IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2020, accepted June 6, 2020, date of publication June 16, 2020, date of current version June 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002844

Model-Based Deep Encoding Based on USB Transmission for Modern Edge Computing Architectures

LI-QUN YANG[1], SHANQ-JANG RUAN[1], (Senior Member, IEEE), KAI-HAN CHENG[2], AND YAN-TSUNG PENG[2], (Member, IEEE)

[1]Department of Electronics and Computer Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan
[2]Pervasive Artificial Intelligence Research (PAIR) Laboratories, Department of Computer Science, National Chengchi University, Taipei 116, Taiwan

Corresponding author: Yan-Tsung Peng (ytpeng@cs.nccu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan through MOST AI Biomedical Research Center at NCKU under Grant MOST 109-2634-F-019-001 and through Pervasive Artificial Intelligence Research (PAIR) Labs under Grant MOST 109-2634-F-004-001.

ABSTRACT With the advance of deep neural networks (DNNs), artificial intelligence (AI) has been widely applied to various applications in our daily lives. These DNN-based models can be stored in portable storage disks or low-power Neural Compute Sticks. They can then be deployed in edge devices through the USB interface for AI-based applications, such as Automatic Diagnosis Systems or Smart Surveillance Systems, which provides solutions to incorporating AI into the Internet of Things (IoT). In this work, based on our observation and careful analysis, we propose a model-based deep encoding method built upon Huffman coding to compress a DNN model transmitted through the USB interface to edge devices. Based on the proposed lopsidedness estimation approach, we can exploit a modified Huffman coding method to increase the USB transmission efficiency for quantized DNN models while reducing the computational cost entailed by the coding process. We conducted experiments on several benchmarking DNN models compressed using three emerging quantization techniques, which indicates that our method can achieve a high compression ratio of 88.72%, with 93.76% of the stuffing bits saved on average.

INDEX TERMS Network compression, Huffman coding, USB transmission, edge computing.

I. INTRODUCTION

In recent years, the demand for internet-connected devices or Internet-of-Things (IoT) has been grown drastically. With the advance of deep-learning technology used in Computer Vision (CV) and Natural Language Processing (NLP), migrating various deep-learning-based CV and NLP applications to the Internet of Things (IoTs) attracts more attention [1]. Such applications usually require massive data transmission to/from the cloud from/to IoT devices, mostly relying on cloud computing for running deep-learning models. To save on cloud computation costs and preserve data privacy, using edge computing to analyze or process data using deep neural network models are pervasive nowadays. Edge computing can also help reduce the amount of data transmitted to the cloud to avoid long latency [2]. For instance, autonomous driving service provides drivers with assistance systems based on CV applications based on Deep-Neural-Network (DNN) models, including lane recognition [3], road signs recognition [4], pedestrian detection [5], etc. In a smart city, DNN models can be deployed on edge devices for deep-learning-based applications, such as crowd density estimation [6], abnormal crowd event detection [7], and person re-identification [8].

In a modern edge computing architecture for AI [9], as shown in Fig. 1, edge devices can utilize a hardware inference accelerator (HIA) to process deep-learning tasks. The inference model can be trained on local GPU servers and transmitted to edge devices through wired/wireless interfaces, which is called deployment. However, emerging deeper DNN models, often used as a feature extractor with high accuracy, impose a heavy burden of computing power on the hardware inference accelerator. Besides, a considerable number of weight parameters in a deeper DNN model severely

The associate editor coordinating the review of this manuscript and approving it for publication was Nizam Uddin Ahamed.

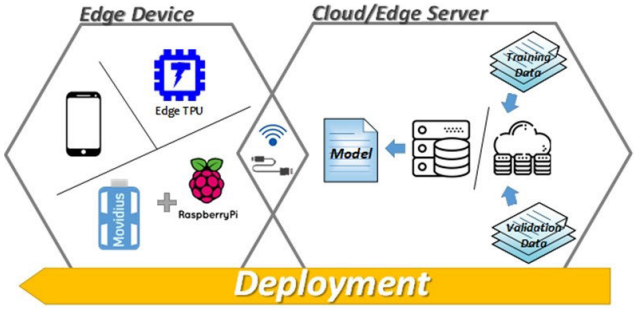


FIGURE 1. Modern edge computing architecture.

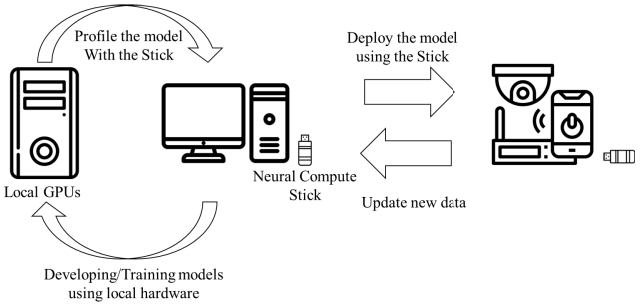


FIGURE 2. A general workflow for neural compute stick. (Icons made by https://www.flaticon.com/authors/freepik, Nikita Golubev, surang, and itim2101).

impact the transmission efficiency during deployment and the access efficiency of the hardware inference accelerator [10]. Therefore, many methods for model compression have been proposed to cut down the hardware resource need and data transmission load. He *et al.* [11] proposed an interactive method, called "channel pruning," to reduce the number of channels in network layers by a Least Absolute Shrinkage and Selection Operator (LASSO) regression based on channel selection and least square reconstruction. Lin *et al.* [12] proposed to quantize the weights of a model from 32-bit floating-point to 16-bit fixed-point representation to reduce the data usage. Han *et al.* [13] proposed pruning and trained quantization techniques to quantize the weights of deep-neural-network (DNN) models for compression. They also adopted the conventional Huffman coding as their final compression step.

