ReLeQ: An Automatic Reinforcement Learning Approach for Deep Quantization of Neural Networks

Ahmed T. Elthakeb Prannoy Pilligundla Fatemehsadat Mireshghallah Amir Yazdanbakhsh*
Sicun Gao Hadi Esmaeilzadeh
Alternative **C**omputing **T**echnologies (**ACT**) Lab
University of California San Diego *Google Brain
{a1yousse, ppilligu, fmireshg}@eng.ucsd.edu ayazdan@google.com {sicung, hadi}@eng.ucsd.edu

Abstract

Deep Neural Networks (DNNs) typically require massive amount of computation resource in inference tasks for computer vision applications. Quantization can significantly reduce DNN computation and storage by decreasing the bitwidth of network encodings. Recent research affirms that carefully selecting the quantization levels for each layer can preserve the accuracy while pushing the bitwidth below eight bits. However, without arduous manual effort, this deep quantization can lead to significant accuracy loss, leaving it in a position of questionable utility. As such, deep quantization opens a large hyper-parameter space (bitwidth of the layers), the exploration of which is a major challenge. We propose a systematic approach to tackle this problem, by automating the process of discovering the quantization levels through an end-to-end deep reinforcement learning framework (ReLeQ). We adapt policy optimization methods to the problem of quantization, and focus on finding the best design decisions in choosing the state and action spaces, network architecture and training framework, as well as the tuning of various hyper-paramters. We show how ReLeQ can balance speed and quality, and provide an asymmetric general solution for quantization of a large variety of deep networks (AlexNet, CIFAR-10, LeNet, MobileNet-V1, ResNet-20, SVHN, and VGG-11) that virtually preserves the accuracy ($\leq$ 0.3% loss) while minimizing the computation and storage cost. With these DNNs, ReLeQ enables conventional hardware to achieve 2.2$\times$ speedup over 8-bit execution. Similarly, a custom DNN accelerator achieves 2.0$\times$ speedup and energy reduction compared to 8-bit runs. These encouraging results mark ReLeQ as the initial step towards automating the deep quantization of neural networks.

1. Introduction

Deep Neural Networks (DNNs) have made waves across a variety of domains, from image recognition [16] and synthesis, object detection [25, 28], natural language processing [7], medical imaging, self-driving cars, video surveillance, and personal assistance [9, 17, 11, 18]. DNN compute efficiency has become a major constraint in unlocking further applications and capabilities, as these models require rather massive amounts of computation even for a single inquiry. One approach to reduce the intensity of the DNN computation is to reduce the complexity of each operation. To this end, quantization of neural networks provides a path forward as it reduces the bitwidth of the operations as well as the data footprint [13, 27, 14]. Albeit alluring, quantization can lead to significant accuracy loss if not employed with diligence. Years of research and development has yielded current levels of accuracy, which is the driving force behind the wide applicability of DNNs nowadays. To prudently preserve this valuable feature of DNNs, accuracy, while benefiting from quantization the following two fundamental problems need to be addressed. (1) learning techniques need to be developed that can train or tune quantized neural networks given a level of quantization for each layer. (2) Algorithms need to be designed that can discover the appropriate level of quantization for each layer while considering the accuracy. This paper takes on the second challenge as there are inspiring efforts that have developed techniques for quantized training [32, 33, 22].

This paper builds on the algorithmic insight that the bitwidth of operations in DNNs can be reduced *below eight bits* without compromising their classification accuracy. However, this possibility is manually laborious [20, 21, 30] as to preserve accuracy, the bitwidth varies across individual layers and different DNNs [32, 33, 19, 22]. Each layer has a different role and unique properties in terms of weight distribution. Thus, intuitively, different layers display different sensitivity towards quantization. Over-quantizing a more sensitive layer can result in stringent restrictions on subsequent layers to compensate and maintain accuracy. Nonetheless, considering layer-wise quantization opens a rather exponentially large hyper-parameter space, specially when quantization below eight bits is considered. For example, ResNet-20 exposes a hyper-parameter space of size $8^l = 8^{20} > 10^{18}$, where $l = 20$ is the number of layers and 8 is the possible quantization levels. This exponentially large hyper-parameter space grows with the number of the layers making it impractical to exhaustively assess and determine the quantization level for each layer.

