Auxiliary Data Selection in Percolative Learning Method for Improving Neural Network Performance

Masayuki Kobayashi[a], Shinichi Shirakawa[b] and Tomoharu Nagao[c]
Faculty of Environment and Information Sciences, Yokohama National University,
79-7 Tokiwadai Hodogaya-ku Yokohama, Japan

Keywords: Neural Network, Feature Selection, Auxiliary Data.

Abstract: Neural networks have been evolved significantly at the cost of requiring many input data. However, collecting useful data is expensive for many practical uses, which can be barrier for practical use in real-world applications. In this work, we propose a framework for improving the model performance, in which the model leverages the auxiliary data that is only available during the training. We demonstrate how to (i) train the neural network to perform as though auxiliary data are used during the testing, and (ii) automatically select the auxiliary data during training to encourages the model to generalize well and avoid overfitting to the auxiliary data. We evaluate our method on several datasets, and compare the performance with baseline model. Despite the simplicity of our method, our method makes it possible to get good generalization performance in most cases.

1 INTRODUCTION

Neural networks have advanced significantly and been used in various of tasks. This is owing to technical and architectural innovations as well as the availability of data that allows applying them in many practical problems. Many of these approaches, especially those that perform well, require enormous amounts of input data. However, these requirements not only can be a barrier to the adoption in application, but also come at cost of obtaining useful data. This is best illustrated by soft sensing applications where many of useful inputs are available only in the laboratory environment. With limited inputs, the complicated relationships between the inputs and outputs might be difficult to learn, thus this leads to overfitting and low generalization.

In this paper, we propose a new framework for training neural networks to solve the above issues. The key strategy is to leverage the auxiliary data that is only available during the training; that is, better and generalized feature representations might be obtained. The idea is encouraged by previous work on percolative learning* (Yanagimoto and Nagao, 2017; Takaishi et al., 2018), which established that the model can be trained as though both the main and aux data are provided during the testing. With non-efficient aux data, the model can be easily overfitted. We therefore propose to automatically select the efficient aux data for training. To this end, we introduce real-valued gate parameters and optimize them over the training set. This approach encourages to train models that generalize well rather than models that overfit to aux data.

We applied our method to several classification datasets, and empirically showed that our approaches achieved better performance than baseline models in most cases. The innovations and contributions of this paper are summarized as follows:

- Our method achieves good generalization performance, but also alleviates the need for inputs by leveraging the aux data during the training.
- Despite the simplicity of our method, our method demonstrates the advantage over the baseline.

2 RELATED WORKS

Our method leverages the additional inputs to improve the neural network performance. In this section, we briefly review the two directions of the most

[a] https://orcid.org/0000-0002-5882-8359
[b] https://orcid.org/0000-0002-4659-6108
[c] https://orcid.org/0000-0002-2841-9538
*This algorithm is patent pending in Japan.

Kobayashi, M., Shirakawa, S. and Nagao, T.
Auxiliary Data Selection in Percolative Learning Method for Improving Neural Network Performance.
DOI: 10.5220/0010825700003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 381-387
ISBN: 978-989-758-547-0; ISSN: 2184-433X

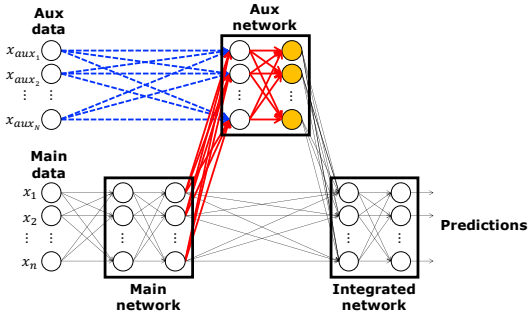


Figure 1: Overview of the percolative learning. The model comprises three sub-networks, all of which are convolutional / fully connected neural networks.

related work: multimodal learning and percolative learning.

Multimodal learning has shown good performance and been studied in the field of deep learning. These methods leverage the additional inputs, however, the inputs from both modalities must be available even in the testing phase. To make the best of multimodal data, researcher have studied to efficiently learn a shared feature representation across modalities.

Ngiam *et al*. (Ngiam et al., 2011) trained the encoder-decoder model to reconstruct the inputs from both modalities. This allows the model to extract a shared feature representation across different modalities. Although this method has assumption that the input modalities are desired to have strong correlations, this method performs well using the input from one modality. A weakly shared deep transfer networks (DTNs) are proposed in (Shu et al., 2015) to generate both domain-specific features and the shared features across domains. Although this method showed the potential of multimodal learning, there is still room for flexibility.

