Structural exploration of stochastic neural networks for severely-constrained 3D memristive devices

Paniti Achararit\textsuperscript{1a)}, Itaru Hida\textsuperscript{2}, Takao Marukame\textsuperscript{3}, Tetsuya Asai\textsuperscript{2}, and Yuko Hara-Azumi\textsuperscript{1}

\textsuperscript{1} Tokyo Institute of Technology, Japan
\textsuperscript{2} Chome-12-1 Ookayama, Meguro, Tokyo 152-8550, Japan
\textsuperscript{3} Corporate Research and Development Center, Toshiba Corporation, Japan

\textsuperscript{a)} paniti@cad.ict.e.titech.ac.jp

Received February 27, 2018; Revised June 8, 2018; Published October 1, 2018

Abstract: Neural networks architectures will be soon required to be deployed on IoT-edge systems, which will demand small and low-power circuits and application-dependent customizability. Focusing on the power-efficiency and internal structures of 3D memristive devices, this paper proposes neural networks architectures that are implementable in those devices, together with an existing transfer learning technique. Our architectural model is stochastic and device-conscious in handling noise/variation-vulnerability and suppressing inter-layer connections. Through experiments, we quantitatively explored appropriate structures under device constraints and demonstrated comparable performance as conventional work holding complex connections. Also, we revealed an important challenge on device technologies for further performance improvement.

Key Words: stochastic neural networks, locally-connected convolutional deep belief network, 3D memristive devices

1. Introduction

Technologies for neural networks, including deep learning, are rapidly developed and expected to resolve problems even human experts cannot handle well. Since those technologies are currently feasible only on powerful computing architectures with rich resources and storage, their use is limited in servers and data centers. However, these technologies will be also required to be deployed in Internet of Things (IoT)-edge systems in near future – while the IoT-edge systems are currently used as sensing devices, etc., the technologies for neural networks will let them have “intelligence” and turn so-called edge-AI systems [1], enabling to process a variety of novel applications in real time, such as
health-care and disaster prevention/mitigation.

Since most IoT-edge systems are battery-constrained, combination of both architectural and device ingenuity will be a key breakthrough to realize practically power/energy-efficient edge-AI systems. Among a number of studies on devices, memristive devices have been intensively studied to alternate the current silicon devices [2–4]. Interestingly, memristive devices work as both of an arithmetic unit and memory by exploiting their resistive wires that express different values by different resistances [5]. Moreover, 3D memristive devices are further efficient in integration (i.e., circuit footprint) than 2D counterparts. Therefore, this “3D structure” should be leveraged when considering novel architectures. For example, focusing on the fact that most material objects, including human brains, have the 3D structure by nature, neural networks can be efficiently realized by exploiting the internal structures of such 3D memristive devices.

Motivated by the backgrounds above, in this paper, we propose a neural networks model that is conscious on 3D memristive devices, for the sake of realizing small, power-efficient edge-AI systems. This model, named Locally-Connected Convolutional Deep Belief Network (LCCDBN), is developed based on Convolutional Neural Network (CNN) and Deep Belief Network (DBN) with some extensions. Because 3D memristive devices have several severe constraints in fabrication and implementation, we explore a variety of deployable structures in those devices while achieving sufficient performance. The important features and contributions of our LCCDBN are as follows:

• For realizing neural networks in 3D memristive devices, inter-layer connections in convolutions are realized by conductive wires between electrodes. In order to avoid defects in the devices, the complexity of these wires should be mitigated. More specifically, overlapping wires are restricted. In our model, this problem is resolved by limiting convolutions only to neighboring data (i.e., local data).

• From the implementation perspective, layer-wise unsupervised training is preferable over wire-by-wire supervised training. While fine-tuning (or supervised training) requires an additional device technique, such as a controller in each layer to manage the weights by utilizing off-chip memories, unsupervised training to pre-set the weights layer by layer does not require such techniques and thus is implementation-friendly. We not only consider a variety of structures under this constraint (i.e., using unsupervised training) but also evaluate the impacts of relaxing this constraint (i.e., allowing supervised training).

