarantee the preservation of important knowledge expressed by neurons. Even if the learning algorithm consolidates some of the connections to an important neuron, the neuron may have remaining free incoming connections, the change of which may severely affect the knowledge carried by the neuron.

Figure 1 illustrates the limitation of the connection-level consolidation in deep neural networks more clearly. In the figure, although the neuron X is important for Task 1, changing the value of θ_1 or θ_2 individually may not change the output for task 1 significantly if the values of both θ_1 and θ_2 are close to zero. In such a case, due to their low importance values, connection-level algorithms would consolidate neither of the two connection parameters. Nonetheless, the neuron X can be seriously affected during the subsequent learning, when both parameters are rapidly increased, because they are highly correlated and increasing one of them can boost the importance of the other. This problem can be particularly serious in convolution layers in which the same filters are shared between multiple positions. Thus, even if the concept of connection-level consolidation is perfectly implemented, catastrophic forgetting cannot be eliminated completely.

To overcome this problem, we propose controlling the plasticity at the neuron level rather than at the connection level as shown in Figure 1(c). The proposed algorithm, NPC, collectively consolidates all the incoming connections of important *neurons*, including the connections that might not individually be evaluated as important. As a result, NPC protects the important neurons more effectively from the change of unimportant neurons than the connection-level consolidation algorithms. Note that the connection from an unimportant neuron to an important neuron is likely to be small, because otherwise, the evaluation algorithm would not have determined the source neuron to be unimportant. In the example shown in Figure 1, NPC consolidates all the input connections of X, and as a result, the value of θ_1 remains small, preventing the change of θ_2 from affecting X severely. On the other hand, NPC does not consolidate a

connection whose destination neuron is unimportant, even if the source neuron is important. Therefore, the total number of consolidated connections in the whole network remains acceptable.

Neuron-level Plasticity Control

Evaluation of Neuron Importance

To evaluate the importance of each neuron, we adopt a criterion based on Taylor expansion that has been used in the field of network pruning (Molchanov et al. 2016). Taylor criterion is simple but powerful and remains one of the state-of-the-art methods for network pruning.(Molchanov et al. 2019) Furthermore, the Taylor criterion is more computationally efficient than other methods such as (Yu et al. 2018; Luo and Wu 2017; Luo, Wu, and Lin 2017), since Taylor criterion is computed from the gradient of the loss function with respect to the neurons, which is computed during back-propagation. Therefore, it can easily be integrated into the training process with minimal additional computation. We have also attempted a few alternatives including the square of gradient as (Kirkpatrick et al. 2017), but could not obtain an improved result during experiment.

We measure the importance of each neuron by the normalized Taylor criterion as shown in eq. (2) and (3). Then, we take their exponential moving averages (EMA) as eq. (4) to preserve the importance from the previous tasks. The EMA also reduces the fluctuation of the measurements and improves learning stability.

$$c_i^{(t)} = \underset{batch}{average} \, |n_i^{(t)} \frac{dL^{(t)}}{dn_i^{(t)}}| \tag{2}$$

$$\bar{c_i}^{(t)} = \frac{c_i^{(t)}}{\sum_{layer} c_j^{(t)} / N_{layer}} \tag{3}$$

$$C_i^{(0)} = 0, \; C_i^{(t)} = \delta C_i^{(t-1)} + (1-\delta) \, \bar{c_i}^{(t)} \tag{4}$$

$C_i^{(t)}$ is the importance value of the i-th neuron at training step t, n_i denotes the activation of the i-th neuron. L is the loss, and N_{layer} is the number of nodes on the layer, and δ is a hyperparameter that should be set to a value close to one. The importance value of a convolution filter is computed by averaging the importance values of the neurons on the corresponding feature map before computing its absolute value, following the original paper(Molchanov et al. 2016). However, we use the arithmetic mean as in eq. (3), instead of the L2-norm, in order to enforce stricter balance among the layers composed of different number of neurons.

