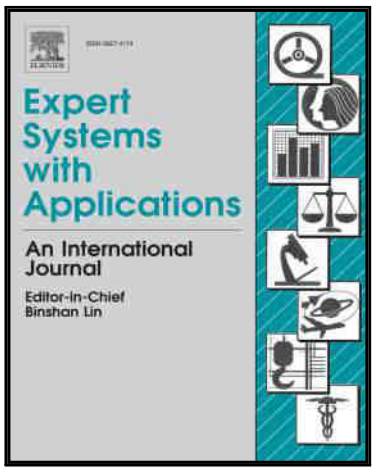
Accepted Manuscript



Difficulty-Weighted Learning: A Novel Curriculum-Like Approach Based on Difficult Examples for Neural Network Training

Tomoumi Takase

PII: S0957-4174(19)30417-8
DOI: https://doi.org/10.1016/j.eswa.2019.06.017
Reference: ESWA 12729

To appear in: *Expert Systems With Applications*

Received date: 20 December 2018
Revised date: 6 June 2019
Accepted date: 6 June 2019

Please cite this article as: Tomoumi Takase , Difficulty-Weighted Learning: A Novel Curriculum-Like Approach Based on Difficult Examples for Neural Network Training, *Expert Systems With Applications* (2019), doi: https://doi.org/10.1016/j.eswa.2019.06.017

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

HIGHLIGHTS

- We prioritize the classification of difficult examples over easy examples.
- We proposed difficulty-weighted learning (DWL) for neural network training.
- DWL uses a loss function weighted by the neural network outputs.
- We evaluated the performance of DWL on several benchmark datasets.
- DWL has better generalization performance for MLP or a small CNN

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Difficulty-Weighted Learning: A Novel Curriculum-Like Approach Based on Difficult Examples for Neural Network Training

Author names and affiliations

Tomoumi Takase

Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology, 2-4-7 Aomi, Koto-ku, Tokyo, Japan

takase_316@yahoo.co.jp

Corresponding Author

Tomoumi Takase

Abstract

Curriculum learning, in which training examples gradually proceed from easy to difficulty, has been applied to various tasks and demonstrated better performance than other machine learning approaches. However, identifying the difficulty level in advance often requires domain knowledge and is a time-consuming process. We dynamically decide the difficulty of examples based on outputs from neural networks during training and propose a loss function to promote training with difficult examples. Experimental results verify that the proposed method improves the generalization ability across several datasets.

Keywords

Neural network; Curriculum learning; Supervised learning; Deep learning; Multilayer perceptron; Classification

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

1. Introduction

Neural networks have been demonstrating excellent classification performance for various datasets of images, audio, language, among others. This performance has relied on the development of robust training methods such as fine-tuning (Hinton and Salakhutdinov, 2006; Mesnil et al., 2012; Yosinski et al., 2014) and generative adversarial networks (Goodfellow et al., 2014; Radford, Metz, and Chintala, 2015). Curriculum learning, proposed by Bengio et al. (2009), is another powerful training method, in which learning gradually proceeds from easy to difficult examples, aiming to resemble human learning. Its proponents successfully applied curriculum learning to classification of geometric shapes and language processing.

In this paper, we prioritize the classification of difficult examples over easy examples. Therefore, we focus on the training of difficult examples and employ the conventional curriculum learning (Bengio et al., 2009) to train easy examples. A training strategy based on difficulty can be easily implemented in neural networks, because the classification outputs represent the degree of confidence, that is, the difficulty of the examples. To increase the weight of difficult examples over easy ones, we use a loss function weighted by the network outputs. As the loss function is determined at each iteration, it can reflect the varying difficulty of examples, establishing the proposed method, which we call difficulty-weighted learning (DWL).

DWL is strongly related to expert systems because it automatically retrieves the difficulty level of examples based on the devised loss function, whereas conventional methods, such as curriculum learning (Bengio et al., 2009), require domain knowledge for each task. Furthermore, as DWL is supported by neural networks, which are powerful intelligent systems, the DWL implementation can be regarded as an expert and intelligent system.

The contributions of the proposed DWL are summarized as follows:

(1) DWL is a novel training method for neural networks and can be easily implemented without a considerable burden in computation time.

(2) DWL improves curriculum learning by adopting positive training based on a loss function targeting difficult examples.

(3) The high performance of DWL is demonstrated by training a multilayer perceptron (MLP) and convolutional neural networks (CNNs) on the MNIST (LeCun et al., 1998a), CIFAR-10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), Fashion-MNIST (Xiao, Rasul, and Vollgraf, 2017), and several datasets from the UCI Machine Learning Repository (Dua and Karra, 2017).