Percolative learning (Yanagimoto and Nagao, 2017) can be viewed as multimodal learning, but is more straightforward to solve above issues. Consider the situation where all the inputs are available during the training but some of the inputs can be used only for testing, this method allows the model to perform well as though all inputs are provided during the testing. The key strategy is to use both the main and aux data and efficiently learn shared feature representations between these data. Percolative learning has proven their effectiveness on various tasks. Yanagimoto and Nagao have tested this method on image classification tasks using the MNIST dataset (LeCun et al., 1998). Takaishi *et al*. (Takaishi et al., 2018) have extended this method and applied it to time-series prediction tasks. Our method is inspired by the recent success of percolative learning, and we exploit the potential of leveraging the aux data.

3 OUR METHOD

Our method makes use of the aux data that is available only during the training. The training process we use in this work is based on the Percolative Learning framework proposed by (Yanagimoto and Nagao, 2017), in which the model is trained to perform as though aux data are provided even during the testing phase. The main contribution of this work is the training process, such that the model is trained to generalize well to tasks. The inspiration is that feature engineering uses domain knowledge of data to achieve good performance. Our percolative learning also relies on domain knowledge to consider which aux data should be used for percolating; otherwise, the model can be easily overfitted to the aux data. These observations suggest that it might be possible to get improvement by selecting the aux data for training; see Fig. 2 for its overview.

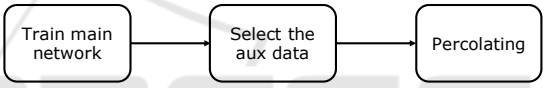


Figure 2: The overall pipeline of our method.

In our method, the aux data is automatically selected so as to get improvements in terms of the final generalization performance. To this end, we present two ways to implement aux data selection, simple gradient based method and natural gradient based method. Both approaches allow the model to jointly optimize the network weights and the aux data selection using the gradient descent.

In this section, we describe our training framework. We first explain the details of percolative learning, and then describe the network architecture that we use in our method. Lastly, we will explain the training process in detail.

3.1 Background

We briefly explain the percolative learning framework that we proposed in (Yanagimoto and Nagao, 2017). In this method, we train the model using two types of data: main and aux data. The main data is standard data for training neural networks, whereas the aux data is additional data that supports the model training process, but only available during the training phase. The basic architecture of this method is shown in Fig. 1. As can be seen in Fig. 1, the network comprises three sub-networks: an aux network, a main

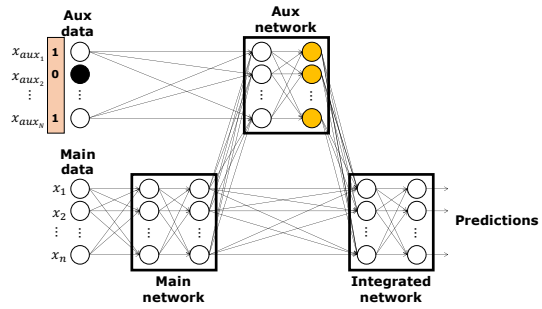


Figure 3: Overview of our method. We introduce real-valued parameters to determine which aux data to use for percolating.

network, and an integrated network, all of which are convolutional / fully connected neural networks and end-end trainable. The aux and integrated networks take two feature maps and concatenate them in the channel dimension to use some beneficial aux data for training. To efficiently train our model, we decompose the training process of this method into two phases: pre-training and percolating.

Pre-training Phase: We use both the main and aux data and optimize the weights of the whole networks, and in turn the shared feature representations, called *percolative features*, are obtained the main and aux data.

