HitNet: Hybrid Ternary Recurrent Neural Network

Peiqi Wang[1,2,4], **Xinfeng Xie**[4], **Lei Deng**[4], **Guoqi Li**[3], **Dongsheng Wang**[1,2]*, **Yuan Xie**[4]

[1]Department of Computer Science and Technology, Tsinghua University
[2]Beijing National Research Center for Information Science and Technology
[3]Department of Precision Instrument, Tsinghua University
[4]Department of Electrical and Computer Engineering, University of California, Santa Barbara
wpq14@mails.tsinghua.edu.cn, {liguoqi, wds}@mail.tsinghua.edu.cn
{xinfeng, leideng, yuanxie}@ucsb.edu

Abstract

Quantization is a promising technique to reduce the model size, memory footprint, and computational cost of neural networks for the employment on embedded devices with limited resources. Although quantization has achieved impressive success in convolutional neural networks (CNNs), it still suffers from large accuracy degradation on recurrent neural networks (RNNs), especially in the extremely low-bit cases. In this paper, we first investigate the accuracy degradation of RNNs under different quantization schemes and visualize the distribution of tensor values in the full precision models. Our observation reveals that due to the different distributions of weights and activations, different quantization methods should be used for each part. Accordingly, we propose HitNet, a hybrid ternary RNN, which bridges the accuracy gap between the full precision model and the quantized model with ternary weights and activations. In HitNet, we develop a hybrid quantization method to quantize weights and activations. Moreover, we introduce a sloping factor into the activation functions to address the error-sensitive problem, further closing the mentioned accuracy gap. We test our method on typical RNN models, such as Long-Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Overall, HitNet can quantize RNN models into ternary values of {-1, 0, 1} and significantly outperform the state-of-the-art methods towards extremely quantized RNNs. Specifically, we improve the perplexity per word (PPW) of a ternary LSTM on Penn Tree Bank (PTB) corpus from 126 to 110.3 and a ternary GRU from 142 to 113.5.

1 Introduction

Recurrent Neural Networks (RNNs) yield great results across many natural language processing applications, including speech recognition, machine translation, language modeling, and question answering [1, 2, 3, 4, 5]. The emergence of various RNN architectures, such as Long-Short-Term Memory (LSTM) [6] and Gated Recurrent Units (GRU) [7], has achieved state-of-the-art accuracy in these applications. In order to further improve the model accuracy, researchers always build deeper internal recurrence by increasing the number of recurrent sequence length, hidden units, or layers. However, these methods usually introduce large models resulting in practical hardness, especially on embedded devices due to inferior resources for computation and memory. Model compression is a promising method to alleviate aforementioned problems. There exist several techniques to reduce the amount of model weights, such as low-rank approximation [8], sparsity utilization [9, 10, 11], and

[*] Corresponding Author

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

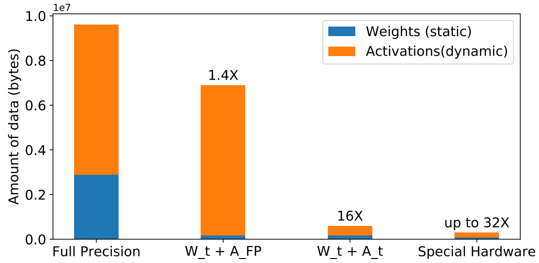


Figure 1: Theoretical memory saving for an LSTM model through quantization. W_t denotes ternary weights, A_t denotes ternary activations, and A_FP denotes full precision activations.

data quantization [12, 13]. Among these compression techniques, quantizing data into lower bitwidth is quite promising.

In previous studies that attempt to quantize RNNs to extreme low-bit precision, they always focus on weights while retaining activations in full precision [14, 15, 16]. Activations include all dynamic tensors during processing, and weights are learned and fixed after training. As shown in Figure 1, quantizing weights into ternary values {-1, 0, 1} can only save 1.4× memory consumption in the training phase. While quantizing both weights and activations can achieve up to 16× memory saving. Ternary values can be further compressed by using only 1 bit to represent -1 and +1 without storing the zero values on some specialized devices [17]. Memory requirements for this further compression can be reduced up to 32×. For the inference phase, although weights occupy the most of memory, the quantization of activations can further simplify the floating-point computations. Under the ternary scenario, the costly full precision matrix arithmetic can be transformed into simple logical operations, which are friendly for efficient implementations on customized hardware[18].

