dsODENet: Neural ODE and Depthwise Separable Convolution for Domain Adaptation on FPGAs

Hiroki Kawakami, Hirohisa Watanabe, Keisuke Sugiura, and Hiroki Matsutani
Dept. of ICS, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan 223-8522
Email: {kawakami, watanabe, sugiura, matutani}@arc.ics.keio.ac.jp

Abstract—**High-performance deep neural network (DNN)-based systems are in high demand in edge environments. Due to its high computational complexity, it is challenging to deploy DNNs on edge devices with strict limitations on computational resources. In this paper, we derive a compact while highly-accurate DNN model, termed dsODENet, by combining recently-proposed parameter reduction techniques: Neural ODE (Ordinary Differential Equation) and DSC (Depthwise Separable Convolution). Neural ODE exploits a similarity between ResNet and ODE, and shares most of weight parameters among multiple layers, which greatly reduces the memory consumption. We apply dsODENet to a domain adaptation as a practical use case with image classification datasets. We also propose a resource-efficient FPGA-based design for dsODENet, where all the parameters and feature maps except for pre- and post-processing layers can be mapped onto on-chip memories. It is implemented on Xilinx ZCU104 board and evaluated in terms of domain adaptation accuracy, training speed, FPGA resource utilization, and speedup rate compared to a software counterpart. The results demonstrate that dsODENet achieves comparable or slightly better domain adaptation accuracy compared to our baseline Neural ODE implementation, while the total parameter size without pre- and post-processing layers is reduced by 54.2% to 79.8%. Our FPGA implementation accelerates the inference speed by 27.9 times.**

I. Introduction

To improve the accuracy of CNNs (Convolutional Neural Networks) in image recognition tasks, a typical approach is to build deeper models by stacking more convolutional layers [1]. Although such image recognition tasks are in high demand in edge environments, computation resources are strictly limited in edge devices, making it difficult to use high-performance CNN models. To reduce the amount of parameters and mitigate this issue, light-weight neural network models have been developed [2]–[4]. Their key idea is to employ DSC (Depthwise Separable Convolution) that decomposes a conventional convolutional layer into two smaller convolutional steps.

ResNet [1] is one of conventional CNN models that stacks a lot of layers for a higher accuracy. To reduce the parameter of ResNet, by utilizing a similarity to ODE (Ordinary Differential Equation), Neural ODE [5] repeatedly uses weight parameters instead of having a lot of different parameters. Thus, Neural ODE becomes significantly small compared to ResNet, and can be implemented in resource-limited edge devices. Recently its implementation on a low-end FPGA (Field-Programmable Gate Array) device has been reported in [6]. However, its performance improvement is limited since only one or two building blocks are implemented on the programmable logic, and it does not employ any other parameter reduction techniques. For example, FPGA-based neural network accelerators and their optimization techniques, such as binarization and quantization, are surveyed in [7]. FPGA-optimized multipliers for DNNs that minimize information loss from quantization are studied in [8]. DSC is applied to an FPGA-based CNN accelerator in [9].

In this paper, a combination of Neural ODE and DSC, called dsODENet, is proposed and implemented for FPGAs to fully utilize on-chip memory resources. As a practical use case, dsODENet is applied to domain adaptation, which is useful in a common edge AI deployment scenario. When a trained model at server side is deployed to edge devices, the distribution difference between training data and inference data acquired at edge devices often causes a performance degradation, which can be dealt with domain adaptation techniques. Note that our approach is basically orthogonal to quantization techniques and can be combined with them. dsODENet is implemented on Xilinx ZCU104 board and evaluated in terms of the domain adaptation accuracy using image classification datasets, training speed, FPGA resource utilization, and speedup rate compared to a software execution.

Section II introduces baseline technologies behind our proposal. Section III introduces our domain adaptation method and Section IV proposes dsODENet for FPGA-based domain adaptation. Section V shows evaluation results and Section VI concludes this paper.

