A Single Filter CNN Performance for Basic Shape Classification

Kenya Murata
Graduate School of Integrated Arts and Sciences, Kochi University
2-5-1 Akebono-cho, Kochi
780-8520 Japan
kemurata@is.kochi-u.ac.jp

Masataka Mito
Graduate School of Integrated Arts and Sciences, Kochi University
2-5-1 Akebono-cho, Kochi
780-8520 Japan
masamito@is.kochi-u.ac.jp

Daisuke Eguchi
Graduate School of Integrated Arts and Sciences, Kochi University
2-5-1 Akebono-cho, Kochi
780-8520 Japan
daeguchi@is.kochi-u.ac.jp

Yuichiro Mori
Graduate School of Integrated Arts and Sciences, Kochi University
2-5-1 Akebono-cho, Kochi
780-8520 Japan
ymori@is.kochi-u.ac.jp

Masahiko Toyonaga
Graduate School of Integrated Arts and Sciences, Kochi University
2-5-1 Akebono-cho, Kochi
780-8520 Japan
toyonaga@is.kochi-u.ac.jp

***Abstract*—IoT cameras and sensors collect images and sensing data from everywhere in the world to transmit them via the Internet. These collected images are stacked into the servers, and an image recognition system on the server, such as CNN (Convolutional Neural Net), mines valuable information. In the near future, when the enormous number of IoTs collect images at various places, these servers would reach an overflow. Hence, if IoTs would send not only images but also analyzed results to the server, it would reduce server loads; however, the conventional CNN is too large to implement this.**

We propose a single-filter CNN model that can be implemented even on a small IoT. Our CNN model is of minimal configuration with an input layer, an affine transformation layer, a convolution layer, a pooling layer, and a fully connection layer.

We evaluate our proposed CNN model with two experiments. First, we check whether it can learn the eleven basic shapes, i.e., a circle, a triangle, a square, etc. Second, we check whether it can classify the basic shapes against their shape reduction and their noise mixture. Results of the first experiment show that our system can classify all the basic shapes perfectly, results of the second experiment show that accuracy depends on the types of filters for both the scaled-shape classification and the inverse-pixel noise-shape classification.

Keywords—Convolutional Neural Network(CNN), Feature Map, IoT, Pooling Layer.

I. INTRODUCTION

Recently, an enormous number of cameras and sensors have come into use to collect various data, and those are used as big data for promoting business or social prosperity, i.e., in forming the strategy for store location decisions or traffic optimization. Most of these cameras are implemented as IoTs, and the collected data are analyzed by an image recognition system, such as a CNN, and accumulated on many servers.

According to the increase in big data usage, the number of IoTs in use has increased. The IT white paper of the Japanese Ministry of Internal Affairs and Communications (2017) estimates that more than 30 billion IoT devices will be shipped in the market in 2020.

This trend could exacerbate the bottlenecks of both Internet traffic and server overflow. To avoid Internet traffic jams and server processing overflows, we need more intelligent IoTs that can recognize a lot of images and sensor data instantly and abstract them. However, reliable image recognition systems, such as CNN, need high-end processing performance for the enormous processing steps generally involved.

The convolution neural network (CNN) captures an input image layer and repeats the pairs of convolution layer and pooling layer several times, and then finally outputs the classified results of the image by the fully connected layer shape. Fig.1 shows AlexNet illustrated by Krizhevsky et al.[1]. It is a typical CNN model, where the numbers in several layers correspond to the number of filters, and the number of feature maps, and their numbers of pixels. AlexNet comprises more than hundred thousand neural nets, i.e., more than ten million nodes and connections within it. Though AlexNet shows higher performance in classifying the ImageNet Benchmark[1], it will consume a lot of CPU time and memory size, performance that cannot be expected on a cheap IoT. Therefore, we need a new CNN model for image recognition that consumes much smaller computational resources for an IoT implementation.