To that end, this paper sets out to automate effectively

navigating this hyper-parameter space using Reinforcement Learning (RL). We develop an end-to-end framework, dubbed ReLeQ, that exploits the sample efficiency of the Proximal Policy Optimization [26] to explore the quantization hyper-parameter space. The RL agent starts from a full-precision previously trained model and learns the sensitivity of final classification accuracy with respect to the quantization level of each layer, determining its bitwidth while keeping classification accuracy virtually intact. Observing that the quantization bitwidth for a given layer affects the accuracy of subsequent layers, our framework implements an LSTM-based RL framework which enables selecting quantization levels with the context of previous layers' bitwidths. Rigorous evaluations with a variety of networks (AlexNet, CIFAR, LeNet, SVHN, VGG-11, ResNet-20, and MobileNet) shows that ReLeQ can effectively find heterogenous deep quantization levels that virtually preserve the accuracy ($\leq$0.3% loss) while minimizing the computation and storage cost. The results (Table 2) show that there is a high variance in quantization levels across the layers of these networks. For instance, ReLeQ finds quantization levels that average to 6.43 bits for MobileNet, and to 2.81 bits for ResNet-20. With the seven benchmark DNNs, ReLeQ enables conventional hardware [5] to achieve 2.2$\times$ speedup over 8-bit execution. Similarly, a custom DNN accelerator [15] achieves 2.0$\times$ speedup and 2.0$\times$ energy reduction compared to 8-bit runs. These results suggest that ReLeQ takes an effective first step towards automating the deep quantization of neural networks.

2. Related Work

ReLeQ is the initial step in utilizing reinforcement learning to automatically find the level of quantization for the layers DNNs such that their classification accuracy is preserved. As such, it relates to the techniques that given a level of quantization, train a neural network or develop binarized DNNs. Furthermore, the line of research that utilizes RL for hyper-parameter discovery and tuning inspires ReLeQ. Nonetheless, ReLeQ, uniquely and exclusively, offers an RL-based approach to determine the levels of quantization.

Training algorithms for quantized neural networks. There have been several techniques [32, 33, 22] that train a neural network in a quantized domain after the bitwidth of the layers is determined manually. DoReFa-Net [32] trains quantized convolutional neural networks with parameter gradients which are stochastically quantized to low bitwidth numbers before they are propagated to the convolution layers. [22] introduces a scheme to train networks from scratch using reduced-precision activations by decreasing the precision of both activations and weights and increasing the number of filter maps in a layer. [33] performs the training phase of the network in full precision, but for inference uses ternary weight assignments. For this assignment, the weights are quantized using two scaling factors which are learned during training phase. PACT [6] introduces a quantization scheme for activations, where the variable α is the clipping level and is determined through a gradient descent based method.

ReLeQ is an orthogonal technique with a different objective: automatically finding the level of quantization that preserves accuracy and can potentially use any of these training algorithms.

Binarized neural networks. Extensive work, [13, 24, 19] focuses on binarized neural networks, which impose accuracy loss but reduce the bitwidth to lowest possible level. In BinaryNet [12], an extreme case, a method is proposed for training binarized neural networks which reduce memory size, accesses and computation intensity at the cost of accuracy. XNOR-Net [24] leverages binary operations (such as XNOR) to approximate convolution in binarized neural networks. Another work [19] introduces ternary-weight networks, in which the weights are quantized to -1, 0, +1 values by minimizing the Euclidian distance between full-precision weights and their ternary assigned values. ReLeQ aims to utilize the levels between binary and 8 bits to avoid loss of accuracy while offering automation.

Reinforcement learning for hyper-parameter tuning. Few works leverage RL in the context of hyper-parameter search. Two of these inspiring efforts [34, 3] use RL to determine the architecture of the neural network and its kernels. Another research [10] employs an RL policy gradient method to automatically find the compression ratio for different layers of a network.