Plasticity Control

The stability-plasticity dilemma is a well-known constraint in both artificial and biological neural systems(Mermillod, Bugaiska, and Bonin 2013). Catastrophic forgetting can be seen as a consequence of the same trade-off problem: attempting to determine the optimal point that maximizes the performance of the neural network for multiple tasks. We control the plasticity of each neuron by applying different learning rates η_i for each neuron. If η_i is high, the neuron actively learns the new knowledge at the cost of rapidly losing existing knowledge. On the other hand, if η_i is low, existing knowledge can be preserved better; however, the neuron will be reluctant to learn new knowledge.

In order to encourage the neural network to find a good stability-plasticity balance, we define two costs as functions of η_i that play opposite roles; subsequently, we combine them. The first is the stability-wise cost to minimize the forgetting of existing knowledge. It should be a monotonically increasing function of η_i starting at $\eta_i = 0$ and bounded above by the amount of current knowledge. We heuristically defined this function as $l_{stability} = a_1 C_i \, \sigma(b_1 \eta_i)$, where C_i is the importance of the neuron, and σ is the activation function increasing from zero to one, and a_1, b_1 are constants to control the scale and the slope, respectively.

The second function is the plasticity-wise cost to decrease the reluctance against new knowledge. It is a decreasing function of η_i that starts from the maximum value at $\eta_i = 0$ and decreases monotonically to zero. The upper bound in this case does not regard existing knowledge, and therefore, is unrelated to C_i. We thus define the plasticity-wise cost as $l_{plasticity} = a_2(1 - \sigma(b_2 \eta_i))$

To find the balance between stability and plasticity, we choose the η_i that minimizes the combined cost function eq. (5).

$$\begin{aligned} \eta_i^* &= \underset{\eta_i}{argmin} \left(l_{stability}(\eta_i) + l_{plasticity}(\eta_i) \right) \\ &= \underset{\eta_i}{argmin} \{ a_1 C_i \sigma(b_1 \eta_i) + a_2(1 - \sigma(b_2 \eta_i)) \} \end{aligned} \tag{5}$$

For σ, we considered hyperbolic tangent, sigmoid, and error function. But we could not find a huge difference among the choices of σ. We use sigmoid function, which is slightly better than the others after the dense hyperparameter search.

By solving this minimization problem we get eq.(6), where $\alpha = \sqrt{4/(b_2^2 - b_1^2)}, \beta = a_2 b_2 / a_1 b_1$ are hyperparameters.(Please see appendix for details.)

$$\eta_i^* = \begin{cases} \alpha \sqrt{\frac{\beta}{C_i} - 1} & \text{if } C_i \leq \beta \\ 0 & \text{if } C_i > \beta \end{cases} \tag{6}$$

As our intuition, a larger C_i draws a smaller η_i^*, thereby consolidating the important neurons in the subsequent learning. However, if $C_i = 0$, then η_i^* diverges. This is explainable from the perspective of the plasticity-stability dilemma: if a neuron has no knowledge at all, it is desirable to learn the new knowledge as much as possible without considering the cost of existing knowledge. However, this cannot be applied to reality. Therefore, we set an upper bound of the learning rate η_{max}.

$$\eta_i^* = \begin{cases} min(\eta_{max}, \alpha \sqrt{\frac{\beta}{C_i} - 1}) & \text{if } C_i \leq \beta \\ 0 & \text{if } C_i > \beta \end{cases} \tag{7}$$

We heuristically set $\eta_{max} = 0.1$ for all experiments, which is a common upper bound for neural network training.

