IEICE TRANS. INF. & SYST., VOL.E103–D, NO.10 OCTOBER 2020

PAPER

Construction of an Efficient Divided/Distributed Neural Network Model Using Edge Computing

Ryuta SHINGAI[†a)], **Yuria HIRAGA**[†], **Hisakazu FUKUOKA**[†], **Takamasa MITANI**[†], ***Student Members***, **Takashi NAKADA**[†], ***Member***, ***and*** **Yasuhiko NAKASHIMA**[†], ***Fellow***

SUMMARY Modern deep learning has significantly improved performance and has been used in a wide variety of applications. Since the amount of computation required for the inference process of the neural network is large, it is processed not by the data acquisition location like a surveillance camera but by the server with abundant computing power installed in the data center. Edge computing is getting considerable attention to solve this problem. However, edge computing can provide limited computation resources. Therefore, we assumed a divided/distributed neural network model using both the edge device and the server. By processing part of the convolution layer on edge, the amount of communication becomes smaller than that of the sensor data. In this paper, we have evaluated AlexNet and the other eight models on the distributed environment and estimated FPS values with Wi-Fi, 3G, and 5G communication. To reduce communication costs, we also introduced the compression process before communication. This compression may degrade the object recognition accuracy. As necessary conditions, we set FPS to 30 or faster and object recognition accuracy to 69.7% or higher. This value is determined based on that of an approximation model that binarizes the activation of Neural Network. We constructed performance and energy models to find the optimal configuration that consumes minimum energy while satisfying the necessary conditions. Through the comprehensive evaluation, we found that the optimal configurations of all nine models. For small models, such as AlexNet, processing entire models in the edge was the best. On the other hand, for huge models, such as VGG16, processing entire models in the server was the best. For medium-size models, the distributed models were good candidates. We confirmed that our model found the most energy efficient configuration while satisfying FPS and accuracy requirements, and the distributed models successfully reduced the energy consumption up to 48.6%, and 6.6% on average. We also found that HEVC compression is important before transferring the input data or the feature data between the distributed inference processes.

key words: *convolutional neural network, edge computing, distributed neural network, video compression*

1. Introduction

Recently, machine learning has been progressed and integrated into many applications, such as object detection and pattern matching. These applications are used in a wide variety of situations. Automatic operation is an example. Deep Convolutional Neural Network (DCNN) achieved better recognition accuracy than that of human capability (5.1%) [1].

Deep Convolutional Neural Network (DCNN) consists of 2 main functionalities, feature extractor, which is implemented by a combination of convolutional and pooling layers, and classification, which is implemented by fully-connected layers. DCNN achieved 10% or more improvement of the recognition accuracy than conventional techniques. However, DCNN commonly needs a large amount of calculation. Due to enormous computational costs, mobile devices may not meet their real-time requirements. Therefore, the calculation of DCNN is usually processed on servers, having a huge amount of computing resources.

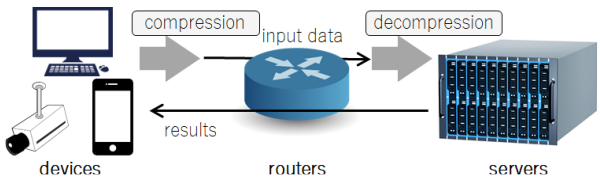


Fig. 1 Inference process in DCNN using servers

Here are two famous architectures of DCNN: AlexNet [2] and VGG16 [3]. Both are excellent Neural Network for object recognition. AlexNet is composed of 5 convolutional layers, 3 pooling layers, and 3 fully-connected layers. The VGG16, which is deeper than AlexNet, is composed of 13 convolutional layers, 5 pooling layers, and 3 fully-connected layers. Image recognition using DCNN is implemented, as shown in Fig. 1 and processes as follows.

1. The sensors get the data, compress, and send them to the server.
2. The server receives and decompresses the data and executes the inference process.
3. The server sends the results back to the user.

However, recently proposed models of DCNN became much deeper and bigger than previous models. They require huge computation power. In the conventional method, all of DCNN are processed on servers, which have computing resources in abundance. As the size of DCNN grows and IoT devices increase [4], traffic congestion, and the power consumption of network switches and servers are drastically increased. To cope with this overconcentration, other computational infrastructures that are independent of the cloud server are required.

As a solution to these problems, many researchers and companies recently have paid great attention to edge computing [5], [6] to process the data near their sources. So, we introduce that the process of a part of the neural network model could be executed by using edge computing.

Manuscript received December 14, 2019.
Manuscript revised May 18, 2020.
Manuscript publicized July 2, 2020.
[†]The authors are with Nara Institutet of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: shingai.ryuta.sm0@is.naist.jp
DOI: 10.1587/transinf.2019EDP7326

Our target is to implement the system to process machine learning in the divided/distributed neural network model. Since we need to find which optimal split boundary from the viewpoints of FPS, object recognition accuracy, and energy consumption, we did a preliminary evaluation. Firstly, we confirm the results when applying HEVC compression to intermediate data of each layer, and then summarize the indicators of the energy consumption in communication and inference process. Finally, in the comprehensive evaluation, we evaluate which split layer is optimal and whether compression is necessary.

The rest of this paper is organized as follows. Section 2 summarizes related work on approximation models, edge computing, and applications that require the object recognition task. Section 3 shows preliminary evaluations to find the optimal split boundary. Section 4 indicates a method of applying compression to the intermediate data. Section 5 shows the relationship between compression and recognition accuracy, and comprehensive evaluation. Section 6 shows the experimental results. A conclusion is described in Sect. 7.