• Last but not least, our model holds stochastic nature, which stems from DBN, in order to implicitly consider the vulnerability of 3D memristive devices to noise/variation, etc.

We assume that our LCCDBN will be implemented in 3D memristive devices as a feature extractor, whose outputs (i.e., features) are fed to a classifier that will be developed by a reconfigurable hardware or software, for providing the application-dependent customizability. Thus, an existing transfer learning technique should be reasonable to be utilized for the feature extractor in order to efficiently construct the whole edge-AI systems.

Through our experiments, we quantitatively explored feasible and appropriate LCCDBN structures, varying a variety of parameters. Even with the severe constraint of using layer-wise unsupervised training, we demonstrated that our LCCDBN provides comparable performance as conventional DBN, in spite of significant reduction in inter-layer connections (i.e., conductive wires). Also, we revealed that relaxing this constraint can largely enhance the performance, at the sacrifice of implementation costs. Although difficulties in devices/implementations will be incurred, this relaxation is worth being tackled.

The rest of the paper is organized as follows. First, Section 2 briefly explains fundamental technologies, i.e., 3D memristive devices and transfer learning. Then, Section 3 presents our proposed stochastic neural networks. Section 4 provides experimental setup and results of structural exploration. Finally, Section 5 concludes this paper.
2. Preliminaries
This section briefly explains 3D memristive devices and transfer learning.

2.1 3D memristive devices
As an alternative to CMOS devices, a variety of new types of memristive devices have been studied in both academia and industry, such as Phase-Change Memory (PCM) [6], Magnetic Random-Access Memory (MRAM) [7], and Resistive Random-Access Memory (ReRAM) [8]. These memories are non-volatile and low-power as well as conventional flash memories, yet as fast to access data as conventional DRAM. More interestingly, they can work not only as fast random-access memories but also as computing units [5] by utilizing their analog behaviors.

In these memristive devices, data are stored exploiting conductive wires between two electrodes. Figure 1 displays a fabrication mechanism of controlling the conductive wires. A conductive wire is controlled under sufficiently high voltage. Its conductance is controlled by the voltage to express a different data or value for arithmetic operations (particularly, weights for multiplication operations) between two electrodes, since the conductance is determined by $v = R(w)i$, where $v$ is voltage, $i$ is current, and $R$ is a generalized resistance depending on the state variable $w$, which has a relationship between time $t$ and electric charge $q$ (i.e., $\frac{dw}{dt} \propto \frac{dq}{dt}$) [9]. More specifically, resistance (i.e., the inverse of conductance) will be varied by the amount of change of the current (or the voltage) in short duration [10–12]. These characteristics represent the basis of memristive devices. In terms of how these conductive wires are controlled, memristive devices can be roughly classified into two: step-by-step control and one-time control. In the former devices such as PCM [13], the weights are updated in a step-by-step manner by voltage pulses during iterative increment and decrement of the weights in the training process. On the other hand, in the latter such as ReRAM, the weights are predetermined and applied to the wires by voltage pulse at one time. For both types of memristive devices, the weight at the initial state (i.e., with no wires) is regarded as 0, while the weight after programmed to a certain value expresses a real number in the range of 0 to 1. Note that in this paper, we do not assume specific memristive devices.

Currently in existing memristive devices, a conductive wire can be usually realized between two electrodes in a one-to-one manner as shown in Fig. 2. Thus, those devices need to be implemented in a grid structure with a large number of crossbar junctions to perform convolution of neural networks [5] – actually most crossbar junctions are not used in CNN due to zero weights. These wastes can be resolved if conductive wires are realized to bridge between multiple electrodes as shown in Fig. 2(a), which are being studied in a variety of 3D memristive devices, such as [14]. By exploiting this technology and appropriately controlling the conductive wires (Fig. 2(a)), the overall memristive device can efficiently perform convolution (Fig. 2(b)) – the conductance of each wire represents a weight ($w_i$), and the product of the weight and input signals ($v_j$) is calculated when the signal is passed from the upper to lower electrodes, e.g., $h_2 = w_2 \times v_1 + w_3 \times v_2$.