II. Related Work

A. Depthwise Separable Convolution

CNNs typically stack a set of convolutional layers for a higher image recognition accuracy, and each convolutional layer contains a lot of parameters. Let N, M, and K be the number of input channels, the number of output channels, and the kernel size, respectively. The weight parameter size of a conventional convolutional layer is NMK^2.

DSC decomposes the operation in a convolutional layer into two simpler convolution steps: depthwise convolution step and pointwise convolution step. In depthwise convolution step, a convolution operation involving only spatial direction (the size is K^2) is applied for each of an input feature map. Different weight parameters are used for each of N input channels; thus its weight parameter size is NK^2. Then, an output feature map of the depthwise convolution step is fed to the pointwise convolution step as an input. A 1×1 convolution operation is applied for each of an input feature map and for each of an output channel, thus its weight parameter size is NM. The weight parameter size of DSC is $NK^2 + NM$ in total, which is approximately K^2 times reduction, assuming that $N, M \gg K$.

B. Neural ODE

ResNet is a well-known neural network architecture that can increase the number of stacked layers or building blocks by introducing shortcut connections. Using a shortcut connection, an input feature map to a building block is temporarily saved and then it is added to the original output of the building block to generate the final output of the block.

ODE is composed of an unknown function and its ordinary derivatives. To obtain an approximate numerical solution, an ODE solver such as first-order Euler method and higher-order Runge-Kutta method can be used. Based on a similarity found in shortcut connection and an ODE solver, one building block can be interpreted as one step in the ODE solver [5]. Assuming that Euler method is used as an ODE solver, it can be interpreted that a first-order approximation is applied to solve the output of the building block. In this paper, one building block is called ODEBlock and the whole network architecture consisting of ODEBlocks is called ODENet.

Each ODEBlock is repeatedly executed C times in ODENet, while in ResNet, C different building blocks are executed once. Let $O(L)$ be the parameter size of one building block or ODEBlock. Total parameter sizes of convolutional layers in ResNet and ODENet are $O(CL)$ and $O(L)$, respectively; thus ODENet can significantly reduce the parameter size.

C. Edge Domain Adaptation

Domain adaptation is a kind of transfer learning, where knowledge obtained at a source domain is transferred to a different domain (target domain). It is typically assumed that the source domain has enough labeled training data while the target domain does not. As explained in Section I, it is useful in a common edge AI deployment scenario. MobileDA [10] is a domain adaptation technique for edge devices based on knowledge distillation and DeepCORAL [11]. DeepCORAL is a method to reduce the distance between domains. Since the target domain is an edge environment in the edge domain adaptation scenario, the target model should be further reduced in parameter size and computation cost. Although pruning, quantization, and distillation are very common model compression techniques, in this paper we propose a combination of ODENet and DSC to further reduce the number of parameters of the target domain model.

III. Domain Adaptation Method

In this paper, a modified version of MobileDA is used as an edge domain adaptation procedure to gain a higher performance. While MobileDA uses one teacher model and one student model as knowledge distillation, our approach uses one teacher model and two student models. ResNet is used as the teacher model, while the combination of ODENet and DSC, called dsODENet, is used in the two student models. In Section V, our approach will be compared to the original MobileDA.

The training phase consists of three steps. First, a teacher model is trained with source domain data in Step 1, and then two student models are trained in Steps 2 and 3. Finally, the student model makes inferences. Steps 2 and 3 of the learning process are shown in Figure 1.

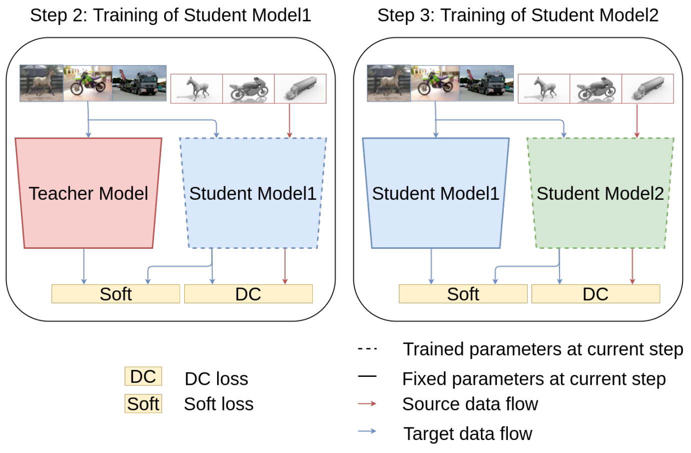
Step 2: Training of Student Model1
Step 3: Training of Student Model2
Teacher Model
Student Model1
Student Model1
Student Model2
Soft
DC
Soft
DC
DC loss
Soft loss
Trained parameters at current step
Fixed parameters at current step
Source data flow
Target data flow