Quantization achieved impressive success and satisfied the accuracy requirement on convolutional neural networks (CNNs) even with only ternary weights and activations [19, 20, 21, 22, 23], while the results on RNNs are still unsatisfactory. The huge accuracy gap between the quantized models and the original models prevents RNNs from using extreme low-bit precision. This motivates us to investigate the reason for huge loss of accuracy in aggressive quantizations on RNNs. There are several differences between RNNs and CNNs. These differences could cause the practical success of state-of-the-art quantization methods on CNNs while similar methods cannot work well on RNNs. One major reason is the recurrent structure in RNN models. Activations in RNNs are iteratively updated along the temporal dimension through the same network structure and weights. This structure makes quantizing RNNs easier to cause an error accumulation. In addition, the strongly nonlinear activation functions (e.g. $sigmoid$ or $tanh$) in RNN are more sensitive to small errors compared to the piecewise linear functions in CNNs (e.g. $ReLU$). Although previous studies [14, 24, 25] tried to bridge this gap in various ways, the accuracy of quantized RNNs, especially the ones using extremely low-bit precision, is far from being satisfied.

In this paper, we analyze the value distributions of weights and activations to reveal the reason behind the accuracy degradation. Based on our observation, we propose a hybrid ternary quantization method called HitNet to significantly bridge the accuracy gap. In HitNet, we take full advantages of various quantization methods. Specifically, we use threshold ternary quantization method for weight quantization and Bernoulli ternary quantization method for activation quantization. Moreover, to further decrease the accuracy gap between the quantized models and the original models, we introduce a sloping factor into the activation functions to address the error-sensitive problem.

Our contributions can be summarized as follows:

- To the best of our knowledge, our work is the first to present a comprehensive analysis regarding the impact of different quantization methods on the accuracy of RNNs.
- Motivated by the above analysis, we propose a hybrid ternary quantization method, termed as HitNet, which adopts different quantization methods to quantize weights and activations according to their statistical characteristics.
- To further narrow the accuracy gap, we introduce a sloping factor into the activation functions. HitNet achieves significant accuracy improvement compared with previous work, like the

perplexity per word (PPW) of a ternary LSTM on Penn Tree Bank (PTB) corpus decreasing from 126 to 110.3 and a GRU from 142 to 113.5.

2 Related Work

Model compression is widely used to reduce the size of RNN models with redundancy, like low-rank approximation [8], sparsity utilization [9, 10, 11], and data quantization [12, 13]. All of these approaches are independent and we focus on data quantization in this work. The biggest challenge of quantization is to maintain application accuracy. Some methods [24, 26, 19] propose various transformation functions before uniform quantization or design different threshold decision approaches, trying to balance the distribution of quantized data and utilize limited quantized states efficiently. Zhou *et al.* [26] recursively partitioned the parameters by percentiles into balanced bins before quantization to balance data distribution. He *et al.* [24] introduced a parameter dependent adaptive threshold to take full advantage of the available parameter space. A few prior studies [14, 26] introduce preconditioned coefficients or employ various transformation functions before and after quantization to decrease the accuracy degradation. Some others formulated quantization into an optimization problem and then used different approaches to search optimal quantization coefficients, like greedy approximation [27] and alternating multi-bit quantization [25]. A few studies [16, 28] added extra components to reorganize the original RNN architecture and fine-tune the quantized models, but they introduce extra computation resulting in negative impacts on the performance for both inference and training phases. Most of these studies still present a big accuracy gap when using extreme low-bit precision. Our HitNet addresses this problem by quantizing both weights and activations of RNNs into ternary states with impressive accuracy improvement compared to previous work.

3 Analysis of Quantized RNNs

In this section, we provide a comprehensive study on the accuracy loss of quantized RNNs under different state-of-the-art quantization methods. We take LSTM as a case study to understand how these quantization methods work. The basic unit is the LSTM cell which can be described as

$$\begin{aligned} i_t, f_t, g_t, o_t &= \sigma(W_x x_t + b_x + W_h h_{t-1} + b_h) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= o_t * \sigma(c_t) \end{aligned} \tag{1}$$

where σ denotes the activation function. The computation contains: input gate i_t, forget gate f_t, candidate cell state g_t, output gate o_t, cell state c_t, and hidden state h_t. The activation function is $\sigma(x) = sigmoid(x) = 1/(1+e^{-x})$ for i_t, f_t, and o_t, and $\sigma(x) = tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ for g_t and h_t. We refer to the first four items as gate computations in the following sections due to the similar formats. Weights in gate computations can be separated into two parts: one is related to current input information (i.e. $W_x x_t + b_x$) and the other is related to previous state (i.e. $W_h h_{t-1} + b_h$). Here W_x (W_h) merges W_{xi} (W_{hi}), W_{xf} (W_{hf}), W_{xg} (W_{hg}), and W_{xo} (W_{ho}) for clarity. Activations represent the intermediate data propagated in the forward pass.