Algorithm 1 summarizes the NPC algorithm. Considering that C_i is also simply computed from the activation and

Algorithm 1 Neuron-level Plasticity Control (NPC)

f : neural network model
n_i : i-th neuron in f
w_{ji} : weight of connection from n_j to n_i
η_{max} : upper bound of learning rate
α, β : hyperparameters controlling learning rate
δ : decay rate of the importance value of each neuron
C_i : importance value of i-th neuron

$C_i \leftarrow 0,\ \forall i$
for $input$, $label$ **in** full training dataset **do**
  $y \leftarrow f(input)$
  $L \leftarrow CrossEntropy(y, label)$
  for n_i **in** f **do**
    $c_i \leftarrow \underset{batch}{average} |n_i \frac{dL}{dn_i}|$
    $\bar{c_i} \leftarrow \frac{c_i^{(t)}}{\sum_{layer} c_j^{(t)} / N_{layer}}$
    $C_i \leftarrow \delta C_i + (1-\delta)\, \bar{c_i}$
    $\eta_i \leftarrow min(\eta_{max}, \alpha \sqrt{max(\frac{\beta}{C_i} - 1, 0)})$
    $w_{ji} \leftarrow w_{ji} - \eta_i \frac{dL}{dw_{ji}},\ \forall j$
  end for
end for

the gradient, which are computed by the back-propagation algorithm, the overhead to implement NPC is minimal.

Instance Normalization

Batch normalization (BN) plays a key role in the training of deep neural networks(Ioffe and Szegedy 2015). However, the vanilla batch normalization does not work well in continual learning environments, because the mean and the variance are heavily affected by the transition of tasks. There are a few alternatives available in such cases, such as conditional batch normalization(De Vries et al. 2017) and virtual batch normalization(Salimans et al. 2016). However, they are not appropriate for NPC since they maintain task-specific information. Therefore, we apply a instance normalization(Ulyanov, Vedaldi, and Lempitsky 2016) with the affine transforms removed. As instance normalization is applied to each sample independently, it operates without any special manipulation of model parameters not only at the training time but also at the test time.

Experiments

We experimented on an *incremental* version of MNIST(LeCun et al. 1998) and CIFAR100(Krizhevsky and Hinton 2009) datasets, where the datasets containing X classes were divided into K subsets of X/K classes, each of which is classified by the k-th task. We set K to 5 for MNIST and 10 for CIFAR100. For preprocessing, we applied random cropping with padding size of 4 for both datasets. We also applied random horizontal flip for the incremental CIFAR100 (iCIFAR100) dataset. Additionally, we experimented on sequential tasks with heterogeneous datasets, which is composed of MNIST(LeCun et al. 1998), fashion-MNIST(fMNIST)(Xiao, Rasul, and Vollgraf 2017), EMNIST(balanced dataset)(Cohen et al. 2017), and smallNORB (LeCun 2004).

For consistency, we redefined the unit of *one epoch* in all experiments as the cycle in which the total number of train data was seen. For example, as the original MNIST dataset has 60,000 training samples, we defined one epoch of the iMNIST dataset as the processing of the 12,000 task-specific samples five times. With this definition of an epoch, we trained the models for 30 epochs on each task. All experiments were performed on a server with 2 NVIDIA Tesla P40 GPUs.

We used a simple CNN with 3 convolutional layers with (128, 512, 256) channels, and 2 fully connected layers with (512, *number of classes*) nodes. Each convolutional layer consists of convolution, Instance normalization, ReLU activation, and (2,2) max pooling. Dropout(Srivastava et al. 2014) of rate 0.2 is applied between two fully connected layers. The cross-entropy loss for each task was computed from only the output nodes belonging to the current task.

For direct comparison between neuron-level and parameter -level consolidation, we implemented an alternative algorithm called 'Connection-level Plasticity Control(CPC)' that is almost the same as NPC except that it consolidates the network at the connection level. Note that producing a neuron-level counterpart of weight consolidation is not trivial, since restricting individual neurons to stay close to certain values is not appropriate.

We compared our methods with EWC(Kirkpatrick et al. 2017), SI(Zenke, Poole, and Ganguli 2017), MAS(Aljundi et al. 2018), SSL(Aljundi, Rohrbach, and Tuytelaars 2018), and baseline SGD with simple L2 regularization. For hyperparameters of various continual algorithms. We first searched with a unit of 10 based on average validation accuracy on iCIFAR100 dataset. After that, we searched in space of one significant digit. For example, if we found that $\lambda = 10$ is better than $\lambda = 1$ or $\lambda = 100$, we searched for best value of λ in (5,6,7,8,9,10,20,30,40,50).