2. Related Works

2.1 Approximate Computing for CNNs

As Neural Networks evolve, a recognition based on CNN needs a large amount of memory and computational power. For that reason, while they work well on expensive machines, including high-end GPU, they are often unsuitable for smaller devices like cell phones and embedded devices. Recently, the construction of a neural network with reduced calculation cost and an equivalent model scale has been developed. Lightweight CNNs reduced precision computation have been developed. In Binary-Weight-Networks [7], [8], all the weight values are approximated with binary values. A convolutional neural network with binary weights is smaller (1/32×) than the same network with single-precision floating-point weight values. Since weight values are binary, convolutions can be performed by only addition and subtraction (without multiplication), resulting in more than 2× speed up. Also, XNOR-Nets [8] approximates both the weights and the inputs in the convolutional and fully connected layers to binary values. Binary weights and binary inputs allow an efficient calculation of convolutional operations. Since all of the operands of the convolutions are binary, the convolutions can be performed by XNOR and Bitcounting operations [7].

Ternary Weight Networks (TWNs) [9] is a model compression method that has neural networks with weights constrained to +1, 0, and −1. TWNs have a more precise expression than the Binary-Weight-Networks. TWNs achieved higher accuracy than that of the Binary-Weight-Network. However, it is still lower than that of the full precision models. Like XNOR-Nets, there is a Neural Network model replacing floating-point matrix multiplications with ternary convolutions (based on sparse binary kernels), with both activations and weights restricted to values of {−1, 0, +1} [10]. This model can be calculated using a masked Hamming distance, an XOR/XNOR operation followed by a popcount, and reduce the computational cost by up to 1/16. This approach does not require high computational performance but needs exploring the theoretical advantages of explicit sparsity promotion to reduce the risk of overfitting. To solve these problems, we propose a faster, more accurate, and more efficient distributed model without excessive precision reduction.

2.2 Distributed DCNN

To reduce computation costs, BranchyNet [11] have adopted early exits from additional side branches. BranchyNet allows exiting the network early via these branches when results are already inferred with high confidence. The original longest path is only used when any early exits are not confident.

Based on this idea, Distributed Deep Neural Network [12] have been proposed. Only early exits are implemented on the edge device, and the rest of the longest path is implemented on the server. If the ratio of early exit is high enough, both computation and communication costs can be reduced. However, additional training is required to make early exits. Moreover, data compression is not considered at all.

In contrast to these ideas, since our idea is independent of the neural network models, any additional training is not required. The video compression is effective for any model whose input is time-series data.

2.3 Edge Computing

Owing to a huge amount of computing resources, DCNN is usually processed in the cloud server. However, rapidly increasing demand brings traffic congestion and overconcentration of power consumption. To deal with these problems, edge computing [5], [6] has been recently paid great attention.

We propose processing the inference process of DCNN within this edge device as much as possible. When most processes of DCNN is executed within the edge device, traffic congestion and overconcentration in the cloud server are resolved.

Additionally, it is possible to process with lower power consumption than the process on the server. Recently, Jetson TX2 [13] and Jetson Nano [14] developed by NVIDIA are good candidates for edge computing. Jetson TX2 and nano are modular AI supercomputers with NVIDIA Pascal and Maxwell architectures, respectively. These devices can execute the inference process with low energy consumption.

2.4 Possible Applications

One of the applications that we assumed, for example, is automobiles detecting the objects used the model on DCNN

that had been learned for well detecting the passersby and the other automobiles. The datasets they have been evaluated upon [15]–[17] have been targeted at the approximate frame rate of 30 Frames Per Second (FPS). Therefore, a processing speed of 30 FPS is required for the implementation of a practical application. By eliminating the learning phase, it is possible to process the inference within the distributed neural network model. In this paper, we focus on the inference phase of DCNN.

Also, the object detection used in automobiles is considered to be insufficient for future implementation in a 4G environment with the delay of 4G. Therefore, it is expected that a communication band (5G [18]) with high-speed communication will be installed.

3. Preliminary Evaluation

Figure 1 shows the inference process of DCNN on the server. Conventional implementation causes problems of network congestion and overconcentration of the calculation load on the server, as described in Sect. 1. We split the layers of DCNN into two and assign them to multiple devices. That is, we introduce a model divided/distributed method, which calculates DCNN more efficiently.

We explain the flow of the process of divided DCNN using edge computing. When DCNN is implemented on two devices, edge device processes data from the input data to a certain layer in DCNN, and the other device processes the remained layers. The input size is 224×224, and input bit-depth is 8 for each RGB color.

Table 1 shows the average execution times of all models on a server GPU. The calculation time of resnet152, which is the largest, is 16.4 times larger than that of AlexNet, which is the smallest. This causes the overconcentration of computation. In order to balance the amount of computation, the split boundary should be the middle of the layers.

As split boundaries, 1) we mainly chose after pooling layers, which reduce the amount of data largely. Additionally, in some models, there are branch and marge in the data flow. If we divide the models during branched computation, we should transfer multiple data streams. Therefore, 2) we should not divide during branched computation. According to these policies, we divided the models as follows.

For VGG16 and GoogLeNet, the split boundary is selected from after 5 and 4 pooling layers, respectively. For AlexNet, it is selected from after 5 convolution layers. For MobileNet, it is selected from after 7 inverted residual blocks. For SqueezeNet and ResNet, it is selected from 8 fire modules and 4 residual blocks and before the first of these modules or blocks, respectively.

If a pooling layer follows these boundaries, the boundaries are moved to just after the pooling layer, according to the first policy.

Figure 2 shows the amount of output of each boundary and amount of computation before and after each boundary. In addition to AlexNet and VGG16, we also evaluate

Table 1 Calculation time on a server GPU

model	calculation time [ms/frame]
alexnet	0.300
vgg16	3.864
mobilenet	0.941
squeezenet	0.653
googlenet	1.112
resnet18	0.749
resnet50	2.016
resnet101	3.435
resnet152	4.925

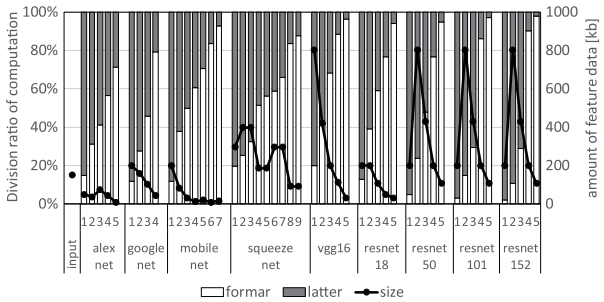


Fig. 2 Division ratio of computation (formar and latter) and amount of output (size) on each split boundary

GoogLeNet, MobileNet v2, SqueezeNet, and ResNet18, 50, 101, and 152. For example, if we choose the split boundary 2 on AlexNet, the computation costs can be divided by 3:7.