Techniques for selecting quantization levels. Recent work ADMM [31] runs a binary search to minimize the total square quantization error in order to decide the quantization levels for the layers. Then, they use an iterative optimization technique for fine-tuning. NVIDIA also released an automatic mixed precision (AMP) [23] which employs mixed precision during training by automatically selecting between two floating point (FP) representations (FP16 or FP32).

There is a concurrent work HAQ [29] which also uses RL in the context of quantization. The following highlights some of the differences. ReLeQ uses a unique reward formulation and shaping that enables simultaneously optimizing for two objectives (accuracy and reduced computation with lower-bitwidth) within a unified RL process. In contrast, HAQ utilizes accuracy in the reward formulation and then adjusts the RL solution through an approach that sequentially decreases the layer bitwidths to stay within a predefined resource budget. This approach also makes HAQ focused more towards a specific hardware platform whereas we are after a strategy than can generalize. Additionally, we also provide a systemic study of different design decisions, and have significant performance gain across diverse well known benchmarks. The initial version of our work [2], predates HAQ, and it is the first to use RL for quantization[1].

3. RL for Deep Quantization of DNNs

Overview. ReLeQ trains a reinforcement learning agent that determines the level of deep quantization (below 8 bits) for each

[1]We have disclosed our arXiv paper to the program committee chairs while anonymizing it in the references.

Table 1. Layer and network parameters for state embedding.

	Layer Specific	Network Specific
Static	Layer Index	N/A
	Layer Dimensions	
	Weight Statistics (standard deviation)	
Dynamic	Quantization Level (Number of bits)	State of Quantization
		State of Accuracy

layer of the network. ReLeQ agent explores the search space of the quantization levels (bit width), layer by layer. To account for the interplay between the layers with respect to quantization and accuracy, the state space designed for ReLeQ comprises of both static information about the layers and dynamic information regarding the network state during the RL process (Section 3.1). In order to consider the effects of previous layers' quantization levels, the agent steps sequentially through the layers and chooses a bitwidth from a predefined set, e.g., {2,3,4,5,6,7,8}, one layer at a time (Section 3.2). The agent, consequently, receives a reward signal that is proportional to its accuracy after quantization and its benefits in term of computation and memory cost. The underlying optimization problem is multi-objective (higher accuracy, lower compute, and reduced memory); however, preserving the accuracy is the primary concern. To this end, we shape the reward asymmetrically to incentivize accuracy over the benefits (Section 3.3). With this formulation of the RL problem, ReLeQ employs the state-of-the-art Proximal Policy Optimization (PPO) [26] to train its policy and value networks. This section details the components and the research path we have examined to design them.

3.1. State Space Embedding to Consider Interplay between Layers

Interplay between layers. The final accuracy of a DNN is the result of interplay between its layers and reducing the bitwidth of one layer can impact how much another layer can be quantized. Moreover, the sensitivity of accuracy varies across layers. We design the state space and the actions to consider these sensitivities and interplay by including the knowledge about the bitwidth of previous layers, the index of the layer-under-quantization, layer size, and statistics (e.g., standard deviation) about the distribution of the weights. However, this information is incomplete without knowing the accuracy of the network given a set of quantization levels and state of quantization for the entire network. Table 1 shows the parameters used to embed the state space of ReLeQ agent, which are categorized across two different axes. (1) "Layer-Specific" parameters which are unique to the layer vs. "Network-Specific" parameters that characterize the entire network as the agent steps forward during training process. (2) "Static" parameters that do not change during the training process vs. "Dynamic" parameters that change during training depending on the actions taken by the agent while it explores the search space.