Wires reduction between layers is essential not only from the perspective of better integration but also from that of a device fabrication issue. The former should be obvious (i.e., the smaller footprint...
Transfer learning is a method to focus on a network that has been already trained to extract input features and reuse it for different but related problems [15]. This network works as a feature extractor, whose outputs are fed to a classifier that can be customized depending on the problem (or target application). For example, in Fig. 3, the feature extractor is first built using “Dataset A.” Then, the feature extractor is used to classify “Dataset B”, which is a different but related dataset than “Dataset A.” Also, for Neural Network B, the classifier is rebuilt accordingly.

Since customizability of edge-AI systems is mandatory for practical use, it is realistic to employ transfer learning. In this work, we also assume that a feature extractor is implemented in a hard-wired 3D memristive device while the user-defined classifier is realized to be customizable by software or reconfigurable hardware, as noted in Fig. 3. The weight (conductance) of wires in the feature extractor can be pre-set by training, i.e., hard-wired.

Although there exist a variety of practically-used feature extractors for transfer learning, such as AlexNet [16] and VGG16 [17], they are too complex and heavy to be realized in 3D memristive devices. More specifically, too many connections (e.g., wires in ReRAMs) between layers will overlap, causing short or serious defects if implemented in the devices. In the next section, we will present a kind of stochastic neural network structures that are simple enough to be implemented in 3D memristive devices and resolve the aforementioned wiring issue.
3. Locally-connected convolutional deep belief network

The goal of our work is to realize neural networks in low-power 3D memristive devices for embedded systems, so-called edge-AI systems. This paper presents a first attempt to explore the structure of such neural networks considering severe constrains in these devices, and proposes a stochastic neural network called `locally-connected convolutional deep belief network (LCCDBN)` by integrating the advantages of Convolutional Neural Network (CNN) and Deep Belief Network (DBN).

3D memristive devices have several severe constraints in fabrication. Out of them, wire complexity and conductance control are particularly severe ones. First, we focus on CNN (a neural network widely used in a variety of applications in AI) in order to learn its weights setting between layers. In CNN, each layer uses a single or multiple small-sized weighted matrix/matrices (shared weights as shown in Fig. 4(a)) applied to the input for creating a particular feature map (shared weights). However, shared weights are not feasible in 3D memristive devices, particularly for networks composed of fully-connected layers, as they result in overlapping wires as depicted in Fig. 4(a). Hence, we extend CNN to be able to create multiple feature maps while considering the fabrication limitation by allowing the local connected unshared weights (Fig. 5(b)). For example, for an input image of $28 \times 28$ pixels, while shared weights creates one feature map in one convolutional layer, our extension creates 28 feature maps when a 3x3 weighted matrix is employed with one zero-padding. By using unshared weights, fabrication friendliness and multiple feature maps are both realized.

Next, we focus on another stochastic neural network, DBN, due to its training method. Training methods are roughly classified into two approaches: supervised and unsupervised. For 3D memristive devices, employing the supervised training tunes the weights wire by wire. This will lead to intervention from an off-chip controller so that weights are stored apart from the arithmetic units (e.g., in an off-chip memory) and fed back, as well as have been done in conventional von Neumann architectures. This may bring not only better classifications but also fabrication difficulties and computing performance degradation. Contrary, employing the unsupervised training tunes the weights layer by layer, meaning that the weights are adjusted by the local connections without off-chip intervention. This is equivalent to a so-called “processing in-memory” in architectures nowadays [5] and is obviously more fabrication-friendly and efficient from computing and energy perspectives. Therefore, layer-wise training is more preferable considering the topology of 3D memristive devices. Actually, [15, 18, 19] have demonstrated that greedy layer-wise training successfully works on DBN to gain sufficient performance.

