# Mixed Precision Quantization Scheme for Re-configurable ReRAM Crossbars Targeting Different Energy Harvesting Scenarios

Md Khan, Nicholas Anton Jao, Changchi Shuai, Keni Qiu, Mehrdad Mahdavi, Vijaykrishnan Narayanan

# Mixed Precision Quantization Scheme for Re-configurable ReRAM Crossbars Targeting Different Energy Harvesting Scenarios

Md Fahim Faysal Khan*[1], Nicholas Anton Jao*[1], Changchi Shuai[2], Keni Qiu[2], Mehrdad Mahdavi†[1], and Vijaykrishnan Narayanan†[1]

†Department of Computer Science and Engineering
*Department of Electrical Engineering
[1]Pennsylvania State University, University Park, PA
[2]Capital Normal University, Beijing, China

**Abstract.** Crossbar arrays with non-volatile memory have recently become very popular for DNN acceleration due to their In-Memory-Computing property and low power requirements which makes them suitable for deployment on edge. Quantized neural network (QNNs) enables us to run inference with limited hardware resource and power availability and can easily be ported on smaller devices. On the other hand, to make edge devices self sustainable a great deal of promise has been shown by energy harvesting scenarios. However, the power supplied by the energy harvesting sources is not constant which becomes problematic as a fixed trained neural network requires a constant amount of power to run inference. This work addresses this issue by tuning network precision at layer granularity for variable power availability predicted for different energy harvesting scenarios.

**Keywords:** Quantization, Deep Learning, ReRAM, Crossbar, Energy Harvesting, Power Predictor

## 1 Introduction

Deep Neural Networks (DNNs) have gained enormous popularity by solving tasks such as object recognition and detection at human level accuracy over the past few years [10]. One of the key factors responsible for the rapid progress, is the availability of compute power such as GPUs. Although these hardware facilitate the training and running of the DNNs at high precision, they consume a huge amount of power. Excessive power consumption is a major bottleneck while running inference using DNNs on edge devices such as smartphones, smartwatches or training them in a distributed fashion using edge nodes. Quantization of these networks offer lower state representation, transforming expensive floating point computations into integer arithmetic. Hence, quantization makes the networks more amenable for execution on systems with low compute capability and limited power. Consequently, research in Quantized Neural Networks (QNNs) is promising and is the focus of this work.

Non-volatile memories such as Resistive Random Access Memory (ReRAM) arranged in a crossbar structure permit intrinsic and efficient computation of multiply-accumulate (MAC) operations which dominate the run-time of convolutional neural networks (CNNs). This is due to the notion of leveraging Kirchoff's current law using the ReRAM cells as the weight modulators to perform analog current summation and exploit the crossbar's intrinsic parallelism to compute MAC [3, 17]. However, this analog method of computation is impractical on full-precision floating point data representations and therefore more suitable to run low-precision fixed point computations. This work leverages ReRAM based crossbars as the underlying hardware platform for the proposed low-precision neural networks.

The scenarios of deploying energy harvesting processors and accelerators have drawn researchers' interest in the area of Internet of Things (IoTs). Various techniques have been proposed to reconfigure tasks or execution patterns to match the widely fluctuating harvested energy and thus achieve the optimal energy efficiency [12, 23, 7]. For an energy harvesting system, it is important to have the ability of power prediction to be aware of the future power in advance. In this work, with the ability of power prediction, we can proactively configure the last incomplete inference's network structure to fit the next coming power cycle, but not discarding the obtained partial results.

Inference using a given deep network requires a fixed amount of power. As a result, when the amount of harvested power varies due to inherent fluctuations in an energy-harvesting environment, there exists a mismatch between the power producer and the deep network consumer. It can happen that a high precision fixed network cannot be executed in many power cycles due to its high power demand. It can also happen that a low precision fixed network may achieve high throughput but very low accuracy. In order to adapt to this variable power scenario, dynamic precision quantization could be a solution. That is, we accommodate different mix-precision network structures to different power supply levels to achieve a balance among accuracy, performance and power.

This work makes the following contributions:

- We propose a mixed precision quantization scheme to find different network configurations to support operation at different levels of harvested power. Our approach builds upon an existing multi-precision framework and modifies it to incorporate power-aware tuning.

- A high-accuracy power predictor is designed to be able to predict multiple power levels. Specifically, a SMOTE algorithm is embedded into the predictor to pre-process the data set so as to achieve balanced data group density for the training phase.

- Both QNN learning results and the system-level results (*throughput, energy efficiency, energy utilization* etc.) show that the proposed mixed precision quantization scheme can manage a good trade-off among throughput, energy efficiency and accuracy.

The remainder of the paper is organized as follows. Section 2 describes background information and related work to QNN, ReRAM crossbar and energy harvesting systems. Section 3 presents the quantization schemes for different power levels. Section 4 shows the ReRAM circuit parameters and different network configurations. In Section 5, we present a power predictor which can predict multi-level future power with a high accuracy. Section 6 shows the experimental results. Finally, we conclude the paper in Section 7.

## 2 Background and Related work

### 2.1 Quantized Neural Networks

Quantized Neural Networks (QNNs) enable us to run inference using low compute capability and power availability. One approach towards QNN investigates post training quantization, where the training is done in full precision and then the trained model is quantized [13, 1]; Another approach deals with quantization during training [5, 16, 25, 14, 4, 24]. Both approaches result in a compressed version of the model with reduced bit precision to have a lighter inference requirement. However, all of the above mentioned works offer a uniform precision quantized network i.e. bit precision of each layers is fixed.