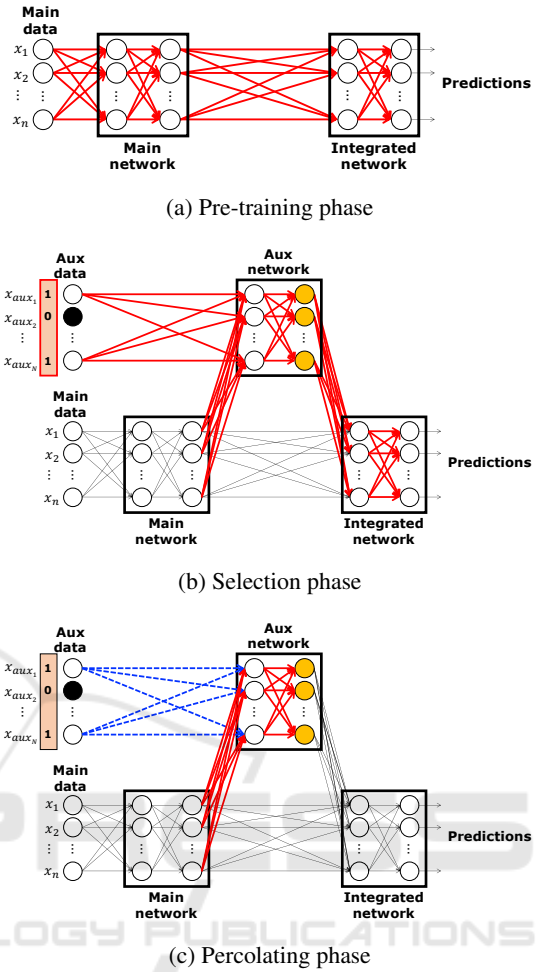
Percolative Phase: We train the network to perform well while reducing the magnitude of the aux data. Specifically, we introduce a parameter $\alpha (0 \leq \alpha \leq 1.0)$, and gradually reduce the magnitude of the aux data by element-wise multiplication αx_{aux}. The initial value of α is 1.0 and slowly decayed to zero. We then update the weights of only aux network in order not to change the percolative features. After this phase is completed, the aux data is no longer available, but the network is considered to be able to represent the same percolative features as those obtained during the pre-training phase. To this end, we use the following training loss $\mathcal{L}_{\text{perc}}$ to update the weights of aux network:

$$\mathcal{L}_{\text{perc}} = \frac{1}{|\mathcal{T}|} \sum_{j=1}^{|\mathcal{T}|} \| f_{\text{perc}}(x_{\text{main}_j}, \alpha x_{\text{aux}_j}) - F_{\text{perc}}(x_{\text{main}_j}, x_{\text{aux}_j}) \|_2^2$$

where we denote the network's percolative features by $f_{\text{perc}}(\cdot)$; $F_{\text{perc}}(\cdot)$ is the percolative features obtained in the pre-training phase; x_{main} is the main data; x_{aux} is the aux data; $\mathcal{T}$ is the training set.

3.2 Network Architecture

The network architecture of our method is shown in Fig. 3. Although the network architecture is one of the most important aspects that affect the performance



(a) Pre-training phase

(b) Selection phase

(c) Percolating phase

Figure 4: The training process is divided into three phases: (a) Pre-training, (b) Selection and (c) Percolating.

of our percolative learning (Takaishi et al., 2018), we used a standard network architecture; the network architecture comprising three sub-networks.

3.3 Training Process

In our method, we divide the training process into three phases: pre-training, selection and percolating. In each phase of training, different parts of network are involved for training.

3.3.1 Pre-training

In the pre-training phase, we use only the main data as input and train the network; see Fig. 4a. As with the standard neural network training, we update the model weight parameters by a stochastic gradient descent method through back-propagation.

3.3.2 Selection

In the selection phase, we fix the weights of the main network and optimize the weights of aux network. Meanwhile, the decision of which aux data to use is jointly optimized. To this end, we introduce real-valued gate parameters $\{\theta_i\}$ which force N-dimensional aux data to be active at training; see Fig. 4b. Therefore, the goal of the selection phase is to learn two learnable parameters by minimizing the training loss $\mathcal{L}(W,\theta)$ which is determined by both the network weights W and the gate parameters θ.

Beside the performance, the correlation between the main and aux data is important objective; otherwise the model can be overfitted to the aux data (i.e. the model might select the aux data to produce the percolative features that are difficult to represent during the percolating phase). To avoid this, we introduce a simple penalty term to capture correlation across main and aux data. Specifically, given main and aux data x_{main}, x_{aux}, our loss function penalizes the difference between the network's percolative features f_{perc} with / without percolation procedure. As such, we have the expected correlations between the main and aux data as:

$$\ell_{\text{corr}} = \frac{1}{|\mathcal{T}|}\sum_{j=1}^{|\mathcal{T}|} \| f_{\text{perc}}(x_{\text{main}_j}, \rho(x_{\text{aux}_j}, p_{\text{perc}})) - f_{\text{perc}}(x_{\text{main}_j}, x_{\text{aux}_j}) \|_2^2$$