Another reason of focusing on DBN is its stochastic nature. Referring to recent research trends on (2D) memristive devices [5], our target 3D memristive devices will be also likely to be used as analogue circuits, which are vulnerable to dynamic disturbance like environmental noise, device variation, etc. Considering these effects, neural networks for 3D memristive devices must be modelled stochastically rather than deterministically. By partially extending DBN for our work, we will implicitly deal with the stochastic nature of DBN, which is composed of layers of stochastic variables.

To sum up, our LCCDBN is a stochastic neural network model with local connections only, developed from CNN (unshared weights) and DBN (greedy layer-wise training and stochastic nature). This LCCDBN will be built in 3D memristive devices in order to construct a feature extractor, followed by a fully-connected classifier (realized as a reconfigurable hardware or software). Figure 6 describes...
an example of a 3-layer feature extractor and a 1-layer classifier. In this paper, we assume that the feature extractor is composed of relatively few layers (i.e., one, two, or three layers) with a small (i.e., 3x3) filter considering the device immaturity in fabrication. However, these limitations may be relaxed in future along with the advance of device fabrication technologies. It should be also noted that such simple settings are important to understand the impact of exploring and varying network structures.

4. Experiments
This section evaluates our proposed LCCDBN varying its structural parameters.

4.1 Setup
Here, we describe the evaluation setup on datasets, comparison neural networks structures, and metrics.

4.1.1 Datasets
We utilized two well-known databases, MNIST and CIFAR-10, in our evaluation.

- MNIST [20] (Fig. 7) is a handwritten digits database which contains 10 classes of digits from 0 to 9. 50,000 training images, 10,000 validation images, and 10,000 testing images are included. Each image is a 28×28 pixel grayscale image. MNIST is often used as the first step of evaluating a neural network for the image classification task because of its simplicity.

- CIFAR-10 [21] (Fig. 8) contains a picture database of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). 50,000 training images and 10,000 testing images are included. Although validation images are not provided in the dataset, they can be easily created by splitting some images from training images. Each image is a 32×32 pixel color image, but converted in grayscale in our evaluation for simplicity without loss of generality. This dataset is also frequently used for image classification but more realistic than MNIST.
4.1.2 Comparison methods

In our experiments, four neural network structures, including three different structures of LCCDBN, are evaluated. Since the concept of LCCDBN was partially derived from DBN [18], we used the Sigmoid function and the normal distribution as the activation function and weight initialization, respectively. The summaries of each method are shown below and in Table I.

1. **DBN**: DBN is composed of a fully-connected network of one to three hidden layers. No classifier needs to be attached since the fully-connected DBN itself can work as both of a feature extractor and a classifier. We followed the settings employed in [18], including neurons counts (500, 500, and 2,000 in the first, second, and third layers, respectively) as shown in the table.

2. **LCCDBN$_{2,000}$**: This LCCDBN was constructed under the severe device constraints in that only greedy layer-wise training (unsupervised training) is feasible. LCCDBN needs a classifier afterwards. We assume that the classifier is composed of one layer because the classifier is also required to be as simple (or small) as possible in embedded systems or edge-AI systems. 2,000 indicates the neurons counts in the classifier, representing a resource-rich classifier.

3. **LCCDBN$_{100}$**: While the feature extractor is the same as LCCDBN$_{2,000}$, the classifier has only 100 neurons, representing a resource-poor classifier such as a case with a limited memory capacity, etc. This structure was built to evaluate the performance of LCCDBN more clearly.

4. **LCCDBN$_{\text{supervised}}$**: This LCCDBN utilized back propagation (i.e., supervised training), assuming that the device constraint is relaxed enough to overcome the fabrication difficulty. Along with such relaxation, ReLU [22] and the method of He [23] were employed as the activation function and initialization, respectively, to enhance the effect of back propagation.