Existing quantization methods can be classified into three categories: uniformed quantization [26, 25], deterministic quantization (threshold-based quantization) [19, 21], and stochastic quantization [12, 13]. We use $Q(X)$ to represent the quantization method, where X denotes the input tensor. Hence, the computation of its corresponding gradient in the back propagation is governed by

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Q}\frac{\partial Q}{\partial X} \tag{2}$$

where E denotes the loss function. For the non-differentiable part in $Q(X)$, we adopt the straight-through estimator (STE) method [29]. An STE actually can be regarded as an operator owning arbitrary forward operations but unit derivative in the backward pass. In the rest of this section, we will conduct detailed analysis for different quantization methods. All evaluations in this section adopt an LSTM model with one hidden layer of 300 units. The sequence length is set to 35, and it is applied on Penn Tree Bank (PTB) corpus [30]. The accuracy is measured in perplexity per word (PPW), and a lower value in PPW means a better accuracy.

Table 1: The testing perplexity per word (PPW) on quantized LSTM over PTB by using different quantization methods. W_2bit represents 2-bit weights, W_FP represents full precision weights, W_t represents ternary weights. The symbol A for activations has the similar meaning. UQ denotes Uniform Quantization, TTQ denotes Threshold Ternary Quantization, and BTQ denotes Bernoulli Ternary Quantization.

(a) UQ

	W_2bit	W_FP
A_2bit	638.9	108.9
A_FP	103.3	97.2

(b) TTQ

	W_t	W_FP
A_t	125.5	108.7
A_FP	98.6	97.2

(c) BTQ

	W_t	W_FP
A_t	189.5	108.3
A_FP	110.9	97.2

3.1 Uniform Quantization

The basic and straightforward way to quantize a full precision value x to a k-bit fixed-point one is the uniform quantization (UQ) [31], which can be implemented as

$$q_k(x) = \frac{round[(2^k - 1)x]}{2^k - 1}. \tag{3}$$

This quantization essentially reduces the representation precision rather than changing the dynamic range, thus x needs to be mapped into [0,1] before quantization and then recovered back to the original range. Thus, the practical UQ is usually modified to

$$UQ(X) = 2max(|X|)[q_k(\frac{X}{2max(|X|)} + \frac{1}{2}) - \frac{1}{2}]. \tag{4}$$

In order to observe how UQ affects the accuracy of RNNs, we evaluate a 2-bit quantized LSTM on PTB and the results are shown in Table 1(a). We find that using 2-bit representation is relatively acceptable for just quantizing either weights or activations by UQ method. However, the accuracy degrades dramatically when we apply this quantization method on both weights and activations.

3.2 Threshold Ternary Quantization

Deterministic quantization assigns the quantized states arbitrarily according to an estimated threshold, and threshold ternary quantization (TTQ) [21] is such a method to convert full precision to ternary values. To retain as much information as possible, an estimated threshold θ and an non-negative scaling factor α are often used to minimize the quantization error. The TTQ method is implemented as