Zeiler et al. [2] claim that the meaning of each layer in the deeper CNN is not clearly understood, and they therefore propose a method to visualize the feature map in the middle layers [2,4]. Yosinski et al. [3] propose a quantization method of CNN layer parameters to show the dominant layer for accuracy [3]. Young et al. [5] propose an MNS (Multi-node Evolutionary Neural Networks for Deep Learning) as an optimization method of network parameters, in addition to proposing an automatic parameter size determination. Rastegari [6] proposes high-speed, high-precision, and further

compactness as an XNOR network to implement CNN on a circuit.

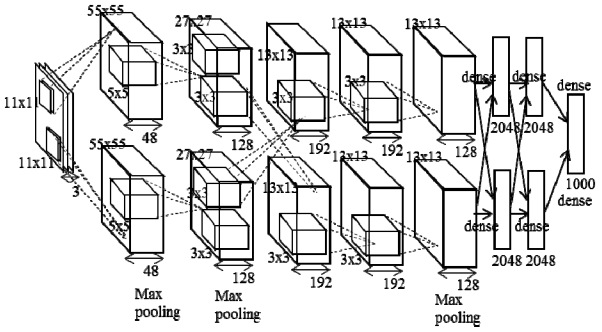


Fig. 1. An Example of Deep CNN model, the Alex Net[1] (Redraw)

Umuroglu [7] proposes a model called Finn that uses floating point converted to binary by focusing on the redundancy of CNN, and implements it in FPGA. Truong [8] claims that a small amount of data is not enough to improve the parameters in the deeper CNN, and as a result, a smaller-scale CNN is proposed.

In this paper, we analyze a single-filter CNN model to optimize for IoT implementation. Our model comprises an input layer, an affine transform layer, a single convolution layer with the specified filter, a pooling layer, a fully connected layer, and a soft-max output layer. As our model is smaller than other CNN models that comprise only a few thousand nodes and connections, we can evaluate the lower bound of the CNN facility.

We evaluate the performances of our model with four primary filters by feeding into it the eleven basic shapes of 28 × 28 pixels, such as a circle, a triangle, a square, a star, etc., and examining two kinds of verification test data. The data from one test are the three different size basic shapes scaled from ×1.0 to ×0.80 and from the other test are the three different added basic shapes from 0% to 13%.

The experimental results show that our models can learn all basic shapes perfectly and that each filter has its own suitable shapes for classification. Because our models can classify those shapes with an extremely small number of operations, it could help the IoT to conduct an affordable CNN model investigation.

The following section describes the motivation for our proposal. The single-filter CNN model is described in Section 3. The experiments are explained in Section 4, and Section 5 covers the conclusion of our proposal.

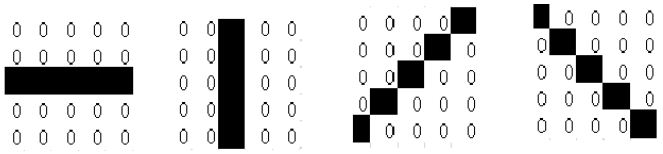
II. Motivation

AlexNet in Fig. 1 has three RGB images input layers by using 48 × 2 feature maps, subjected to pooling processing, 128 × 2 feature maps, and 192 sheets in the pooling process, repeating the filtering process and the pooling process 5 times and repeating the fully connected layer twice. In these calculation processes, the number of nodes required is hundreds of thousands or more and the connection weight calculation between the nodes requires tens of millions or more. Consequently, such large processing operations are not realistic in small IoTs; therefore, a CNN model with a simpler configuration is desired.

We propose the following policy to consider the simplest CNN.

1) The convolution layer consists of one specified filter for feature map.
2) The pooling and the affine layer for abstracting the position and shapes of shapes.
3) The fully connected layer is only the layer for learning.

