Journal of Signal Processing Systems (2023) 95:909–920
https://doi.org/10.1007/s11265-020-01520-7

On-Device Partial Learning Technique of Convolutional Neural Network for New Classes

Cheonghwan Hur[1] · Sanggil Kang[1]

Received: 2 May 2018 / Revised: 2 December 2019 / Accepted: 16 January 2020 / Published online: 30 January 2020

Abstract
In general, Convolutional Neural Networks (CNNs) have a complex network structure consisted of heavy layers with huge number of parameters such as the convolutional, pooling, relu-activation, and fully-connected layers. Due to the complexity and computation load, CNNs are trained on a cloud environment. There are a couple of drawbacks on learning and performing on the cloud such as security problem of personal information and dependency of communication state. Recently, CNNs are directly trained at the mobile devices in order to alleviate those two drawbacks. Due to the resource limitation of the mobile devices, the structure of CNNs needs to be compressed or to reduce training overhead. In this paper, we propose an on-device partial learning technique with the following benefits: (1) does not require additional neural network structures, and (2) reduces unnecessary computation overhead. We select a subset of influential weights from a trained network to accommodate the new classification class. The selection is made based on the information of the contribution of each weight to output, which is measured using the entropy concept. In the experimental section, we demonstrate and analyze our method with a CNN image classifier using two datasets such as Mixed National Institute of Standards and Technology image data and Microsoft Common Objection in Context data. As a result, the computational resources at LeNet-5 and AlexNet showed performance improvements of 1.7× and 2.3×, respectively, and memory resources demonstrated performance improvements of 1.4× and 1.6×, respectively.

Keywords Convolutional neural network · Information · On-device partial learning · Qualitative entropy · Weight

1 Introduction

Deep-neural networks (DNNs) have been creating tremendous change and advance in various academic or industry fields. In particular, Convolutional Neural Networks (CNNs) [1] have made rapid progress in academic field due to the ImageNet dataset and many advanced networks such as GoogleNet [2], ResNet [3], and VGGNet [4]. Recently, for the applications that provide intelligent services using these developments, CNNs are getting be imbedded to various mobile devices such as smartphone, tablet personal computer, drone, and embedding board. The CNNs perform well on the image classification such as detecting dangerous situations by predicting people's behavior from videos captured through mobile device cameras, and taking pictures on a mobile device and giving information of the pictures. However, in general, the CNNs have a complex network structure which consists of heavy layers and a lot of parameters such as the convolutional, pooling, relu-activation, and fully-connected layers. Due to the complexity and computation load, the CNNs are learnt and run on large-scale cloud networks. However, there are two obstacles on learning and performing on the cloud network: 1) Security of personal information becomes vulnerable because an interchange of information is getting easy, and the possibility of personal information leakages by hacking operations. 2) Learning and running can be unstable because cloud communications have a high dependency of mobile network state. Thus, some researchers [5–12, 18–33] have studied for making the structure of CNNs compressed and efficient and light learning of CNNs in order to learn and run the network ("network" can be exchangeable with "network of CNN or DNN" from here) directly at the mobile devices. The representative methods for the compression of network structure are weight matrix reconstruction [5–8], quantization [9, 10], and pruning [11, 12]. Also, a popular method for efficient and light learning is the transfer learning that trains the networks by utilizing the information of an existing network. It reduces learning cost when new

✉ Sanggil Kang
sgkang@inha.ac.kr

[1] Department of Computer Engineering, Inha University, Inha-ro 100, Nam-gu, Incheon 22212, South Korea

classes or patterns, which are not known during training the existing network, happen.

A variety of approaches for the transfer learning, such as the incremental-transfer [18], the parameter-transfer [19–23], and the feature-transfer [24–33], have been developed by transferring additional network structures, feature values, classifier weights, etc. for new classes. The incremental-transfer method (transferring an additional network structure) reconfigures an existing network by connecting an additional network for new classes in parallel with the existing network. However, the method has a disadvantage that requires the structure overhead for designing the additional network structure. The parameter-transfer (transferring characteristic values) constructs a network by utilizing the common features extracted from the feature filters of an existing network. The feature-transfer (transferring the weights) trains an existing network by using only the common weights obtained from analysis of the differences of the weights. The analysis is done based on the distribution of input data of existing classes and the new class. However, the methods require unnecessary computation overhead of analysis of the weight differences.

To solve the problems issued from the transfer learning, we propose an on-device partial learning technique with no requiring additional network structures and reducing the unnecessary computation overhead. First, we select the weights which need to be trained from an existing network when a new class happens. The weights are analyzed using the information of importance of their contribution to outputs. The importance of weights means that the weights have big contribution to outputs for existing classes. The information of weights is measured by modifying the entropy concept which is widely used in information theory [34–39]. The entropy is defined as the expected amount of information or the potential amount of information. Thus, it is calculated by the probability of an event occurring. However, the entropy formula cannot be directly applied to the networks because the magnitude of weights plays a critical role on whether the weights are important to the output or not. Thus, we combine the entropy concept with the magnitude of weights in order to develop a new importance metric which is named as the qualitative entropy. Based on the qualitative entropy, we partially learn the existing network by training only the little important weights on the existing classes. In the experimental section, we demonstrate and analyze the qualitative entropy technique for the CNN image classifier. We also show that our method outperforms in learning time and performance to the conventional transfer leaning using two data sets such as Mixed National Institute of Standards and Technology (MNIST) [40] and ImageNet dataset.

The remainder of this paper is organized as follows. Section 2 introduces various transfer leaning techniques which are highly related to our work. Also, we explain about various entropy applications using CNNs to help understand the derivation of the qualitative entropy metric. In Section 3, we derive the quantitative entropy metric step by step. Section 4 demonstrates and analyze our method and compare the classification performance with the transfer learning method. Section 5 concludes this paper.

2 Related Works

2.1 Transfer Learning for CNN

In order to reduce cost for rebuilding an existing CNN as new classes occur, many researchers [18–33] worked on developing the transfer learning techniques. The transfer learning makes it easier to learn new classes by using knowledge of an existing network. There are three approaches for the transfer learning such as the incremental-transfer, parameter-transfer, and feature-transfer.