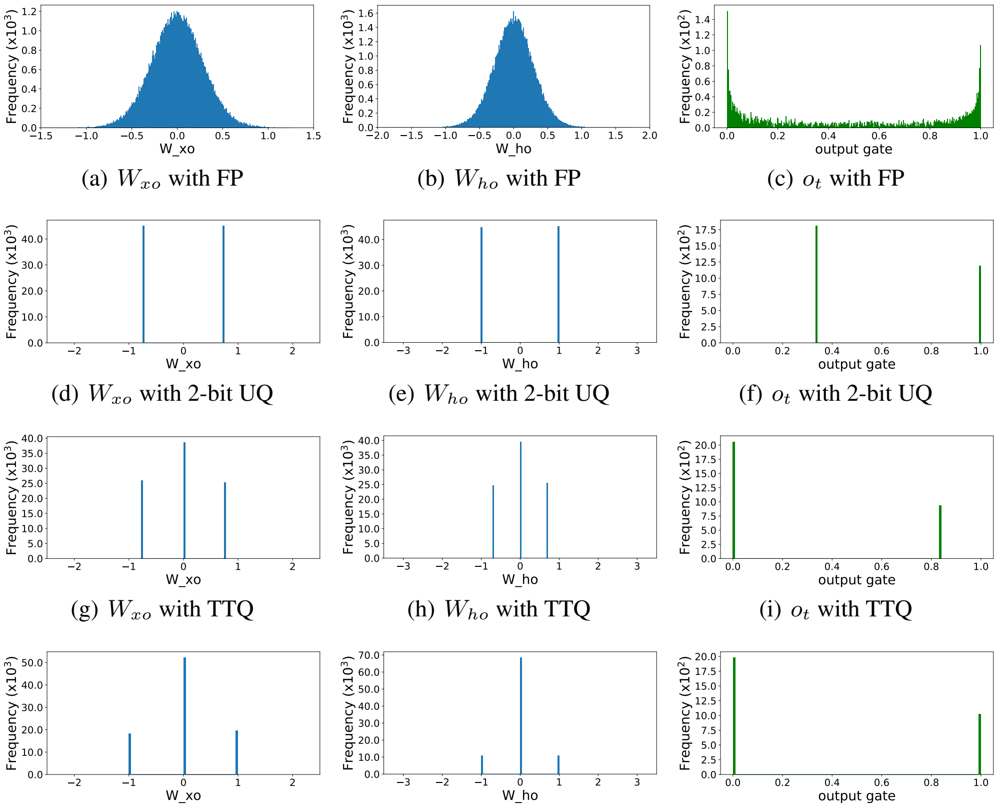
$$f(x) = \begin{cases} +1 & x > \theta \\ 0 & |x| \leq \theta \\ -1 & x < -\theta \end{cases} \tag{5}$$

$$TTQ(X) = \alpha f(X). \tag{6}$$

Assume that the data in RNNs obey a normal distribution $X \sim N(0, \sigma^2)$. We define a set $\Theta = \{i||x_i| > \theta\}$, and I_Θ be the indicator function of the set Θ. Therefore, approximately solving an optimization problem of minimizing L2 distance between the original full-precision values and the quantized ternary values, we can get an sub-optimal threshold θ and the scaling factor α [21] calculated by

$$\begin{aligned} \theta &\approx \frac{2}{3}E(|X|) \\ \alpha &\approx \frac{1}{|\Theta|}\sum |x| I_\Theta(x). \end{aligned} \tag{7}$$

We quantize both weights and activations in RNNs to ternary values using TTQ and the evaluation results are shown in Table 1(b). Quantizing weights has small accuracy loss compared to the original model while quantizing activations suffers from larger accuracy loss. Quantizing both of them also shows large accuracy degradation although it is much better than the result in the UQ case.



(a) W_{xo} with FP (b) W_{ho} with FP (c) o_t with FP

(d) W_{xo} with 2-bit UQ (e) W_{ho} with 2-bit UQ (f) o_t with 2-bit UQ

(g) W_{xo} with TTQ (h) W_{ho} with TTQ (i) o_t with TTQ

(j) W_{xo} with BTQ (k) W_{ho} with BTQ (l) o_t with BTQ

Figure 2: The distribution of weights (i.e. W_{xo}, W_{ho}) and activations of output gate (i.e. o_t) in LSTM under different quantization methods. (a)-(c) show the distributions in full precision (FP), (d)-(f) use the 2-bit uniform quantization (UQ), (g)-(i) use the threshold ternary quantization (TTQ), (j)-(l) use the Bernoulli ternary quantization (BTQ). The model used here is an LSTM trained on Penn Tree Bank (PTB) corpus.

3.3 Bernoulli Ternary Quantization

Stochastic quantization performs well in CNNs [13, 23]. Specifically, for the ternary case, the quantized value is sampled from a Bernoulli distribution. Thus, a Bernoulli ternary quantization (BTQ) is formulated as

$$f(x) = \begin{cases} +1 & \text{with probability } p \text{ if } x > 0 \\ 0 & \text{with probability } 1-p \\ -1 & \text{with probability } p \text{ if } x < 0 \end{cases}, \text{ where } p \sim Bernoulli(|x|) \tag{8}$$

$$BTQ(X) = f(X). \tag{9}$$

We also present the quantization test for BTQ, as shown in 1(c). For the weights outside the range of [-1,1], we apply a $tanh$ function for magnitude scaling before quantization [19]. We can see that the activation quantization obtains slightly better accuracy than quantizing weights. Quantizing both of them still suffers from unacceptable accuracy loss although it is also better than the UQ result.