4. How to Apply Compression

For practical applications of image recognition in the IoT era, a popular type of input data is images that are continuously captured (i.e., video) rather than many independent images. In the technical competitions, huge numbers and a wide variety of pictures are suitable for measuring the average recognition accuracy. However, this kind of dataset is too far from reality. In this paper, for practical uses, we treat the video stream obtained from surveillance or object inspection cameras.

Since DCNN commonly requires a huge amount of computation, it is hard for a single device on the edge device to process them. At the same time, the divided approach requires network communication between devices. This overhead is not negligible.

The conventional implementation of DCNN requires data transmission from the sensors to the server. If the size of data is large and the communication costs are not negligible, the total performance is decreased. In many cases, we can observe that as the amount of edge computing is increased, the amount of data of the former layer is decreased. Note that there are some exceptions, such as split boundaries 2 and 3 of ResNet 50, 101, and 152. At these boundaries, the amount of data is significantly larger than that of input data or split boundary 1. Therefore, these boundaries may not be suitable for distributed execution.

In previous research [19], H.265/HEVC compression [20] was adopted as a compression method for inter-

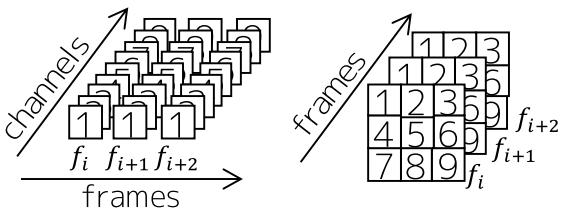


Fig. 3 Frame composition from multi-channel image

Table 2 Execution enviroment

GPU (tx2)	Jetson TX2 (NVIDIA Pascal, 256 CUDA Cores)
GPU (nano)	Jetson Nano (NVIDIA Maxwell, 128 CUDA Cores)
GPU (server)	GTX1080TiOC (NVIDIA Pascal, 3584 CUDA Cores)
input video	224×224

mediate data of the neural network model. We thought that HEVC is optimal in this experiment because it achieves a high video compression rate. HEVC compression has a minimum CTU size, which is 16×16. The output data from the pooling layer in the latter half of AlexNet or some models are smaller than that, so it cannot be compressed directly. Therefore, we compose multiple channel images by tiling, as shown in Fig. 3 in order to fit the input of HEVC compression.

The higher the recognition accuracy, the higher the reliability of the user as an actual application. In the previous study, only AlexNet was targeted, but in this experiment, the other eight models with a large number of layers are also evaluated. By implementing a divided/distributed neural network model, the overall energy consumption is expected to be lower than with a conventional way of processing the entire inference process on the server.

5. Experimental Setup

In this experiment, we applied HEVC compression as a video compression method. And, AlexNet and GoogLeNet, MobileNet v2, SqueezeNet, VGG16, and resnet18, 50, 101, and 152 are adopted as the distributed neural network models. We investigate the relationship among the FPS, object recognition accuracy, and the energy consumption in divided/distributed neural network models. We validated a comprehensive evaluation based on the inference process, the encoding/decoding process of the compress method, and communication.

5.1 Object Recognition and Compression Method

We investigate the efficiency of the video compression for the feature data extracted from the convolution layers of DCNN and the trade-off between the compression ratio and recognition accuracy. At first, Intermediate data of DCNN is extracted by the executing former layer. The first half of the model processed on the edge device. After encoding, transferring, and decoding, the server continues with the inference process and executes the object recognition task.

We used a caffe Model Zoo [21] via torchvision as the trained models of AlexNet and the other eight models in the experiment. These models are trained for the task of classifying 1000 classes using the ImageNet data set published in the ILSVRC competition. As the verification data, we used the video captured by a fixed-point camera of elephants, jaguar, buffalo, zebra, and panda. These videos have 622, 520, 1256, 590, and 429 frames, respectively. There are three labels classified as elephants in the training dataset, "African elephant," "Indian elephant," and "tusker." We evaluated the accuracy when one of the three labels was included in Top5. For the other four animals, only one label exists for each animal. Unfortunately, some models cannot recognize some animals. For example, with VGG16, the recognition accuracy of pandas is less than 30% without any split nor compression. Therefore, such combinations are excluded before evaluation.

First of all, we compared these models constructed as distributed neural network models. For the distributed model, we applied H.265/HEVC to intermediate data extracted from the former layers. HEVC compression has a Quantization parameter (Qp). It manages the trade-off between compression ratio and image quality. HEVC has another parameter that specifies the GOP width of motion compensation (interframe prediction). The GOP is a group unit composed of multiple consecutive frames (I picture, P picture, B picture), and I picture is placed in the first frame. In this experiment, the GOP width is set to 30 frames (I: 1, P: 29, B: 0). NvEnc encoder does not support creation for B pictures in many GPUs to realize real-time encoding. Therefore, the compressed data is composed of I pictures and P pictures. In the distributed model, we evaluated the compression ratio and object recognition accuracy. Also, the details of the execution environment are described in Table 2. As edge GPUs, we evaluated with Jetson TX2 and Jetson Nano.

5.2 Comprehensive Evaluation

Based on the above experiments, we evaluate the overall distributed neural network models. In the distributed neural network models, the following steps proceed step by step (1) extraction of intermediate data, (2) compression process, (3) communication, (4) decompression process, and (5) rest of the inference process. We assumed that the five processes of the distributed model are performed by pipeline processing.