In addition to model compression, optimization based on transmission media is another way to improve transmission efficiency. Much research has been done to increase transmission efficiency based on transmission media, such as the 5G network [14] and WiFi [15], by examining a better deployment scenario in terms of data transmission. However, the techniques used are mostly regarding distributed computing for mobile devices, which usually does not apply to edge computing of artificial intelligence (AI) deployment. Additionally, most commercial hardware inference accelerators use the USB interface for data transmission, such as Intel Movidius Neural Compute Stick [16]. To provide feasible solutions for edge devices to adopt deep-learning models, using a specialized low-power Neural Compute Stick is very popular. As far as we know, no work has been done on gating the transmission behavior of a DNN model through the USB protocol. The proposed work aims to increase the USB transmission efficiency, which can directly work with the Movidius Neural Computing Stick for better compression.

Fig. 2 shows that a general workflow for using a Neural Compute Stick to develop deep-learning models for edge devices. As can be seen, at the model developing/training/fine-tuning stage, one has to profile the performance of the model multiple times. After deploying the model to an edge device, it still needs periodic updates on new data. All of the transmissions are through the USB interface.

Based on our careful experiment and analysis, we observed that the weights of quantized DNN models conform to a biased distribution. The characteristic allows us to compress the weights through Huffman encoding [17], [18]. Furthermore, we propose a Model-based Deep Encoding (MDE) method to further optimize transmission efficiency for DNN models deployed via the USB interface to edge devices. The proposed MDE incorporates the method [19], which modifies Huffman encoding based on the Non-Return-to-Zero Inverted (NRZI) encoder, to reduce the bit stuffing and thus the amount of data transmission. We, moreover, proposed to estimate the lopsidedness of a Huffman tree to decrease the computational overhead incurred by modified Huffman encoding.

Overall, we make three primary contributions as follows:

- We propose the first DNN model-based encoding scheme for the USB transmission by a critical observation that biased distributions of weights of quantized DNN models exist for the modified Huffman coding to reduce the bit stuffing.
- We develop an approach that can estimate the lopsidedness of a Huffman tree to avoid unnecessary tree modifications before coding weights of DNN models for computational reduction.
- We conduct extensive experiments on three benchmarking DNN models using three emerging quantization techniques to demonstrate the superior performance of our proposed scheme for the USB transmission.

The remainder of this paper is organized as follows. Sec. II describes related work, including network compression techniques, non-return-to-zero inverted encoding, and the modified Huffman coding for data transmission. In Sec. III, we detail the proposed MDE method. The experimental results are demonstrated and discussed in Sec. IV. At last, Sec. V concludes the paper.

II. RELATED WORK

A. MODEL COMPRESSION

Applications using DNN models require more computations and memory, making deployment to low-power devices difficult due to low computational and memory resources. Therefore, it is common to apply model compression to

DNN models to relax these hardware constraints for the practicality and portability. In general, model compression techniques can be classified into three categories: compact network design, network parameters reduction, and network quantization [20]. Compact network design aims at building efficient networks where convolutional layers can be replaced with more economical designs, such as Inception Module in GoogLeNet [21], Bottleneck structure in ResNet [22], depth-wise convolution in MobileNet [23] and point-wise group convolution in ShuffleNet [24]. Although using these compact network architectures can not only make performance better but reduce the number of the weights in the model, it is not appliable to a pre-trained DNN model. Network-parameter-reduction techniques, such as pruning feature map connections [25], [26], learning a more efficient representation by low-rank approximation [27], and structured sparsity regularization [28], [29]. However, dimension-reduced representation methods could hardly achieve a high compression ratio while keeping the fidelity of the data. Network quantization techniques can be used to scale down the size of the model by quantizing data with lower precision [30], [31]. Therefore, it can achieve a larger compression ratio than the other two types of techniques.

B. NON-RETURN-TO-ZERO INVERTED (NRZI) ENCODING

Non-return-to-zero inverted encoding [32] is a commonly-used encoding technique widely used over transmission media due to its stability and simplicity. This method transforms the logic transition as a transmission code without requiring a separate clock signal delivered with the data. As shown in Fig. 3, logical "0" in the original data is transmitted as a signal which represents a logic transition in the NRZI data, while logical "1" is transmitted as no transition. However, a long series of no-transition signals in NRZI data (i.e., a long consecutive logical 1 in original data) is hard for the receiver to count the bits accurately without a reference clock. To ensure the entirety of transmission for the receiver, a stuffing bit "0" is used and inserted in the original data stream when meeting five consecutive 1 bits to force a transition.