3.4 Characteristic Analysis

In order to understand the accuracy gap between the original full precision and quantized low-bit precision with different quantization schemes, we first visualize the distributions of weights (i.e. W_{xo} and W_{ho}) and activations (i.e. o_t) in Figure 2. Here we just take output gate as an analysis example. The distributions of full-precision data are shown in Figure 2(a)-2(c), and the rest figures represent models using various aforementioned quantization methods. These distributions are consistent with our prior knowledge in terms of weights. The distribution of weights in RNNs follows the normal distribution although the range of W_{ho} is slightly different from W_{xo}, which is shown in Figure 2(a) and Figure 2(b). However, the distribution of activations is very different from weights. The activation range is limited within [0,1] due to the activation functions. More importantly, most of the values locate near to the two poles instead of the middle of this range.

The distributions of the quantized data under 2-bit UQ scheme are shown in Figure 2(d)-2(f). It is found that this quantization method does not fully utilize the representation ability of 2 bits with 4 states and most values in this tensor concentrated to 2 states due to the unbalanced data distribution (other 2 states are too small to see in the figures). For quantized activations shown in Figure 2(f), only 2 states are valid due to the activation functions. This greatly degrades the model's expressive power. Some previous studies [26, 32] try to balance the distribution of quantized values to sufficiently utilize these 4 states for better representation. Usually, they introduce new transformation functions before mapping them into a 2-bit value to force them into a balanced distribution. However, such methods change the original data distribution to some extent, resulting in an unavoidable accuracy gap between the quantized model and the original model.

Figure 2(g)-2(i) show the distributions of quantized data under TTQ scheme, which provides good explanation for the comparable accuracy of just quantizing the weights. First, the distribution of quantized weights is similar to the original normal distribution. Second, all 3 states are fully utilized in the weight quantization. However, for the activation quantization, TTQ doesn't present good bipolar distribution as the original FP data. Therefore, although TTQ achieves better accuracy than the UQ, it still degrades the accuracy obviously when quantizing both weights and activations. Note that there are only 2 states in quantized activations, this is because we take o_t with only positive values as the example, which is similar to the following BTQ scheme.

Figure 2(j)-2(l) present the distributions under BTQ scheme. Different from TTQ, activations in BTQ present bipolar distribution similar to the original FP data. Through probabilistic sampling, it performs a better approximation of the middle data in the stochastic sense. However, although the range of values in weights is preserved, 3 ternary states are not balanced since the lack of global statistic information (e.g. θ and α in TTQ). Therefore, BTQ scheme still suffers from similar accuracy loss when quantizing both the weights and activations.

In summary, these distributions under different quantization schemes provide insightful observations:

- TTQ is preferable for weights. Among these three quantization schemes, the TTQ utilizes 3 states efficiently and presents a balanced distribution.
- BTQ approximates the bipolar distribution of the activations better, and the probabilistic sampling preserves the middle data in the stochastic sense.

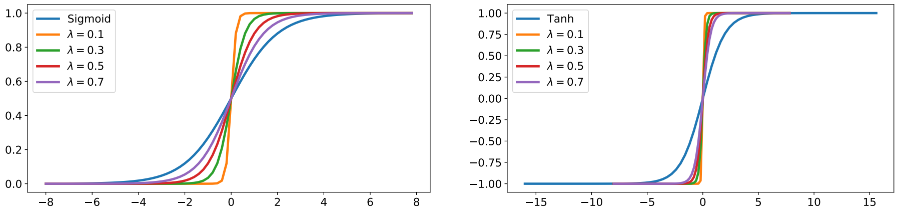
Although both of them still suffer from obvious accuracy degradation, the above comprehensive analysis motivates us to propose a hybrid ternary quantization method that takes full advantages of the different quantization schemes.

4 HitNet

Based on the above comprehensive comparisons, we propose HitNet in this section. We first introduce a constant sloping factor into activation to guide activations towards two poles, and then apply a hybrid ternary quantization method. Specifically, we use TTQ to quantize weights and BTQ to quantize activations to fully leverage the advantages of both. The hybrid quantization is governed by

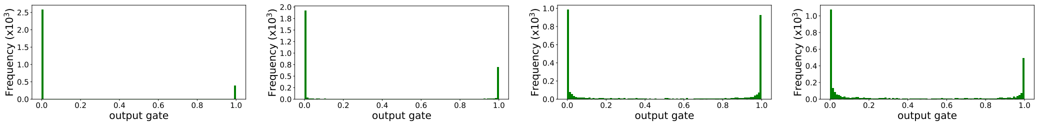
$$\begin{aligned} i_t, f_t, g_t, o_t &= \sigma(TTQ(W_x)x_t + TTQ(b_x) + TTQ(W_h)h_{t-1} + TTQ(b_h)) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= BTQ(o_t * \sigma(c_t)) \end{aligned} \tag{10}$$

We quantize the weights and hidden states since they are the only part involved in costly matrix operations, and the remained lightweight vector operations can be kept in high precision. In the training phase, the weights can be quantized right after each gradient step, while the activations need to be quantized every time the activation computation occurs. The threshold θ and the scaling factor α in TTQ are updated according to the real value distribution of the data to be quantized.