In this experiment, the HEVC compression/decompression process is executed by a server GPU and edge GPU (tx2 and nano). NvEnc [22] is adopted to use hardware encoder integrated with these GPUs. In the overall evaluation, we used all the processing time and the power consumption for the distributed model. The energy consumption includes the total energy consumption per frame for the inference process on the edge device and the server, the compression/decompression process, and communication.

Table 3 Energy consumption in communication

Communication standard	Energy Consumption [J/bit]
Wi-Fi (40M [bps])	5.3×10^{-9}
3G (1M [bps])	3.5×10^{-7}
5G (50M [bps])	6.7×10^{-8}

In communication, we assumed Wi-Fi (40M[bps]) [23], 3G (1M[bps]) [23], and 5G (50M[bps]) as the effective communication speed of the channel.

Our requirements are setting to 30 FPS or higher for the processing speed and to 69.7% or higher achieved by XNOR-Nets, for object recognition accuracy.

5.3 Performance and Energy Models

Here, the calculation method of FPS and energy consumption used in a comprehensive evaluation is described. In the inference process, t_{former} is the processing time executed on the GPU of the edge device, and t_{latter} is the processing time executed on the GPU of the server. The HEVC compression time is t_{comp}, and the decompression time is t_{decomp}. We measured these values with real equipment. Also, the processing time required for communication is obtained by Eq. (1).

$$t_{comm} = size/throughput \tag{1}$$

We assume these five processes can be done in parallel by pipeline manner. Therefore, Eq. (2) shows how to calculate FPS.

$$FPS = 1/\max(t_{former}, t_{comp}, t_{comm}, t_{decomp}, t_{latter}) \tag{2}$$

Finally, the measurement method of energy consumption is described below. The energy consumption J for the inference process and the compression/decompression process is calculated by Eq. (3),

$$J = P \cdot t \tag{3}$$

which is a product of the power consumption P of GPU and the processing time t. We measured voltage and current of power lines with a voltmeter and an ammeter. Table 3 summarizes the energy consumption for each communication standard in a J/bit. The energy consumption for communication is calculated by multiplication of the values in Table 3 and the size of the intermediate data.

6. Experimental Result

6.1 Distributed Models Using H.265/HEVC

Table 4 shows the raw and compressed data size when HEVC is applied to each split boundary. raw is not applied any compression. Thus, the compression ratio is 1.0. 8bit is applied 8bit quantization for 32 bit data. Thus, the compression ratio is 0.25. qpX indicates the compressed data size when HEVC compression is applied with the Quantization parameter (Qp) value is X. As qp value is increased, a higher compression ratio is realized.

Table 4 Compression ratio

compression		compression ratio
raw		1.000
8bit		0.250
hevc	qp0	0.624
	qp10	0.376
	qp20	0.173
	qp25	0.0982
	qp30	0.0421
	qp35	0.0148
	qp40	0.00385
	qp45	0.00201
	qp50	0.00104

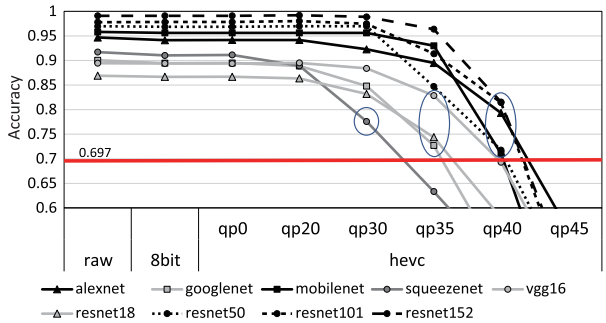


Fig. 4 Compressions and accuracy

Figure 4 shows the object recognition accuracy when HEVC is applied and not applied to each split boundary. The X-axis represents the compression method and Qps for HEVC. Each line chart represents each model. This result clearly shows that the larger Qp significantly degraded recognition accuracy. The horizontal red line shows the target accuracy of 69.7%.

In many models, the recognition accuracy is maintained up to Qp20, but the accuracy decreases sharply when Qp is 35 or more. However, for example, when HEVC Qp40 is applied to the intermediate data of AlexNet, we found that the recognition accuracy is still higher than 69.7%.

From these results, we found that the highest compression while maintaining recognition accuracy is higher than 69.7% is Qp30 for SqueezeNet, Qp35 for GoogLeNet, VGG16, and ResNet18, Qp40 for AlexNet, MobileNet, ResNet50, 101, and 152. With these compressions, the amount of data becomes 4.21% (Qp30) to 0.2% (Qp40) of original data. These compression methods are marked with ovals in the figure. When comparing ResNet18, 50, 101, and 152, the larger model achieved higher accuracy. Consequently, when target accuracy is fixed, higher compression can be applied for the larger model.

6.2 Energy Consumption for Inference, Compression and Communication

Table 5 shows the processing time and the energy consumption of GPU in the edge device and the server used in AlexNet. Table 6 shows those of GPU in the edge device

Table 5 Inference processing time & energy consumption of the GPU for AlexNet

split boundary	Time [ms/frame]		Energy [mJ/frame]	
	Edge	Server	Edge	Server
server	-	0.300	-	72.0
1	0.77	0.253	6.2	60.8
2	1.65	0.178	13.3	42.7
3	2.20	0.162	17.7	38.8
4	2.78	0.117	22.4	28.0
5	3.26	0.071	26.3	17.2
edge	4.29	-	34.6	-

Table 6 Inference processing time & energy consumption of GPU for VGG16

split boundary	Time [ms/frame]		Energy [mJ/frame]	
	Edge	Server	Edge	Server
server	-	3.86	-	927
1	14.2	3.49	114	838
2	25.2	2.21	203	531
3	42.0	1.22	339	293
4	58.3	0.44	470	106
5	63.9	0.14	515	33
edge	65.5	-	528	-