State of quantization and relative accuracy. The "Network-Specific" parameters reflect some indication of the state of quantization and relative accuracy. State of Quantization is a metric to evaluate the benefit of quantization for the network and it is calculated using the compute cost and memory cost of each layer. For a neural network with L layers, we define compute cost of layer l as the number of *M*ultiply-*Acc*umulate (*MAcc*) operations (n_l^{MAcc}), where ($l = 0,...,L$). Additionally, since ReLeQ only quantizes weights, we define memory cost of layer l as the number of weights (n_l^w) scaled by the ratio of *Memory Access Energy* ($E_{MemoryAccess}$) to *MAcc Computation Energy* (E_{MAcc}), which is estimated to be around 120× [8]. It is intuitive to consider the that sum of memory and compute costs linearly scales with the number of bits for each layer (n_l^{bits}). The n_{max}^{bits} term is the maximum bitwidth among the predefined set of bitwidths that's available for the RL agent to pick from. Lastly, the State of Quantization ($State_{Quantization}$) is the sum over all layers (L) that accounts for the total memory and compute cost of the entire network.

$$State_{Quantization} = \frac{\sum_{l=0}^{L}[(n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}) \times n_l^{bits}]}{\sum_{l=0}^{L}[n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}] \times n_{max}^{bits}}$$

Besides the potential benefits, captured by $State_{Quantization}$, ReLeQ considers the State of Relative Accuracy to gauge the effects of quantization on the classification performance. To that end, the State of Relative Accuracy ($State_{Accuracy}$) is defined as the ratio of the current accuracy (Acc_{Curr}) with the current bitwidths for all layers during RL training, to accuracy of the network when it runs with full precision (Acc_{FullP}). $State_{Accuracy}$ represents the degradation of accuracy as the result of quantization. The closers this term is to 1.0, the lower the accuracy loss and more desirable the quantization levels.

$$State_{Accuracy} = \frac{Acc_{Curr}}{Acc_{FullP}}$$

Given these embedding of the observations from the environment, the ReLeQ agent can take actions, described next.

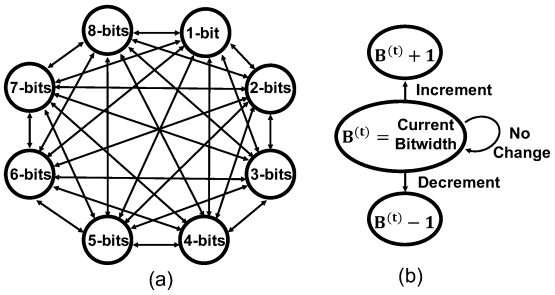

Figure 1. (a) Flexible action space (used in ReLeQ). (b) Alternative action space with restricted movement.

3.2. Flexible Actions Space

Intuitively, as calculations propagate through the layers, the effects of quantization will accumulate. As such, the ReLeQ agents steps through each layer sequentially and chooses from the bitwidth of a layer from a discrete set of quantization levels which are provided as possible choices. Figure 1(a) shows the representation of action space in which the set of bitwidths is {1,2,3,4,5,6,7,8}. As depicted, the agent can flexibly choose to

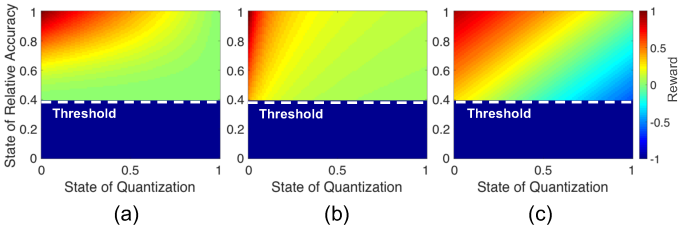

Figure 2. Reward shaping with three different formulations. The color palette shows the intensity of the reward.

change the quantization level of a given layer from any bitwidth to any other bitwidth. The set of possibilities can be changed as desired. Nonetheless, the action space depicted in Figure 1(a) is the possibilities considered for deep quantization in this work. As illustrated in Figure 1(b), an alternative that we experimented with was to only allow the ReLeQ agent to increment/decrement/keep the current bitwidth of the layer ($B^{(t)}$). The experimentation showed that the convergence is much longer than the aforementioned flexible action space, which is used.