As a result, we used $\alpha_{NPC} = 0.05, \beta_{NPC} = 0.5, \delta_{NPC} = 1 - 2 \cdot 10^{-4}, \lambda_{EWC} = 900, \lambda_{MAS} = 3.0, \lambda_{SI} = 0.08, \lambda_{SSL} = 2 \cdot 10^{-6}$. In a baseline experiment, we used L2 regularization with $\lambda = 10^{-4}$. Experiments are averaged over 3 runs, But training curves are visualization of single run.

In all experiments, NPC performed significantly better than the connection-level consolidation algorithms. In particular, we found that CPC is very inefficient at preserving old knowledge, even though it consolidates similar numbers of weights with NPC. This shows that the neurons are more appropriate than the connections as the units of neural network consolidations.

We tuned our hyperparameters in the iCIFAR100 experiment(Figure 2), based on average accuracy. Therefore each algorithm has different balance points. NPC has achieved average validation accuracy 72.43% in iCIFAR100 experiment, while that of connection-level consolidation algorithms is between $50 \sim 60\%$.

iMNIST experiment(Figure 3) is a relatively easy environment with only five binary classification tasks. We used larger confidence decay($\delta = 1 - 10^{-3}$) in this experiment. All algo-

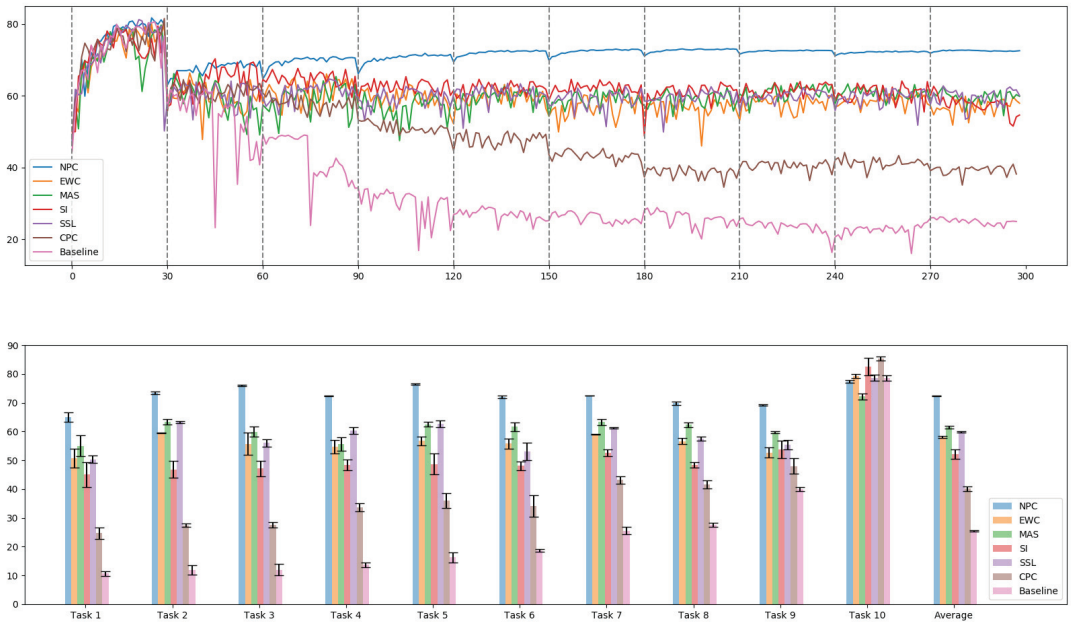

Figure 2: Training results of continual learning algorithms on the iCIFAR100 dataset. (top) Average validation accuracy of tasks that have been trained up to each moment. (bottom) Average validation accuracy and standard error(*standard deviation* $/\sqrt{\#\ of\ runs}$) for each task after finished training for all 10 tasks. NPC exhibits the best performance for all tasks after finished training except for the last task. CPC was inefficient at preserving old knowledge, even though it consolidates similar numbers of weights with NPC.