The incremental-transfer builds an architecture consisting of a basic inference network and a small incremental network. The basic inference network is fixed after learning with large data set. The small incremental network is learned with user-customized data set that users provide. The learned small incremental network is connected with the basic inference network in parallel so that the network can classify the new classes without hurting the existing network. The approach increased the classification accuracy from 76.3% to 93.2% in experiments using the 19 hand-printed character image data sets provided from National Institute of Standards and Technology [41] and the users' customized dataset. However, the approach requires computation overhead to learn the new networks connected to the existing networks. The parameter-transfer extracts the common feature of weight information in a classifier by comparing and analyzing principal components of the weights. In order to transfer the common feature of the network for the new class, Killian et al. [19] used a Hidden Parameter Markov Decision Process (HiP-MDP). Also, Shin et al. [20] used an unsupervised CNN pre-training with supervised fine-tuning. The found features are used to train the network for new classes. Fernandes et al. [21] used a method that encourages the source and target to share the same coefficient signs. However, it is difficult to achieve real benefits with these techniques because they use the computational resource as a whole matrix-based operation. To solve the problem, Long et al. [25] proposed a feature-transfer technique that uses domain adaptation in neural networks, which can jointly learn adaptive classifiers and features from labeled data in the source and unlabeled data in the target. Once the adaptation is done, the network is

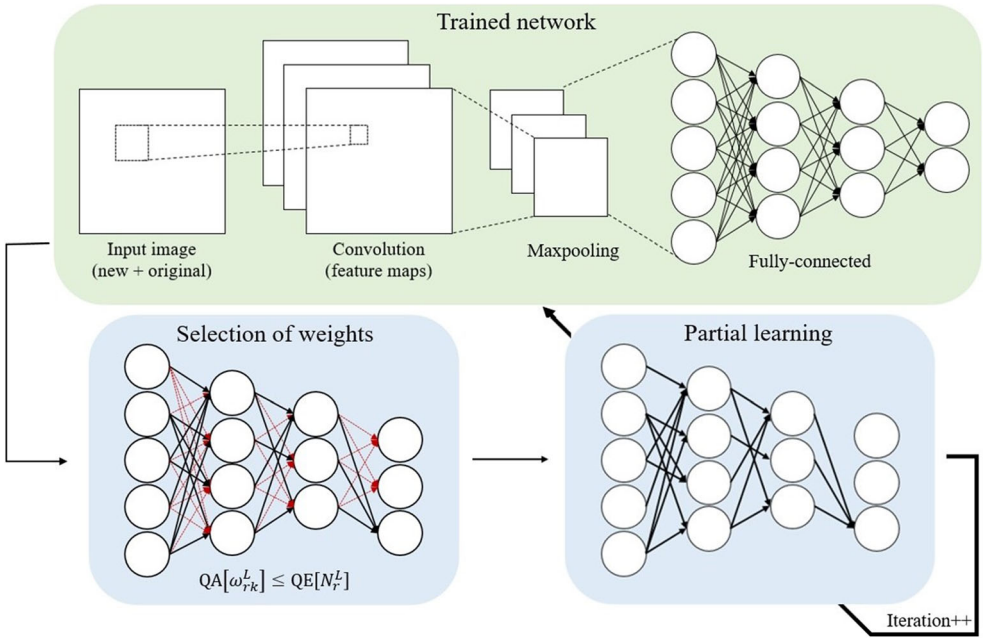


Fig. 1 The overall process schema for partial learning based on qualitative entropy.

elaborately trained using residual transfer network [42, 43]. They used 20 newsgroups [44] dataset to model the approximate network, and then the network is trained using SRAA 2 [45] and Reuters–21,578 [46]. Tjandra et al. [27] proposed a feature transfer learning to assist the training process of the end-to-end network. First of all, the method removes the data of existing classes which are not categorized in the existing network and learns the network by giving more benefit to the weights of the new class. They evaluate the accuracy performance of the method using the part-of-speech tagging [47], the named entity recognition [48], the relation extraction [49], and the semantic role labeling [50, 51]. Also, they demonstrated that integrating and leveraging information from the new class is more useful for improving the performance than excluding misleading training cases from existing classes. Even though the feature transfer approaches show good performance with respect to the accuracy, unnecessary computational costs are incurred by fine-tuning all weights of the existing networks. Thus, their methods cannot be applicable to learning directly in mobile devices as mentioned in the previous section.

In order to solve the problem, we learn partial weights in the existing network without an additional structure and training full weights for new classes. To do that, we apply the concept of entropy to select the weights for the partial learning. We mention about the applications of entropy for deep learning systems in the following section.

2.2 Applications of Entropy for CNN

Entropy is defined as the expectation of amount of information under uncertain circumstance, which has been used as a criterion for detecting and selecting important or unimportant information in many fields such as data mining, pruning, weight quantization, and sampling in CNN. Bereziński et al. [52] proposed an entropy-based approach for detecting malware based on abnormal patterns in the computer network. They use the entropy of probes passing bi-directionally across the router in order to estimate the traffic characteristics such as flows duration, packets, and in(out)-degree. The detection of abnormal probes is made by comparing the amount of information of each characteristic with the estimated entropy. The phenomenon of a computer network is usually very small, and the abnormal information of network traffic represented by packet or byte number is hidden. In this situation,

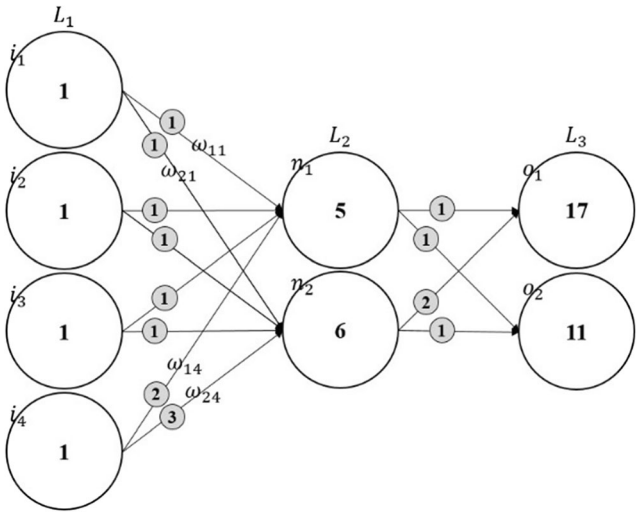


Fig. 2 An example for calculating the qualitative entropy.