(a) $sigmoid(\cdot)$ function under different λ (b) $tanh(\cdot)$ function under different λ

Figure 3: Activation functions under different λ.



(a) $\lambda = 0.1$ (b) $\lambda = 0.4$ (c) $\lambda = 0.7$ (d) $\lambda = 1$

Figure 4: The distribution of output gate in LSTM with different sloping factor λ.

Because the activations present not only bipolar distribution but also with many middle data, none of the prior methods can avoid significant accuracy loss in an extreme low-bits quantization. The middle data between the two poles also play a non-negligible role on the model accuracy. If we can control the bipolarity of the activation distribution, we may have an opportunity to reduce the quantization error. Fortunately, prior work [33] proposed a temperature factor for the $sigmoid(\cdot)$ function in deep belief networks to control the activation distribution for better accuracy. It is also applicable to RNNs since the similar formats of activation functions. Therefore, we introduce a constant sloping factor in the activation functions to guide the activation distribution concentrating to the two poles as much as possible. Furthermore, based on previous analysis, we propose a hybrid ternary quantization method that uses TTQ for weight quantization and BTQ for activation quantization.

4.1 Hybrid Ternary Quantization

We introduce a constant sloping factor λ to RNN cells for controlling the activation distribution:

$$\hat{sigmoid}(x) = \frac{1}{1+e^{-\frac{x}{\lambda}}}, \quad \hat{tanh}(x) = \frac{e^{\frac{x}{\lambda}} - e^{-\frac{x}{\lambda}}}{e^{\frac{x}{\lambda}} + e^{-\frac{x}{\lambda}}}. \tag{11}$$

Figure 3 shows the variation tendency of activation functions under different λ. As prior work[33] pointed out that a properly small λ ($\lambda < 1$) can scale the propagated gradients thus solve the gradient vanish problem to some extent to improve the accuracy; whereas, too small λ will lead to extreme bipolar activation distribution that harms the model expressive power. Actually, a smaller λ can also produce more bipolar activation distribution in RNNs as shown in Figure 4. This will reduce the middle data, and consequently improve the ternarization error from BTQ. Therefore, it is possible to obtain better accuracy on the quantized models if we carefully configure a properly small λ. We apply the modified activation functions during both the training and inference phases and evidence this prediction through the experimental results in Table 2. We use the same LSTM model as the one used in Section 3 for comparing the existing quantization schemes. The accuracy under different λ for the full-precision model, quantized model, and the gap between them are clearly shown in the table. We get the following observations:

- A properly small λ (0.3-0.6) can slightly improve the accuracy of the full-precision model, while too small λ (<0.2) will obviously degrade the model functionality.
- A smaller λ generates more bipolar activation distribution that reduces the ternarization error. It usually brings better accuracy on the quantized model except for smaller λ with much worse baseline model in full precision. Jointly considering these two points, we configure λ=0.3-0.4 in this network.

Table 2: The testing PPW of LSTM models on PTB with different sloping factor λ.

λ	0.1	0.2	**0.3**	**0.4**	0.5	0.6	0.7	0.8	0.9	1
Full precision	123.5	100.1	**96.4**	**96.3**	95.8	96.7	97.1	97.1	97.1	97.2
HitNet	136.4	115.7	**110.6**	**110.3**	112.8	114.6	115.6	116.3	119.9	120.1
PPW degradation	-12.9	-15.6	-14.2	-14	-17	-17.9	-18.5	-19.2	-22.8	-22.9

Table 3: The testing PPW of HitNet on the PTB dataset.

Models	Kapur *et al.*[28]	He *et al.*[24]	Zhou *et al.*[26]	**HitNet(ours)**	FP(baseline)
LSTM	152.2	152	126	**110.3**	97.2
GRU	N/A	150	142	**113.5**	102.7

- Actually, our hybrid ternary quantization method using TTQ for weights and BTQ for activations can improve the accuracy of the quantized model even without introducing the sloping factor (i.e. λ=1) compared to previous schemes. This can be seen if we combine Table 2 and Table 1.