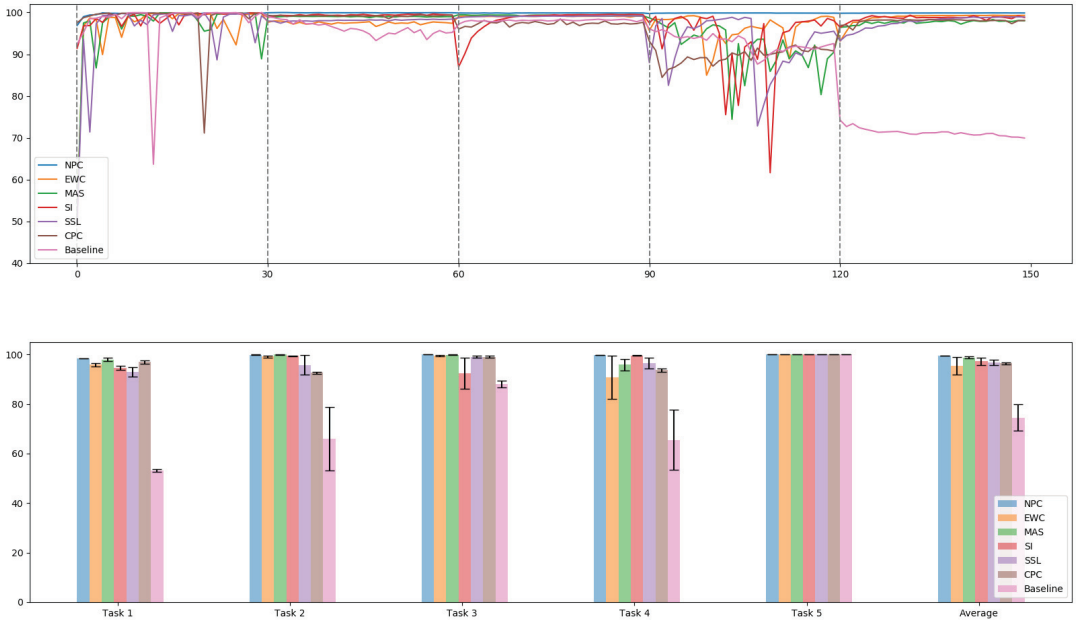

Figure 3: Training results of continual learning algorithms on the iMNIST dataset. (top) Average validation accuracy of tasks that have been trained up to each moment. (bottom) Average validation accuracy and standard error for each task after completing training for all five tasks. All algorithms achieved 100.0% validation accuracy for the last(5th) task. NPC preserved old knowledge almost perfectly(99.63% average validation accuracy), but all other algorithms also worked well.