Recently, there have been a few studies on training mixed precision networks as well where each layer inside the network can have different bit representation. It has recently been showed by [22] that different layers inside a neural network serves different purposes and should not be treated as same. The mixed precision approaches support the statement [20, 6, 21, 19] by showing that the accuracy can be preserved even though a good number of layers are quantized to lower precision while keeping a few at higher precision. HAQ [20] and ReLeQ [21] are reinforcement learning based approaches where the agent learns the bit precision for each of the layers after a large number of training episodes. HAWQ [6] finds mixed precision configurations by using second order information like calculating hessian. C2Q [19] takes a full precision or quantized network at higher state and quantizes it to a lower bit representation gradually layer by layer based on a competitive-collaborative approach.

### 2.2 ReRAM crossbar-based accelerator

ReRAM crossbar is a promising device to perform MAC operations in a In-Memory Computing style. PRIME [3] presents the architecture-level design of ReRAM crossbar-based accelerator where the ReRAMs function as dual modes of both computing module and storage module. custom peripheral circuits are designed to achieve the reusability of the ReRAM crossbars. The ISAAC [17] architecture supports a pipeline execution to boost the MAC throughput based on the ReRAM tiles. The hybrid ReRAM structure [15] is proposed to combine sequential and parallel execution fashions to meet some power budget.

### 2.3 Energy harvesting system

Energy harvesters accumulate energy from the surrounding environment, such as solar energy, piezo electricity, thermal gradients, radio frequency (RF) radiation,

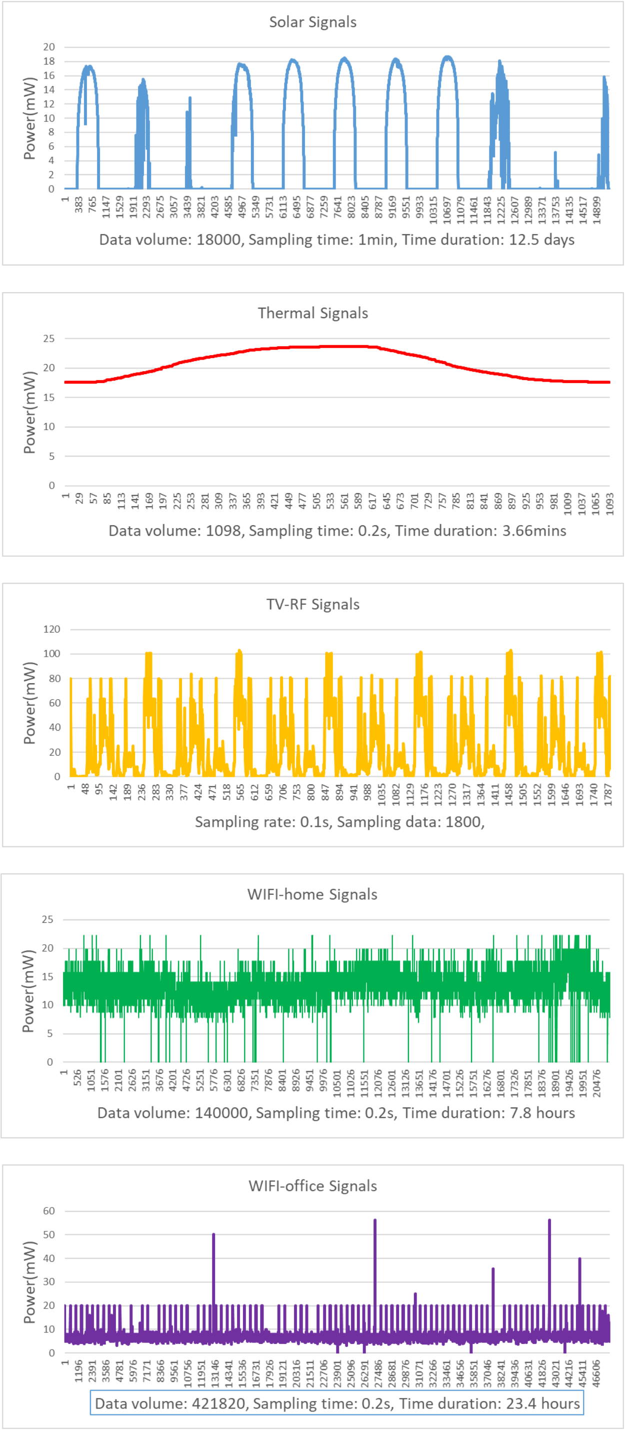

**Fig. 1.** Five energy harvesting sources.

as shown in Figure 1. The harvested energy can be first charged to energy stores or directly fed to devices. In this work, we consider the ”Harvest-Direct Use” architecture to use the harvested energy. Because the environmental energy is not stable, the system may suffer frequently power-down and have to restart. Even with power on, the system has to operate under a fluctuating powering condition.

It is known that power consumption requirements of different system architectures vary. K. Ma et al. proposed three hardware structures to fit the changing power of energy harvesters [11]. This work targets the ReRAM crossbar-based CNN deployment. The goal is to accommodate different quantization solutions to the changing harvested power. Since we need to reconfigure the ReRAMs given a power level and a corresponding quantization solution, it is highly demanded to know the power level of future power cycles.

## 3 Quantization for different power levels

In this section, we briefly describe our methodology for training mixed precision networks. We choose a layer by layer gradual quantization strategy to find networks with different bit precision granularity following C2Q [19]. The quantization framework can incorporate any of the existing quantization strategies and deliver a quantized model within a targeted size, power or accuracy threshold. Figure 2 gives an overview of this framework.

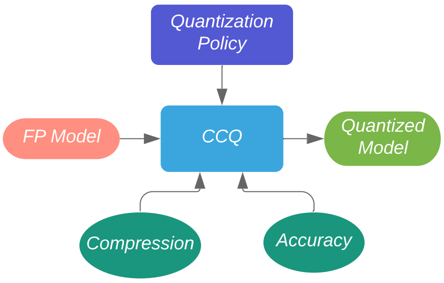

**Fig. 2.** Competitive-Collaborative Quantization (CCQ) framework. It takes a full precision model and gives a quantized model under different size, power and accuracy constraints.