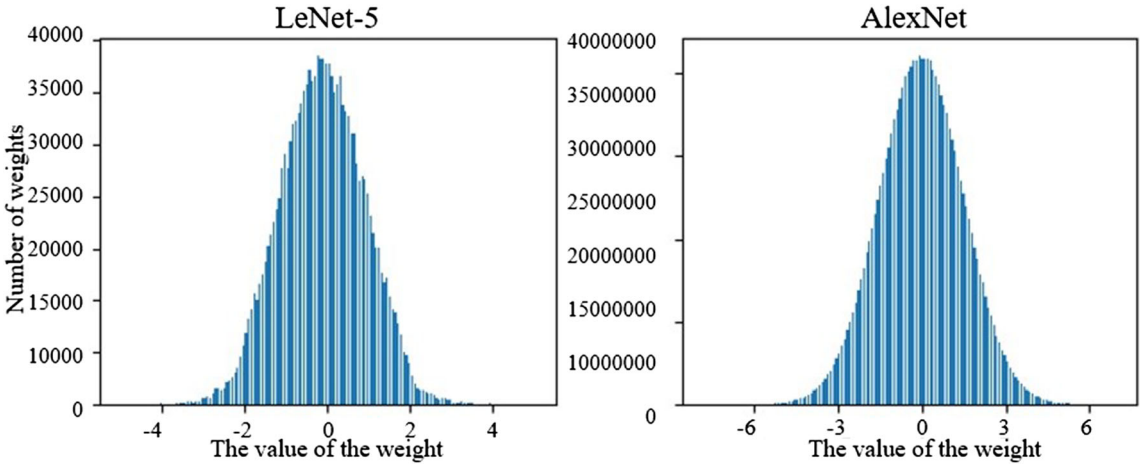


Fig. 3 Distribution of weights of fully-connected layer1 of LeNet-5 and AlexNet.

the entropy concept can work well on detecting the abnormal patterns because it tells potential amount of information which is hidden between packets. Han et al. [53] proposed an entropy-based filter pruning method to accelerate and compress existing CNNs. The importance of each filter in the convolution layer is evaluated by the entropy. The filter with a low entropy is considered to hold less information, then the filter is considered as less important. They remove the filter with the smaller value of entropy according to the evaluated filter value by considering the amount of information transferred from the filter to the next layer. The entropy technique showed good pruning performance on the CNN structure of VGGNet and ResNet trained with ImageNet [54] dataset.

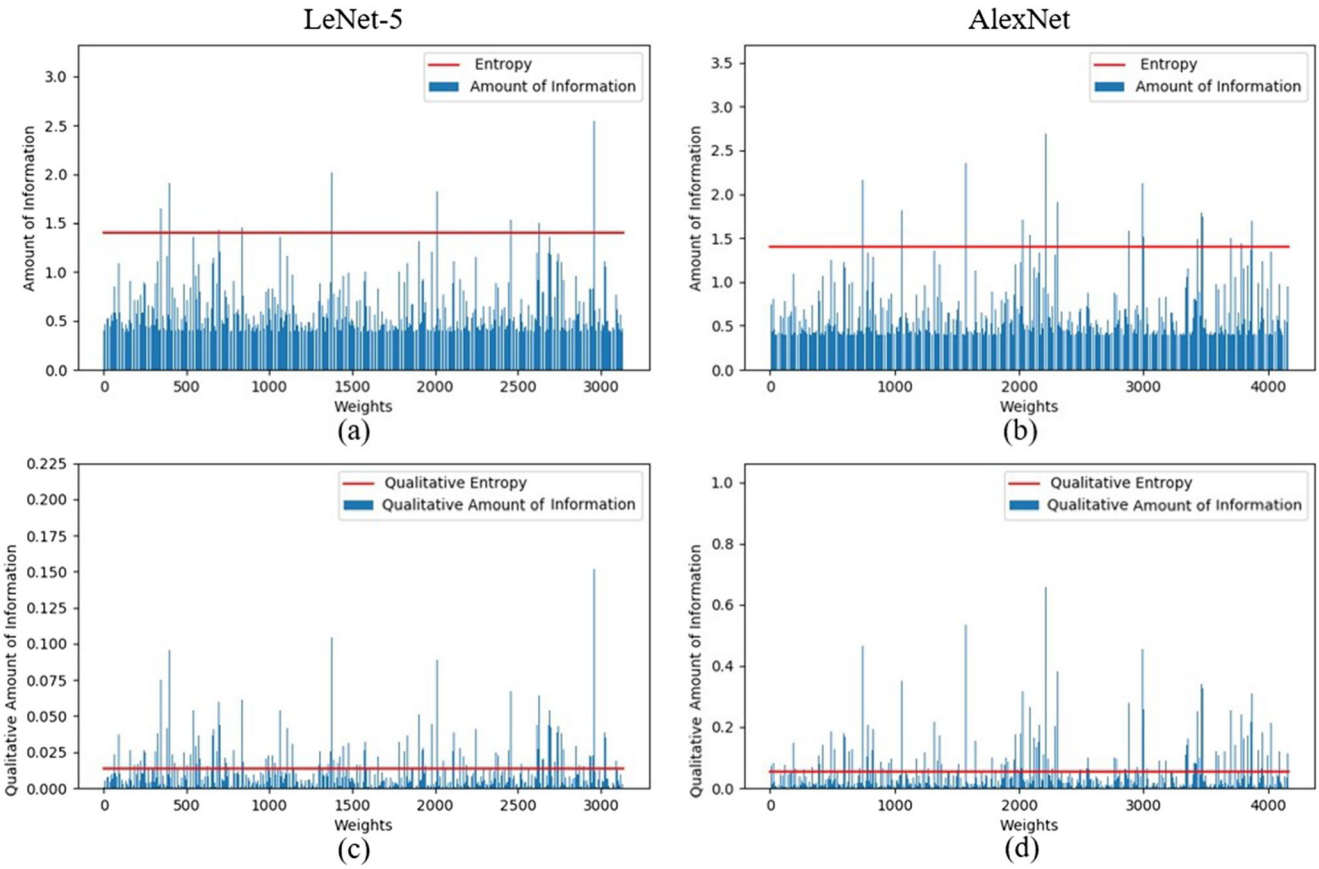


Fig. 4 The entropy of a node and the amount of information of weights connected to the node in LeNet-5 (a), in AlexNet (b), The qualitative entropy of a node and the qualitative amount of information of weights connected to the node in LeNet-5 (c), in AlexNet (d).