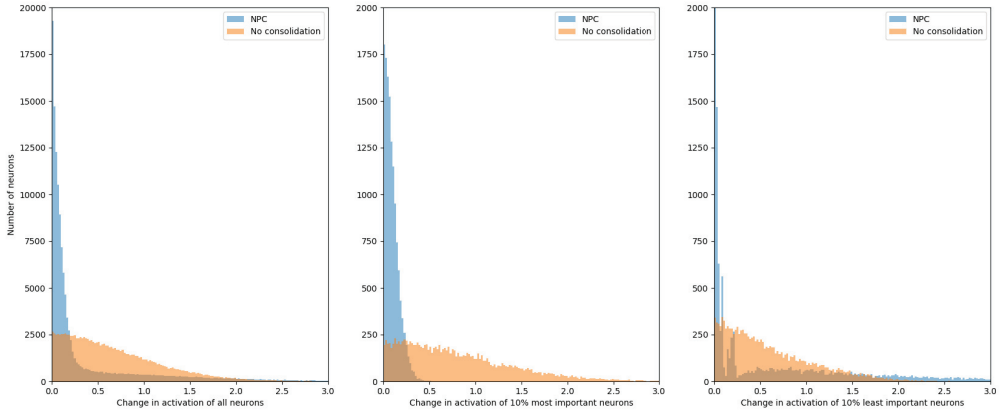

Figure 4: Change in the activation of the neurons on the second top layer after learning a subsequent task. Using NPC, the average change of the important neurons(0.094, center) was significantly smaller than the average change of all the neurons(0.383, left), while the average change of less important neurons(0.667, right) was much larger than that of all neurons. Without consolidation, there was no meaningful correlation between the change in neuron activation and their importance values.

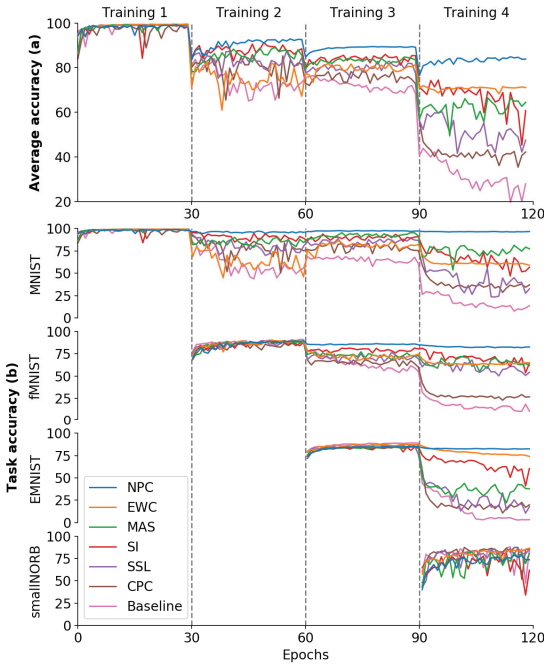

Figure 5: Training curves of continual learning algorithms on the MNIST, fMNIST, EMNIST, smallNORB dataset. (a) Average validation accuracy of tasks that have been trained up to each moment. (b) Training curves of the four tasks according to the learning algorithms. We used the same hyperparameter with experiment on iCIFAR100(Figure 2) without modification. Similar to other experiments, NPC exhibits the best performance for all tasks after completing training except for the last task.

rithms achieved 100.0% validation accuracy for the last(5th) task. NPC preserved old knowledge almost perfectly(99.63% average validation accuracy), but all other algorithms also worked well.

In experiments with four different datasets(MNIST, fMNIST, EMNIST, smallNORB)(Figure 5), NPC also exhibits the best performance. Some algorithms have had difficulty maintaining performance when learning from smallNORB, which is the most heterogeneous dataset of the four, but NPC was hardly affected.

Additionally, we measured the change in the activation of the neurons on the second top layer after learning a subsequent task to see whether NPC successfully consolidates important neurons. First, we trained a CNN for Task 1 of iCIFAR100 for 30 epochs and recorded the neuron activation values of the second top neurons(just before the final classifier) extracted from randomly chosen 256 samples. (512 neurons $\times$ 256 samples = 131,072 data points in total.) Then, we trained the CNN for Task 2 for another 30 epochs. Finally, we measured the change in the neuron activation values from the same sample. Figure 3 displays the results. Without consolidation, there was no meaningful correlation between the change in neuron activation and their importance values. However, using NPC, the average change of activation of all neurons was 0.383. The average change of activation of the most important 10% of the neurons was only 0.094, while that of the least important 10% of neurons was 0.667. These results suggest that NPC successfully preserved the neurons important for Task 1, while Task 2 was learned mainly by the neurons less important for Task 1.