where we denote our modified percolation procedure and the percolating probability by $\rho(\cdot)$ and p_{perc}. In the procedure $\rho(\cdot)$, the aux data x_{aux} is stochastically dropped out with a probability p_{perc} that is linearly increased during the selection phase. We empirically found that these penalizes help to improve generalization and avoid overfitting. Thus, the total loss $\mathcal{L}_{\text{sel}}$ used for aux data selection can be written as:

$$\mathcal{L}_{\text{sel}} = \ell_{\text{CE}}(x_{\text{main}}, x_{\text{aux}}) + \ell_{\text{corr}}(x_{\text{main}}, x_{\text{aux}})$$

where ℓ_{CE} is cross entropy between the model predictions and the training labels.

Unlike the network weight parameters, the gate parameters θ cannot be updated by using the standard gradient descent. Therefore, we approximate the gradients, with respect to the gate parameters θ, to directly optimize its corresponding parameters. In this work, we present two implementations of aux data selection, although we think that other optimization techniques could also be employed. The first method, simple gradient based method, approximately estimates the gradients with respect to its gate parameters. The second one, natural gradient based method, formulates the optimization task in a probabilistic manner. After the selection phase is completed, we deterministically select the aux data based on θ (i.e. $\text{argmax}_p p(g\,|\,\theta)$).

Simple Gradient based Method: In this approach, we determine which aux data to use stochastically. As with the optimization in (Courbariaux et al., 2015), we constraint the parameters to either 0 or 1 to determine whether or not to use aux data for percolative learning. To be specific, the gate parameters $\{\theta_i\}$ are transformed to binarized weights $\{g_i\}$ stochastically:

$$g_i = \begin{cases} 1 & \text{with probability } p_i = \sigma(\theta_i), \\ 0 & \text{with probability } 1 - p_i. \end{cases}$$

where we denote σ as the hard sigmoid function:

$$\sigma(x) = \text{clip}(x, 0, 1)$$

Although we think other functions could also be employed, for simplicity we use this simple hard sigmoid function. Since the gradients $\partial\mathcal{L}/\partial\theta_i$ cannot be calculated through backpropagation, we simply update the gate parameters θ using $\partial\mathcal{L}/\partial g_i$ instead of $\partial\mathcal{L}/\partial\theta_i$. To this end, we compute the "*masked aux data*" and use them as inputs to aux network to ensure the binarized weights g are involved in the computational graph:

$$x'_{\text{aux}} = g x_{\text{aux}}$$

The gradients $\partial\mathcal{L}/\partial g_i$ can be computed using the backpropagation, thus we can analogously learn the gate parameters θ.

Natural Gradient based Method: As with the optimization method proposed in (Shirakawa et al., 2018; Saito et al., 2018), we consider the gate parameter g that determines which aux data to use for percolating. The gate parameter g is sampled from the probabilistic distribution $p(g|\theta)$ which is parameterized by a distribution parameter $\theta \in \Theta$. Under the Bernoulli distribution $p(g|\theta) = \prod_{i=1}^{N} \theta_i^{g_i}(1-\theta_i)^{1-g_i}$, we minimize the following loss function:

$$\mathcal{G}(W,\theta) = \int \mathcal{L}(W,g)p(g|\theta)\,dg$$

We optimize both the W and θ by computing the gradient and the natural gradient with respect to W and θ, respectively.

$$\nabla_W \mathcal{G}(W,\theta) = \int \nabla_W \mathcal{L}(W,g)p(g|\theta)\,dg$$

$$\tilde{\nabla}_\theta \mathcal{G}(W,\theta) = \int \mathcal{L}(W,g)\tilde{\nabla}_\theta \ln p(g|\theta)\,dg$$

where $\tilde{\nabla}$ is natural gradient (Amari, 1998) which can be computed by the product of the inverse of Fisher information matrix and the gradient $F(\theta)^{-1}\nabla_\theta$. We follow (Shirakawa et al., 2018) and approximate these gradient by using Monte-Carlo methods with λ samples from $p(g|\theta)$. Specifically, we use the analytical natural gradients of the log-likelihood $\tilde{\nabla}_\theta \ln p(g|\theta) = g - \theta$, thus the total gradients as:

$$\mathcal{G}(W,\theta) = \sum_g \mathcal{L}(W,g)(g-\theta)$$

Here, we employ the Monte-Carlo method to approximate the gradient using λ samples from $p(g|\theta)$. As a result, two parameters W and θ are updated as follows:

$$\nabla_W \mathcal{G}(W,\theta) \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} \nabla_W \overline{\mathcal{L}}(W,g_i)$$

$$\tilde{\nabla}_\theta \mathcal{G}(W,\theta) \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} u_i(g_i-\theta)$$

where u is the ranking-based utility proposed in (Shirakawa et al., 2018); top $[1/4]$ of the samples are $u_i = 1$, bottom $[1/4]$ of the samples are $u_i = -1$, and $u_i = 0$ otherwise.

3.3.3 Percolating

As with the standard percolative learning (Yanagimoto and Nagao, 2017), we update the weights of only aux network in order not to change the percolative features; see Fig. 4c. However, we discovered a modified version of percolative learning works well in our framework. Our modified version is different from the original one in the following ways.

Firstly, we modified the percolation process. In the original percolative learning, the aux data are decayed by element-wise multiplication during the training. In our modified approach, the aux data x_{aux} is stochastically dropped out with a percolating probability p_{perc} that is linearly increased during the training.

Secondly, we use the mean absolute error (MAE) between the output of the main network and the percolative features instead of the mean squared error (MSE). This is because MAE is less sensitive to outliers and might lead to better feature extraction.

Lastly, we introduce a consistency cost between the two percolative features. This allows the model to give consistent predictions around the percolative feature points. Following (Tarvainen and Valpola, 2017), we use mean squared error (MSE) as the consistency cost during the percolating phase. Thus, the total loss function $\mathcal{L}_{\text{perc}}$ can be written as:

$$\mathcal{L}_{\text{perc}} = \frac{1}{|\mathcal{T}|}\sum_{j=1}^{|\mathcal{T}|} |f_{\text{perc}}(x_{\text{main}_j}, \rho(x_{\text{aux}_j}, p_{\text{perc}})) - F_{\text{perc}}(x_{\text{main}_j}, x_{\text{aux}_j})| + \frac{1}{|\mathcal{T}|}\sum_{j=1}^{|\mathcal{T}|} \|f(x_{\text{main}_j}, \mathbf{0}) - F(x_{\text{main}_j}, x_{\text{aux}_j})\|_2^2$$

where we denote the percolative features and the network predictions by $f_{\text{perc}}(\cdot)$ and $f(\cdot)$; $F_{\text{perc}}(\cdot)$ and $F(\cdot)$ are the percolative features and the network predictions obtained in the selection phase.

4 EXPERIMENTS AND RESULTS

4.1 Dataset

In this experiment, we applied out method on several classification datasets, which can be obtained from UCI Machine Learning Repository (Dua and Graff, 2017) and Kaggle[†]. We split each dataset into 80% of training set and 20% testing set. In all dataset, the randomly sampled 20% of inputs are used as main inputs and the remaining inputs are used as aux inputs to validate effectiveness of our method.

Breast Cancer Wisconsin Dataset: Breast cancer Wisconsin Dataset is one of the classic binary classification dataset. It contains 569 data, in which each data has features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.

Heart Disease UCI Dataset: Heart Disease UCI Dataset is also a classic binary classification dataset. It contains 303 data, in which each data contains 14 attributes of heart disease patients.

4.2 Training Settings

In the pre-training, we used SGD with Nesterov momentum (Sutskever et al., 2013) of 0.9, mini-batch size of 32, weight decay of 1.0×10^{-4}, and initial learning rate of 0.1. In the selection and the percolating phase, the Adam optimizer (Kingma and Ba, 2015) with learning rate $\alpha = 0.001$, and momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$. In all phases, the network was trained for 200 epochs, in which the learning rate was reduced by a factor of 10 at $1/2$ and $3/4$ of the total training epochs. We updated the gate parameters using the Adam optimizer with learning rate of 1.0×10^{-4} for the simple gradient based method, and used the sample size $\lambda = 2$, learning rate $\eta = 1/N$, and the initial theta value $\theta_{\text{init}} = 0.5$ for the natural gradient based method. Throughout our experiments, we use a small neural network illustrated in Fig. 2 with a hidden size of 8 units. Then we compare the performance of the baseline neural network with our model to demonstrate the potential of our model. This baseline model with same architecture has the same number weight parameters and expressiveness as our model. For a fair comparison, baseline model was trained for 600 epochs (pre-training: 200 epochs + selection: 200 epochs + percolating: 200 epochs), in which the learning rate was reduced by a factor of 10 at $1/2$ and $3/4$ of the total training epochs.