Table 7 Processing time & energy consumption with compression/ decompression for AlexNet

split boundary	Time [ms/frame]		Energy [mJ/frame]	
	Comp	Decomp	Comp	Decomp
input	1.40	1.53	1.34	1.53
1	1.63	1.07	1.57	1.07
2	1.64	0.71	1.57	0.71
3	2.18	0.99	2.10	0.99
4	1.89	0.85	1.82	0.85
5	1.63	0.28	1.56	0.28

Table 8 Processing time & energy consumption with compression/ decompression for VGG16

split boundary	Time [ms/frame]		Energy [mJ/frame]	
	Comp	Decomp	Comp	Decomp
input	1.40	1.53	1.34	1.53
1	13.54	4.42	13.00	4.42
2	7.49	4.03	7.19	4.03
3	3.48	2.69	3.34	2.69
4	2.33	1.53	2.24	1.53
5	1.63	0.62	1.56	0.62

and the server used in VGG16. We chose 5 split boundaries for both AlexNet and VGG16. server and edge of Tables 5 and 6 means that execute the entire process on the server and on the edge device, respectively. We measured the energy consumption for the former and latter layers when executing at each split boundary. We also evaluated the other seven models in the same manner.

After the former layers are over, the compression process is applied to the intermediate data to reduce communication costs. Tables 7 and 8 show the results of processing time and energy consumption when the compression is applied to intermediate data for AlexNet and VGG16, respectively. "Comp" and "Decomp" indicate compression and decompression, respectively. In this experiment, HEVC

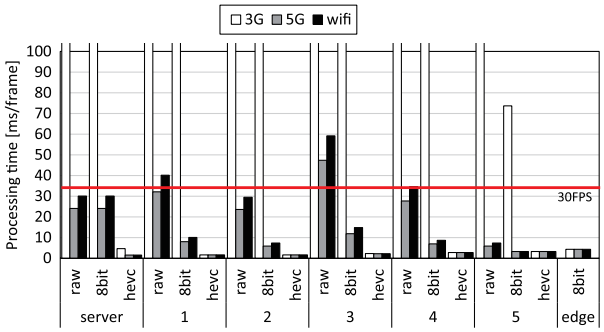


Fig. 5 FPS in distributed AlexNet

is used for compression/decompression. The energy consumption of the 8-bit quantization process is so small that it can be negligible. "input" of these tables is a case in which HEVC is applied to input video data. We also evaluated the other seven models in the same manner.

For edge, which executes the entire process on the edge device, the output is object IDs, and the output size is only a few bytes. Therefore, both the processing time and the energy of compression/decompression and communication are negligibly short. In the overall evaluation shown later, these six tables are used to estimate the total energy consumption.

6.3 Results of Comprehensive Evaluation

Based on the results obtained in Sects. 6.1 and 6.2, we performed a comprehensive evaluation. In Sect. 5.2, our requirements are 30 FPS or higher for the processing speed and 69.7%, which achieved XNOR-Nets, or higher for object recognition accuracy. Wi-Fi, 3G and 5G communication standards were used in this experiment.

Figure 5 shows the FPS in distributed models of AlexNet using TX2 (edge) and GTX1080 (server). The red horizontal line in Fig. 5 represents the target value of 30 FPS (=33.3ms/frame). From left to right, the results of the entire process on the server, split boundaries, the entire process on the edge device in AlexNet are shown. We compared the communication speed by Wi-Fi, 3G, and 5G, with their compression methods. raw represents the data of 32bit-float type, 8bit represents unsigned 8-bit quantization. hevc represents HEVC. For HEVC, Qp is set to 40, as explained in Sect. 6.1.

From the viewpoint of compression methods, 8bit always achieves higher FPS than raw, because the processing time of 8bit quantization is negligibly short, and its impact of accuracy degradation is very small as shown in Fig. 4. hevc also always achieves higher FPS than raw because hevc can drastically reduce data size, and its processing time is short enough. From the viewpoint of communication standards, wifi always achieves higher FPS than 3G and 5G always achieve higher FPS than 3G.

In this result, any split boundary can satisfy 30 FPS by applying 8-bit quantization or HEVC with Wi-Fi or 5G.

In the 3G environment, without HEVC compression cannot satisfy the requirements.

Figure 6 shows the total energy consumption, which includes the inference process of the edge device and the server, the compression/decompression process, and communication in distributed models of AlexNet. In this result, the conditions which cannot satisfy 30FPS are drawn with hatched lines.

From the viewpoint of compression methods, similarly to FPS, 8bit and hevc always achieve lower energy consumption than raw. From the viewpoint of communication standards, 5G always achieve lower energy consumption than 3G and wifi always achieve lower energy consumption than 5G.

We found that edge with wifi, which indicated with a down arrow, can realize the lowest energy consumption while satisfying both FPS and accuracy requirements.

Through these results, we found that there is a trade-off between FPS and energy for wifi and 5G. We also found that 8bit is suitable for small data, and hevc is suitable for large data. Therefore, we conclude that raw and 3G have no advantage at all, and both raw and 3G are excluded from further evaluation.

Figure 7 shows the energy consumption of all nine models using TX2 (edge) and GTX1080 (server). In this result, the conditions which cannot satisfy the 30FPS are omitted. The down allows indicate the best conditions in each model. In addition to AlexNet, edge with 8bit and wifi achieved the lowest energy consumption for GoogLeNet, MobileNet, SqueezeNet, ResNet18, and ResNet50. For VGG16, ResNet101, and ResNet152, edge cannot satisfy the FPS condition. Since the energy efficiency of Jetson TX2 is better than GTX1080, it is better to process as much as layers on edge. Therefore, the latest possible split boundary (2 for VGG16, 3 for ResNet101 and 152) can realize the lowest energy consumption. At these boundaries, the amount of the feature data is large. Thus, HEVC compression must be required to satisfy the FPS requirement. After compression, wifi is better than 5G to minimize energy consumption.