Since we are interested in evaluating how well the feature extractor works, for each of these four structures, performance of transfer learning was quantitatively evaluated as follows:

- **Only classifier**: No feature extractor is used, and thus only the classifier is trained via back-propagation for the target dataset with 50 epochs. This can be regarded as a baseline to evaluate the effects of feature extractors.

- **With feature extractor (FE)**: Feature extractors were built using the aforementioned four structures in two steps; First, for each structure, a feature extractor is built and trained using the original 10 classes of the target dataset with 50 epochs; Then, while the feature extractor is reused for test, only the classifier was trained by back-propagation using only five classes out of 10 – even numbers were selected from MNIST, and airplane, bird, deer, frog, and ship were selected from CIFAR-10.

4.1.3 Metrics

In our evaluation, three metrics were utilized to compare the aforementioned four structures, varying the hidden layers counts from one to three.

- **Wire complexity**: We compare the wire complexity of DBN [18] and our LCCDBN, which are composed of fully-connected and locally-connected hidden layers, respectively.

---

---

2 For DBN, no hidden layers are inserted, and the input and output layers are fully-connected.
Table I. Structures of neural networks for comparison.

|                             | DBN | LCCDBN _2,000 | LCCDBN _100 | LCCDBN _supervised |
|-----------------------------|-----|---------------|-------------|--------------------|
| Hidden layers #             | 1, 2, 3 |               |             |                    |
| Hidden neurons #            | 1st:500, 2nd:500, 3rd:2,000 | Same as input pixels |           |                    |
| Classifier layers #        | -   | 1 Layer       |             |                    |
| Classifier neurons #       | 2,000 | 100           | 2,000       |                    |
| Receptive field size       | -   | 3 × 3         |             |                    |
| Zero-padding size          | -   | 1             |             |                    |
| Stride                     | -   |               |             |                    |
| Learning method            | Greedy layer-wise training | Back prop. |             |                    |
| Activation function        | Sigmoid | ReLU          |             |                    |
| Weight init.               | Normal distribution | He init.   |             |                    |

Table II. Comparison on wire complexity.

| Dataset       | DBN                              | LCCDBN                          |
|---------------|----------------------------------|---------------------------------|
| MNIST         | 28×28×500 = 392,000 1×(3×3×28×28) = 7,056 | 392,000 + (500×500) = 642,000 2×(3×3×28×28) = 14,112 |
| CIFAR-10      | 32×32×500 = 512,000 1×(3×3×32×32) = 9,216 | 512,000 + (500×500) = 762,000 2×(3×3×32×32) = 18,432 |
|               | 762,000 + (500×2,000) = 1,762,000 3×(3×3×28×28) = 27,648 |

- **Image reconstructability**: How well DBN and LCCDBN can reconstruct the input images, varying the hidden layers counts, are visually compared.

- **Performance of transfer learning**: For each of four structures, performance of transfer learning is quantitatively evaluated for MNIST and CIFAR-10 datasets.

### 4.2 Results

We first evaluate the effect of reduction in wire complexity by comparing the wire counts in DBN and LCCDBN. Note that wire complexity is the same for three different LCCDBN structures. The upper and lower parts of Table II tabulate the wire counts for MNIST and CIFAR-10, respectively, varying the hidden layers counts. In DBN, the wires counts in the first hidden layer is determined by the product of the input image size and the neuron counts (i.e., 500). As the layers increase, additional wires, determined by the product of neurons counts in the precedent and successive layers, are added. Whilst, in LCCDBN, each layer has the same wires counts, determined by the product of the receptive field size (i.e., 3×3) and the input image size. From these results, it is clear that the wires reduction by LCCDBN is considerable. This will accelerate to realize the neural networks in 3D memristive devices.