Table 1 The weights selected by Equation (7) for LeNet-5

Layer	Weights	Selected weights	Selected weights(%)
Fully-connected layer1	400,000	272,195	68.04%
Fully-connected layer2	5000	3634	72.68%
Total	405,000	275,829	68.10%

Park et al. [55] proposed an entropy-based quantization technique to reduce the inference cost of neural networks. They cluster weights according to the importance of the weights using the entropy concept to improve quantization quality. The weights are grouped so as the entropy of weights of each cluster is uniform unlike the random binarization technique such as the binary quantization [13–15]. By using the entropy for the quantization, the compression performance can be improved because the hidden information in weights is considered during clustering. The entropy technique provided good compression performance on the CNN structure of AlexNet [16], GoogleNet, and ResNet with ImageNet dataset. Zilly et al. [56] proposed an entropy sampling technique to reduce the computational complexity in retinal image segmentation using a CNN structure based on ensemble learning. They estimate the probability for pixels with a histogram of 256 bins represented using the retinal image of each pixel. The entropy of the pixel is calculated using the estimated probability. The entropy is used to find the important pixels that have larger entropy than the average entropy of whole pixels. They showed outperformance of their method on the CNN trained with DRISHTI-GS [17] dataset.

From above methods, the entropy has been proved that it can be a useful approach for determining importance of information in many fields. However, the entropy has a limit for being directly expanded to various fields because the information depends on network structure or dataset, so the entropy should be modified. From the following we show how the entropy is modified for finding weights suitable for the partial learning.

3 Qualitative Entropy Based Partial Learning

Figure 1 shows the schematic of overall process of the qualitative entropy-based partial learning method which enables to classify the part of weights to train the existing network for the new classes. The entropy is employed for calculating the expectation of amount of information of weights for generating outputs as shown in Equation (1). If the amount of information of a weight is lower than the entropy of a set of weights, the weight is considered as less important.

$$E\left[N_r^L\right] = \sum_k Pr\left(\omega_{rk}^{L-1}\right) \cdot \mathrm{A}\left[\omega_{rk}^{L-1}\right] \tag{1}$$

where, N_r^L is the r^{th} node in the L^{th} layer. $E\left[N_r^L\right]$ is the entropy of N_r^L, ω_{rk}^L is a weight connected from the k^{th} node in the L^{th} layer to the r^{th} node of the next layer. $Pr\left(\omega_{rk}^L\right)$ is the probability of ω_{rk}^L among the weights connected to N_r. $\mathrm{A}\left[\omega_{rk}^L\right]$ is the amount of information that ω_{rk}^L holds. Each entropy means the expected value of the amount of information held by ω_{rk}^{L-1} connected to N_r^L. The amount of information of each ω_{rk}^{L-1} can be evaluated by Equation (2).

$$\mathrm{A}\left[\omega_{rk}^L\right] = -\log Pr\left(\omega_{rk}^L\right) \tag{2}$$

If each $\mathrm{A}\left[\omega_{rk}^{L-1}\right]$ has a smaller value than $E\left[N_r^L\right]$, it is meant that less information is transmitted to N_r^L. It is deciphered that ω_{rk}^{L-1} is not important for generating outputs during learning. However, as shown in Equation (1) and (2), the entropy is produced by products based on probability distributions of weights only. If the probability distributions of the weights are similar, the weights have the similar entropy. Even though the weights have the similar entropy, the influence on the output node can be different because the weights are trained in a black-box pattern [57].

For example, as shown in Fig. 2, four nodes (i_1, i_2, i_3, i_4) have the same value of 1 and are completely connected to the node of L_2. If the values of the weights connected to n_1 of L_2

Table 2 The weights selected by Equation (7) for AlexNet

Layer	Weights	Selected weights	Selected weights(%)
Fully-connected layer1	38,000,000	23,943,187	63.01%
Fully-connected layer2	17,000,000	11,053,214	65.02%
Fully-connected layer3	4,000,000	2,800,745	70.02%
Total	59,000,000	37,797,146	64.06%

Table 3 Performance of partial learning as adding new class at a time for MNIST datase

	Classes	Accuracy (%)	Classes	Accuracy (%)
Initial	5	99.2	–	–
Trained on	1	98.7	4	97.7
Tested on	6	99.06	9	93.19
Trained on	2	98.2	5	94.8
Tested on	7	98.25	10	89.74
Trained on	3	98.1	–	–
Tested on	8	97.61	–	–

are 1, 1, 1, and 2, then the entropy of the nodes is 0.431. However, if the values of weights connected to n_2 of L_2 are 1, 1, 1, and 3, the entropy is 0.431, too. The entropy values of n_1 and n_2 are the same. It is because the entropy is calculated using the probability distribution only. Even though the entropy of n_1 and n_2 is the same, n_2 has a greater impact on o_1 and o_2 in L_3. In order to solve the problem, we consider the quality of the weights to avoid the case that the weights with much information are misunderstood as insignificant due to their high probability.