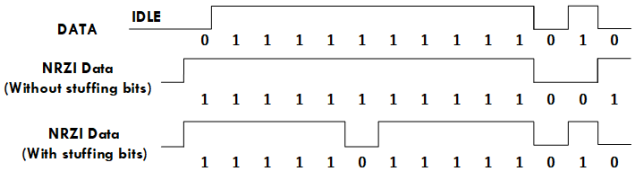


FIGURE 3. The NRZI code with and without stuffing bits.

C. MODIFIED HUFFMAN TREE

With the rapid growth of multimedia and internet popularity, image compression attracts much attention. Huffman coding [17] is a lossless compression method with variable-length codes that can reduce the average code length to achieve a higher compression ratio. It is a very commonly used coding technique in JPEG or many video compression methods. In [20], Y. Pai *et al.* proposed an improved method to speed up data transmission based on the NRZI encoder by modifying the Huffman coding in the JPEG standard.

As previously mentioned, in the NRZI coding scheme, a "0" bit is transmitted as a signal transition, whereas a "1" bit no change. The NRZI data transmitter and receiver use bit stuffing to prevent clocks from losing synchronization by inserting a "0" bit after a long run of "1" bits and to verify the data entirety. Based on that, one can avoid the bit stuffing by reducing the occurrence of over consecutive six 1s events (OCF1). In [20], it proposed to reduce the OCF1, which has two parts: sub-trees partition (STP) and sub-trees swap (STS). Given a Huffman tree, STP transverses each node of the tree to create a list of sub-trees, where OCF1 may occur, for swapping in STS until the node is found to belong to a binary tree or the depth of the tree is less than six. The list stores sub-trees where their depths are more than six and are deeper than the corresponding sibling sub-tree. For STS, the sub-tree would swap with the corresponding sibling sub-tree if the total expected value of OCF1 in both sub-trees is greater than that after swapping. The expected value of OCF1 in a sub-tree t is calculated as:

$$E(t) = \sum_{k} \{O_k \cdot P(L_k)\} \tag{1}$$

where L_k is the k_{th} leaf nodes included in the sub-tree and O_k represents the occurrence of OCF1 event defined as:

$$O_k = \begin{cases} x, & \text{if } x \text{ OCF1 events occur in } L_k; \\ 0, & \text{if no OCF1 events occur in } L_k. \end{cases} \tag{2}$$

The $P(L)$ in (2) is the occurrence probability of a leaf node L and is denoted as:

$$P(L) = 2^{-[length(L)]}. \tag{3}$$

where $length(L)$ is the level length of L.

The Huffman tree would be partitioned into several groups based on the skewness of the sub-tree and the possibility of OCF1. By comparing the expected values of OCF1 before and after swapping, the Huffman tree is modified to reduce the OCF1 occurrences. Therefore, the modified Huffman tree can still keep a comparable compression ratio while achieving high efficiency on data transmission because of less bit stuffing used.

III. PROPOSED METHOD

A. QUANTIZED DNN MODELS

Compared to compact network design and network parameters reduction, network quantization can shrink operation units by using a lower precision to achieve more efficient computations and hardware-friendly implementation. That is, network model weights can be quantized to meet the requirements of speed and memory on resource-limited devices with only a slight loss on accuracy. Han *et al.* [13] proposed to generate the codebook as a look-up table to quantize the

weights of network models by mapping them to several centroids of clusters derived by k-means. Their experiment indicated that the weights of a quantized model conform to a biased probability distribution. With Huffman coding applied, the compression ratio of a model can be further improved (e.g., 20% - 30% on the AlexNet model). Inspired by this work [13], we proceed further to look into the distributions of quantized weights in all the layers of a network model and conduct a series of experiments over several DNN models quantized using different techniques to examine whether a biased probability distribution exists concerning the network layers.

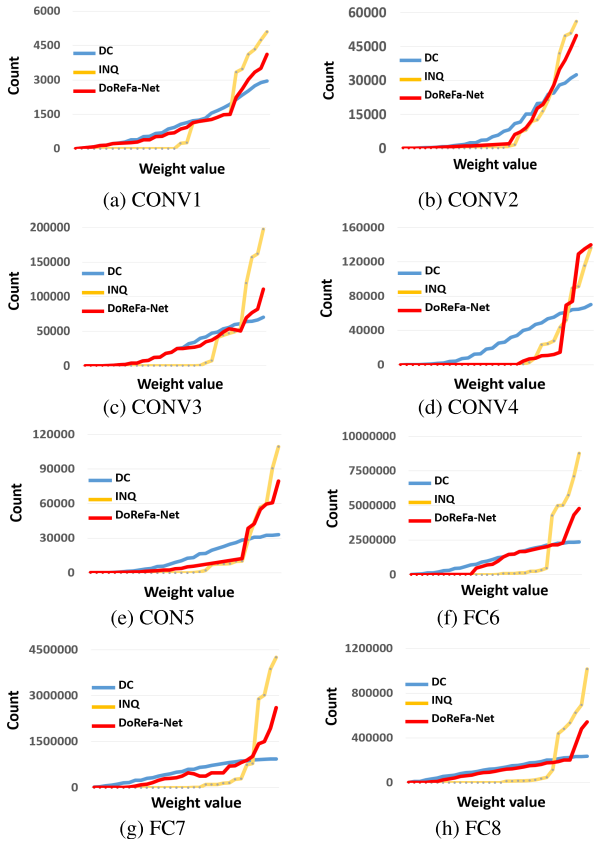


FIGURE 4. The sorted weight distributions of the AlexNet for each layer quantized by DC, INQ [33], and DoReFa-Net [30].