Fig. 1. Training steps of our approach.

In Step 2, the teacher model is fixed, and parameters of student model1 are trained by the help of the teacher model. In Step 3, the student model1 is fixed, and parameters of student model2 are trained by the help of the student model1. Step 3 is optional, and either student model1 or model2 can be used for the prediction. As shown in Figure 1, two loss functions are used in this domain adaptation: L_{Soft} and L_{DC}. L_{Soft} is a soft target loss of knowledge distillation and it is defined in Equation 1. L_{DC} is a loss function borrowed from DeepCORAL [11] and it is defined in Equation 2.

$$L_{Soft} = \mathbb{E}_{x_t \sim X_t} \sum_k [L(M_T(x_t), M_S(x_t))] \tag{1}$$

$$L_{DC} = \frac{1}{4d^2} ||C_s - C_t||_F^2, \tag{2}$$

where $L(\cdot)$ is a loss function, $M_S(x_t)$ is an output when target domain data is fed to a student model, $M_T(x_t)$ is an output when target domain data is fed to a teacher model, C_s is a covariance matrix of $M_S(x_s)$, C_t is a covariance matrix of $M_S(x_t)$, d is degree of the covariance matrix (e.g., the number of samples), and $|| \cdot ||_F^2$ is Frobenius norm, respectively. Given that target domain labels are produced by a teacher model, L_{Soft} is a loss value computed by comparing the generated target domain labels and those predicted by a student model. L_{DC} is computed by the distance between the covariance matrices of the two domains. The final loss function combines L_{Soft} and L_{DC} as shown in Equation 3.

$$L = L_{Soft} + \lambda L_{DC} \tag{3}$$

L_{DC} is weighted by a hyper-parameter λ that controls the strength of domain confusion. A smaller λ increases the importance of class prediction results by a teacher model, which was trained by the source domain data. On the other hand, a larger λ increases the importance of domain invariant representation.

Here, target domain samples to be trained are selected by a given threshold value. Specifically, target domain samples are first fed to the teacher model in Step 2 or the student model1 in Step 3. Softmax function is then applied to the class prediction results so that the sum of the probability of each class is 1.0. If the highest class probability value of a sample is greater than a given threshold value, the sample is used for the student model training. This can prevent situations that incorrect labels produced by the teacher model in Step 2 or student model1 in Step 3 are used for the student model training. The flow of Step2 is summarized in Algorithm 1. Step 3 is done in the same way.

```
Algorithm 1 Domain adaptation method (Step 2)
  Pretrain: Training of teacher model
  for each epoch do
    1) Obtain $x_t$ from teacher model if the highest predicted probability value is higher than threshold
    2) Calculate the Soft Target Loss (Equation 1)
    3) Calculate the DC Loss (Equation 2)
    4) Train student model1 by the loss function (Equation 3)
  Output: student model1
```

IV. Depthwise Separable Neural ODE for FPGA

A. Models

Here, we propose a resource-efficient and lightweight DNN model, termed dsODENet, that takes advantages of both ODENet and DSC for resource-limited FPGAs. Figure 2 shows dsODENet models with three ODEBlocks.