The quality of the weights needs to be normalized into [0, 1] by using the sigmoid function because it helps to avoid over-emphasizing on the contribution of weights which are outside of a certain range of the distribution of weights. In this paper, the normalized quality of the weight as shown in Equation 3 is called as the qualitative characteristics for convenience.

$$\mathrm{Q}\left[\omega_{rk}^{L}\right] = {}^{1}\Big/_{\left(1+e^{-\omega_{rk}^{L}}\right)} \tag{3}$$

where, $\mathrm{Q}\left[\omega_{rk}^{L}\right]$ is the quality of ω_{rk}^{L}. The qualitative characteristic depends on the degree of the importance of each

Table 4 Performance of partial learning as adding new class at a time for ImageNet dataset

	Classes	Accuracy (%)	Classes	Accuracy (%)
Initial	500	62.8	–	–
Trained on	50	61.5	300	59.3
Tested on	550	62.2	800	56.3
Trained on	100	61.7	350	58.2
Tested on	600	61.5	850	52.2
Trained on	150	60.8	400	56.3
Tested on	650	61.4	900	48.6
Trained on	200	59.1	450	55.7
Tested on	700	61.0	950	42.8
Trained on	250	60.2	500	50.6
Tested on	750	59.5	1000	32.1

weight connected to one node. The formula of the qualitative information of each weight is obtained by plugging Equations (2) and (3) into Equation (4).

$$\mathrm{QA}\left[\omega_{rk}^{L}\right] = \mathrm{Q}\left[\omega_{rk}^{L}\right] \cdot \mathrm{A}\left[\omega_{rk}^{L}\right] \tag{4}$$

where, $\mathrm{QA}\left[\omega_{rk}^{L}\right]$ is the qualitative information amount of ω_{rk}^{L}. Using the Equation (4), the qualitative entropy can be derived as Equation (5).

$$\begin{aligned}\mathrm{QE}\left[N_r^{L}\right] &= -\sum_{k} Pr\left(\omega_{rk}^{L-1}\right) \cdot logPr\left(\omega_{rk}^{L-1}\right) \cdot \mathrm{A}\left[\omega_{rk}^{L-1}\right] \\ &= -\sum_{k} Pr\left(\omega_{rk}^{L-1}\right) \cdot \mathrm{QA}\left[\omega_{rk}^{L-1}\right]\end{aligned} \tag{5}$$

where, $\mathrm{QE}\left[N_r^{L}\right]$ is the qualitative entropy of N_r^{L}. For most CNNs. The probability distribution of the weights can converge into a Gaussian distribution with bell-shape by the central limit theorem [58] because CNN has a huge number of weights in the fully-connected layer at least 5000. Therefore, Equation (5) can be simplified as Equation (6).

$$\begin{aligned}QE\left[N_r^{L}\right] &= -\sum_{k}\left(\frac{1}{\sigma_{\omega}^{L-1}\sqrt{2\pi}} e^{\left(-\frac{\left(\omega_{rk}^{L-1}-\mu_{\omega}^{L-1}\right)^2}{2\sigma_{\omega}{}^2}\right)}\right) log\left(\frac{1}{\sigma_{\omega}\sqrt{2\pi}} e^{\left(-\frac{\left(\omega_{rk}^{L-1}-\mu_{\omega}^{L-1}\right)^2}{2\sigma_{\omega}{}^2}\right)}\right) \\ \cdot \mathrm{A}\left[\omega_{rk}^{L-1}\right] &= {}^{1+ln\left(2\pi\sigma_{\omega}^{L-1^2}\right)\cdot\sum_k \mathrm{A}\left[\omega_{rk}^{L-1}\right]}\Big/_{2} = ln\left(\sigma_{\omega}^{L-1}\sqrt{2\pi e}\right) \cdot \sum_{k} \mathrm{A}\left[\omega_{rk}^{L-1}\right]\end{aligned} \tag{6}$$

where, σ_{ω}^{L} is the standard deviation of the weights in the L^{th} layer, μ_{ω}^{L} is the average of the weights in the L^{th} layer. If Equation (6) is applied for evaluating the entropy of Figure 2, n_1 and n_2 become 0.116 and 0.212 respectively. From the result, it is concluded that n_2 has a greater impact than n_1. As shown in Figure 2, if values of nodes at lower layer are changed, its subsequent nodes are affected. Consequently, by partially learning about the weights that have a little influence on the output, the classification performance for the existing classes can be preserved. The weights with little amount of information can be selected by Equation (7) obtained from Equation (4) and (6).

$$\mathrm{QA}\left[\omega_{rk}^{L}\right] \leq \mathrm{QE}\left[N_r^{L}\right] \tag{7}$$

However, the partial learning does not apply the above method in the last layer. Since the last layer has no next nodes to transmit information, the weights influence on the output node independently. Therefore, it is efficient to learn only the weights associated with the new classes. Revisiting to the example of Figure 2, in the node n_1, the qualitative amount of information of ω_{11} obtained from Equation (7) is 0.091, and the qualitative amount of information of ω_{14} is 0.530. The qualitative entropy of the n_1 is 1.789. In the node n_2, the qualitative amount of information of ω_{21} is 0.091, and the qualitative amount of

Table 5 Training time of partial learning and transfer learning for LeNet-5

Additional class numbers	Partial learning(s)	Transfer learning(s)	Gap(s)
1	45	252	207
2	53	294	241
3	56	330	274
4	61	370	309
5	64	412	348

information of ω_{24} is 0.573. The qualitative entropy of the n_2 is 4.011. Because of the qualitative entropy, both qualitative and probabilistic characteristics are taken into consideration, so that we can identify the weights with little amount of information to generate outputs and train them only. Our algorithm is summarized as below.

Algorithm Partial Learning based on Qualitative Entropy

Input:

Dataset (MNIST, ImageNet dataset*), trained network, the weights in each layers W, the number of layers L*

Output:

The new class adapted network

Initialization:

Epoch	*Number of iteration*
Total_batch	*Number of training per epoch*
$QE[N_r^L]$	*Qualitative entropy*
Sum	*Sum of weights*

Procedure:

$$QE[N^L] = \ln\left(\sigma_\omega^{L-1}\sqrt{2\pi e}\right) \Big/ \sum_k Q[\omega^{L-1}]$$

```
for i=1 to L do
    if QA[ω^i] ≤ QE[N^i] then
        Select ω^i
    end if
end for
for j=0 to Epoch do
    for k=0 to Total_batch do
        for r=0 to L do
            Only update seleted weight ω^i
        end for
    end for
end for
```

4 Experiment

We analyze the mechanism of our method and show the performance using networks such as Lenet-5 [59] and AlexNet. Lenet-5 consists of two convolutional layers and two fully-connected layers. Each layer has weights of 500, 25,000, 400,000, and 5000 respectively. The network is trained with the MNIST data set having 10 handwritten image classes, 6000 pieces of each. AlexNet consists of five convolutional layers and three fully-connected layers. Each layer has weights of 3500, 307,000, 885,000, 663,000, 442,000, 38,000,000, 17,000,000, and 4,000,000 respectively. The network is trained with the ImageNet dataset having 1 M images classified into 1000 categories. We modify the Tensorflow framework [60] by adding a mask to ignore the weights not selected. Also, we use the NVIDIA Titan X Pascal graphics processing unit and NVIDIA Jetson TX1.