2. Related Work

Curriculum learning (Bengio et al., 2009) requires grouping examples into several sets before training according to their difficulty, which is established by prior knowledge. Hence, the criteria for deciding difficulty depend on tasks. For instance, in shape recognition of squares, circles, and

ACCEPTED MANUSCRIPT

equilateral triangles, Bengio et al. (2009) used the shape complexity as criterion and generated the BasicShapes and GeomShapes datasets, which have low and high variability, respectively. Hence, training proceeded from the BasicShapes to the GeomShapes dataset. Spitkovsky, Alshawi, and Jurafsky (2009) defined a short sentence as easy for language processing. In addition, they proposed "baby-step" learning, which improves conventional curriculum learning by training a model including previously used examples, whereas curriculum learning replaces an easy group by a difficult one during training. Nevertheless, curriculum learning and baby-step learning require domain knowledge and manual preprocessing before training.

In contrast, self-paced learning (Kumar, Packer, and Koller, 2010) automatically creates a boundary surface between difficult and easy examples from a training loss. It starts by training easy examples and gradually adds examples as training proceeds. The difficulty of examples is flexible, although it is fixed during training in the conventional curriculum learning. Self-paced learning with diversity, proposed by Jiang et al. (2014), improves the conventional approach by an automatic curriculum based on both diversity and difficulty. Then, Jiang et al. (2015) proposed self-paced curriculum learning, which considers both prior knowledge and learning progress. Although these methods do not require prior knowledge, they include hyperparameters that are difficult to adjust (e.g., training pace).

The abovementioned methods aim for efficiency by training easy examples before proceeding with difficult ones. Although this idea resembles human learning, it may not lead to improved classification accuracy. Unlike these methods, we aim to improve classification performance by reducing the training error of difficult examples. In this sense, our approach is similar to a pioneering algorithm called adaptive boosting (Freund and Schapire, 1997), which assigns an importance degree to each example and increases the degree for misclassified examples. The final prediction is decided by an ensemble of classifiers. However, as adaptive boosting has a training algorithm different from

Table 1 Comparison of related learning approaches.

Method	Domain knowledge	Manual preparation before training	Hyperparameter adjustment	Target examples for training
Curriculum learning (Bengio et al., 2009)	Yes	Yes	No	Easy
Baby-step learning (Spitkovsky, Alshawi, and Jurafsky, 2009)	Yes	Yes	No	Easy
Self-paced learning (Kumar, Packer, and Koller, 2010)	No	No	Yes	Easy
Adaptive boosting (Freund and Schapir, 1997)	No	No	No	Difficult
Difficulty-weighted learning (Ours)	No	No	No	Difficult

ACCEPTED MANUSCRIPT

that of neural networks, it is used in other contexts compared to DWL. Table 1 summarizes the characteristics of DWL and related approaches.

3. Difficulty-Weighted Learning

DWL mainly relies on (1) outputs of neural networks to determine the difficulty of examples and (2) a loss function weighted by difficulty. Regarding point (1), output z_i from class i of a neural network is determined using the softmax function as

$$z_i = \frac{\exp(a_i)}{\sum_{j=1}^{N} \exp(a_j)}, \quad (1)$$

where N is the number of classes and a_i is the input to unit i of the output layer. The sum over every a_j of z_i is 1, and each z_i represents the degree of confidence in the example. For instance, large z_i indicates that an example belongs to a class with high confidence, and therefore, it can be regarded as an easy example. In contrast, small z_i indicates low confidence and increased difficulty.

An application of the difficulty level is knowledge distillation (Hinton, Vinyals, and Dean, 2015), which uses it to transfer the performance from a larger (teacher) to a smaller (student) model. The outputs of examples obtained from the teacher model are considered as soft labels, which take values in [0, 1] (in contrast, hard labels are represented by {0, 1}), and the student model is trained using these examples.

Regarding point (2), we use a loss function weighted by the example difficulty. The weights are expressed as

$$w_d = 1 - f(\theta; x_d)|_{\theta=\theta_t}, \quad (2)$$

where f denotes a neural network model, x_d represents the inputs of the d-th example in a minibatch, and θ_t denotes parameters θ at the t-th update. Then, a weighted cross-entropy function can be given by the weighted mean using the difficulty of each example:

$$L(\theta) = \frac{\sum_d^D w_d \cdot \left(y_d \cdot \ln\left(f(\theta; x_d)\right)\right)}{\sum_d^D w_d}, \quad (3)$$

where D is the number of examples in a minibatch and y_d is the one-hot label of the d-th example. Although weighted cross-entropy is useful for imbalanced training data and widely used given its suitability for deep learning, as demonstrated by its use in libraries such as PyTorch and TensorFlow, the proposed cross-entropy in Eq. (3) weighted by example difficulty constitutes a novel function.