[†]https://www.kaggle.com/

Table 1: Results on the Breast Cancer Wisconsin Dataset and Heart Disease UCI Dataset.

(a) Breast Cancer Wisconsin Dataset

Model	Classification Performance (%)										
	1	2	3	4	5	6	7	8	9	10	Average
Baseline Model	90.90	93.01	89.51	**95.80**	93.01	93.71	87.41	93.01	**95.80**	95.11	92.73
Simple Gradient Based Method	95.11	**93.71**	93.01	95.11	**96.50**	95.11	**92.31**	**94.41**	**95.80**	**96.50**	**94.76**
Natural Gradient Based Method	**95.80**	**93.71**	**93.71**	**95.80**	**96.50**	**95.80**	91.61	**94.41**	**95.80**	**96.50**	**94.96**

(b) Heart Disease UCI Dataset

Model	Classification Performance (%)										
	1	2	3	4	5	6	7	8	9	10	Average
Baseline Model	**68.42**	71.05	57.90	48.68	71.05	**80.26**	**64.47**	71.05	**64.47**	64.47	66.18
Simple Gradient Based Method	**68.42**	**72.37**	**59.21**	48.68	**72.37**	**80.26**	**64.47**	**82.90**	61.84	**65.79**	**67.63**
Natural Gradient Based Method	67.11	**72.37**	**59.21**	**53.95**	**72.37**	**80.26**	**64.47**	77.63	63.15	**65.79**	**67.63**

4.3 Results

A summary of the accuracy is provided in Table 1. We ran our method ten times with different random seeds (i.e., different input splits) and reported the classification performances. We also reported the mean classification accuracy over all splits. As can be seen from this table, our method outperformed the baseline model in most cases. It is observed overall that our method demonstrated the advantage over the baseline by 2.0% on both datasets.

We additionally conducted Wilcoxon signed rank test to analyze the efficiency of our approach. The p-values for simple gradient based method and the baseline model on the breast cancer Wisconsin dataset and heart disease UCI dataset were 0.0078 and 0.28, respectively, and the ones for natural gradient based method and the baseline model were 0.0078 and 0.055. As the p-values were less than 0.05 in most cases, our percolative approach can improve the performance over the baseline models. It should be also noted that our method can achieve good performance, despite the fact that we use no regularizations such as Dropout (Srivastava et al., 2014). This suggests that it is possible that further performance gain could be achieved by leveraging these techniques in our method.

5 CONCLUSION

In this paper, we propose a new framework for training neural network via a percolate process, in which the model is trained by leveraging the aux data that is only available only training. Our method also automatically selects suitable aux data for percolating. This encourages the models to obtain generalized percolative features during the training. We evaluated our method on several datasets, and demonstrated that our model outperforms the baseline model. However, our method employed simple neural network built upon standard components to achieve this performance. This implies that our method can be made more effective. Another direction of future research is to apply our method to practical datasets such as time-series prediction, thereby extending the capabilities of our method.

ACKNOWLEDGEMENTS

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

REFERENCES

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems 28*, volume 28.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representaions*, pages 2452–2459.

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.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Proc. the*

28th International Conference on Machine Learning (ICML '11), pages 689–696.

Saito, S., Shirakawa, S., and Akimoto, Y. (2018). Embedded feature selection using probabilistic model-based optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (Student workshop at GECCO 2018)*, pages 1922–1925.

Shirakawa, S., Iwata, Y., and Akimoto, Y. (2018). Dynamic optimization of neural network structures using probabilistic modeling. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4074–4082.

Shu, X., Qi, G.-J., Tang, J., and Wang, J. (2015). Weakly-shared deep transfer networks for heterogeneous-domain knowledge propagation. In *Proc. the 23rd ACM International Conference on Multimedia (ACMMM1'15)*, pages 35–44.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 1139–1147.

Takaishi, K., Kobayashi, M., Yanagimoto, M., and Nagao, T. (2018). Percolative learning: Time-series predictions from future tendencies. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, pages 1195–1204.

Yanagimoto, M. and Nagao, T. (2017). Neural networks percolating information available only in training. In *Proc. Mathematical Modeling and Problem Solving*, pages 1–6.