In this particular problem, the power supplied by the energy harvesting sources is not constant. A constant fixed network requires a specific amount of power to execute. When the energy harvesting sources can deliver that amount of power, the network can operate. However, when the available power is less than that required, it cannot operate. Interestingly, if the energy harvesting sources deliver more power, the network cannot make use of that either without an energy storage device. This work tries to find out a way to make use of that extra available power to boost the accuracy. In general, quantization makes the network run using integer arithmetic enabling it to be deployed on edge devices. Existing policies are capable of quantizing the network to lower precision levels such as 8 bits, 4 bits, 2 bits or even 1 bit. The lower the bit precision, the smaller the required compute, requiring less power for execution. However, using lower precision causes degradation in accuracy. Hence, there is a clear trade-off between precision and accuracy. As higher precision networks requires comparatively higher power to operate, we can easily draw a connection among power→precision→accuracy providing us a very interesting knob to tune. This work proposes a scheme where we can boost the accuracy upon availability of sufficient power by increasing the precision of the network.

While quantizing, most of the prior works used uniform bit precision i.e. used same precision such as 2 bits, 4 bits etc. for all the layers. There are however a couple of issues of using the whole network operating at constant precision such as 4 bits or 8 bits. First of all, the separation among these levels in terms of power is quadratic and we cannot scale the precision linearly. On the other hand, mixed precision networks where different layers operate using different precision levels can offer better linearity. Moreover, C2Q argued that the uniform bit-precision may not be optimal representation for a network. According to them, some layers might need higher precision to preserve the accuracy while others can operate at much lower precision levels. In order to find out these mixed precision networks they use a quantization framework which can deliver a mixed precision network under different constraints such as size, power, accuracy etc. Consequently, this framework becomes a perfect choice for our desired goal. We use it to find different mixed precision networks working at different power levels providing accuracy numbers accordingly. In brief the quantization framework works as follows,

- Starts from a uniform precision network working at higher accuracy, and then quantizes each layer gradually one by one
- During quantization, the layers compete with each other in order to get quantized. In competition, each layers gets quantized and a score is calculated for that layer based on the network's performance on a small portion of the validation set. Once the scores for all the layers are obtained, a probability for each of the layers gets calculated. For each layer, the corresponding probability are calculated using the following equation:

$$p^{(t)} = \frac{\alpha_m^{(t)}}{\sum_{i=1}^{m} \alpha_i^{(t)}}, \tag{1}$$

  where, $p^{(t)}$ is the probability at $t$th quantization step, $\alpha$ is the score for each layer and $m$ is the number of layers. Finally, a layer gets selected based on the probability vector $p^{(t)}$. This is called the *competition* stage.
- The selected layer is quantized to the next level (usually, the steps are 8 bits → 6 bits → 4 bits → 2 bits). Due to quantization, the accuracy gets degraded which is then recovered by retraining of the network where all the layers participate. This is called *collaboration* stage.

- The whole procedure is repeated until a desired compression level (in terms of size or power) is achieved.

We follow a similar strategy to find out different configurations of a network for different power levels. In order to take the compute power into account, we modify the probabilities calculated using equation 1 by introducing a parameter $\lambda$. The final probability becomes:

$$p_{new}^{(t)} = (1-\lambda)p^{(t)} + \lambda \frac{|U_m^{(t)}|}{\sum_{i=1}^{m} |U_i^{(t)}|} \tag{2}$$

where, $p_{new}^{(t)}$ is the new probability at $t_{th}$ quantization step and $U_m^{(t)}$ is the compute power for $m_{th}$ layer at that quantization step. The parameter $\lambda$ determines how we aggressively the layers requiring higher power will get quantized. A higher value will try to quantize the expensive layers first. Typical values for $\lambda$ is around 0.6 to 0.7. Following this, we find out different mixed precision configurations for different power levels. One of these networks gets employed into the ReRAM crossbar depending on the power availability predicted by the power predictor. This makes the accuracy boosting based on power availability possible.

## 4 Precision Configurable ReRAM Crossbar

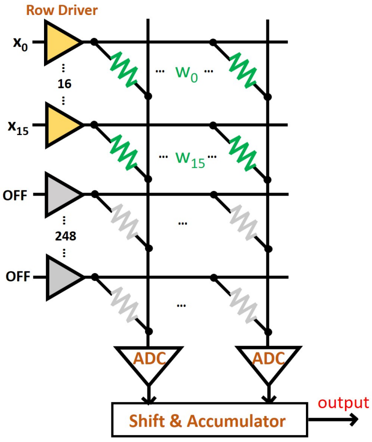

**Fig. 3.** Circuits of ReRAM-based Neural Network Accelerator.

We evaluate the network on a ReRAM crossbar to perform the MAC operation at variable precision. We perform HSPICE circuit simulation to evaluate the latency and energy consumed by the components of a ReRAM crossbar.

Each crossbar comprises of a row driver, ReRAM cells, analog-digital converters (ADC) and shift-add output accumulators illustrated in Figure 3.

The row driver logic is designed to switch between one-hot and MAC modes. During one-hot mode, the row driver acts as a row decoder for the ReRAM arrays to behave as a basic read/write memory. To enable the MAC operation, the drivers select 16 rows of the array simultaneously as a bit-serial input. By bit-slicing the data in to processing one bit at a time, the driver circuits avoid costly digital-to-analog converters proposed in prior works [3].

For the crossbar array, ReRAM memory devices are employed to store the weights of the quantized neural networks. To reduce the variation challenges associated with the technology, each ReRAM cell only supports two resistance states, high resistance and low resistance. The SET operation sets the device into low resistance and the RESET operation switches it back to high resistance state. Therefore, multiple ReRAM cells are used to represent the weights in binary format.