The process of DWL is detailed in Algorithm 1, where (x, y) are training examples, θ_0 are initial neural network parameters, T denotes the number of updates, D denotes the number of examples

ACCEPTED MANUSCRIPT

Algorithm 1 Difficulty-Weighted Learning

1: **Input:**
   (x, y), θ_0, T, D, f, g
2: **Output:**
   θ_T
3: **for** $t = 0$ to $T - 1$ **do**
4: $l(\theta), w \leftarrow 0$
5: **for** $d = 0$ to $D - 1$ **do**
6: $w_d \leftarrow 1 - f(\theta; x_d)|_{\theta=\theta_t}$
7: $l(\theta) \leftarrow l(\theta) + w_d \cdot (y_d \cdot \ln(f(\theta; x_d)))$
8: $w \leftarrow w + w_d$
9: **end for**
10: $L(\theta) \leftarrow l(\theta)/w$
11: $\theta_{t+1} \leftarrow g(L(\theta))$
12: **end for**
13: **return** θ_T

in a minibatch, f is the feedforward neural network function, and g is the backward neural network function. The proposed weighted cross-entropy is applied to the training phase but not to the test phase. During training, computation using Eqs. (2) and (3) is performed after forward propagation and before backpropagation. Step 6 corresponds to Eq. (2), and steps 7, 8, and 10 correspond to Eq. (3). For step 11, parameters are updated using a weighted loss function $L(\theta)$.

Although DWL is similar to adaptive boosting by focusing training on difficult examples, these methods differ because DWL objectively retrieves difficulty based on the network outputs, and classifiers from previous epochs are discarded for final prediction.

Figure 1 illustrates the expected effect of DWL, where the easiness of examples is depicted as a dynamic process during training. An example is considered as correctly classified when it retrieves classification values within the region of highest output, that is, the output corresponding to the example class is the highest among the outputs of all classes. When the output for a difficult example

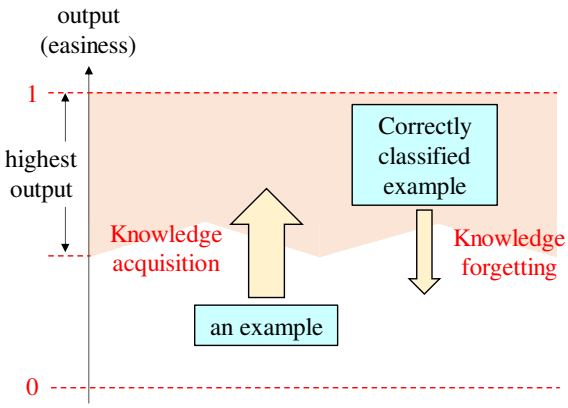


Fig. 1 Knowledge acquisition and forgetting of examples for classifier. Highest output is the highest classification value among classes in the output layer for each example.

ACCEPTED MANUSCRIPT

with a large training loss increases due to the weighted loss function in Eq. (3) and the example is correctly classified, we consider that knowledge acquisition occurs in the classifier. As DWL considerably adjusts weights for difficult examples, we assume that classifying them with high confidence requires many acquisitions. Therefore, misclassified examples tend to be correctly classified as training proceeds, thus reducing the classification error. Although this approach is applied during training, it is expected to have a similar effect for testing. In addition, an example that has been correctly classified may become a misclassification as training proceeds, in a phenomenon we call knowledge forgetting in the classifier. Still, we expect that acquisition occurs more often than forgetting during DWL.

4. Evaluation

4.1 Experimental Setup

To evaluate the performance of the proposed DWL, we conducted experiments on several benchmark datasets for machine learning, namely, MNIST (LeCun et al., 1998a), CIFAR-10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), Fashion-MNIST (Xiao, Rasul, and Vollgraf, 2017), and several datasets from the UCI Machine Learning Repository (Dua and Karra, 2017).