When comparing the best of all combination and the best of edge or server, the distributed models successfully reduce the energy consumption up to 19.9% (VGG16), and 4.6% on average.

Figure 8 shows the energy consumption of all nine models using Jetson Nano (edge) and GTX1080 (server). Similar to the result with TX2, in addition to AlexNet, edge with 8bit and wifi achieved the lowest energy consumption for GoogLeNet, MobileNet, SqueezeNet, and ResNet18. For VGG16, ResNet50, 101, and 152, edge cannot satisfy the FPS requirement. Since the computational performance of Nano is smaller than that of TX2, Nano can compute fewer layers than TX2. As a result, the amount of the feature data becomes larger, as shown in Fig. 2. Therefore, the latest possible split boundary (1 for VGG16, 2 for ResNet50, 101, and 152) consume more energy than server, and server can realize the lowest energy consumption. At server, the amount of communication is the same as the input video. Thus, HEVC compression can successfully minimize the data size and minimize the energy consumption of communication. After compression, wifi is better than 5G to minimize energy consumption. In this evaluation, either of the best of edge or server is the best.

In addition to two real environments, to confirm the generality of our model, we also evaluate with an imaginary edge device, whose performance and power consumption is a quarter of Jetson TX2 and named TX2/4. Figure 9 shows the energy consumption of all nine models using TX2/4 (edge) and GTX1080 (server). Due to its limited performance, only AlexNet can satisfy the FPS requirement by edge with 8bit and wifi and achieved the lowest energy consumption. For VGG16, only server can satisfy the FPS requirement, and hevc and wifi achieved the lowest energy consumption. For other models, there are complicated trade-offs related to compression and communication meth-

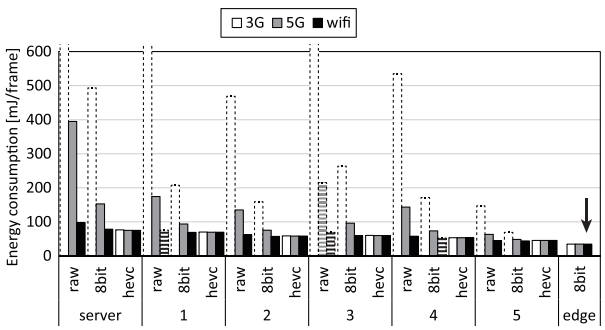


Fig. 6 Energy consumption in distributed AlexNet

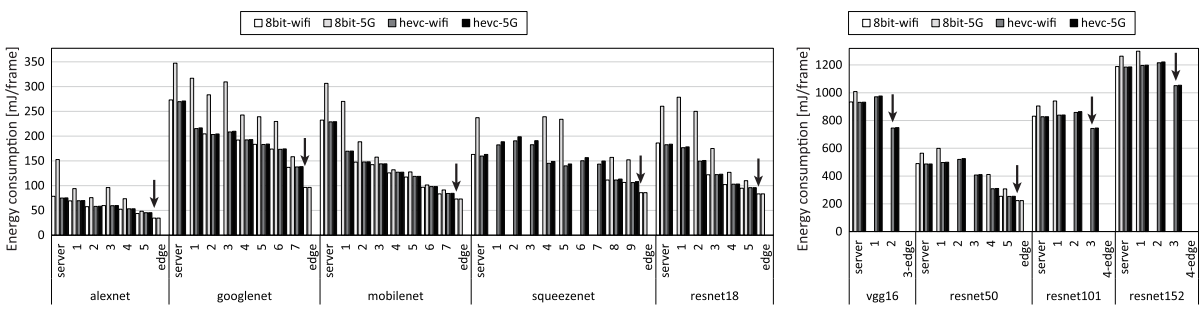


Fig. 7 Energy consumption in distributed models with TX2 (edge) and GTX1080 (server)

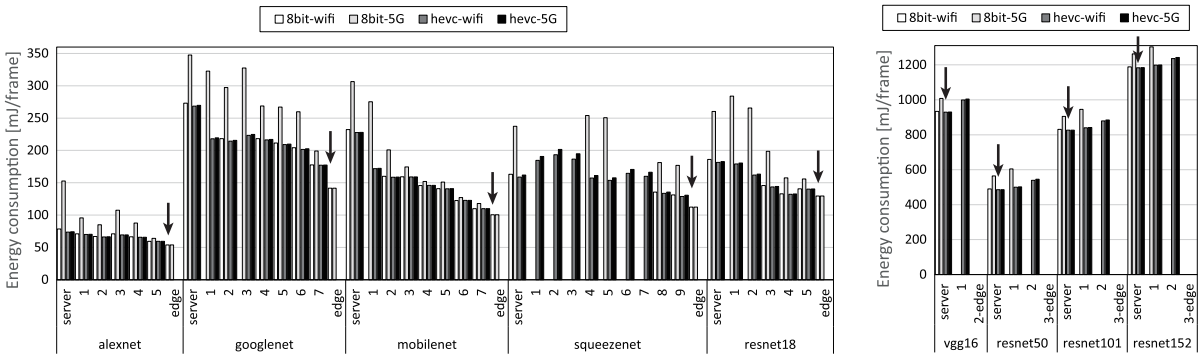


Fig. 8 Energy consumption in distributed models with Nano (edge) and GTX1080 (server)