The ADC unit is composed of a current sense amplifiers (CSA) used to quantize the output current from the ReRAM crossbar coupled with a feedback circuit made of reference ReRAM devices and registers. The CSA is a comparator circuit which compares the two current inputs and outputs which input is greater than the other. The feedback control logic is employed for the reference input to perform a binary search on the ReRAM current like a successive-approximation register ADC (SAR-ADC).

**Table 1.** Power breakdown of each component of a crossbar for different weight configurations.

| | **Crossbar Power Breakdown (per convolution)** | | | | | | |
|---|---|---|---|---|---|---|---|
| Config. | Row Driver | ReRAM | CSA | ADC Logic | Shift-Add | Output Reg. | **Total** |
| | (nW) | ($\mu$W) | ($\mu$W) | ($\mu$W) | ($\mu$W) | ($\mu$W) | ($\mu$W) |
| 1-bit | 414 | 20.2 | 110.7 | 148.6 | 28.2 | 124 | 432 |
| 2-bit | 414 | 40.5 | 221.3 | 269.5 | 51.8 | 124 | 718 |
| 3-bit | 414 | 60.7 | 332 | 410.4 | 74.1 | 124 | 1000 |
| 4-bit | 414 | 80.9 | 442.7 | 541.3 | 94.5 | 124 | 1280 |
| 5-bit | 414 | 101.2 | 553.3 | 672.2 | 113 | 124 | 1560 |
| 6-bit | 414 | 121.4 | 664 | 803.1 | 131 | 124 | 1840 |
| 7-bit | 414 | 141.6 | 774.7 | 934 | 148 | 124 | 2100 |
| 8-bit | 414 | 161.8 | 885.3 | 1060 | 163.7 | 124 | 2400 |

Finally, the shift-add output accumulator is required to add the partial product results generated from the ADC units to generate the final product of the MAC operation [17][8]. When the weights are represented by multiple ReRAMs, multiple ADC units are employed to digitize the current summation from least significant to most significant bits. Consequently, the shift and add units are the key logic components to ensure mathematical correctness of the analog computation of both bit-sliced weights and activations.

Table 1 shows the measured power breakdown of each component of a crossbar for different weight configurations. The crossbar's peripheral components employ power gating circuit techniques on ADC units to support lower precision weights as the weight quantization varies the number of columns that are active during convolution. Consequently, the inputs of the shift-add units for inactive columns are shut-off, reducing the dynamic power of the partial product compression. Lastly, the power consumed by the row driver and output registers do not change with weight precision and are measured to be constant as long as the crossbar is active.

Understanding the QNN quantization principles and the ReRAM circuit design paradigm, we study five different quantization configurations of *Network-1*, *Network-2*, *Network-3*, *Network-4* and *Network-5*. Each of these network structures consumes different amounts of power and incurs different latency and achieves different output accuracy as shown in Table 2. The *Network-1* configuration with the highest data precision can achieve the best accuracy, while consuming the highest power and longest latency, and vice versa. Motivated by this reconfigurable design possibility, this paper proposes to apply different quantization designs to accommodate to the unstable harvested power while achieving as high as possible output accuracy. In other words, when a minimum amount of power is available from the energy harvesting sources, the smallest configuration can be used and upon availability of more power, other configurations can be used to boost the accuracy.

**Table 2.** Five precision-fixed quantization schemes

| | **Network-1** | **Network-2** | **Network-3** | **Network-4** | **Network-5** |
|---|---|---|---|---|---|
| Feature.0 (bit) | 8 | 8 | 6 | 4 | 2 |
| Feature.3 (bit) | 8 | 2 | 2 | 2 | 2 |
| Feature.7 (bit) | 8 | 6 | 2 | 2 | 2 |
| Feature.10 (bit) | 8 | 4 | 2 | 2 | 2 |
| Feature.14 (bit) | 8 | 2 | 2 | 2 | 2 |
| Feature.17 (bit) | 8 | 6 | 4 | 2 | 2 |
| Feature.20 (bit) | 8 | 8 | 6 | 4 | 2 |
| Feature.24 (bit) | 8 | 4 | 2 | 2 | 2 |
| Feature.27 (bit) | 8 | 4 | 2 | 2 | 2 |
| Feature.30 (bit) | 8 | 8 | 6 | 4 | 2 |
| Feature.34 (bit) | 8 | 6 | 6 | 4 | 2 |
| Feature.37 (bit) | 8 | 6 | 4 | 4 | 2 |
| Feature.40 (bit) | 8 | 8 | 4 | 2 | 2 |
| Classifier (bit) | 8 | 6 | 4 | 2 | 2 |
| Accuracy | 90.92% | 89.58% | 89.16% | 89.11% | 87.54% |
| Power (mW) | 21.4 | 14.1 | 10.6 | 8.5 | 7.0 |
| Latency (s) | 0.0049 | 0.00313 | 0.00202 | 0.00158 | 0.00125 |

# 5 Power Prediction

In energy harvesting systems, a key issue is to give the system the ability of predicting the future harvested power in advance. In this work, if we can understand the power level in the next power cycle in advance, we may be able to transfer the incomplete computations to the next power cycle, instead of discarding them. For high sample rate power sources, it is of great value, because only a few inferences can be completed in one power cycle and it is worthwhile to save one more inference through an early action.

Power prediction is not easy for multi-level situations. Prior efforts have presented that the existing neural network algorithms can predict three harvesting power levels with an accuracy of 80%. However, those techniques are not able to predict greater power levels while maintaining a high accuracy. This work augments an existing machine learning-based approach to predict 5-/6-level power with a very high accuracy in order to accommodate 4/5 quantization configurations. This can be realized through a proposed SMOTE algorithm to re-sample and provide more friendly training data set. Our augmented power predictor can achieve an accuracy of up to 90% for several power sources.