Table 6 Training time of partial learning and transfer learning for AlexNet

Additional class numbers	Partial learning(s)	Transfer learning(s)	Gap(s)
50	39,024	165,833	126,809
100	40,857	180,916	140,059
150	42,973	197,478	154,505
200	49,312	210,191	160,879
250	51,676	225,124	173,448
300	54,448	241,834	187,386
350	56,134	255,369	199,235
400	57,812	270,604	212,792
450	60,678	286,971	226,293
500	62,902	321,173	258,271

4.1 Analysis of our Learning Method

Figure 3 shows the distribution of weights connected from fully-connected layer1 to fully-connected layer2 of LeNet-5 and AlexNet, respectively. Since the distribution of weight has a bell-shape, the qualitative entropy is calculated using Equation (6) resulting in Figure 4.

Figure 4(a) and 4(b) show the entropy and the amount of information obtained by using Equations (1) and (2) at a node, in which the qualitative property is not applied yet. Figure 4(c) and 4(d) show the qualitative entropy and the qualitative amount of information obtained by using Equations (4) and (6) at a node.

As shown in Figure 4 (a) and (b), more than 95% of the weights is selected because the weights with smaller amount of information than entropy are the majority. In this case, the weights with high probability can be misunderstood as unimportant to the outputs because Equation (1) and (2) consider the probability distribution only, regardless of the magnitude of weights. In other words, most of weights are selected as not retraining for the new cases. The entropy of (a) and (b) is almost equal to 1.45 because the probability distribution of both networks are almost the same as the Gaussian distribution as shown in Figure 3. On the other hand, the qualitative entropy of (c) and (d) in Figure 4 is 0.015 and 0.09, respectively. Although the probability distributions of (c) and (d) are almost same, unlike (a) and (b), the weights are properly divided by the qualitative entropy as seen in Table 1 and Table 2.

Table 1 shows the number of weights selected for each layer in LeNet-5. The selected weight in fully-connected layer1 and fully-connected layer2 is 272,195 out of 400,000, 3634 out of 5000 respectively. As the result, 68.10% of the weights in the fully-connected layer is selected. Table 2 also shows the number of weights selected for each layer in AlexNet. The selected weights in fully-connected layer1, fully-connected layer2, and fully-connected layer3 is 23,943,187 out of 38,000,000, 11,053,214 out of 17,000,000, and 2,800,745 out of 4,000,000 respectively. As the result, 64.06% of the total weights of the fully-connected layer is selected. As the size of network is bigger, the selected weight ratio is smaller because Gaussian distribution is getting close to bell-shape by the central limit theorem.

From the following section, we show the performance of partial learning that trains the weights selected by qualitative entropy.

4.2 Performance of Partial Learning

Table 3 shows the performance of classification accuracy of the partial learning by adding a new class, using MNIST. First, five classes are trained using LeNet-5 as an initial network structure.

The classification accuracy of the initial network is 99.20%. From the structure, we analyze the performance by adding one class at a time. The accuracy results in 99.06% (total of six classes), 98.25% (total of seven classes), 97.61% (total of eight classes), 93.19% (total of nine classes), and 89.74% (total of ten classes), respectively, as new class is added at a time. When a new class is added, there is almost no accuracy difference between the six-class network

Table 7 The Number of selected partial learning weight and FLOPs on embedded device.

Network	Layer	Selected parameters	Total	Initial	Our(FLOPs)
LeNet-5	Fully-connected1	232,195	235,229	810 K	470.5 K
	Fully-connected2	3034			
AlexNet	Fully-connected1	17,523,673	8,844,199	117 M	51.2 M
	Fully-connected2	6,112,835			
	Fully-connected3	2,175,981			

Table 8 Training time performance of partial learning and transfer learning for LeNet-5 on embedded device.

Additional class numbers	Partial learning(s)	Transfer learning(s)	Gap
1	1121s	6309 s	5188 s
2	1328s	7356 s	6028 s
3	1404s	8251 s	6847 s

structure and the original network structure. When two and three new classes are added, the accuracy difference is less than 1% and 1.6%, respectively. From adding the fourth new class, the performance degrades because the network runs out of information resources. The network should adopt new classes as well as keep the performance of the existing ones.

For ImageNet dataset, as shown in Table 4, the accuracy is 62.8% when the 500 classes are trained using AlexNet as an initial network structure. Again, we analyze the performance by adding one class at a time. For the classes 500 to 700, the accuracies result in 62.2% (total of 550 classes), 61.5% (total of 600 classes), 61.4% (total of 650 classes), 61.0% (total of 700 classes), where there is almost no accuracy degradation. When 250 and 300 classes are added to the network, the accuracies are 59.5% (total of 750 classes) and 56.3% (total of 800 classes) with accuracy loss of 3.3% and 6.5%, respectively. From 350 new classes, the performance starts degrading gradually.

Through the experiment, the partial learning gives better performance than the transfer learning when adding up to three new classes to the existing network. Our method can be acceptable for the partial retraining about 40% of additional new classes from an existing network.

Table 5 shows the training time required for partial learning and transfer learning on LeNet-5 using MNIST. The time gaps between our method and transfer learning are 207 s, 241 s, 274 s, 309 s, and 348 s, respectively, as adding a new class to the existing network up to five new classes. The time gap increases as the network size gets bigger because the partial learning gradually effects on reduction of the computational complexity. For the transfer learning, the training time increases as new classes are added due to the increase in the size of the network structure. As the network gets bigger, the number of weights increases exponentially. Table 6 shows the training time performance for AlexNet trained on ImageNet dataset. As in Table 5, the time gap increases linearly because the time required for learning is usually determined by the number of data. This experiment takes more time than previous because the size of AlexNet is 140× bigger than LeNet-5 and ImageNet dataset is more complex than MNIST. In larger networks, the difference can be even greater. Tables 5 and Tables 6 show how unnecessary computations exist for traditional transfer learning because it initializes all the information of the weight in the network and relearns upon addition of a new class to the learned network. The tables show our partial learning technique reduces computational overhead.