MNIST, CIFAR-10, SVHN, and Fashion-MNIST are image classification datasets containing 10 classes, whose inputs are intensity values (from 0 to 255). Specifically, the MNIST dataset consists of handwritten numerical digits (from 0 to 9) and contains 60,000 training and 10,000 test examples of size 28×28 pixels. The CIFAR-10 dataset consists of images from objects in natural scenes (32×32 pixels) with 50,000 training and 10,000 test examples. The SVHN dataset consists of images showing digits in natural scenes (32×32 pixels) with 73,252 training and 26,032 test examples. The Fashion-MNIST dataset consists of black and white clothing images and has the same structure as the MNIST dataset.

We also used the Car Evaluation, Wine, Letter Recognition, and Epileptic Seizure Recognition (Andrzejak et al., 2001) datasets from the UCI Machine Learning Repository (Dua and Karra, 2017) for multiple classification. The Car Evaluation dataset contains 1,728 examples and four classes for overall evaluation. It comprises six features such as buying price and number of doors. The wine dataset contains 178 examples and three classes. It comprises 13 features from chemical analyses such as those for alcohol and malic acid. The Epileptic Seizure Recognition dataset contains 11,500 examples, five categories regarding the conditions under which the subjects had an epileptic seizure, and 178 features corresponding to data chunks from brain activity. We used 35% of the training examples from these datasets for testing. The Letter Recognition dataset contains 20,000 examples, and we used the first 16,000 examples as training data and the remaining 4,000 examples for testing, as recommended by Dua and Karra (2017). The dataset contains 26 classes, from A to Z in the

ACCEPTED MANUSCRIPT

Table 2 Structures of convolutional neural networks (CNNs). (a: image size before flattening)

(a) Small CNN.

Layer type	Channels/Units
Input	1 or 3
3×3 convolutional	32
3×3 convolutional	32
2×2 max pooling	32
3×3 convolutional	64
3×3 convolutional	64
2×2 max pooling	64
Flattened	$a \times a \times 64$
Fully connected	512
Fully connected	10

(b) CNN similar to VGGNet (Simonyan and Zisserman, 2015).

Layer type	Channels/Units
Input	3
3×3 convolutional	64
3×3 convolutional	64
2×2 max pooling	64
3×3 convolutional	128
3×3 convolutional	128
3×3 convolutional	128
2×2 max pooling	128
3×3 convolutional	256
3×3 convolutional	256
3×3 convolutional	256
3×3 convolutional	256
2×2 max pooling	256
Flattened	$a \times a \times 256$
Fully connected	1024
Fully connected	10

(c) AlexNet (Krizhevsky, Sutskever, & Hinton, 2012).

Layer type	Channels/Units
Input	3
3×3 convolutional	64
3×3 max pooling	64
3×3 convolutional	192
3×3 max pooling	192
3×3 convolutional	384
3×3 convolutional	384
3×3 convolutional	256
3×3 max pooling	256
Flattened	$a \times a \times 256$
Fully connected	4096
Fully connected	4096
Fully connected	10

English alphabet, and 16 features mainly related to statistical moments and edge counts.

For evaluation, we used an MLP with one hidden layer and CNNs with the structures listed in Table 2. A rectified linear unit (Glorot, Bordes, and Bengio, 2011) was inserted after each layer except for the output layer, which used a softmax function. The Glorot uniform initializer (LeCun et al., 1998b; Glorot and Bengio, 2010) established the first set of weights in the neural network. In addition, we used stochastic gradient descent with initial learning rate of 0.01 for optimization. In the CNNs, the convolutional layers in Table 2(b) maintain the size of input data by padding. To avoid overfitting, we inserted batch normalization (Ioffe and Szegedy, 2015) after each layer in the CNNs. We did not use dropout (Srivastava et al., 2014) to follow the recent findings by Li et al. (2018), who noted that using both dropout and batch normalization can degrade performance. Moreover, we did not perform data augmentation.

Training proceeded with a batch size of 64 for 1,000 epochs when using either the MLP or small CNN (Table 2(a)) and with a batch size of 128 for 200 epochs when using the large CNN (Table 2(b)). The test accuracies were calculated at every epoch. We represented the standard error over five trials for different initial weights using error bars. The simulation was implemented and conducted using the Theano and Numpy Python libraries.

We compared the results for four methods, namely, DWL, default learning (without curriculum), conventional curriculum learning (Bengio et al., 2009), and baby-step learning (Spitkovsky, Alshawi, and Jurafsky, 2009). Although comprehensive and adaptable, self-paced learning (Kumar, Packer, and Koller, 2010) is difficult to implement given its task-dependent hyperparameter selection. Instead, we used the conventional curriculum learning with a curriculum based on training loss.