Next, in order to confirm that LCCDBN sufficiently works as a feature extractor, we perform the evaluation on the simple dataset, MNIST, in terms of the image reconstructability and quantitative performance. For some selected images among MNIST, Fig. 9 compares original images and reconstructed images through feature extractors only. Note that both DBN and LCCDBN were trained by greedy-layer wise training (unsupervised training). Although some noises were added to the outputs of LCCDBN, we can find that the image reconstruction is sufficiently done in spite of the significant reduction in the wire complexity (Table II).
Similarly, Table III compares the performance of feature extractor on the image classification task. Here again we can find that LCCDBN achieves comparable performance as DBN even under the severe device constraint (by comparing LCCDBN\_2,000 and DBN as they have the same neurons counts in the last layer). However, DBN and the unsupervised LCCDBNs (i.e., LCCDBN\_2,000 and LCCDBN\_100) have performance degradation than the classifier only. This would be because the image classification task on MNIST is simple enough only for the classifier. This presumption will be further evaluated on a more complex dataset (CIFAR-10) later.

We also conducted the evaluations on CIFAR-10 dataset as well as have been done on the MNIST dataset. Since the images become more complex than those in MNIST, as described in Fig. 10, the reconstructed images of both DBN and LCCDBN slightly blur. This may reveal the limitation of unsupervised training, especially in LCCDBN where significant wire reduction is done.

For all the structures, Table IV describes the performance results of feature extractors against the classifier only. Although one might have thought that image reconstruction (Fig. 10) from LCCDBN is poorer than those from DBN, better performance is achieved from LCCDBN as comparing DBN and LCCDBN\_2,000. This indicates that LCCDBN can efficiently extract features required for classification even on complex images. Although the performance of using the LCCDBN-based feature extractor is still less than that of using a resource-rich classifier (i.e., 2,000 neurons), these results are reversed as the classifier becomes resource-poor (i.e., 100 neurons). Also, recalling the performance results on the MNIST dataset, we can conclude that the LCCDBN-based feature extractor is of help for the classifier in realistic situations that classification tasks are too complex only for the classifier. Finally, we evaluated the supervised LCCDBN assuming that the implementation of off-chip intervention mechanisms may become feasible as the device fabrication technologies get mature. As can be seen from the table, LCCDBN-based feature extractors considerably improve the performance by relaxing this constraint.

To sum up, we obtained following three essential findings through the above structural explorations:

- In spite of significant reduction in wire complexity, our LCCDBN provides comparable performance and image reconstruction as DBN. In order to further demonstrate our feature extractors (i.e., LCCDBN), here we show the performance of only feature extractors, excluding the classifier, by t-Distributed Stochastic Neighbor Embedding (t-SNE) [24] in Fig. 11 and Table V. Only the results of 1 hidden layer feature extractors on the MNIST dataset are shown, as we observed
Table IV. Performance comparison for the CIFAR-10 dataset.

|                      | DBN | LCCDBN | LCCDBN | LCCDBN |
|----------------------|-----|--------|--------|--------|
|                      | 2,000 | 100 | supervised |
| Only Classifier      | 41.08% | 49.78% | 41.14% | 57.68% |
| With 1-layer FE      | 40.96% | 47.74% | 41.56% | 65.06% |
| With 2-layer FE      | 43.82% | 47.70% | 41.78% | 64.56% |
| With 3-layer FE      | 40.94% | 47.54% | 41.92% | 62.96% |

Fig. 10. Comparison in image reconstruction for the CIFAR-10 dataset.

similar results for the other structures. Figure 11 describes how well the features are extracted and distinguished. The denser each class, the better the features are extracted. While features partially overlap in the original data as no feature extraction is done (Fig. 11(a)), features are better extracted through feature extractors (Fig. 11(b)–(d)), showing the usefulness of DBN and LCCDBN. Especially focusing on the results in Fig. 11(b) and Fig. 11(d), we can observe that the features are well extracted by LCCDBN, in spite of having much fewer inter-layer wires than DBN. Additionally, as a quantitative metric of feature extraction, we introduced angle, which is calculated by using the summation of all vectors of each class from the center of the clusters in the t-SNE coordinate (Table V). As seen from the table, each angle is almost equally apart, meaning that all classes are well distinguished. These results, in addition to the evaluations above, well-demonstrate the effectiveness of our LCCDBN as a feature extractor.