## 5.1 NN-based power prediction

In this work, we use a lightweight full-connected neural network (NN) algorithm to do power prediction.

**Feature extraction** To train the NN algorithm of the power predictor, the following parameters are used for training inputs and output.

(1) *Power level classification*: Recalling that the ambient energy keeps changing as shown in Figure 1, so we need to partition power levels to indicate different favorable quantization schemes. In this work, we address a six-level classification scenario.

(2) *Energy intensity*: The energy intensity indicates the strength of the power signal. It is calculated by the product of power and sample rate.

(3) *Average energy intensity*: Corresponding to each energy intensity of a power cycle, this parameter gives the average value of its former five energy intensity. It can smooth the instantaneous changes on the power trace.

(4) *Energy standard deviation*: For each power cycle, this parameter denotes the standard deviation of the energy intensity of the former five samples, showing the stability of energy changes.

Before data are fed to the neural network, normalization is done as shown in Equation 3 to improve the convergence.

$$F_{nor} = (F_{org} - Min(F_{org}))/(Max(F_{org}) - Min(F_{org})) \tag{3}$$

Here $F_{nor}$ and $F_{org}$ denote the normalized feature vector and the original feature vector, respectively.

**NN structure** The NN structure used for our power predictor is shown in Figure 4. The deep neural networks used in our power predictor can be illustrated by the layers of input layer, hidden layer, and output layer.

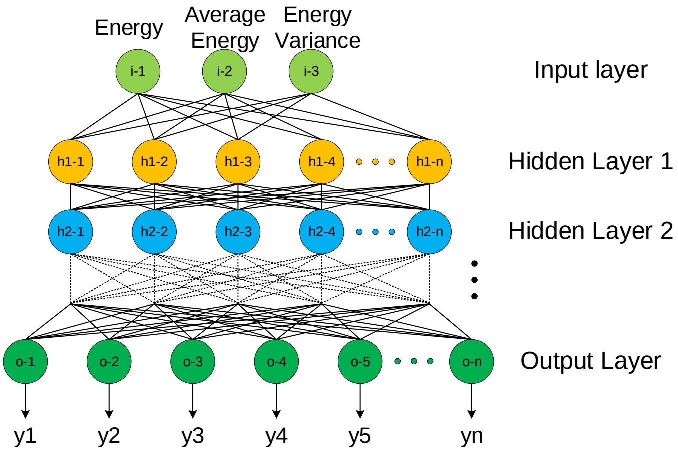

**Fig. 4.** NN structure of the power predictor.

The input layer receives the feature vectors of the energy intensity, average energy intensity and energy standard deviation.

The hidden layer is the key part of the deep neural network as shown in Equation 4. The term $x_{l,n}$ denotes the output of the $n$-$th$ neuron in the $l$-$th$ layer, $w_{l,n,s}$ denotes the $s$-$th$ weight value of the $n$-$th$ neuron in the $l$-$th$ layer, and $b_{l,n}$ denotes the offset of the nth neuron in the $l$ layer. The hidden layer contains the activation function, $s_l$ denotes the number of weights of the $l$-$th$ layer, and $f_{act}$ is the activation function. This work uses *Sigmoid* as the activation function as shown in Equation 5. In this work, we set 30 neurons for the first hidden layer, and 10 neurons for the remaining hidden layers.

$$x_{l,n} = f_{act}(\sum_{s=0}^{S_t} x_{l-1,s} \times w_{l,n,s} + b_{l,n}) \tag{4}$$

$$sig(x) = \frac{1}{1+e^{-x}} \tag{5}$$

When dealing with multi-classification problems, *one-hot* is used as the output of the neural network. In other words, only one of the N-bit status registers outputs 1, while the others all output 0. The output is processed by the *Softmax* algorithm as shown in Equation 6. The value of $S_i$ denotes the normalized probabilities of the $i$-$th$ class. The maximal $S_i$ represents the inferred class.

$$S_i = \frac{e^i}{\sum_j e^j} \tag{6}$$

### 5.2 Augmented power prediction

As discussed before, if we directly apply the existing neural network algorithm to predict the multi-level (¿3) power, the accuracy is not ideal. For multi-level power prediction, we find that the uneven data distribution is the cause of the accuracy problem. Motivated by this, this paper proposes to use the SMOTE algorithm to pre-process the data set and then obtain friendly training data. Finally, a high prediction accuracy can be obtained. The training, validation and accuracy assessment are implemented in the *Keras* machine learning framework.

**SMOTE algorithm** As Figure 1 depicts, most power distributions are significantly uneven. Take the *WiFi-office* power as an example as shown in Table 3, the data sets are very unevenly distributed in different power levels. The harvested power numbers are pretty large in the 1st level while very small in the last level, showing a variance of more than 100×. The imbalance property may cause the model to over-fit and induce low prediction accuracy.

**Table 3.** Power level range and sample number of *WiFi-office* source

| Power Level | Upper bound ($\mu$W) | Lower bound ($\mu$W) | sample number |
|---|---|---|---|
| 1 | 920 | 400 | 385914 |
| 2 | 1440 | 921 | 30954 |
| 3 | 1960 | 1441 | 3299 |
| 4 | 2480 | 1961 | 1623 |
| 5 | 3000 | 2481 | 36 |

This paper exploit the SMOTE algorithm to upsample the data and increment the small data set [2]. The SMOTE algorithm can well solve the problem of imbalanced data sets. The underlying principle is to create new data from the original data so as to increase the distribution density of low-density group.