4.2 Data Preprocessing for Comparison Methods

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

In conventional curriculum and baby-step learning, difficulty is decided before training and examples must be grouped based on the difficulty, whereas in DWL, difficulty can be dynamically decided during training, as described in Section 3. Hence, we conducted the preprocessing illustrated in Fig. 2 for the comparison methods. First, training data were divided into two datasets, A and B, containing equal number of examples. For example, in the MNIST dataset, the first 30,000 examples defined dataset A and the last 30,000 defined dataset B. Then, a neural network was trained using dataset A, and dataset B was input in the resulting trained neural network. The difficulty levels of the examples in dataset B can be obtained from the network outputs. Finally, dataset B is divided into groups containing easy and increasingly difficult examples. Training for evaluation proceeds from easy to difficult examples, by gradual example replacement for conventional curriculum learning and example addition for baby-step learning. In this study, we used two groups of difficult and easy examples according to the experimental conditions described in (Bengio et al., 2009).

DWL and default learning do not require this time-consuming preprocessing, and all examples can be used for training, whereas only half of the training examples can be used for conventional curriculum and baby-step learning. For fair comparison, we used half of the examples for DWL and default learning, besides using all the examples for a complete evaluation.

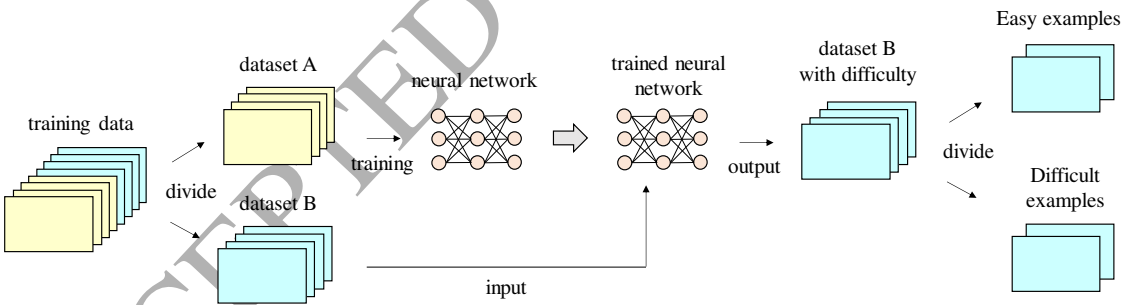
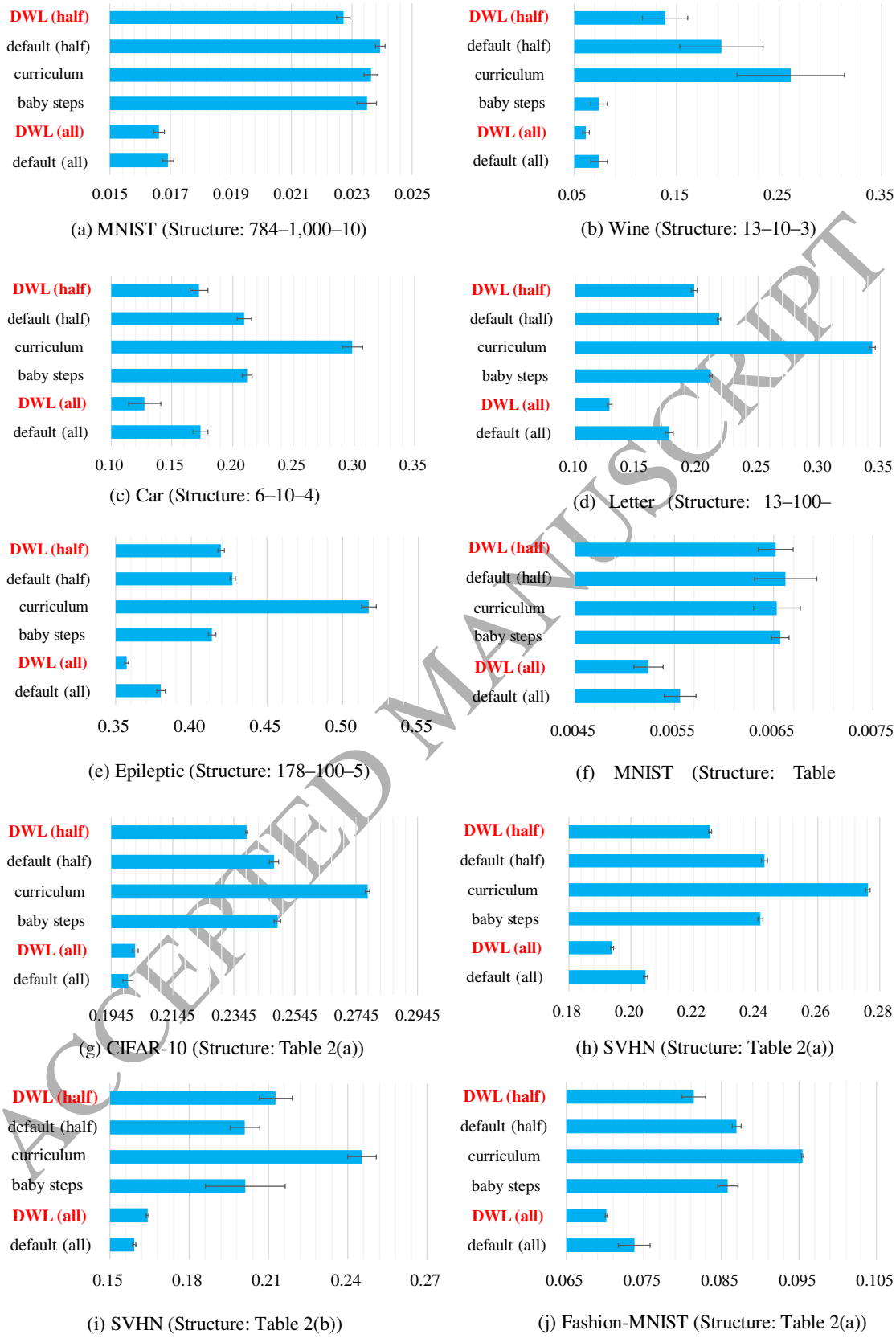