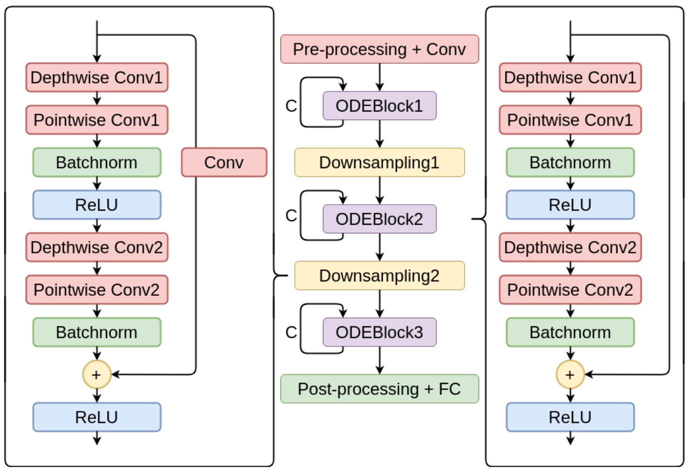

Fig. 2. Model with three ODEBlocks

In Figure 2, the right box shows an internal structure of ODEBlocks and the left box shows that of a downsampling block. The structures of ODEBlocks and downsampling blocks are similar, but in the ODEBlocks, input and output feature map sizes are the same (i.e., $M = N$), while in downsampling block, input feature map size is scaled down to $1/2 \times 1/2$ and thus $M = 2N$. Each ODEBlock is executed C times, while the downsampling block is executed once. In the downsampling blocks, a 1×1 convolutional operation with stride 2 (denoted as Conv) is additionally applied to the shortcut connection. We observed that the accuracy is sometimes sensitive to the DSC on the downsampling block that rescales the feature map. Considering the stability, it is not applied to the Downsampling1. As a smaller version, we consider a model with two ODEBlocks consisting of only ODEBlock1, Downsampling1 and ODEBlock2.

Note that the total parameter size of ODEBlock1, Downsampling1, ODEBlock2, Downsampling2, and ODEBlock3 in the ODENet without DSC is 2,695,168. In dsODENet, the total parameter size is 544,000, which is 79.8% reduction. When a 32-bit fixed-point representation is employed, their sizes are 86.2Mb and 17.4Mb, respectively. This parameter size reduction by DSC is significant since these parameters can be implemented on BRAM or URAM of modest FPGA devices for simplicity. In Section V, ResNet, ODENet, and dsODENet are used as student models of the domain adaptation as mentioned in Section III.

B. FPGA Implementation

As an implementation target, we assume SoC-type FPGA devices that consist of PL (programmable logic) and PS (processing system) parts, and we select Xilinx ZCU104 evaluation board in this paper. The proposed dsODENet models are designed with C/C++ language, and Xilinx Vivado HLS 2020.2 is used as a high-level synthesis tool. The operating frequency of PL part is set to 100MHz. A 32-bit fixed-point format is used to represent numbers.

The weight parameters and feature maps are implemented on BRAM or URAM of the FPGA to fully enjoy benefits of using fast on-chip memories. BRAM and URAM sizes are 11Mb and 27Mb in total, respectively. Their instance sizes are 36kb and 288kb. Depending on the number and sizes of parameter arrays, a part of BRAM and URAM instances is underutilized. In our design, each parameter array is carefully implemented on either BRAM or URAM instances to minimize the underutilized on-chip memories. In the three ODEBlocks case, parameter arrays of normal convolutional layers of Downsampling1, those of depthwise and pointwise convolutional layers of Downsampling2, and those of pointwise convolutional layers of ODEBlock3 are implemented on the URAM instances; and the others are implemented on BRAM instances. Regarding the feature maps, an input feature map array, an output feature map array, and a temporary feature map array that stores the input to be fed to the output directly via a shortcut connection are needed. Their sizes are varied depending on the input image size, which is also varied by the downsampling blocks. In our design, all of them are implemented on BRAM instances for better flexibility.