Conclusion and Discussion

In this paper, we proposed a continual learning algorithm, NPC, that controls the plasticity of a neural network at the neuron level, instead of restricting individual connection

weights to stay close to certain values. NPC is effective in preserving old knowledge since it consolidates all the incoming pathways to important neurons. The experimental results on three different sequential datasets show that NPC is significantly more effective than conventional connection-level consolidation algorithms that do not consider the relation among connections. NPC has an additional benefit that it does not maintain any task-specific information such as the latest set of parameters optimized for each task, which makes it more efficient in terms of memory and computational complexity.

While NPC defines the unit and the method for controlling plasticity, strategies for evaluating and managing the importance value of each neuron leaves room for exploration. Studies on network pruning show us how deep learning models can learn complicated knowledge with a surprisingly small size. However, without explicit intervention, deep neural networks tend to consume more capacity than actually needed. As a result, we found that the model trained with NPC algorithm performed relatively poorly on the last task in sequential training. We believe that NPC will benefit greatly if there is a method to enforce the model to use minimal capacity per task.

Appendix

Derivation of optimal plasticity

Starting from eq. (5),

$$\begin{aligned}\eta_i^* &= \underset{\eta_i}{argmin}\left(l_{stability}(\eta_i) + l_{plasticity}(\eta_i)\right) \\ &= \underset{\eta_i}{argmin}\{a_1 C_i \sigma(b_1\eta_i) + a_2(1-\sigma(b_2\eta_i))\}\end{aligned} \tag{8}$$

To get the minimum, first take $dl/d\eta_i = 0$, if $\sigma(x) \equiv sigmoid(x) = 1/(1+e^{-x})$, we get eq. (10), where $\beta = a_2 b_2 / a_1 b_1$.

$$\frac{a_1 b_1 C_i}{2(1+cosh(b_1\eta))} - \frac{a_2 b_2}{2(1+cosh(b_2\eta))} = 0 \tag{9}$$

$$\iff \frac{1+cosh(b_2\eta)}{1+cosh(b_1\eta)} = \frac{a_2 b_2}{a_1 b_1 C_i} = \frac{\beta}{C_i} \tag{10}$$

The nature of function $(1+cosh(b_2\eta))/(1+cosh(b_1\eta))$ depends heavily on whether $b_1 \geq b_2$ or $b_1 < b_2$. We set $b_1 < b_2$ as a constraint, since otherwise optimal η_i^* becomes a simple step function of C_i.

Let us first assume that $C_i \leq \beta$. we apply the Taylor approximation to solve eq. (10) because there is no closed-form inverse function of $(1+cosh(b_2 x))/(1+cosh(b_1 x))$. Given that $cosh$ is an even function, only the even degree terms remain, as shown in eq. (11).

$$\frac{1+cosh(b_2\eta_i)}{1+cosh(b_1\eta_i)} = 1 + \frac{1}{4}(b_2^2 - b_1^2)\eta_i^2 + O(\eta_i^4) = \frac{\beta}{C_i} \tag{11}$$

Assuming $O(\eta_i^4) \approx 0$, the solution of eq. (11) is the same as eq. (12), where $\alpha = \sqrt{4/(b_2^2 - b_1^2)}$.

$$\eta_i^* = \sqrt{\frac{4}{b_2^2 - b_1^2}\left(\frac{\beta}{C_i} - 1\right)} = \alpha\sqrt{\frac{\beta}{C_i} - 1} \tag{12}$$

In case of $C_i > \beta$, $l(\eta_i)$ increases strictly w.r.t. η_i. Since η is non-negative, we simply get optimal η as $\eta_i^* = 0$. Note that $\eta_i^* = 0$ at $C_i = \beta$ in eq. (9), which makes the two functions continuously connected. Combining the two cases where $C_i > \beta$ and $C_i \leq \beta$, respectively, the solution of eq. (6) is given by eq. (10), where $\alpha, \beta > 0$ are hyperparameters.