Fig. 2 Preprocessing to establish difficulty levels of a dataset to apply conventional curriculum and baby-step learning. Only half of the original training examples can be used for training, whereas all the examples are available for DWL.



(a) MNIST (Structure: 784–1,000–10)

(b) Wine (Structure: 13–10–3)

(c) Car (Structure: 6–10–4)

(d) Letter (Structure: 13–100–

(e) Epileptic (Structure: 178–100–5)

(f) MNIST (Structure: Table

(g) CIFAR-10 (Structure: Table 2(a))

(h) SVHN (Structure: Table 2(a))

(i) SVHN (Structure: Table 2(b))

(j) Fashion-MNIST (Structure: Table 2(a))

Fig. 3 Test error rates when using different methods on evaluated datasets. Error bars represent standard error. Half and all indicate the amount of examples used for training of DWL and default

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5. Results and Discussion

Figure 3 shows the test error rates using the evaluated methods on the selected datasets. In many cases with MLP, judging from the overlapping error bars, DWL with half the training examples shows significantly lower error rates than the comparison methods, namely, default learning with the same number examples, curriculum learning, and baby-step learning. Moreover, with all the training examples, DWL shows significantly lower error rates than default learning. For the CNNs, DWL tends to retrieve lower error rates than the comparison methods, but when using the large CNN model, DWL does not perform well, as seen by comparing Fig. 3(h) and (i).

The preprocessing to obtain easy and difficult example groups based on outputs might not be appropriate for conventional curriculum learning, as shown by the high error rates. However, this preprocessing is not always inappropriate for baby-step learning, and difficulty determination is suitable for DWL, which weights the loss functions by the outputs.

We also trained the VGG-like CNN with the architecture detailed in Table 2(b) and AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) with the architecture detailed in Table 2(c) using all data from the CIFAR-10 dataset. In this case, the test error rates for default learning were 0.219 for the VGG-like CNN and 0.207 for AlexNet, whereas the rate was 0.200 for the small CNN (Table 2(a)). Hence, larger CNNs do not necessarily improve performance, and hence such models are not appropriate for evaluating DWL.