V. Evaluations

The proposed dsODENet for FPGAs is evaluated with an edge domain adaptation scenario using image recognition datasets. For accuracy evaluations, it is implemented with Pytorch 1.8.1 and torchvision 0.9.1. For resource utilization and performance evaluations, it is implemented with Xilinx Vivado v2020.2 for ZCU104 FPGA board. The board runs the Pynq Linux (Ubuntu 18.04) with ARM Cortex-A53 @ 1.5GHz, 2GB DRAM, and Zynq UltraScale+ XCZU7EV-2FFVC1156.

A. Datasets

For accuracy evaluations, Office-31 dataset (Office-31 [12]) and digit datasets (SVHN [13] and MNIST [14]) are used. Their input image sizes are 256×256 and 32×32, respectively. The MNIST images are grayscale, while SVHN images are RGB colored; thus the MNIST images are duplicated for three channels so that they are compatible with the 3-channel SVHN images. Office-31 is a popular dataset used for domain adaption tasks. It contains 4,110 images, and they are divided into three domains: Amazon (A-domain), Webcam (W-domain), and DSLR (D-domain). A$\rightarrow$W means a domain adaptation scenario in which A-domain and W-domain are the source and target, respectively. A$\rightarrow$W and A$\rightarrow$D scenarios are examined in this paper because domain adaptation from a domain with more images to that with less images is a typical use case. The edge domain adaptation procedure proposed in Section III is used. All the labeled source domain data and all the unlabeled target domain data are used for the training phase. The accuracy is then evaluated with all the labeled target data.

B. *Accuracy*

Either SGD or Adam is selected as an optimizer. The learning rate is reduced based on Equation 4.

$$\eta = \frac{\eta_0}{(1+\alpha p)^{\beta}}, \tag{4}$$

where $\eta_0 = 0.01$, $\alpha = 10$, $\beta = 0.75$, and p is linearly changed from 0 to 1. In the evaluations, the same experiments are executed three times and their accuracy values are averaged and reported. As for a teacher model, an initial model was pre-trained with ImageNet dataset, and using this initial model without the final fully-connected layer, the teacher model is trained. The learning rate is reduced to 1/10 for these pre-trained layers. Different student models that use ResNet-50, ODENet, and dsODENet are compared in terms of accuracy. ODENet and dsODENet use three ODEBlocks as shown in Figure 2. The number of executions C of an ODEBlock is set to 10. They are also compared to a teacher model of ResNet-50 and other domain adaptation techniques [10], [15]–[17].

1) Office-31 Dataset: As a counterpart, MobileDA uses AlexNet as a student model and ResNet-50 as a teacher model. 80% of target domain data is used for the training. As another counterpart, CDAN [15] is also considered, which is a domain adaptation technique based on adversarial training. Table I shows the evaluation results of CDAN, MobileDA, and our approach with different student models.

TABLE I
DOMAIN ADAPTATION ACCURACY OF OFFICE-31 DATASET

Model	A$\rightarrow$W	A$\rightarrow$D
CDAN [15]	77.9	75.1
MobileDA [10]	71.5	75.3
Teacher model ResNet-50	75.8	78.3
Student model1 ResNet-50	80.6	**80.9**
Student model1 ODENet	71.3	78.8
Student model1 dsODENet	80.4	79.1
Student model2 dsODENet	**83.2**	79.1

"Student model1 dsODENet" and "Student model2 dsODENet" are our proposed models, where dsODENet is used for student model1 and student model2, respectively. As shown in the table, the accuracy of A$\rightarrow$W is improved by 4.6% and 2.8% for the teacher model to "Student model1 dsODENet" and "Student model1 dsODENet" to "Student model2 dsODENet", respectively. Also, the accuracy of A$\rightarrow$D is improved by 0.8% for the teacher model to "Student model1 dsODENet". However, there is no improvement in accuracy from the student model1 to the student model2.

Note that the loss function used in the proposed method is different from that of MobileDA. The difference is that a hard target loss, which is computed by labeled source domain data, is used in MobileDA. The soft target loss learns output values of a teacher model in target domain data, while the hard target loss learns one-hot vectors of labeled source domain data. This indicates that the hard target loss may cause an overfitting to the source domain. Also, in edge domain adaptation, the model is considered to be smaller, which weakens the feature learning. These make it easy to overfit in the source domain. Therefore, only soft target loss is used in the proposed method.