In Fig. 4, it shows that the quantized weight distribution of the AlexNet for each layer after sorting. The weights are quantized at 5-bit precision on the ImageNet dataset [34] by three representative quantization techniques, Deep Compression (DC) [13], Incremental Network Quantization (INQ) [33], and DoReFa-Net [30], involving pre- and post-training quantization. Note that the source code of these quantization methods is made available by their authors for people to download. We applied these quantization methods to the AlexNet with the same hyperparameters as in the original work. We found that all the distributions present biased shapes no matter which of these quantization techniques is used, which could indicate that Huffman coding can be applied to different quantized DNN models for further compression. The quantized models would have smaller data sizes. Additionally, we verified this observation and also found a biased distribution of quantized weights for an often-used object detector model proposed in [35] with the VGG-16 backbone. In our work, we exploit this characteristic in a low-power edge computing scenario where network model weights are often transmitted through the USB interface. By modifying Huffman coding, we can further reduce the inserted stuffing bits, used in the USB protocol for data synchronization.

B. MODEL-BASED DEEP ENCODING

Based on our observation, a bias probability distribution generally exists in sorted quantized model weights. We propose an encoding pipeline where the quantized weights are coded by Huffman coding specially modified for transmission via the USB interface for further compression. It can work on both pre- or post-training quantization. The proposed encoding pipeline is called Model-based Deep Encoding (MDE), based on network quantization and Huffman coding to improve the transmission efficiency of edge device deployment of deep-neural-network models. MDE utilizes tree modification [18] to reduce the overhead of bit stuffing used in the USB transmission, discovering and reducing the occurrence possibility of Over Consecutive Six 1 (OCS1) events in the Huffman code table. However, tree modification involves traversing the entire Huffman tree, which may not be needed because OCS1 events do not always happen to any Huffman code. To avoid the computation overhead caused by unnecessary tree modification, we propose a method to estimate the lopsidedness of a Huffman tree to determine whether applying tree modification. Fig. 5 illustrates the diagram of the proposed pipeline.

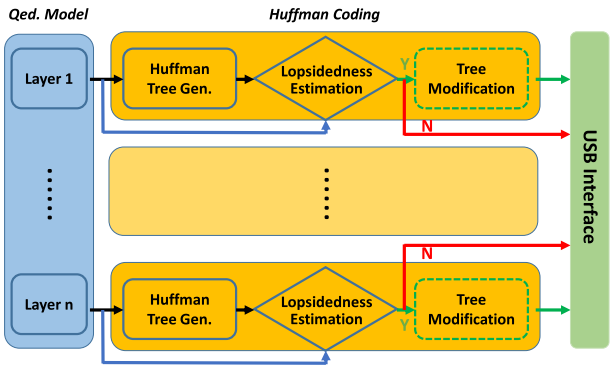
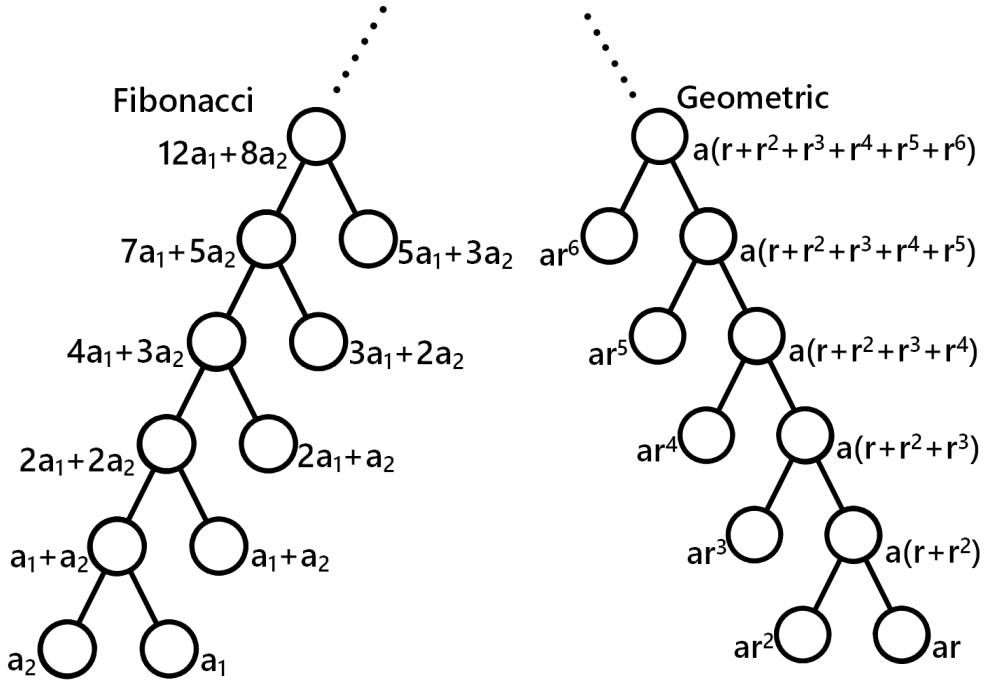


FIGURE 5. The proposed pipeline applied to each layer of a quantized (Qed) model.