Fig.2 shows the basic four filters that emphasize horizontal, vertical, lower-left, and lower-right elements on an input shape, respectively. The four feature maps of "star," shown in Fig.3, are shown in Figs.4 (a), (b), (c), and (d), respectively. The following pooling layer abstracts the size and the position of the convolution feature map. Iteration of the convolution and pooling layer extract the shape emphasized and the position free object. All pixels in the final feature maps are connected to the output nodes in the fully connected layer to compare the node values to the teaching data. In the training step, all the connection weights are tuned to minimize the error between the outputs for the input shapes and the corresponding teach values. In the validation step, the trained connection weights are applied for output.



(a) Horizontal (b) Vertical (c) Lower-Left (d) Lower-Right

Fig. 2. Four filters (7×7[pixels])

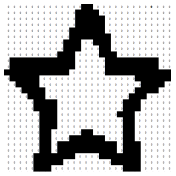
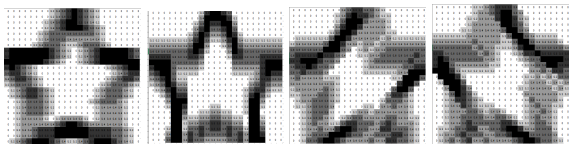


Fig. 3. “Star” (28×28[pixels])



(a) Horizontal (b) Vertical (c) Lower-Left (d) Lower-Right

Fig. 4. Feature maps by the four filters in the convolution (24 × 24 [pixels])

III. A Single-Filter CNN Model

An overview of our model is shown in Fig.5, where an image is set on the input layer (A), then the input image is shifting and fitting to the top left offset in the affine layer (B). Then, the normalized shape is filtered in (C) and output to the convolution layer (D), and the obtained feature maps are

transferred to the final feature maps by pooling layer (E). Finally, all the pixels in the final feature map are fully connected with weights to the output nodes, and the node values of f() in (F) are the sum of the weight and pixel values, and the soft-max values in (G) transform the node values into the probabilities.

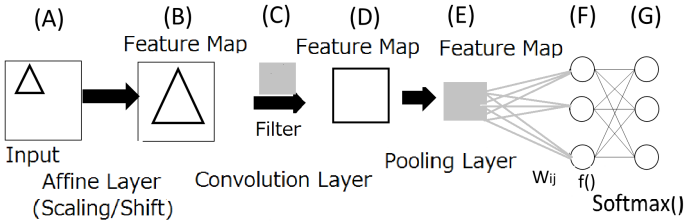


Fig. 5. A single-filter CNN

IV. Experiment

We evaluate our single-filter CNN model in three steps. Step one is to evaluate the learning accuracy of the model with respect to the eleven basic shapes, i.e., a circle, a triangle, a square, etc. Step two is to verify the classification accuracy of scaled basic shapes from x0.9 to x0.8. Step three is to verify the classification accuracy of noise added to basic shapes from 50 dots to 150 dots per 28x28 pixels.

A. Operating environment

We implement our tiny CNN model in C language. The compiler is GCC (MinGW). The execution environment operating system is Windows 7 (64 bit). The feature of tiny CNN in the experiment is shown in Fig.6 as follows:

1) The input layer read image of 28 × 28 pixels as input data.
2) The affine layer detects the size of the shape and enlarges as much as possible and places its offset points in the top left of the image.
3) The convolution layer uses four filters shown in Fig.2. The size of the filter is 5 × 5 pixels and patterns are as shown in Fig.2 and stride is 1 pixel.
4) The pooling layer applies max pooling, and the size is 2 × 2 pixels and stride is 2 pixel.
5) The fully connection layer has 12 × 12 nodes to 11 nodes connections. The weights of connections are decided from the training steps. The number iteration for training a shape is 100 × 10 times.

B. Basic shapes

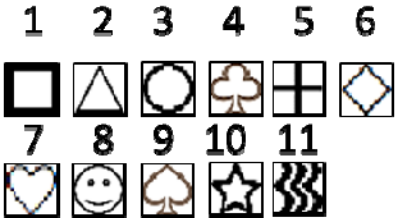


Fig. 6 Basic Shapes

Fig.6 shows the eleven basic shapes—square, triangle, circle, clover, cross, diamond, heart, smile, spade, star, and wave. Each shape size is 28 × 28 pixels, and the value of the pixel is normalized as 1 or 0.