ACCEPTED MANUSCRIPT

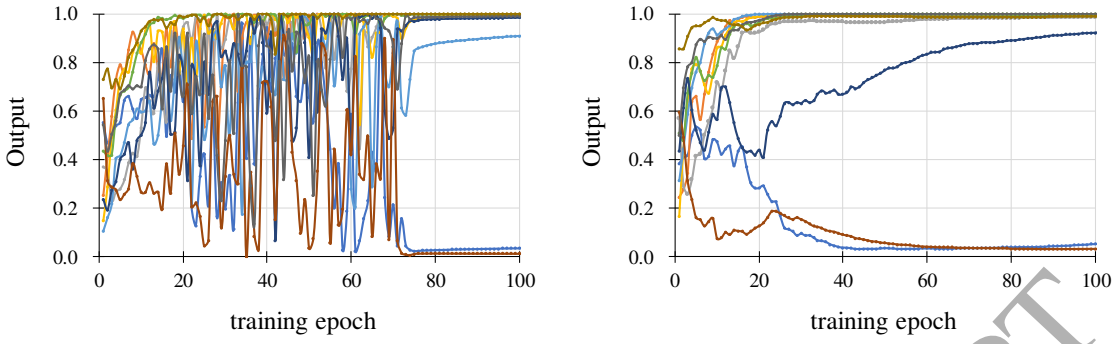
Table 3 Training error rates and training losses when using DWL and default learning (no curriculum). The information in parentheses in column dataset indicates the structures of the MLP or CNN. The numbers in bold in columns training error rate and loss denote significantly smaller values than those of the other methods.

Dataset	Method	Training error rate	Training loss
MNIST (784–1,000–10)	**DWL (half)**	**0**	**29.6**
	Default (half)	0	49.5
	DWL (all)	0	**36.0**
	Default (all)	0	63.9
Wine (13–10–3)	**DWL (half)**	5.52×10^{-2}	37.1
	Default (half)	0.114	**34.5**
	DWL (all)	2.93×10^{-2}	45.1
	Default (all)	3.45×10^{-2}	**32.9**
Car (6–10–4)	**DWL (half)**	**0.145**	4.20×10^{2}
	Default (half)	0.174	$\mathbf{2.44 \times 10^{2}}$
	DWL (all)	**0.111**	7.45×10^{2}
	Default (all)	0.146	$\mathbf{4.07 \times 10^{2}}$
Letter (13–100–26)	**DWL (half)**	**0.163**	9.60×10^{3}
	Default (half)	0.191	$\mathbf{5.73 \times 10^{3}}$
	DWL (all)	$\mathbf{9.68 \times 10^{-2}}$	1.49×10^{4}
	Default (all)	0.150	$\mathbf{8.69 \times 10^{3}}$
Epileptic (178–100–5)	**DWL (half)**	**0.331**	4.10×10^{3}
	Default (half)	0.353	$\mathbf{3.17 \times 10^{3}}$
	DWL (all)	**0.278**	7.16×10^{3}
	Default (all)	0.308	$\mathbf{5.29 \times 10^{3}}$

Dataset	Method	Training error rate	Training loss
MNIST (Table 2(a))	**DWL (half)**	0	2.89
	Default (half)	0	**0.914**
	DWL (all)	0	23.7
	Default (all)	0	**1.41**
CIFAR-10 (Table 2(a))	**DWL (half)**	0	58.7
	Default (half)	0	**17.7**
	DWL (all)	8.00×10^{-5}	1.23×10^{3}
	Default (all)	$\mathbf{3.20 \times 10^{-5}}$	**49.2**
SVHN (Table 2(a))	**DWL (half)**	2.02×10^{-4}	1.66×10^{3}
	Default (half)	$\mathbf{1.31 \times 10^{-4}}$	**61.6**
	DWL (all)	6.28×10^{-4}	7.92×10^{3}
	Default (all)	$\mathbf{2.87 \times 10^{-4}}$	$\mathbf{1.35 \times 10^{2}}$
SVHN (Table 2(b))	**DWL (half)**	2.29×10^{-3}	1.16×10^{4}
	Default (half)	$\mathbf{8.19 \times 10^{-5}}$	**43.5**
	DWL (all)	1.39×10^{-3}	2.16×10^{4}
	Default (all)	$\mathbf{5.46 \times 10^{-5}}$	**43.5**
Fashion-MNIST (Table 2(a))	**DWL (half)**	0	20.5
	Default (half)	0	**7.88**
	DWL (all)	**0**	1.09×10^{2}
	Default (all)	1.67×10^{-5}	**27.4**

Table 3 lists the training error rates and training losses when using DWL and default learning (without curriculum). DWL tends to retrieve lower training error rates for MLP and higher training losses for MLP and CNN than default learning. The lower training error rates in DWL are expected, because DWL focuses training on examples classified with small confidence. The higher training loss in DWL is also expected, because the loss function weighs more heavily examples with high loss. However, using CNN, the training error rate tends to become larger in DWL than in default learning, indicating that DWL does not always retrieve the lower test error rates, as shown in Fig. 3.

ACCEPTED



(a) DWL outputs

(b) Default learning outputs

Fig. 4 Output transitions of first 10 examples in the CIFAR-10 dataset for 100 epochs during CNN training with weighting and no curriculum.