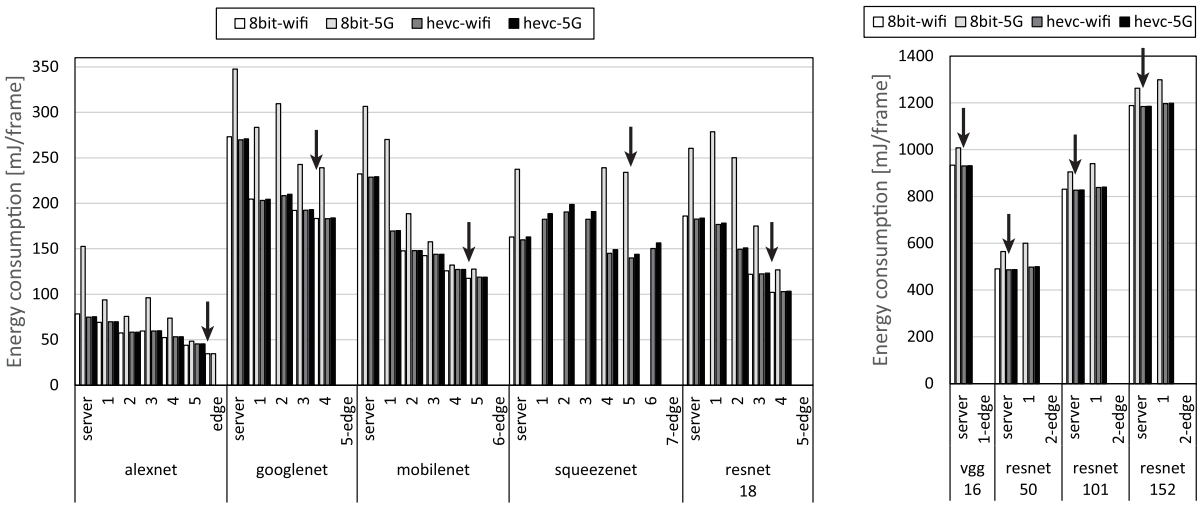


Fig. 9 Energy consumption in distributed models with TX2/4 (edge) and GTX1080 (server)

ods. For SqueezeNet, split boundary 5 is the best. The feature data should be compressed by hevc and communicated with wifi to minimize energy consumption. For GoogLeNet, MobileNet, and ResNet18, split boundary 4, 5, and 4 are the best, respectively. The feature data should be quantized by 8bit and communicated with wifi to minimize energy consumption. For ResNet50, 101, and 152, similar to the result with Nano, processed by server with hevc and wifi is the best combination to minimize the energy consumption while satisfying both FPS and accuracy requirements.

When comparing the best of all combination and the best of edge or server, the distributed models successfully reduce the energy consumption up to 48.6% (MobileNet), and 15.3% on average.

Table 9 shows breakdowns of energy consumption of selected models and conditions using TX2/4 (edge) and GTX1080 (server). In this table, the values within parentheses indicate that conditions do not meet the FPS requirement. 0.00 and blanks indicate smaller than 0.005 and not applicable, respectively.

This result also clearly shows that AlexNet can satisfy the FPS requirement by edge with 8bit and wifi and achieved the lowest energy consumption. For GoogLeNet, in edge, the computation time of the edge device cannot satisfy the FPS requirement. Thus, split boundary 4 with 8bit and wifi is the best. For SqueezeNet, in addition to the computation time of the edge device, some of the communication times cannot meet the FPS requirement. Thus, split boundary 5 with hevc and wifi is the best.

When comparing GoogLeNet and SqueezeNet, there is a trade-off between 8bit and hevc. It comes from a trade-off between compression time and communication time. In general, for small data, the compression time of hevc becomes larger than the reduction of the communication time, and 8bit is beneficial, and vice versa.

For VGG16, any edge cannot satisfy the FPS requirement. Thus, server is the only choice. Since the input data is large, hevc is beneficial. Finally, server with hevc and wifi is the best combination to minimize the energy consumption while satisfying both FPS and accuracy requirements.

In summary, when edge can satisfy the FPS requirement, edge is the first choice. When the whole computation is done on edge, the output, which includes classification results, is very small. Thus, no compression is required and transmit by the most energy efficient method wifi. When edge cannot satisfy the FPS requirement, both server and the distributed architecture should be considered. If the data size of the split boundary is smaller than that of input,

Table 9 Breakdown of energy consumption [mJ/frame]

model	split boundary	compress method	computation		compression		communication		total energy	
			edge (†, ‡)	server (†, ‡)	comp (†, ‡)	decomp (†, ‡)	wifi (†)	5G (‡)	wifi (sum of †)	5G (sum of ‡)
Alex Net	server	8bit		72.01			6.38	80.68	78.40	152.70
		hevc		72.01	1.34	1.53	0.02	0.31	74.92	75.20
	5	8bit	26.25	17.21			0.39	4.93	43.87	48.42
		hevc	26.25	17.21	1.56	0.28	0.00	0.01	45.33	45.35
	edge	8bit	34.59				0.00	0.00	**34.60**	34.60
GoogLe Net	server	8bit		266.80			6.38	80.68	273.18	347.48
		hevc		266.80	1.34	1.53	0.09	1.20	269.76	270.87
	4	8bit	58.12	129.80			4.34	54.89	**192.26**	242.81
		hevc	58.12	129.80	2.41	1.95	0.06	0.81	192.35	193.10
	edge	8bit	(96.42)				0.00	0.00	(96.42)	(96.42)
Squeeze Net	server	8bit		156.70			6.38	80.68	163.08	237.38
		hevc		156.70	1.34	1.53	0.27	3.39	159.84	162.96
	5	8bit	55.16	78.86			(7.91)	100.03	(141.93)	234.05
		hevc	55.16	78.86	3.59	2.09	0.33	4.21	**140.03**	143.91
	6	8bit	62.69	79.48			(12.55)	(158.62)	(154.73)	(300.80)
		hevc	62.69	79.48	5.12	2.47	0.53	6.67	150.30	156.44
	edge	8bit	(85.75)				0.00	0.00	(85.75)	(85.75)
VGG16	server	8bit		927.32			6.38	80.68	933.70	1008.00
		hevc		927.32	1.34	1.53	0.09	1.20	**930.28**	931.39
	1	8bit	(114.34)	838.35			(34.04)	(430.31)	(986.73)	(1383.00)
		hevc	(114.34)	838.35	13.00	4.42	0.51	6.39	(970.62)	(976.50)
	edge	8bit	(527.83)				0.00	0.00	(527.83)	(527.83)

the distributed architecture has an advantage of energy efficiency. Otherwise, the decision should be made carefully. In such a situation, the compression method is one of the important choices. Since there is a trade-off between compression energy and communication energy, the best method depends on the data size. As a result, both of the split boundary and the compression method should be considered simultaneously. From the viewpoint of communication standards, in most cases of our evaluation, wifi is better than 5G because the difference in the communication speed is small, but that of energy efficiency is large. If additional requirement, such as communication distance between the edge and the server, is given, the best method may be changed.