In conclusion, our HitNet tries to find the best quantization strategies for different components in RNN models benefit from the hybrid quantization paradigm. Besides, the introduced sloping factor can guide the activation distribution more bipolar without affecting the accuracy of the full-precision model (actually slight improvement) and bridges the accuracy gap between the full-precision model and the quantized model. For the rest of the conducted experiments, we adopt the sloping factor λ with the best accuracy (i.e. λ=0.4).

4.2 Overall Results

In this section, we evaluate the effectiveness of the proposed HitNet on language modeling with two typical RNN models (LSTM [6] and GRU [7]). The target is to predict the next word, thus we use perplexity per word (PPW) to measure the accuracy. A lower PPW means a better accuracy. For all RNN models in our experiments, a word embedding layer is used on the input side. We initialize the learning rate as 20 and decrease it by a factor of 4 at the epoch if the validation error exceeds current best record. The sequence length is set to 35 and the gradient norm is clipped into the range of [-0.25, 0.25]. In addition, we set the maximum epoch to be 40 and set dropout rate to be 0.2. The baseline of our experiment does not catch up with the state-of-the-art result, which is not within the scope of this paper. It can be further improved by optimizing the hyper-parameter configuration.

We first use the Penn Tree Bank (PTB) corpus [30], which contains 10K vocabulary. We conduct experiments for both LSTM and GRU, and provide comparisons to prior work. For a fair comparison, we use the model with one hidden layer of 300 hidden units, which is the same with previous counterparts. We use a batch size of 20 for training, and the results are shown in Table 3. We take the accuracy of 2-bit quantized models in previous work for comparison. In prior work [28], the result on 2-bit quantized GRU on PTB is not reported, thus we leave it. Although there is still an accuracy gap between our HitNet and the original full-precision model, HitNet outperforms existing quantization methods significantly. Furthermore, the 2-bit quantization in previous work actually has 4 states, while HitNet just has 3 states. Compared to these existing studies, HitNet achieves much better accuracy even using one less state for quantizing both the weights and activations.

We also apply our model to other larger datasets. Wikidata-2 is a larger corpus owning 33K vocabulary. It is roughly 2x larger in dataset size and 3x larger in vocabulary than PTB. We train both LSTM and GRU with one 512-size hidden layer and set the batch size to 50. Text8 is also a large corpus which is composed of a pre-processed version of the first 100 million characters from Wikipedia dump. We follow the same setting in [34] and the resulting vocabulary size is about 44K. On this dataset, we train the models with one hidden 1024-size layer and set the batch size to be 50. The results are shown in Table 4. HitNet is still able to achieve acceptable results on these larger datasets. We didn't compare the results with previous work because few of them focused on extremely low-bit quantization on these datasets.

Table 4: The testing PPW of HitNet on Wiki-2 and Text8 datasets.

Dataset	LSTM		GRU	
	HitNet	Full Precision	HitNet	Full Precision
Wikidata-2	126.72	114.37	132.49	124.5
Text8	169.1	151.4	172.6	158.2

Compared to prior work, HitNet demonstrates several advantages:

- HitNet adopts hybrid quantization methods for different data types to achieve better accuracy, providing a good match between the data distribution and the quantization scheme.
- HitNet introduces a sloping factor that can control the activation distribution to reduce the ternarization error.
- HitNet uses only 3 data states (less than 2 bits) that results in significant memory reduction via data compression and speedup via transforming full-precision matrix arithmetics to simple logic operations. Actually, the 0 values in HitNet do not need computation and can be skipped for further efficiency optimization on specialized platforms [9, 35, 36].

5 Conclusion and Future Work

We introduce HitNet, a hybrid ternary quantization method on RNNs to significantly bridge the accuracy gap between the full-precision model and the quantized model, outperforming existing quantization methods. The reason why previous extremely low-bit quantization on RNNs fails is comprehensively analyzed. Our observation reveals that the distributions of weights and activations are completely different. Prior work considers them equally leading to huge accuracy degradation in low-bit quantization. In HitNet, we quantize RNN models into ternary values $\{-1, 0, 1\}$ by using TTQ (threshold ternary quantization) for weight quantization and BTQ (Bernoulli ternary quantization) for activation quantization. In addition, a sloping factor is introduced into the activation functions, which guides the activation distribution more bipolar and further reduces the ternarization error. Our experiments on both LSTM and GRU models over PTB, Wikidata-2, and Text8 datasets demonstrate that HitNet can achieve significantly better accuracy compared to previous work, and resulting in $>$16x memory saving via only 3 data states and extreme compute simplification via only logic operations.