C. LOPSIDEDNESS ESTIMATION

We propose an effective approach to determine whether to apply the tree modification, where the depth of the generated Huffman tree is estimated. As one may know, the depth of a Huffman tree is measured by traversing the entire tree



(a) (b)

FIGURE 6. The Huffman tree of (a) Fibonacci sequaence, $a_n = a_{n-1} + a_{n-2}$ where $a_1 > 0$, $a_2 > 0$, $n = \{3, 4, \ldots\}$, and (b) Geometric sequence, $b_n = a \cdot r^n$ where $a > 0$, $r > 1$, $n \in \mathbb{N}$.

after generating the tree, which is time-consuming and inefficient. According to Huffman coding, a lopsided Huffman tree features a larger depth compared to a balanced one. Here, the lopsidedness of a Huffman tree is determined based on the pattern of the probability distribution of input symbols (iPD), which can also be considered as the skewness of a Huffman tree. Therefore, the lopsidedness of a tree can be utilized to estimate the depth of a tree.

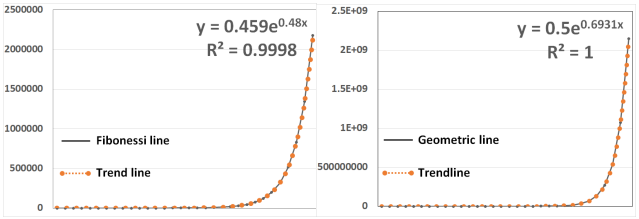
A Huffman tree is generated by iteratively combining two nodes with the two least probabilities in the ascending-order-sorted queue as a new node. The most lopsided Huffman tree appears when the ascending-order-sorted queue Q in each iteration satisfies the following conditions:

$$a_1 + a_2 \leq a_4, \tag{4}$$

where a_1, a_2 and a_4 denote the first, second and fourth elements respectively in Q. For example, two representative distributions which can generate such tree type are Fibonacci Sequence and Geometric Sequence. Fig. 6 shows Huffman trees for Fibonacci Sequence and Geometric Sequence with their iPDs. We observed that the generated Huffman tree is a full binary tree in which each node is either a leaf node or processes exactly two child nodes. The deep of the tree achieves the ultimate, that is:

$$N_{symbol} - 1, \tag{5}$$

where N_{symbol} denotes the number of input symbols. Furthermore, these two iPDs are well modeled by exponential function as shown in Fig. 7 where the R^2 score is approximately



(a) (b)

FIGURE 7. The distribution of (a) Fibonacci sequence given that $a_1 = 1$, $a_2 = 1$, and (b) Geometric sequence given that $a = 0.5$, $r = 2$.

equal to one. The lopsidedness estimation in the proposed pipeline exploits the regression degree of the exponential model. It assesses the lopsidedness of the Huffman tree of iPD through R^2 score, also referred to lopsidedness in this work. The formula of a exponential regression model is:

$$y = \alpha \cdot e^{\beta x}, \tag{6}$$

where α and β can be derived by the observed data set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$:

$$\begin{cases} \beta = \dfrac{n \sum_{i=1}^{n} \ln y_i \cdot x_i - \sum_{i=1}^{n} \ln y_i \cdot \sum_{i=1}^{n} x_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}; \\ \alpha = e^{\overline{\ln y} - \beta \overline{x}}. \end{cases} \tag{7}$$

According to the definition of R^2 score, the regression model is regarded as a well-fit model when its score is higher

TABLE 1. The lopsidedness and depth of Huffman tree for each layer of the AlexNet quantized by three different quantization techniques at 5-Bit precision.

layer	DC [13]		INQ [33]		DoReFa-Net [30]	
	R^2	H	R^2	H	R^2	H
Conv1	0.857	11	0.824	8	0.898	14
Conv2	0.939	14	0.839	11	0.844	12
Conv3	0.823	14	0.830	12	0.899	12
Conv4	0.796	14	0.808	11	0.853	11
Conv5	0.820	14	0.859	10	0.958	9
FC6	0.806	11	0.870	11	0.832	12
FC7	0.788	11	0.842	11	0.870	10
FC8	0.763	10	0.850	10	0.846	10

than 0.75. Therefore, the threshold of lopsidedness for determining whether to perform tree modification is set to 0.75. Table 1 shows the R^2 score of the exponential regression model and the depth of the Huffman tree, which uses the probability distribution of weights in each layer of the Alexnet. The R^2 score for each iPD is more than 0.75, which indicates that its corresponding Huffman tree has more than 5 layers, and tree modification is needed to avoid OCS1. It also means if a tree with its $R^2 < 0.75$, tree modification can be skipped.