*C. Experiment*1 *: Basic shape training and classification*

We first check whether our system can learn the classification of eleven basic shapes, first by training eleven basic shapes in Fig.6 iteratively as the step 5) in the previous section.

Experimental results are shown in Table 4.1. The soft-max values for the horizontal filter are from 0.61 to 0.94, for the vertical filter are from 0.49 to 0.94, for the lower-left filter are from 0.61 to 0.94, and for the lower-right filter are from 0.73 to 0.95, where every shape is correctly classified.

TABLE 4.1
BASIC SHAPE LEARNING

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.93	0.88	0.72	0.86
triangle	0.94	0.94	0.94	0.94
circle	0.61	**0.49**	0.61	0.73
clover	0.92	0.91	0.89	0.73
cross	0.94	0.94	0.94	0.94
diamond	0.94	0.94	0.92	0.94
heart	0.94	0.93	0.94	0.93
smile	0.94	0.94	0.94	0.95
spade	0.72	0.88	0.82	0.90
star	0.94	0.94	0.94	0.93
wave	0.93	0.94	0.94	0.94

A. Experiment2 : Size scaled basic shapes

We evaluate classification performance of our system for shrinking images after training. The reduced shapes in Fig.7 are generated from the eleven basic shapes in Fig.6 by scaling 0.9 to 0.8. Experimental results of the maximum soft-max values are in Table 4.2, Table 4.3 and Table 4.4.

In the case of scale 0.9, the soft-max values of the horizontal filters are from 0.20 to 0.96, those of the vertical filters are from 0.44 to 0.95, those of the lower-left filters are from 0.23 to 0.95, and those of the lower-right filters are from 0.37 to 0.95.

In case of scale 0.85, the soft-max values of the horizontal filters are from 0.20 to 0.92, those of the vertical filters are from 0.22 to 0.92, those of the lower-left filters are from 0.17 to 0.88, and those of the lower-right filters are from 0.45 to 0.94.

In the case of scale 0.8, the soft-max values of the horizontal filters are from 0.13 to 0.92, those of the vertical filters are from 0.20 to 0.88, those of the lower-left filters are from 0.09 to 0.94, and those of the lower-right filters are from 0.11 to 0.95.

Note that some shapes' soft-max value of less than 0.5 means low reliable classification; these appear in bold in the tables.

It is found that the classification accuracy of the scaled-shape verification results shows that each filter has its suitable shapes clearly. If we combine the filters that are good at the

specified shape classification, then we could design an accurate smallest CNN.

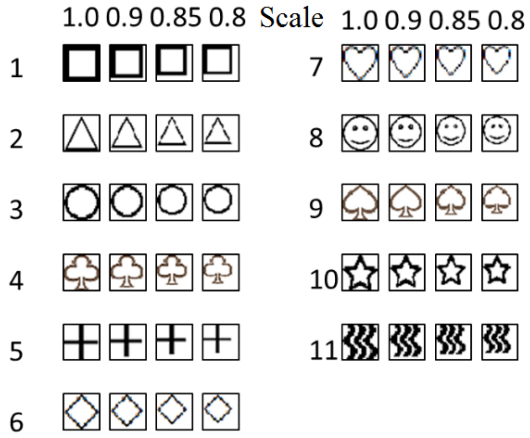


Fig.7. Reduced Basic Shapes

B. Experiment 3 : Noise added basic shapes

We evaluate classification performance of our system for noise added images after training. The noise added shapes in Fig.8 are generated from the eleven basic shapes in Fig.6 by adding 50, 100, and 150 reverse dots. Experimental results of the maximum soft-max values are in Table 4.5, Table 4.6, and Table 4.7.