In future work, we aim to deploy HitNet on specialized hardware to get practical acceleration and efficiency improvement because many characteristics of HitNet can be leveraged to obtain benefits. For example, the full-precision data can be compressed to only 2 bits or less with $>$16x memory saving; all matrix arithmetics are converted into logic operations, which can accelerate the execution and save the energy consumption; all zero values do not need to be stored and computed for further optimization.

Acknowledgement

The authors were supported by the Beijing National Research Center for Information Science and Technology, the Beijing Innovation Center for Future Chip, and the Scalable Energy-efficient Architecture Lab (SEAL) in UCSB. This research was supported in part by the National Key Research and Development Plan of China under Grant No. 2016YFB1000303, the National Science Foundations(NSF) under Grant No. 1725447 and 1730309, and the National Natural Science Foundation of China under Grant No. 61876215. This work was also supported in part by a grant from the China Scholarship Council.

References

[1] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[3] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[5] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III, "A neural network for factoid question answering over paragraphs," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 633–644, 2014.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[8] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6655–6659, IEEE, 2013.

[9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pp. 243–254, IEEE, 2016.

[10] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 75–84, ACM, 2017.

[11] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

[12] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[13] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016.

[14] J. Ott, Z. Lin, Y. Zhang, S.-C. Liu, and Y. Bengio, "Recurrent neural networks with limited numerical precision," *arXiv preprint arXiv:1608.06902*, 2016.

[15] C. Z. Tang and H. K. Kwan, "Multilayer feedforward neural networks with single powers-of-two weights," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2724–2727, 1993.

[16] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[17] Y. van de Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. A. Talin, and A. Salleo, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," *Nature materials*, vol. 16, no. 4, p. 414, 2017.

[18] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pp. 10–14, IEEE, 2014.

[19] 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," *arXiv preprint arXiv:1606.06160*, 2016.

[20] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, pp. 525–542, Springer, 2016.

[21] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[22] C. Zhu, S. Han, H. Mao, and W. J. Dally, “Trained ternary quantization,” *arXiv preprint arXiv:1612.01064*, 2016.

[23] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, “Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework,” *Neural Networks*, vol. 100, pp. 49–58, 2018.

[24] Q. He, H. Wen, S. Zhou, Y. Wu, C. Yao, X. Zhou, and Y. Zou, “Effective quantization methods for recurrent neural networks,” *arXiv preprint arXiv:1611.10176*, 2016.

[25] C. Xu, J. Yao, Z. Lin, W. Ou, Y. Cao, Z. Wang, and H. Zha, “Alternating multi-bit quantization for recurrent neural networks,” *arXiv preprint arXiv:1802.00150*, 2018.

[26] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou, “Balanced quantization: An effective and efficient approach to quantized neural networks,” *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 667–682, 2017.

[27] Y. Guo, A. Yao, H. Zhao, and Y. Chen, “Network sketching: Exploiting binary structure in deep cnns,” *arXiv preprint arXiv:1706.02021*, 2017.

[28] S. Kapur, A. Mishra, and D. Marr, “Low precision rnns: Quantizing rnns without losing accuracy,” *arXiv preprint arXiv:1710.07706*, 2017.

[29] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.

[30] A. Taylor, M. Marcus, and B. Santorini, “The penn treebank: an overview,” in *Treebanks*, pp. 5–22, Springer, 2003.

[31] R. M. Gray and D. L. Neuhoff, “Quantization,” *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[32] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” in *International Conference on Machine Learning*, pp. 1737–1746, 2015.

[33] G. Li, L. Deng, Y. Xu, C. Wen, W. Wang, J. Pei, and L. Shi, “Temperature based restricted boltzmann machines,” *Scientific reports*, vol. 6, p. 19133, 2016.

[34] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, “Learning longer memory in recurrent neural networks,” *arXiv preprint arXiv:1412.7753*, 2014.

[35] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, “Cnvlutin: Ineffectual-neuron-free deep neural network computing,” in *ACM SIGARCH Computer Architecture News*, vol. 44, pp. 1–13, IEEE Press, 2016.

[36] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, “Cambricon-x: An accelerator for sparse neural networks,” in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pp. 1–12, IEEE, 2016.