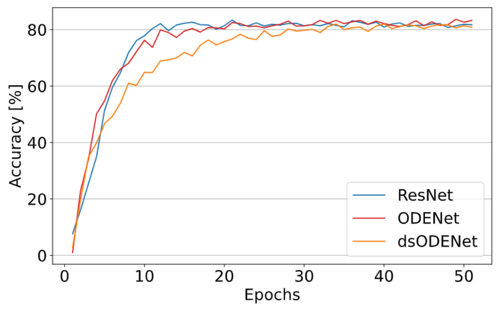

Fig. 3. Training speed of different student models in Offce-31 dataset

Figure 3 shows the training speeds of different student models (ResNet-50, ODENet, and dsODENet) for A$\rightarrow$D scenario. As shown in the figure, the student model of ResNet-50 is converged faster than the others, followed by those of dsODENet and ODENet. In ResNet-50, the number of implemented building blocks is optimized for each feature map size in this experiment. These numbers in ResNet-50 are interpreted as the numbers of continuous executions of ODEBlocks in the cases of ODENet and dsODENet. Reducing the number of executions C of an ODEBlock degrades the approximation performance and reduces the accuracy. To obtain a stable accuracy, the number of continuous executions C of an ODEBlock is equally set to 10 for each feature map size in ODENet and dsODENet cases, though there is still a room for optimization. Note that the training speed of dsODENet is faster than ODENet, because in dsODENet the number of parameters to be trained is reduced by 54.2% to 79.8% by DSC.

2) Digit Dataset: As a counterpart, ADDA [17] which is also a domain adaptation based on adversarial training is considered. Table II shows the evaluation results of ADDA and our approach with different student models. As shown in the table, there is an improvement of 7.0% and 1.7% in accuracy from teacher model to "Student model1 dsODENet" and from

TABLE II
DOMAIN ADAPTATION ACCURACY OF DIGIT DATASETS

Model	SVHN$\rightarrow$MNIST
ADDA [17]	76.0
Teacher model ResNet-50	76.5
Student model1 ResNet-50	82.6
Student model1 ODENet	82.5
Student model1 dsODENet	83.5
Student model2 dsODENet	**85.2**

"Student model1 dsODENet" to "Student model2 dsODENet". We consider that this two-step improvement in accuracy is the result of both the domain adaptation and knowledge distillation. We set a high threshold for knowledge distillation from the teacher model to the student model1. On the other hand, for knowledge distillation from student model1 to student model2, we set a lower threshold. We consider that student model1 is already domain-adapted when learning from the teacher model, and student model2 has the same effect as general knowledge distillation. This results in a further improvement in accuracy just by using a similar architecture.

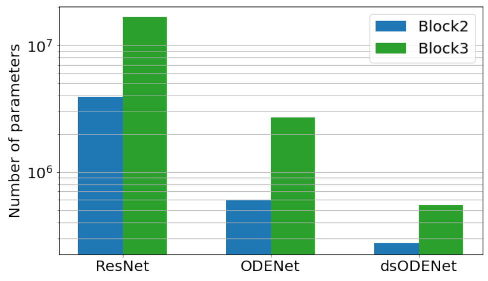

Fig. 4. Comparison of the numbers of parameters of three models

C. FPGA Resource and Performance

Figure 4 compares the number of parameters of the three models: ResNet, ODENet, and dsODENet, with two or three building blocks (denoted as Block2 and Block3). dsODENet has the least number of parameters, followed by ODENet and ResNet, and dsODENet (Block3) has the 97% fewer parameters compared to ResNet (Block3).

dsODENet is implemented on the FPGA and evaluated in terms of resource utilization and execution time. dsODENets with two and three ODEBlocks were implemented on the PL part, while only pre- and post-processing layers were executed on the PS part. Table III shows resource utilizations of the two and three ODEBlocks. In both cases, buffers for weight parameters and feature maps can be implemented only using BRAM and URAM, and thus external DRAM is not required. Table IV compares the execution time of each block for a single data sample between our FPGA implementation and its software counterpart (denoted as FPGA and CPU). In FPGA case, a DMA transfer between PS–PL is additionally required, while it accounts for a negligible portion of the entire inference time. Both ODEBlocks and downsampling blocks were accelerated by 18.7–49.5 times, which contributes to the overall speedup of 27.9 times. The downsampling blocks achieved better speedups than ODEBlocks, as they involve normal convolution, which is more compute-intensive than DSC.