- Generally, memristive devices have large variability. Among the two types of memristive devices, step-by-step conductance-control type takes into account the variability during the conductance-control process, while one-time conductance-control is affected by the variability at both of read and write of the weights – the former thus has less variability but more time-consuming in conductance control than the latter. Considering the conductance-control time, variability-aware training on the latter type of memristive devices is preferable. Thus, we prefer unsupervised training as it is aware of variability in that it does not rely on the predetermined weights. Through the evaluations above, even under severe constraints of using unsupervised training on limited inter-layer wires, we have observed that our LCCDBN successfully extracts features of input images. The more complex the classification tasks will become, the more LCCDBN-based feature extractors will exercise their usefulness.

- However, we have also observed that enabling supervised training will largely improve the performance and emphasize the usefulness of feature extractors. This can be realized by several different approaches: deploying a controller in each hidden layer to manage (fine-tune) the weights, utilizing step-by-step conductance-control type of memristive devices, or realizing the intentional refresh technique\(^3\) [25] to suppress the variability on one-time conductance-control

\(^3\)By refreshing the device when the variations reach a certain value, device variation can be controlled to some extent. With this technique, the possibility of implementations increases.
Fig. 11. Feature visualization of 2,500 data (500 data × 5 classes) of the MNIST dataset: (a) Original data (with no feature extractor), (b) Features extracted from DBN, (c) Features extracted by LCCDBN with unsupervised learning (i.e., LCCDBN_{2,000} and LCCDBN_{100}), and (d) Features extracted by LCCDBN with supervised learning.

Table V. Angles between two neighboring classes on the MNIST dataset.

| Angle between two classes         | Original | DBN | LCCDBN_{2,000} | LCCDBN_{100} | LCCDBN_{supervised} |
|-----------------------------------|----------|-----|----------------|--------------|--------------------|
| Between Class 0 and Class 2       | 80.0     | 80.6| 78.7           | 81.3         |                    |
| Between Class 2 and Class 8       | 64.6     | 59.9| 36.4           | 66.2         |                    |
| Between Class 8 and Class 4       | 74.4     | 73.1| 92.3           | 65.7         |                    |
| Between Class 4 and Class 6       | 68.1     | 75.2| 67.8           | 75.7         |                    |
| Between Class 6 and Class 0       | 72.8     | 71.2| 84.8           | 71.1         |                    |

type of memristive devices. Although this will obviously impose some difficulties in implementations, we conclude that it is worth being tackled.

5. Conclusions

This paper presented a model of stochastic neural networks, called *locally-connected convolutional deep belief network (LCCDBN)*, for the purpose of realizing low-power edge-AI systems using 3D memristive devices. Considering severe fabrication/implementation constraints of memristive devices, our LCCDBN greatly restricts connections (e.g., wires in ReRAMs) in the networks so that it can be implemented as a feature extractor for transfer learning. Through experiments, we explored the LCCDBN structures varying some constitutional parameters and demonstrated that the LCCDBN is comparable to conventional DBN in spite of significant wires reduction. We found that LCCDBN
can be useful as a feature extractor for complex classification tasks, although more improvements are necessary. Furthermore, by relaxing a device constraint to adopt supervised training, the performance can develop rapidly. Through quantitative explorations with holding/relaxing this device constraint, we finally concluded that relaxing this constraint should be worth being tackled in future device technologies. Overall, our work has revealed not only usefulness and limitations of device-oriented LCCDBN structures but also an essential issue to be resolved to realize practical 3D memristive devices.

The subject of our future work is to further explore the LCCDBN structures for performance improvement and quantitatively study their limitations.

Acknowledgments
This work was partially supported by JSPS KAKENHI 17H04677.