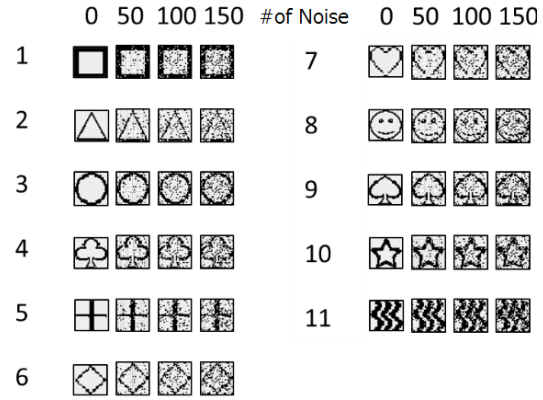


Fig. 8. Noise added basic shapes

In the case of 50 noise added, the soft-max values of the horizontal filters are from 0.33 to 0.90, those of the vertical filters are from 0.31 to 0.91, those of the lower-left filters are from 0.16 to 0.89, and those of the lower-right filters are from 0.31 to 0.94.

In the case of 100 noise added, the soft-max values of the horizontal filters are from 0.13 to 0.88, those of the vertical filters are from 0.08 to 0.89, those of the lower-left filters are from 0.02 to 0.91, and those of the lower-right filters are from 0.17 to 0.92.

In the case of 150 noise added, the soft-max values of the horizontal filters are from 0.06 to 0.86, those of the vertical filters are from 0.05 to 0.82, those of the lower-left filters are from 0.05 to 0.89, and those of the lower-right filters are from 0.05 to 0.91.

Note that some shapes with soft-max values less than 0.5 means also low reliable classification, these values appear in bold in the tables.

It is found that the classification accuracy of the noise added shape verification results depends on the type of filters. Thus, if we select the appropriate filters, the combination of them could classify the specific shapes that the other filter cannot, which makes it possible to design the smallest CNN for IoT.

V. Conclusion

We evaluated a single-filter CNN model for IoT implementation. Our single-filter CNN model is based on a minimal configuration of CNN with an affine transformation layer, a convolution layer, a pooling layer, and a fully connection layer. It is found that the classification accuracy of the reduced shape and the noise added shape verification results depend on the type of filters. Thus, if we select the appropriate filters, the combination of them could classify the specific shapes; thus, through their combination, it could be possible to make the smallest CNN for IoT.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks." In Proc. NIPS, 2012.

[2] M. Zeiler, R. Fergus. "Visualizing and understanding convolutional networks." In: European conference on computer vision. Springer, Cham, 2014. pp. 818-833, 2014.

[3] J. Yosinski, et.al. "How transferable are features in deep neural networks?" In Advances in neural information processing systems, pp. 3320-3328, 2014.

[4] A. Romero et.al. "Fitnets: Hints for thin deep nets." arXiv preprint arXiv:1412.6550, 2014.

[5] S.R.Young et al. "Optimizing deep learning hyper-parameters through an evolutionary algorithm." In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. ACM, p. 4, 2015.

[6] M. Rastegari, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." In: European Conference on Computer Vision. Springer, Cham, 2016. pp. 525-542, 2016.

[7] Y. Umuroglu, et al. "Finn: A framework for fast, scalable binarized neural network inference." In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2017. pp. 65-74, 2017.

[8] T-D.Truong, et.al. "Lightweight Deep Convolutional Network for Tiny Object Recognition." In Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2018), pp.675-682, 2018.