TABLE III
FPGA RESOURCE UTILIZATION OF DSODENET

	BRAM	DSP	FF	LUT	URAM
Block2	462 (74%)	3 (∼0%)	12,762 (2%)	53,886 (23%)	18 (18%)
Block3	584 (93%)	5 (∼0%)	20,839 (4%)	88,055 (38%)	84 (87%)

VI. SUMMARY

In this paper, a combination of Neural ODE and DSC, called dsODENet, is proposed and implemented for FPGAs. dsODENet is applied to a distillation-based edge domain adaptation as student models. All the dsODENet blocks except the pre- and post-processing layers are implemented on PL part of Xilinx ZCU104 FPGA board and the others are executed on PS part. The total parameter size of dsODENets without pre- and post-processing layers is reduced by 54.2% to 79.8%. The FPGA implementation accelerates the prediction tasks by 27.9 times than a software implementation running on PS part.

TABLE IV
EXECUTION TIME OF EACH BLOCK ON FPGA

	FPGA (ms)	CPU (ms)	Speedup
ODEBlock1	13.80	445	32.2
Downsampling1	7.75	384	49.5
ODEBlock2	14.40	400	27.8
Downsampling2	2.72	54	19.9
ODEBlock3	21.20	397	18.7
DMA transfer	0.35	0	-
Total	60.23	1681	27.9

Acknowledgements This work was partially supported by JSPS KAKENHI Grant Number 19H04117, Japan.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385, 2015.
[2] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017, pp. 1800–1807.
[3] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861, 2017.
[4] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *Proceedings of the International Conference on Computer Vision*, Oct 2019, pp. 1314–1324.
[5] R. T. Q. Chen *et al.*, "Neural Ordinary Differential Equations," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NeuroIPS'18)*, Dec 2018, pp. 6572–6583.
[6] H. Watanabe and H. Matsutani, "Accelerating ODE-Based Neural Networks on Low-Cost FPGAs," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops*, Mar 2021, pp. 88–95.
[7] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A Survey of FPGA-Based Neural Network Accelerator," *arXiv:1712.08934v3*, Dec 2018.
[8] J. Faraone *et al.*, "AddNet: Deep Neural Networks Using FPGA-Optimized Multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 115–128, Jan 2020.
[9] L. Bai, Y. Zhao, and X. Huang, "A CNN Accelerator on FPGA Using Depthwise Separable Convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1415–1419, Oct 2018.
[10] J. Yang *et al.*, "MobileDA: Toward Edge-Domain Adaptation," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6909–6918, Aug 2020.
[11] B. Sun and K. Saenko, "Deep CORAL: Correlation Alignment for Deep Domain Adaptation," arXiv:1607.01719, 2016.
[12] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting Visual Category Models to New Domains," *Proceedings of the European Conference in Computer Vision (ECCV'10)*, vol. 6314, pp. 213–226, Sep 2010.
[13] Netzer *et al.*, "Reading Digits in Natural Images with Unsupervised Feature Learning," in *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Dec 2011.
[14] Y. Lecun *et al.*, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[15] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional Adversarial Domain Adaptation," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NeuroIPS'18)*, Dec 2018, pp. 1640–1650.
[16] K. Bousmalis *et al.*, "Domain Separation Networks," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NeuroIPS'16)*, Dec 2016, pp. 343–351.
[17] E. Tzeng *et al.*, "Adversarial Discriminative Domain Adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, Jul 2017, pp. 2962–2971.