Algorithm 1 gives the pseudo code of the SMOTE algorithm. First, we count the sample number of each group (Line 1-3). For each low-density group, we calculate the Euclidean distances between a randomly selected sample and any other samples, and then collect the five nearest samples (Line 4-8). For each sample pair, we generate a new sample between them following the Equation 7 (Line 11-17). And so forth, we can obtain a density-balanced data samples for re-training.

$$X_{new} = x + rand(0,1) \times (\hat{x} - x) \tag{7}$$

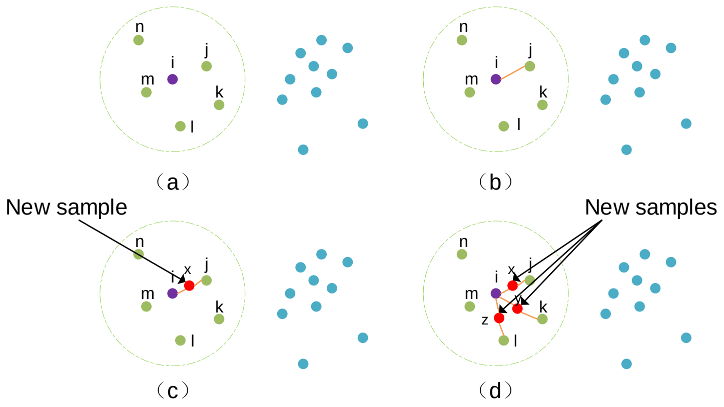

**Fig. 5.** New sample data generation in the SMOTE algorithm.

**Algorithm 1** New sample data generation with the SMOTE algorithm.

```
Input:
    samples(e, e_eva, e_var, label);
    label = [0, m];
    e : energy;
    e_ave : average energy;
    e_var : energy variation;
    k : number of nearest neighbors;
    K : increment rat;
Output:
    smote_samples(e, e_eva, e_var, label);
    find the minority;
count samples group by label;
majo = the amount of majority of samples;
save the rest samples as mino_samples[m-1];
for i = 0 to m-1 do
    randomize the mino_samples[i];
    for each ITEM in mino_samples[i] do
        compute k nearest neighbors for ITEM, and save as neighbor_arr;
        populate(ITEM, i, neighbor_arr);
    end for
end for
Define populate{ITEM, i, neighbor_arr}
while count(mino_samples[i]) ¡= R*majo do
    Choose a random number between 1 and k, set it as x;
    dist = neighbor_arr[x] - ITEM;
    gap = random(0, 1);//random from 0 to 1
    synthetic = ITEM + gap * dist;
    mino_samples[i].add(synthetic)//add the synthetic node to mino_samples
end while
```

**Cross validation** *Ten-Folder* cross validation is used to evaluate the fitting of the to-be-inferred data sets. Specifically, the power traces are divided into ten parts. Each time nine parts are selected as the training set while the remaining one part is for test. Meanwhile, the SMOTE algorithm is applied on the training set to increase the data density of minority groups. Finally, we use the test set to evaluate the model accuracy.

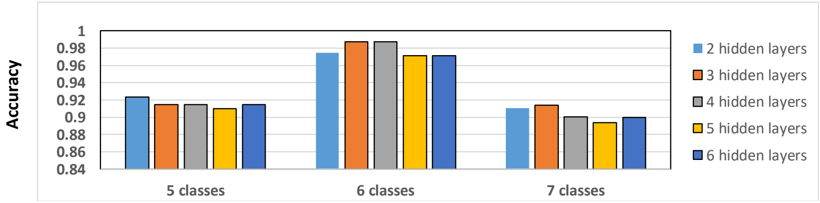

**Fig. 6.** Prediction accuracy of different power levels with 2∼6 hidden layers.

Figure 6 shows the final accuracy of the *Softmax* output layer. It can be seen that more than 90% accuracy can be achieved for 5-/6-/7-level classification with the proposed augmented power predictor, even though that the accuracy is only ∼85% for 3-level prediction. The proposed guarantee high prediction accuracy of power traces in this work.

## 6 Experiments

In this section, we first describe our experimental settings and then move onto discussing our results & findings to demonstrate the plausibility of our idea.

### 6.1 Experimental settings

For our experiments, we use CIFAR10 [9] dataset with VGG16 [18] architecture. CIFAR10 is an image classification dataset of 10 classes with 50000 training and 10000 validation images of size $3*32*32$. VGG16 is a popular DNN architecture consisting of 16 layers (15 convolutional and 1 fully connected). However, to adjust the network for CIFAR10 dataset, two convolution layers are dropped and the size of final classifier is changed.

### 6.2 Results

**Learning Curve** The layer-wise quantization scheme is best illustrated by figure 7 Starting with a fully converged network, the quantization scheme selects one layer at a time, then quantizes it followed by a recovery step. This behavior is very well reflected in the figure. The valleys indicate the accuracy loss after each quantization step and the peaks indicates the recovery following that step. In this experiment we start with uniform 8-bit precision network and then gradually quantize it to 2-bit. In between uniform 8-bit to uniform 2-bit networks, we take few snapshots each of which is basically a mixed precision network. The compute power requirements of these networks decreases gradually as the average precision decreases. Accuracy of these networks follows the same trend which is why we have been able do power aware inference where the accuracy depends on the available power.

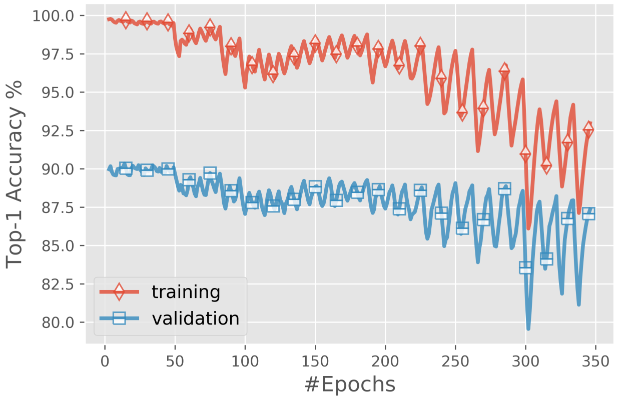

**Fig. 7.** Learning curve for mixed precision models. The zigzag pattern represents the quantization and subsequent recovery steps.