7. Conclusion

In this paper, we proposed a divided/distributed neural network models that increase the processing on the edge device, which realizes better energy efficiency, and reduces the load on the server by dividing the neural network model without excessive approximation such as binarization or ternarization. By applying HEVC compression, the amount of the input videos and any intermediate data has been drastically reduced. In order to construct models with efficient performance and energy, we evaluate the divided/distributed model of AlexNet and the other eight models. In a comprehensive evaluation, we found that HEVC compression is important when dividing the inference process between the edge device and the server. From the perspective of FPS, accuracy, and energy, we found that optimal configurations of all of 9 models. For small models, such as AlexNet, processing entire models in the edge is the best. On the other hand, for huge models, such as VGG16, processing entire models in the server is the best. For medium-size models, the distributed model is a good candidate. To explore many combinations of split points, compression methods, and communication standards and find the best combination, we develop an energy model. We confirmed that our model found the most energy efficient configuration while satisfying FPS and accuracy requirements, and the distributed models successfully reduced the energy consumption up to 48.6%, and 6.6% on average.

Acknowledgments

Part of this research was supported by JSPS KAKENHI JP17H00730.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), vol.115, no.3, pp.211–252, 2015.

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems 25, pp.1097–1105, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol.abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] F. Wortmann and K. Fluchter, "Internet of Things," Business & Information Systems Engineering, vol.57, no.3, pp.221–224, 2015.

[5] Y.C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-a key technology towards 5g," ETSI White Paper, vol.11, no.11, pp.1–16, 2015.

[6] "Fog computing and the internet of things: Extend the cloud to

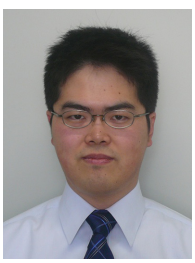
where the things are," http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf
[7] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," Advances in Neural Information Processing Systems 28, pp.3123–3131, 2015.
[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," Computer Vision – ECCV 2016, vol.9908, pp.525–542, 2016.
[9] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016. [Online]. Available: https://arxiv.org/abs/1605.04711
[10] M.P. Heinrich, M. Blendowski, and O. Oktay, "TernaryNet: faster deep model inference without GPUs for medical 3D segmentation using sparse and binary convolutions," International Journal of Computer Assisted Radiology and Surgery, vol.13, no.9, pp.1311–1320, 2018.
[11] S. Teerapittayanon, B. McDanel, and H.T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," 2016 23rd International Conference on Pattern Recognition (ICPR), pp.2464–2469, 2016.
[12] S. Teerapittayanon, B. McDanel, and H.T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp.328–339, 2017.
[13] "JETSON TX2 high performance AI at the edge." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/
[14] "JETSON NANO the power of modern AI to millions of devices." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/
[15] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark," IEEE Trans. Image Process., vol.24, no.12, pp.5630–5644, 2015.
[16] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp.2411–2418, 2013.
[17] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.9, pp.1834–1848, 2015.
[18] J.G. Andrews, S. Buzzi, W. Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, and J.C. Zhang, "What will 5G be?" IEEE J. Sel. Areas Commun., vol.32, no.6, pp.1065–1082, 2014.
[19] T. Mitani, H. Fukuoka, Y. Hiraga, T. Nakada, and Y. Nakashima, "Compression and aggregation for optimizing information transmission in distributed CNN," 2017 Fifth International Symposium on Computing and Networking (CANDAR), pp.112–118, 2017.
[20] G.J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," IEEE Trans. Circuits Syst. Video Technol., vol.22, no.12, pp.1649–1668, 2012.
[21] "Caffe model zoo." [Online]. Available: http://caffe.berkeleyvision.org/model_zoo.html
[22] "NVENC." [Online]. Available: https://developer.nvidia.com/nvidia-video-codec-sdk
[23] A. Striegel, S. Liu, X. Hu, and L. Meng, "LTE and WiFi: Experiences with quality and consumption," Procedia Computer Science, vol.34, pp.418–425, 2014.

Ryuta Shingai received his M.E. degree from Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include video compression, machine learning, deistributed computing, processor architecture.

Yuria Hiraga received his M.E. degree from Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include machine learning and processor architecture.

Hisakazu Fukuoka received his M.E. degree from Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include machine learning and processor architecture.

Takamasa Mitani received his B.S. and M.S. from Tokai University in 2005 and 2008 respectively. He has been a cumputer enginner in Pixela corporation since 2008 and Ph.D. candidate in Graduate School of Information Science, Nara Institute of Science and Technology since 2015. His research interests include distributed computing, video compression and programming methods.



Takashi Nakada received his M.E. and Ph.D. degrees from Toyohashi University of Technology in 2004 and 2007 respectively. He has been an Associate Professor at the Nara Institute of Science and Technology since 2016. His research interests includes Normally-Off Computing, system architecture and related simulation technologies. He is a member of IEEE, ACM and IPSJ.

Yasuhiko Nakashima received the B.E., M.E., and Ph.D. degrees in Computer Engineering from Kyoto University in 1986, 1988, and 1998, respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited from 1988 to 1999. From 1999 to 2005, he was an associate professor in the Graduate School of Economics, Kyoto University. Since 2006, he has been a professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include processor architecture, emulation, CMOS circuitdesign, and evolutionary computation. He is a member of IEEE CS, ACM,and IPSJ.