4.3 Analyzing Embedded Memory

Table 7 shows the selected weights for partial learning on embedded device and the FLOPs. For LeNet-5, there are 232,195 selected parameters in the Fully-connected1 and 3034 in Fully-connected2, with a total of 235,229 weights. As a result, the computational cost decreases for 1.7× due to reducing the existing computational resource of 810 K FLOPs to 470.5 K FLOPs. Whereas for AlexNet, 17,523,673, 6,112,835 and 2,175,981 are selected from Fully-connected1, Fully-connected2, and Fully-connected3, respectively, with a total of 8,844,199 weights is partially learned. As a result, the computational resource of 117 M FLOPs is reduced to 51.2 M FLOPs, which improves the computational performance by 2.3 × .

To test the performance of partial learning on embedded devices, experiments made in Table 5 and Table 6 are repeated for Jetson TX1. Table 8 shows the results training time performance of Lenet-5 on embedded devices. When added up to 3 classes, partial learning takes 1121s, 1328s, and 1404s, respectively. Our method is faster than transfer learning by 5188 s, 6028 s, and 6847 s, respectively. Similarly, Table 9 shows the results of Table 6 performed on embedded devices. The time gaps between our method and the transfer learning are 126,854 s and 140,057 s, respectively. Our experiments are limited to three cases for Table 8 and two cases for Table 9 because Titan X Pascal GPU has an FP32 performance of 12.15 TFLOPs, while Jetson TX1 has only 500 GFLOPs. Therefore, there is a performance difference of about 25×. As seen in experiments, the more complex are the data and the network, the more effective is the partial learning technique.

Table 9 Training time performance of partial learning and transfer learning for AlexNet on embedded device.

Additional class numbers	Partial learning(s)	Transfer learning(s)	Gap
50	126,828	538,957	412,129
100	132,785	587,977	455,192

5 Conclusion

In this paper, we proposed the on-device qualitative entropy-based partial learning method to adapt the new class to an existing network. We derived the qualitative entropy metric to select the weights to be trained for new classes by employing and modifying the entropy concept. In order to derive a mathematical formula for the metric, we assumed that the statistical distribution of weights is a Gaussian. The derivation was achieved by considering not only the probabilistic information but qualitative characteristics of the weights. As shown in the experimental section, the existing network is partially trained based on the qualitative entropy metric and it outperforms the existing transfer learning with no loss of accuracy in terms of learning cost and embedded memory.

Even though our method achieved good performance compared to the existing method, there is a further work. We need to improve our technology by analyzing optimizing algorithm so that it can more practically be used on the mobile device. The other is to generalize our methodology by extending our technology to other types of learning networks such as recurrent neural network and generative adversarial nets.

Acknowledgements This work was supported by Inha University Grant.

References

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

2 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ..., Rabinovich, A. (2015, June). Going deeper with convolutions. Cvpr.

3 He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

4 Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*.

5 Denil, M., Shakibi, B., Dinh, L., & De Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems* (pp. 2148-2156).

6 Ye, J. (2005). Generalized low rank approximations of matrices. *Machine Learning, 61*(1–3), 167–191.

7 Denil, M., Shakibi, B., Dinh, L., & De Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems* (pp. 2148-2156).

8 Yu, D., & Deng, L. (2011). Deep learning and its applications to signal and information processing [exploratory dsp]. *IEEE Signal Processing Magazine, 28*(1), 145–154.

9 Cheng, J., Wu, J., Leng, C., Wang, Y., & Hu, Q. (2017). Quantized CNN: A unified approach to accelerate and compress convolutional networks. *IEEE Transactions on Neural Networks and Learning Systems*.

10 Schneider, P., Biehl, M., & Hammer, B. (2009). Adaptive relevance matrices in learning vector quantization. *Neural Computation, 21*(12), 3532–3561.

11 Polyak, A., & Wolf, L. (2015). Channel-level acceleration of deep face representations. *IEEE Access, 3*, 2163–2175.

12 Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems* (pp. 1135-1143).

13 Machida, H., Yoneda, H., & Kanno, H. (1992). *U.S. Patent No. 5, 109,436*. Washington, DC: U.S. Patent and Trademark Office.

14 Baier, A., & Baier, P. W. (1983). Digital matched filtering of arbitrary spread-spectrum waveforms using correlators with binary quantization. In *Military Communications Conference, 1983. MILCOM 1983. IEEE* (Vol. 2, pp. 418-423). IEEE.

15 Yuan, Z. X., Xu, B. L., & Yu, C. Z. (1999). Binary quantization of feature vectors for robust text-independent speaker identification. *IEEE Transactions on Speech and Audio Processing, 7*(1), 70–78.

16 Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

17 Sivaswamy, J., Krishnadas, S. R., Joshi, G. D., Jain, M., & Tabish, A. U. S. (2014, April). Drishti-gs: Retinal image dataset for optic nerve head (onh) segmentation. In *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on* (pp. 53-56). IEEE. http://cvit.iiit.ac.in/projects/mip/drishti-gs/mip-dataset2/Download.php

18 Harris, B., Moghaddam, M. S., Kang, D., Bae, I., Kim, E., Min, H., ... & Choi, K. (2018, January). Architectures and algorithms for user customization of CNNs. In *Proceedings of the 23rd Asia and South Pacific* Design Automation Conference (pp. 540–547). IEEE Press.

19 Killian, T. W., Daulton, S., Konidaris, G., & Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden parameter markov decision processes. In advances in neural information processing systems (pp. 6250-6261).

20 Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging, 35*(5), 1285–1298.

21 Fernandes, K., Cardoso, J. S., & Fernandes, J. (2017). Transfer learning with partial observability applied to cervical cancer screening. In Iberian conference on pattern recognition and image analysis (pp. 243-250). Springer, Cham.

22 Xu, S., Mu, X., Chai, D., & Wang, S. (2017). Adapting remote sensing to new domain with ELM parameter transfer. *IEEE Geoscience and Remote Sensing Letters, 14*(9), 1618–1622.

23 Afridi, M. J., Ross, A., & Shapiro, E. M. (2018). On automated source selection for transfer learning in convolutional neural networks. *Pattern Recognition, 73*, 65–75.