**System-level results** In the system-level simulation, we built an in-house coarse-grain simulator to evaluate the throughput, energy efficiency and average accuracy on top of different power sources as shown in Figure 1. The parameters in Table 2 are fed into the simulator. In order to support the VGG computation, we assume there are multiple harvesters together to power the ReRAMs. The harvester number for the sources of *Solar, Thermal, TV-RF, WiFi-home* and *WiFi-office* are 18, 8, 4, 25 and 20, respectively. Seven quantization versions are evaluated in total: *Network-1, Network-2, Network-3, Network-4, Network-5, Network-adaptive* and *Network-predictive*. For the first five versions, we use fixed quantization solution as shown in Table 2. The *Network-adaptive* version employs dynamic quantization solutions in different power cycles. That is, we always select the quantization network structure with as high as possible accuracy as long as the harvested power in the power cycle can meet the requirement. The *Network-prediction* perform the same dynamic quantization policy as the *Network-adaptive* based on the predicted power traces.

Figure 8 shows the normalized results of throughput, energy efficiency and energy utilization and output accuracy in (a), (b), (c) and (d) respectively. The normalized results are referred to the *Network-adaptive* version. Further, Table 4 gives the absolute values of the *Network-adaptive* version.

**Table 4.** Throughput, Energy efficiency, Energy utilization and Average accuracy of *Network-adaptive*.

| Power source | Throughput (#inference/s) | Energy efficiency (#inference/J) | Energy utilization | Average accuracy |
|---|---|---|---|---|
| Solar | 12.49 | 2810.6 | 1.4% | 89.35% |
| Thermal | 25.6 | 1241.6 | 42.0% | 90.17% |
| TV-RF | 17.6 | 842.3 | 44.3% | 90.09% |
| WiFi-home | 6.09 | 1911.4 | 42.7% | 89.18% |
| WiFi-office | 44.4 | 7122.4 | 30.8% | 87.98% |

We can make the following observations and analyses from the results:

• It is not surprising that the *Network-5* version always achieves the best throughput because it incurs the shortest latency to complete an inference. However, this version also has the lowest accuracy due to the lowest data precision. It can be seen that the *Network-adaptive* version can achieve the best balance between the throughput and the accuracy. The reason is that the *Network-adaptive* version can well trade throughput of accuracy by adopting dynamic quantization configurations. Consistent with the throughput observation, the proposed *Network-adaptive* can achieve a balanced trade off between energy efficiency and accuracy.

• For each power source, the *Network-adaptive* version always achieves the highest energy utilization. This is because we apply the policy of adopting the quantization degree of the VGG network structure to best meet the the available power level. As a result, the proposed policy can make the best effort to convert the harvested energy to the classification accuracy.

• Overall, the *Network-adaptive* version can manage the tradeoff among throughput, energy efficiency and output accuracy by efficiently utilizing the harvested energy.

As discussed in Section 5, the power predictor can direct us to proactively configure the last incomplete inference's computation with an appropriate network structure, so that the incomplete computation results can be transferred to the next power cycle under a *High-to-Low* power level transition situation. Further, Figure 9 shows the increase in number of inferences when using the power predictor. It is interesting to have the following observations by comparing to the results without power prediction.

• The increased number of inferences for the power sources of *Solar* and *WiFi-office* is much smaller than that of the power sources of *Thermal, TV-RF* and *WiFi-home*. This can be explained by the fact that the power level transitions, especially the *High-to-Low* power level transitions, occur much less