To demonstrate that knowledge acquisition and forgetting in the classifier increase when using DWL, Fig. 4 shows transitions of outputs for correct classifications of the first 10 examples in the CIFAR-10 dataset. The transitions were tracked over 100 epochs during CNN training with DWL and default learning. The outputs for DWL show high variability, whereas those for default learning are mostly constant and convergent.

We further investigated the results of knowledge acquisition and forgetting on the Epileptic Seizure Recognition dataset, which retrieved the largest training error rate among datasets (Table 3). Specifically, we trained an MLP using all the examples and counted the numbers of knowledge acquisition and forgetting occurrences per epoch. The number of acquisitions per example for 1,000 epochs were 7,618 in DWL and 7,022 in default learning, and those of forgetting were 3,526 in DWL and 3,131 in default learning. Thus, the differences between the occurrences of knowledge acquisition and forgetting were 4,092 in DWL and 3,891 in default learning. The larger difference in DWL might explain the lower training and test error rates. These results suggest that the DWL performance strongly depends on the dataset. If the model is overfit to difficult examples using DWL, it correctly classifies such examples but may incorrectly classify others, thus undermining classification accuracy.

6. Summary and future work

Based on a novel curriculum-like approach of positive learning for difficult examples, we propose DWL, which weights the loss function using the difficulty of examples. By focusing on the training of difficult examples, this approach differs from conventional curriculum and baby-step learning.

Experiments using MLP and CNN training with several datasets verify that DWL has better generalization ability for MLP or a small CNN, but not for a large CNN. To correctly classify difficult examples, knowledge acquisition for examples increases with DWL, but knowledge

ACCEPTED MANUSCRIPT

forgetting also increases. The training and test accuracies improve when the difference between occurrences of knowledge acquisition and forgetting for DWL is larger than that of default learning (without curriculum).

DWL can be further improved and extended for realizing more effective neural network training. Directions of future work are provided below:

- Explore more representative expressions for example difficulty than the multiplication of the loss function by difficulty (Eq. (3)).
- As the loss function is defined for each minibatch, the effect of batch size on training with DWL should be investigated.
- Perform a theoretical analysis of DWL that may provide information regarding the conditions required for successful learning.
- Apply DWL to various classification models including binary classification and recurrent neural networks.

Acknowledgement

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

References

Andrzejak, R. G., Lehnertz, K., Rieke, C., Mormann, F., David, P., & Elger C. E. (2001). Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, *Physical Review E*, 64, 061907.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, pp. 41–48

Dua, D. & Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*. 55(1):119–139.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of Artificial Intelligence and Statistics Conference (AISTATS)*, 9, pp. 249–256.

ACCEPTED MANUSCRIPT

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of Artificial Intelligence and Statistics Conference*, pp. 315–323.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pp. 2672–2680.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hinton, G. E., Vinyals, O., & Dean J. (2015). Distilling the knowledge in a neural network. arXiv: 1503.02531.

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. arXiv: 1502.03167.

Jiang, L., Meng, D., Yu, S., Lan, Z., Shan, S., & Hauptmann, A. (2014). Self-paced learning with diversity. In *Advances in Neural Information Processing Systems 27 (NIPS)*.

Jiang, L., Meng, D., Zhao, Q., Shan, S., & Hauptmann, A. G. (2015). Self-paced curriculum learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2694–2700.

Krizhevsky, A. & Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images. *Technical report*, University of Toronto.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25 (NIPS)*.

Kumar, M. P., Packer, B., & Koller, D. (2010). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1189–1197.

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

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. (1998b). Efficient BackProp. *Neural networks: Tricks of the Trade*, pp. 9–48.

Li, X., Chen, S., Hu, X., & Yang, J. (2018). Understanding the disharmony between dropout and batch normalization by variance shift. arXiv: 1801.05134.

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I. J., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A. & Bergstra, J. (2012). Unsupervised and transfer learning challenge: a deep learning approach. *ICML Unsupervised and Transfer Learning*, 27, pp. 97–110.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv: 1511.06434.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv: 1409.1556.

Spitkovsky, V. I., Alshawi, H., & Jurafsky, D. (2009). Baby Steps: How "Less is More" in Unsupervised Dependency Parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*, pp. 1–10.

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

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv: 1708.07747.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3320–3328.

ACCEPTED MANUSCRIPT

Credit Author Statement

Tomoumi Takase: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review & Editing

ACCEPTED MANUSCRIPT