TABLE 2. The lopsidedness and depth of a Huffman tree for each layer of the AlexNet quantized by three different quantization techniques at 3-Bit Precision.

layer	DC [13]		INQ [33]		DoReFa-Net [30]	
	R^2	H	R^2	H	R^2	H
Conv1	0.739	4	0.823	5	0.813	5
Conv2	0.743	4	0.739	4	0.741	4
Conv3	0.727	3	0.759	5	0.727	3
Conv4	0.861	6	0.731	4	0.838	6
Conv5	0.74	4	0.742	4	0.812	5
FC6	0.789	6	0.879	7	0.882	7
FC7	0.891	7	0.818	6	0.794	5
FC8	0.792	5	0.742	4	0.743	4

To assess the effectiveness of the proposed lopsidedness estimation, we experiment with a neural network model quantized at relatively low precision to make the generated Huffman tree of low height for each layer. We measure the time consumed by adopting the modified Huffman coding (Huffman coding with tree modification to avoid OCS1) to demonstrate how much computation time would be saved using the proposed lopsidedness estimation. Table 2 shows both the R^2 score and depth of the Huffman tree generated for weights in each layer of the Alexnet quantized with 3-bit quantization precision. One can see that the low-precision leads to a low R^2 score ($R^2 < 0.75$) for half of the layers' iPD and a small value in depth of the corresponding Huffman trees. Table 3 lists the execution time for Huffman tree construction of each layer with or without using lopsidedness estimation, where you can see the proposed lopsidedness estimation effectively reduces the execution time. Note that

TABLE 3. The execution time (Millisecond) of the Huffman coding part in MDE with and without using lopsidedness estimation based on the AlexNet quantized at 3-bit precision.

layer	DC [13]		INQ [33]		DoReFa-Net [30]	
	w/o	w/	w/o	w/	w/o	w/
Conv1	178	174	177	171	185	178
Conv2	1534	1530	1594	1572	1687	1651
Conv3	3834	3822	3796	3781	3765	3753
Conv4	2888	2888	29217	29188	27261	27261
Conv5	2196	2194	2197	2186	2274	2258
FC6	179132	179077	181219	181219	183972	183972
FC7	72955	72955	72998	72768	81132	79871
FC8	17044	17023	17121	17068	18766	17923
Total	279761	279663	308319	307953	319042	316867

all the experiment was conducted on a desktop with an Intel Core i7-4790 CPU and 24G RAM. It shows that using the proposed lopsidedness estimation, the computational overhead caused by tree modification can be avoided. Hence, using the proposed lopsidedness estimation based on the R^2 score of model regression is more efficient than measuring the depth of the Huffman tree by traversing the entire tree.

IV. EXPERIMENTAL RESULTS

The proposed MDE pipeline has two parts to increase transmission efficiency: 1) model compression and 2) encoding optimization for USB transmission. Thus, there are two types of experimental results presented to assess our method: 1) the compression ratio and 2) bit stuffing reduction ratio. Since Huffman coding is a lossless encoding technique that does not affect the weight distribution, the accuracy of a model after encoded by Huffman coding is the same as that of the quantized model. Therefore, the accuracy of a model would not be discussed in the scope of this work. In the experiment, the MDE pipeline adopts three representative model quantization techniques, DC [13], INQ [33], and DoReFa-Net [30] and works with three benchmarked DNN models, AlexNet, GoogleNet, and ResNet-50. Table 4 lists the model compression ratio $\%\Gamma$ and bit stuffing reduction ratio $\%\Theta$ using the MDE, where Q, H, and $Q + H$ represent the two ratios ($\%\Gamma$ and $\%\Theta$) using model quantization, Huffman coding, and the MDE pipeline. H^* represents the modified Huffman coding with tree modification used.

A. COMPRESSION RATIO

For fair assessment, three DNN models are compressed (quantized) at 5-bit precision using all three quantization techniques. For each layer, the compression ratio would be the same value of 84.38% since the quantization precision used is the same for each method, which is 32 bits quantized to 5 bits. The overall compression ratio of a DNN model is calculated as:

$$\%\Gamma = 1 - \left| \frac{M_{after} - M_{before}}{M_{before}} \right| \cdot 100\%, \tag{8}$$

TABLE 4. The model compression ratio %Γ and bit stuffing reduction ratio %Θ of the MDE in each layer of the AlexNet using three model quantization techniques.