24 Peng, X., Sun, B., Ali, K., & Saenko, K. (2015). Learning deep object detectors from 3d models. In proceedings of the IEEE international conference on computer vision (pp. 1278-1286).

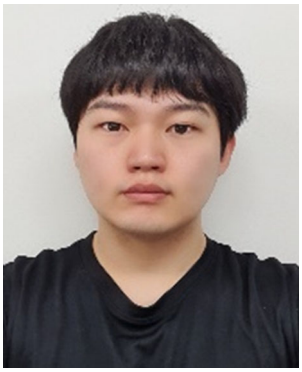
25 Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In advances in neural information processing systems (pp. 136-144).

26 Su, H., Qi, C. R., Li, Y., & Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In proceedings of the IEEE international conference on computer vision (pp. 2686-2694).

27 Tjandra, A., Sakti, S., & Nakamura, S. (2017). Attention-based wav2text with feature transfer learning. In 2017 IEEE automatic speech recognition and understanding workshop (ASRU) (pp. 309-315). IEEE.

28 Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes.

In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1912-1920).
29 Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In proceedings of the IEEE international conference on computer vision (pp. 945-953).
30 Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).
31 Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5648-5656).
32 Kalogerakis, E., Averkiou, M., Maji, S., & Chaudhuri, S. (2017). 3D shape segmentation with projective convolutional networks. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3779-3788).
33 Song, S., Lichtenberg, S. P., & Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 567-576).
34 Lindblad, G. (1973). Entropy, information and quantum measurements. *Communications in Mathematical Physics, 33*(4), 305–322.
35 Föllmer, H. (1973). On entropy and information gain in random fields. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete, 26*(3), 207–217.
36 Borland, L., Plastino, A. R., & Tsallis, C. (1998). Information gain within nonextensive thermostatistics. *Journal of Mathematical Physics, 39*(12), 6490–6501.
37 Nalewajski, R. F. (2005). Partial communication channels of molecular fragments and their entropy/information indices. *Molecular Physics, 103*(4), 451–470.
38 Huerta, M. A., & Robertson, H. S. (1969). Entropy, information theory, and the approach to equilibrium of coupled harmonic oscillator systems. *Journal of Statistical Physics, 1*(3), 393–414.
39 Ebeling, W. (1993). Entropy and information in processes of self-organization: Uncertainty and predictability. *Physica A: Statistical Mechanics and its Applications, 194*(1–4), 563–575.
40 LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2.
41 Mallard, W. G., Westley, F., Herron, J. T., Hampson, R. F., & Frizzell, D. H. (1998). *NIST chemical kinetics database, version 2Q98*. Gaithersburg: National Institute of Standards and Technology. Web address: http://kinetics.nist.gov.
42. Lei, H., Han, T., Zhou, F., Yu, Z., Qin, J., Elazab, A., & Lei, B. (2018). A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning. *Pattern Recognition, 79*, 290–302.
43. Fadaeddini, A., Eshghi, M., & Majidi, B. (2018). A deep residual neural network for low altitude remote sensing image classification. In 2018 6th Iranian joint congress on fuzzy and intelligent systems (CFIS) (pp. 43-46). IEEE.
44. McCallum, A. 20 newsgroups. (2008). http://people.cs.umass.edu/~mccallum/data-/20_newsgroups.tar.gz
45. McCallum, A. SRAA. (2008) http://people.cs.umass.edu/~mccallum/data/sraa.tar.gz
46. Lewis, David, et al. Reuters-21578. *Test Collections*, (1987) http://www.daviddlewis.com/resour-ces/testcollections/reuters21578/
47. Voutilainen, A. (2003). Part-of-speech tagging. *The Oxford handbook of computational linguistics*, 219–232.
48. Mohit, B. (2014). *Named entity recognition*, *In Natural language processing of semitic languages (pp. 221–245)*. Berlin, Heidelberg: Springer.
49. Blanco, E., Castell, N., & Moldovan, D. I. (2008, May). Causal Relation Extraction. In *Lrec*.
50. Björkelund, A., Hafdell, L., & Nugues, P. (2009). Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task* (pp. 43-48). Association for Computational Linguistics.
51. Gupta, S., & Malik, J. (2015). Visual semantic role labeling. *arXiv preprint arXiv:1505.04474*.
52. Bereziński, P., Jasiul, B., & Szpyrka, M. (2015). An entropy-based network anomaly detection method. *Entropy, 17*(4), 2367–2408.
53. Han, B., Zhang, Z., Xu, C., Wang, B., Hu, G., Bai, L., ... & Hancock, E. R. (2017). Deep Face Model Compression Using Entropy-Based Filter Selection. In *International Conference on Image Analysis and Processing* (pp. 127–136). Springer, Cham.
54. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248-255). IEEE. http://image-net.org/download-images
55. Park, E., Ahn, J., & Yoo, S. (2017). Weighted-entropy-based quantization for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
56. Zilly, J., Buhmann, J. M., & Mahapatra, D. (2017). Glaucoma detection using entropy sampling and ensemble learning for automatic optic cup and disc segmentation. *Computerized Medical Imaging and Graphics, 55*, 28–41.
57. Mjalli, F. S., Al-Asheh, S., & Alfadala, H. E. (2007). Use of artificial neural network black-box modeling for the prediction of wastewater treatment plants performance. *Journal of Environmental Management, 83*(3), 329–338.
58. Hoeffding, W., & Robbins, H. (1948). The central limit theorem for dependent random variables. *Duke Mathematical Journal, 15*(3), 773–780.
59. LeCun, Y. (2015). LeNet-5, convolutional neural networks. URL: http://yann.lecun.com/exdb/lenet, 20.
60. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265–283).



Cheongwhan Hur received the B.S and M.S. degrees in Computer Engineering from INHA University, South Korea in 2017 and 2019, respectively. He is currently pursuing a Ph.D. degree in Computer Engineering at Inha University. His research interests include Deep Learning, Neuroscience, Novel Learning Method, etc.

Sanggil Kang received the M.S. and Ph.D. degrees in Electrical Engineering from Columbia University and Syracuse University, USA in 1995 and 2002, respectively. He is currently an associate Professor in the Department of Computer Engineering at INHA University, Korea. His research interests include Machine Learning, Semantic Web, Artificial Intelligence, Multimedia Systems, Inference Systems, etc.