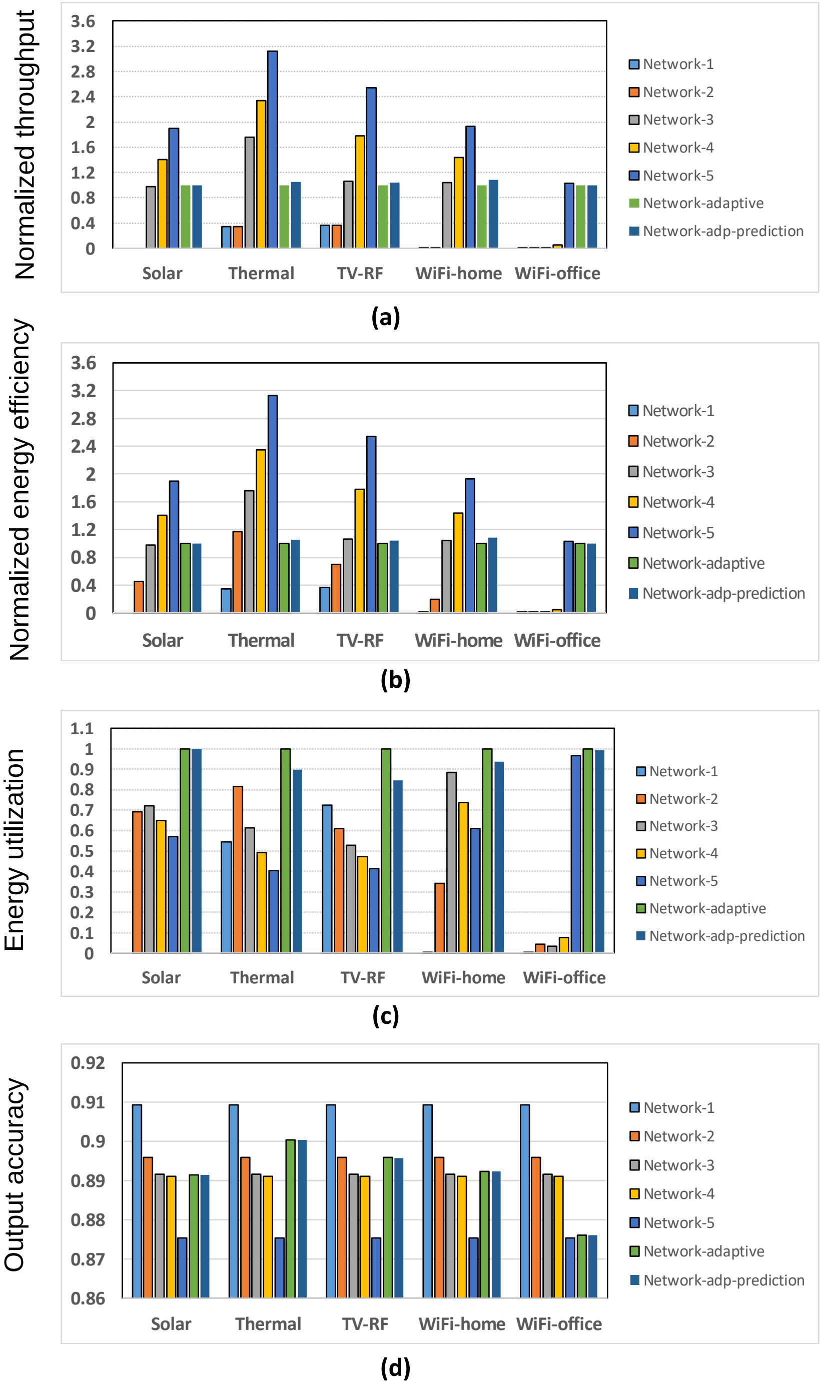

**Fig. 8.** (a) Throughput, (b) Energy efficiency, (c) Energy utilization and (d)Average accuracy across the power sources.

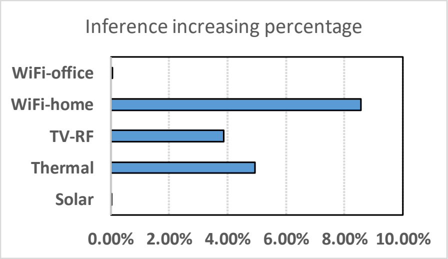

**Fig. 9.** The percentage of inference increasing with power prediction.

with the former two power sources. Therefore, there is less chance to benefit from the incomplete computation saving.

• The energy utilization with the *Network-adaptive* version is always higher than that with the *Network-prediction* version. However, it does not imply that the performance of the *Network-adaption* is better as shown in Figure 8. The real underlying reason is that the computations of the last incomplete inference under the *High-to-Low* power level transitions will be terminated early, directing by the power predictor. This is why we can see paradoxical results from the angles of throughput and energy utilization.

## 7 Conclusion

With the increasing deployment of deep neural networks in edge devices, their operation in energy-harvesting environments with varying power-levels becomes a necessity. Further, in applications without an energy storage device, the ability to dynamically adapt the complexity of the deep neural network becomes essential to best utilize the incoming harvested power. This work deploys a quantized deep network with varying degrees of quantization to meet varying degrees of available power. At execution time, we vary the instantiated network configuration to match the available power. Additionally, we have proposed an approach that predictively ensure that partial results from the network are best retained when power levels change. The results from this work show that the proposed adaptive quantization scheme can exploit the energy to achieve as much as possible high accuracy and maintain good throughput.

## 8 Acknowledgements

This work was supported in part by Semiconductor Research Corporation (SRC), Center for Brain-inspired Computing (C-BRIC), Center for Research in Intelligent Storage and Processing in Memory (CRISP), National Natural Science Foundation of China [NSFC Project No. 61872251] and Beijing Advanced Innovation Center for Imaging Technology.

## References

1. Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Aciq: Analytical clipping for integer quantization of neural networks. *arXiv preprint arXiv:1810.05723*, 2018.
2. Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
3. P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory, June 2016.
4. Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.
5. Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.
6. Zhen Dong, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. *arXiv preprint arXiv:1905.03696*, 2019.
7. Z. Gong, K. Qiu, D. Zhou, W. Chen, Y. Xu, X. Shi, and Y. Liu. Retention state-enabled and progress-driven energy management for self-powered nonvolatile processors. In *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–8, 2017.
8. N. Jao, A. K. Ramanathan, A. Sengupta, J. Sampson, and V. Narayanan. Programmable non-volatile memory design featuring reconfigurable in-memory operations. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2019.
9. Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
10. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
11. Kaisheng Ma, Xueqing Li, Yongpan Liu, John Sampson, Yuan Xie, and Vijaykrishnan Narayanan. Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 670–675, 2015.
12. Kaisheng Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 526–537, 2015.
13. Szymon Migacz. 8-bit inference with tensorrt. In *GPU Technology Conference*, volume 2, page 7, 2017.
14. Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.
15. K. Qiu, W. Chen, Y. Xu, L. Xia, Y. Wang, and Z. Shao. A peripheral circuit reuse structure integrated with a retimed data flow for low power rram crossbar-based cnn. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1057–1062, 2018.

16. Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
17. A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 14–26, June 2016.
18. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
19. Submitted; under review. Learning to quantize deep neural networks: A competitive-collaborative approach.
20. Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.
21. Amir Yazdanbakhsh, Ahmed T Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, and Hadi Esmaeilzadeh. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*, 2018.
22. Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019.
23. M. Zhao, K. Qiu, Y. Xie, J. Hu, and C. J. Xue. Redesigning software and systems for non-volatile processors on self-powered devices. In *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2016.
24. Shu-Chang Zhou, Yu-Zhi Wang, He Wen, Qin-Yao He, and Yu-Heng Zou. Balanced quantization: An effective and efficient approach to quantized neural networks. *Journal of Computer Science and Technology*, 32(4):667–682, 2017.
25. Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.