Layers	Qed Tech.	Size (KB)	Stuffing bits	%Θ			%Γ		
				Q	H (H^*) [1]	Q+H	Q	H (H^*)	Q+H
CONV1	DC	136	302	93.43%	41.55%*	96.16%	84.38%	26.24%*	88.48%
	INQ	136	302	96.24%	34.30%*	97.53%	84.38%	22.02%*	87.82%
	DoReFa-Net	136	302	94.61%	35.80%*	97.22%	84.38%	30.79%*	89.19%
CONV2	DC	1199	54101	76.43%	26.38%*	82.65%	84.38%	30.79%*	89.14%
	INQ	1199	54101	79.35%	75.93%*	82.74%	84.38%	30.47%*	88.70%
	DoReFa-Net	1199	54101	75.93%	28.29%*	82.70%	84.38%	27.65%*	88.97%
CONV3	DC	3456	157029	85.07%	30.40%*	89.61%	84.38%	31.88%*	89.36%
	INQ	3456	157029	85.25%	45.62%*	91.98%	84.38%	30.08%*	89.08%
	DoReFa-Net	3456	157029	86.61%	38.38%*	91.75%	84.38%	27.97%*	88.75%
CONV4	DC	2408	118873	90.04%	35.74%*	93.60%	84.38%	31.17%*	89.25%
	INQ	2408	118873	91.13%	48.47%*	95.43%	84.38%	28.36%*	88.81%
	DoReFa-Net	2408	118873	91.28%	26.72%*	93.61%	84.38%	26.56%*	88.53%
CONV5	DC	1727	78793	92.23%	49.54%*	96.08%	84.38%	31.77%*	89.25%
	INQ	1727	78793	92.11%	38.65%*	95.16%	84.38%	25.03%*	88.29%
	DoReFa-Net	1727	78793	93.11%	42.52%*	96.04%	84.38%	21.19%*	87.69%
FC6	DC	147384	6930804	97.54%	28.86%*	98.25%	84.38%	21.19%*	88.66%
	INQ	147384	6930804	97.76%	38.83%*	98.75%	84.38%	27.40%*	88.42%
	DoReFa-Net	147384	6930804	97.63%	41.35%*	98.63%	84.38%	25.86%*	89.09%
FC7	DC	65504	3230655	96.49%	27.35%*	97.45%	84.38%	29.76%*	89.03%
	INQ	65504	3230655	95.69%	24.12%*	96.73%	84.38%	29.06%*	88.92%
	DoReFa-Net	65504	3230655	96.11%	20.05%*	98.07%	84.38%	26.95%*	88.59%
FC8	DC	15992	571632	91.76%	10.55%*	92.63%	84.38%	27.27%*	88.64%
	INQ	15992	571632	92.75%	14.20%*	95.67%	84.38%	25.16%*	88.31%
	DoReFa-Net	15992	571632	92.07%	23.32%*	94.96%	84.38%	25.60%*	88.38%

[1] H* represents the tree of Huffman coding is modified by tree modification.

where M_{after} and M_{before} denote the data sizes of the model after and before the process. After model quantization, the compression ratio is 84.38%. The Huffman coding can further compress the data up to 30% compared to model quantization. The overall compression ratio by the proposed MDE ($Q + H$) can achieve almost 90%.

B. BIT STUFFING REDUCTION

As mentioned in Sec. II-B, the NRZI coding scheme uses "0" and "1" to represent a signal transition and no change, where a stuffing bit is inserted to prevent clocks from losing synchronization between a transmitter and receiver through inserting a "0" bit after a long run of "1" bits. In MDE, bit stuffing reduction is contributed by model quantization and tree modification. Based on the USB protocol, a DNN model is transmitted by stringing up all the weights as a long bit sequence. A stuffing bit is inserted when the OCS1 occurs in two scenarios. First, OCS1 occurs in a data sequence. Second, it occurs because of the concatenation of two data sequences. For example, given two 5-bit data sequences, 01111 and 11000, OCS1 occurs when the two are transmitted consecutively. Note that the tree modification can only resolve OCS1 in the first scenario when the transmitted data is encoded by Huffman coding. The bit stuffing reduction ratio is calculated as:

$$\%\Theta = 1 - \left| \frac{b_{after} - b_{before}}{b_{before}} \right| \cdot 100\%, \tag{9}$$

TABLE 5. The model compression ratio %Γ and bit stuffing reduction ratio %Θ of the MDE on GoogleNet & ResNet-50 using three model quantization techniques.

Model	Qed Tech.	%Θ	%Γ
GoogleNet	DC	94.74%	88.47%
	INQ	93.45%	87.92%
	DoReFa-Net	93.23%	85.29%
ResNet-50	DC	95.27%	86.48%
	INQ	97.21%	86.49%
	DoReFa-Net	94.27%	88.89%

where b_{after} and b_{before} denotes the number of the stuffing bit inserted in the transmitted data after and before the process. Since model quantization reduces the representation precision and makes data sequences shorter, it can extensively reduce the OCS1 occurrence in both cases.

C. EVALUATION

Table 4 shows the model compression ratio %Γ and bit stuffing reduction ratio %Θ of the MDE in each layer of the AlexNet using three model quantization techniques. The columns of "Size (KB)" and "Stuffing bits" refer to the file size and the number of stuffing bits inserted for the AlexNet model. The average %Γ overall achieves 88.72%, which indicates that using the MDE can provide a better compression ratio than using model quantization only (which is 84.38%).