TABLE 4.2
CLASSIFICATION RESULTS FOR SCALE = 0.9

Scale=0.9

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.96	0.95	0.83	0.95
triangle	0.85	0.87	0.82	0.66
circle	**0.43**	0.60	**0.23**	**0.37**
clover	0.58	0.73	0.68	0.56
cross	0.88	0.86	0.95	0.95
diamond	**0.48**	0.82	0.73	0.83
heart	0.90	**0.44**	0.79	0.81
smile	0.71	**0.46**	**0.30**	0.74
spade	**0.20**	**0.44**	**0.44**	**0.43**
star	0.78	0.74	0.78	0.80
wave	0.88	0.84	0.88	0.90

TABLE 4.3
CLASSIFICATION RESULTS FOR SCALE = 0.85

Scale=0.85

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.92	0.84	0.83	0.87
triangle	0.73	0.92	0.70	0.78
circle	**0.13**	**0.22**	**0.17**	**0.45**
clover	0.88	0.82	0.86	0.80
cross	0.83	0.90	0.88	0.94
diamond	**0.20**	0.51	**0.38**	0.81
heart	0.88	**0.48**	0.57	0.74
smile	0.61	**0.48**	0.51	0.78
spade	0.63	**0.37**	**0.42**	0.58
star	0.80	0.84	0.86	0.80
wave	0.80	0.53	0.76	0.76

TABLE 4.4
CLASSIFICATION RESULTS FOR SCALE = 0.8

Scale=0.8

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.84	0.66	0.65	0.86
triangle	0.85	0.88	0.72	0.75
circle	**0.13**	**0.30**	**0.09**	**0.11**
clover	0.70	0.88	0.63	0.57
cross	0.70	0.85	0.94	0.95
diamond	0.56	0.70	0.80	0.87
heart	0.92	**0.26**	**0.39**	0.85
smile	0.57	0.57	**0.13**	0.92
spade	**0.26**	**0.20**	0.69	**0.48**
star	**0.43**	**0.36**	**0.23**	**0.37**
wave	0.72	0.82	0.82	0.79

TABLE 4.5
CLASSIFICATION RESULTS FOR NOISE = 50

Noise=50

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.86	0.67	0.63	0.69
triangle	0.59	**0.46**	0.71	**0.46**
circle	**0.33**	**0.33**	**0.31**	**0.47**
clover	0.68	0.84	0.79	**0.31**
cross	0.82	0.91	0.86	0.83
diamond	**0.34**	**0.31**	**0.16**	0.63
heart	0.67	0.62	0.77	0.61
smile	0.83	0.59	0.83	0.86
spade	0.67	0.70	**0.29**	0.70
star	0.80	0.80	0.79	0.81
wave	0.90	0.89	0.89	0.94

TABLE 4.6
CLASSIFICATION RESULTS FOR NOISE = 100

Noise=100

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.58	**0.25**	**0.39**	**0.22**
triangle	0.69	**0.46**	**0.47**	**0.41**
circle	**0.13**	**0.42**	**0.21**	**0.40**
clover	**0.22**	0.75	0.71	**0.17**
cross	0.54	0.84	0.83	0.74
diamond	**0.14**	**0.19**	**0.02**	0.70
heart	**0.44**	**0.42**	0.66	**0.36**
smile	0.58	**0.38**	**0.46**	0.74
spade	**0.43**	**0.08**	**0.08**	**0.30**
star	**0.43**	0.58	0.69	**0.48**
wave	0.88	0.89	0.91	0.92

TABLE 4.7
CLASSIFICATION RESULTS FOR NOISE = 150

Noise=150

	Horizon	Vertical	Low-Left	Low-Rigth
square	0.86	0.67	0.63	0.69
triangle	**0.48**	**0.36**	**0.24**	**0.17**
circle	**0.06**	**0.30**	**0.15**	**0.18**
clover	**0.14**	0.67	**0.41**	**0.05**
cross	**0.39**	0.65	0.74	0.51
diamond	**0.14**	**0.17**	**0.04**	0.67
heart	**0.34**	**0.37**	**0.40**	**0.34**
smile	**0.41**	**0.21**	0.54	0.71
spade	**0.24**	**0.05**	**0.05**	**0.17**
star	**0.27**	0.51	**0.50**	**0.41**
wave	0.84	0.82	0.89	0.91