$$\eta_i^* = \begin{cases} \alpha\sqrt{\frac{\beta}{C_i} - 1} & \text{if } C_i \leq \beta \\ 0 & \text{if } C_i > \beta \end{cases} \tag{13}$$

If we use $\sigma(x) \equiv tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, with similar Taylor approximation, we get eq. (14), where $\alpha = 1/\sqrt{b_2^2 - b_1^2}, \beta = a_2 b_2 / a_1 b_1$.

$$\eta_i^* = \begin{cases} \alpha\sqrt{\sqrt{\frac{\beta}{C_i}} - 1} & \text{if } C_i \leq \beta \\ 0 & \text{if } C_i > \beta \end{cases} \tag{14}$$

For error function, since derivative of error function is simple Gaussian, we get eq. (15) without approximation, where $\alpha = 1/\sqrt{b_2^2 - b_1^2}, \beta = a_2 b_2 / a_1 b_1$

$$\eta_i^* = \begin{cases} \alpha\sqrt{log(\beta/C_i)} & \text{if } C_i \leq \beta \\ 0 & \text{if } C_i > \beta \end{cases} \tag{15}$$

Since sigmoid, tanh, and error function are S-shape functions with similar shape, the derived functions of optimal plasticity also have similar shapes and properties. We found that the choice of σ does not critically affect the performance of NPC. We used sigmoid function, which does not outperform but does slightly better than the others.

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Aljundi, R.; Chakravarty, P.; and Tuytelaars, T. 2017. Expert gate: Lifelong learning with a network of experts. 3366–3375.

Aljundi, R.; Rohrbach, M.; and Tuytelaars, T. 2018. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*.

De Vries, H.; Strub, F.; Mary, J.; Larochelle, H.; Pietquin, O.; and Courville, A. C. 2017. Modulating early visual processing by language. 6594–6604.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. PathNet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Gepperth, A., and Karaoguz, C. 2016. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation* 8(5):924–934.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kamra, N.; Gupta, U.; and Liu, Y. 2017. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*.

Kim, H.-E.; Kim, S.; and Lee, J. 2018. Keep and learn: Continual learning by constraining the latent space for knowledge preservation in neural networks. *arXiv preprint arXiv:1805.10784*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 201611835.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

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

LeCun, Y., H. F. J. B. L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2(104).

Lee, J.; Yun, J.; Hwang, S.; and Yang, E. 2017a. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Lee, S.-W.; Kim, J.-H.; Jun, J.; Ha, J.-W.; and Zhang, B.-T. 2017b. Overcoming catastrophic forgetting by incremental moment matching. 4652–4662.

Li, Z., and Hoiem, D. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40(12):2935–2947.

Liu, X.; Masana, M.; Herranz, L.; Van de Weijer, J.; Lopez, A. M.; and Bagdanov, A. D. 2018. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *arXiv preprint arXiv:1802.02950*.

Lopez-Paz, D. 2017. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems* 6467–6476.

Luo, J.-H., and Wu, J. 2017. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. 5068–5076.

Masse, N. Y.; Grant, G. D.; and Freedman, D. J. 2018. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences* 115(44):E10467–E10475.

McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. 24:109–165.

Mermillod, M.; Bugaiska, A.; and Bonin, P. 2013. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology* 4:504.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11264–11272.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2017. Variational continual learning. *arXiv preprint arXiv:1710.10628*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training GANs. 2234–2242.

Serrà, J.; Surís, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. 2990–2999.

Siavash Golkar, Michael Kagan, K. C. 2019. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*.

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

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yu, R.; Li, A.; Chen, C.-F.; Lai, J.-H.; Morariu, V. I.; Han, X.; Gao, M.; Lin, C.-Y.; and Davis, L. S. 2018. NISP: Pruning networks using neuron importance score propagation. 9194–9203.

Zacarias, A. S., and Alexandre, L. A. 2018. Overcoming catastrophic forgetting in convolutional neural networks by selective network augmentation. *arXiv preprint arXiv:1802.08250*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995. JMLR. org.