The precision reduction by model quantization can reduce the occurrence of OCS1, which leads to a reduction ratio of the bit stuffing of 90.77%. Furthermore, using the modified Huffman coding (Huffman coding with tree modification), the bit stuffing reduction ratio achieves a higher value of 93.76%, which indicates that the MDE can further increase the transmission efficiency of the USB transmission. Table 5 shows that the MDE can also achieve a high compression ratio for the GoogleNet and ResNet-50. The results shown represent the average over all layers of the models after applying the MDE.

V. CONCLUSION

In the paper, we propose a model-based deep encoding (MDE) pipeline built upon Huffman coding to compress a DNN model transmitted through the USB interface to edge devices. Based on our observation that a biased distribution exists in the weights of a quantized DNN model, the proposed MDE combines model quantization techniques and the modified Huffman coding as a general encoding pipeline that yields a higher compression ratio in the USB transmission. We also propose a lopsidedness estimation method for a Huffman tree to avoid the computational overhead caused by tree modification. The experimental results on several benchmarking DNN models compressed using three emerging quantization techniques indicate that our method can achieve a high compression ratio of 88.72% with 93.76% of the stuffing bits saved on average.

REFERENCES

[1] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[2] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[3] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 690–703, Mar. 2017.

[4] W. Hu, Q. Zhuo, C. Zhang, and J. Li, "Fast branch convolutional neural network for traffic sign recognition," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 3, pp. 114–126, Jul. 2017.

[5] X. Du, M. El-Khamy, J. Lee, and L. Davis, "Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 953–961.

[6] V. C. Liang, R. T. B. Ma, W. S. Ng, L. Wang, M. Winslett, H. Wu, S. Ying, and Z. Zhang, "Mercury: Metro density prediction with recurrent neural network on streaming CDR data," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 1374–1377.

[7] H. Wei, Y. Xiao, R. Li, and X. Liu, "Crowd abnormal detection using two-stream fully convolutional neural networks," in *Proc. 10th Int. Conf. Measuring Technol. Mechatronics Autom. (ICMTMA)*, Feb. 2018, pp. 332–336.

[8] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3908–3916.

[9] S. B. Calo, M. Touna, D. C. Verma, and A. Cullen, "Edge computing architecture for applying AI to IoT," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2017, pp. 3012–3016.

[10] M. Capra, R. Peloso, G. Masera, M. R. Roch, and M. Martina, "Edge computing: A survey on the hardware requirements in the Internet of Things world," *Future Internet*, vol. 11, no. 4, p. 100, Apr. 2019.

[11] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.

[12] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: http://arxiv.org/abs/1510.00149

[14] J. Zhang, W. Xie, F. Yang, and Q. Bi, "Mobile edge computing and field trial results for 5G low latency scenario," *China Commun.*, vol. 13, no. 2, pp. 174–182, 2016.

[15] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, and M. Satyanarayanan, "Quantifying the impact of edge computing on mobile applications," in *Proc. 7th ACM SIGOPS Asia–Pacific Workshop Syst. (APSys)*, 2016, pp. 1–8.

[16] *Movidius Announces Deep Learning Accelerator and Fathom Software Framework*. Accessed: Apr. 2020. [Online]. Available: https://developer.movidius.com/

[17] C. Pal, S. Pankaj, W. Akram, A. Acharyya, and D. Biswas, "Modified Huffman based compression methodology for deep neural network implementation on resource constrained mobile platforms," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[18] Y.-T. Pai, F.-C. Cheng, S.-P. Lu, and S.-J. Ruan, "Sub-trees modification of Huffman coding for stuffing bits reduction and efficient NRZI data transmission," *IEEE Trans. Broadcast.*, vol. 58, no. 2, pp. 221–227, Jun. 2012.

[19] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 373–390.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[23] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[24] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[25] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.

[26] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[27] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[28] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7370–7379.

[29] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2554–2564.

[30] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*. [Online]. Available: http://arxiv.org/abs/1606.06160

[31] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5918–5926.

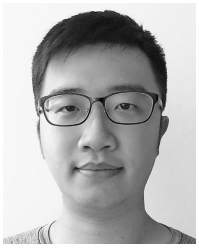
[32] K. Schouhamer-Immink, *Coding Techniques for Digital Recorders*. Upper Saddle River, NJ, USA: Prentice-Hall, 1991.

[33] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*. [Online]. Available: http://arxiv.org/abs/1702.03044

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

LI-QUN YANG received the M.S. degree in electronic and computer engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include low power system design, deep learning, and image processing.

SHANQ-JANG RUAN (Senior Member, IEEE) is a Distinguished Professor with the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology. His research interests include embedded deep neural network processing, energy-efficient image processing, and embedded systems design.



KAI-HAN CHENG received the B.S. degree in computer science and information engineering from Fu Jen Catholic University, Taipei, Taiwan, in 2019. He is currently pursuing the M.S. degree in computer science with National Chengchi University, Taipei.

YAN-TSUNG PENG (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of California at San Deigo, San Diego, CA, USA, in 2017. Following a Senior Engineer at Qualcomm Technologies, Inc., in 2019, he joined the Department of Computer Science, National Chengchi University, Taipei, Taiwan, where he is currently an Assistant Professor. His research interests include image processing, video compression, and machine learning.