References
[1] G. Brotman, Z. Wang, X. Zhang, and L. Zhong, “Panel: Cloud-to-edge ai - applications and user experiences driving the next generation of smartphones and edge devices,” 4G/5G Summit, October 2017.
[2] J.J. Yang, M.D. Pickett, X. Li, D.A. Ohlberg, D.R. Stewart, and R.S. Williams, “Memristive switching mechanism for metal/oxide/metal nanodevices,” Nature nanotechnology, vol. 3, no. 7, p. 429, 2008.
[3] J.J. Yang, D.B. Strukov, and D.R. Stewart, “Memristive devices for computing,” Nature nanotechnology, vol. 8, no. 1, p. 13, 2013.
[4] L. Xia, B. Li, T. Tang, P. Gu, P.Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, “Musim: Simulation platform for memristor-based neuromorphic computing system,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017.
[5] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” Proceedings of the 43rd International Symposium on Computer Architecture, ISCA ’16, IEEE Press, pp. 27–39, 2016.
[6] H.S.P. Wong, S. Raoux, S. Kim, J. Liang, J.P. Reifenberg, B. Rajendran, M. Asheghi, and K.E. Goodson, “Phase change memory,” Proceedings of the IEEE, vol. 98, no. 12, pp. 2201–2227, 2010.
[7] Y. Huai, “Spin-transfer torque mram (stt-mram): Challenges and prospects,” AAPPS bulletin, vol. 18, no. 6, pp. 33–40, 2008.
[8] H.S.P. Wong, H.Y. Lee, S. Yu, Y.S. Chen, Y. Wu, P.S. Chen, B. Lee, F.T. Chen, and M.J. Tsai, “Metal–oxide rram,” Proceedings of the IEEE, vol. 100, no. 6, pp. 1951–1970, 2012.
[9] D.B. Strukov, G.S. Snider, D.R. Stewart, and R.S. Williams, “The missing memristor found,” nature, vol. 453, no. 7191, p. 80, 2008.
[10] G.S. Snider, “Spike-timing-dependent learning in memristive nanodevices,” 2008 IEEE International Symposium on Nanoscale Architectures, pp. 85–92, June 2008.
[11] F. Alibart, L. Gao, B.D. Hoskins, and D.B. Strukov, “High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm,” Nanotechnology, vol. 23, no. 7, p. 075201, 2012.
[12] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K.K. Likharev, and D.B. Strukov, “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” Nature, vol. 521, no. 7550, p. 61, 2015.
[13] A. Fumarola, S. Sidler, K. Moon, J. Jang, R.M. Shelby, P. Narayanan, Y. Leblebici, H. Hwang, and G.W. Burr, “Bidirectional non-filamentary rram as an analog neuromorphic synapse, part ii: Impact of al/mo/pr 0.7 ca 0.3 mno 3 device characteristics on neural network training accuracy,” IEEE Journal of the Electron Devices Society, vol. 6, no. 1, pp. 169–178, 2018.
[14] G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. Spinelli, and D. Ielmini, “Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity,” *Scientific reports*, vol. 7, no. 1, p. 5288, 2017.

[15] S.J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[18] G.E. Hinton, S. Osindero, and Y.W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, pp. 153–160, 2007.

[20] Y. LeCun, “The mnist database of handwritten digits,” http://yann.lecun.com/exdb/mnist/, 1998.

[21] A. Krizhevsky, V. Nair, and G. Hinton, “The cifar-10 dataset,” online: http://www.cs.toronto.edu/kriz/cifar.html, 2014.

[22] V. Nair and G.E. Hinton, “Rectified linear units improve restricted boltzmann machines,” *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[24] L.V.D. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, pp. 2579–2605, November 2008.

[25] K. Ohmori, A. Shinoda, K. Kawai, Z. Wei, T. Mikawa, and R. Hasumuma, “Reduction of cycle-to-cycle variability in reram by filamentary refresh,” *VLSI Technology, 2017 Symposium on, IEEE*, pp. T90–T91, 2017.