3.3. Asymmetric Reward Formulation for Accuracy

While the state space embedding focused on interplay between the layers and the action space provided flexibility, reward formulation for ReLeQ aims to preserve accuracy and minimize bitwidth of the layers simultaneously. This requirement creates an asymmetry between the accuracy and bitwidth reduction, which is a core objective of ReLeQ. The following Reward Shaping formulation provides the asymmetry and puts more emphasis on maintaining the accuracy as illustrated with different color intensities in Figure 2(a).

Reward Shaping:

$reward = 1.0 - (State_{Quantization})^a$

if $(State_{Accuracy} < th)$ **then**

$reward = -1.0$

else

$Acc_{discount} = max(State_{Accuracy}, th)^{(b/max(State_{Accuracy}, th))}$

$reward = reward \times Acc_{discount}$

end if

This reward uses the same terms of State of Quantization ($State_{Quantization}$) and State of Relative Accuracy ($State_{Accuracy}$) from Section 3.1. One of the reasons that we chose this formulation is that it produces a smooth reward gradient as the agent approaches the optimum quantization combination. In addition, the varying 2-dimensional gradient speeds up the agent's convergence time. In the reward formulation, $a = 0.2$ and $b = 0.4$ can also be tuned and $th = 0.4$ is threshold for relative accuracy below which the accuracy loss may not be recoverable and those quantization levels are completely unacceptable. The use of threshold also accelerates learning as it prevents unnecessary or undesirable exploration in the search space by penalizing the agent when it explores undesired low-accuracy states. While Figure 2(a) shows the aforementioned formulation, Figures 2(b) and (c) depict two other alternatives. Figure 2(b) is based on $State_{Accuracy}/State_{Quantization}$ while Figure 2(c) is based on $State_{Accuracy} - State_{Quantization}$. Section 6 provides detailed experimental results with these three reward formulations. In summary, the formulation for Figure 2(a) offers faster convergence.

3.4. Policy and Value Networks

While state, action and reward are three essential components of any RL problem, Policy and Value complete the puzzle and encode learning in terms of a RL context. While there are both policy-based and value-based learning techniques, ReLeQ uses a state-of-the-art policy gradient based approach, Proximal Policy Optimization (PPO) [26]. PPO is a actor critic style algorithm so ReLeQ agent consists of both Policy and Value networks.

Network architecture of Policy and Value networks. Both Policy and Value are functions of state, so the the state space defined in Section 3.1 is encoded as a vector and fed as input to a Long short-term memory(LSTM) layer and this acts as the first hidden layer for both policy and value networks. Apart from the LSTM, policy network has two fully connected hidden layers of 128 neurons each and the number of neurons in the final output layer is equal to the number of available bitwidths the agent can choose from. Whereas the Value network has two fully connected hidden layers of 128 and 64 neurons each. Based on our evaluations, LSTM enables the ReLeQ agent to converge almost $\times 1.33$ faster in comparison to a network with only fully connected layers.

While this section focused on describing the components of ReLeQ in isolation, the next section puts them together and shows how ReLeQ automatically quantizes a pre-trained DNN.

4. Putting it All Together: ReLeQ in Action

As discussed in Section 3, state, action and reward enable the ReLeQ agent to maneuver the search space with an objective of quantizing the neural network with minimal loss in accuracy. ReLeQ starts with a pretrained model of full precision weights and proposes quantization levels of weights for all layers in a DNN. Figure 3 depicts the entire workflow for ReLeQ and this section gives an overview of how everything fits together.

Interacting with the environment. ReLeQ agent steps through all layers one by one, determining the quantization level for the layer at each step. For every step, the state embedding for the current layer comprising of different elements described in Section 3.1 is fed as an input to the Policy and Value Networks of the ReLeQ agent and the output is the probability distribution over the different possible bitwidths and value of the state respectively. ReLeQ agent then takes a stochastic action based on this probablity distribution and chooses a quantization level for the current layer. Weights for this particular layer are quantized to the predicted bitwidth and with accuracy preservation being a primary component of ReLeQ's reward function, retraining of a quantized neural network is required in order to properly evaluate the effectiveness of deep quantization. Such retraining is a time-intensive process and it undermines the search process efficiency.

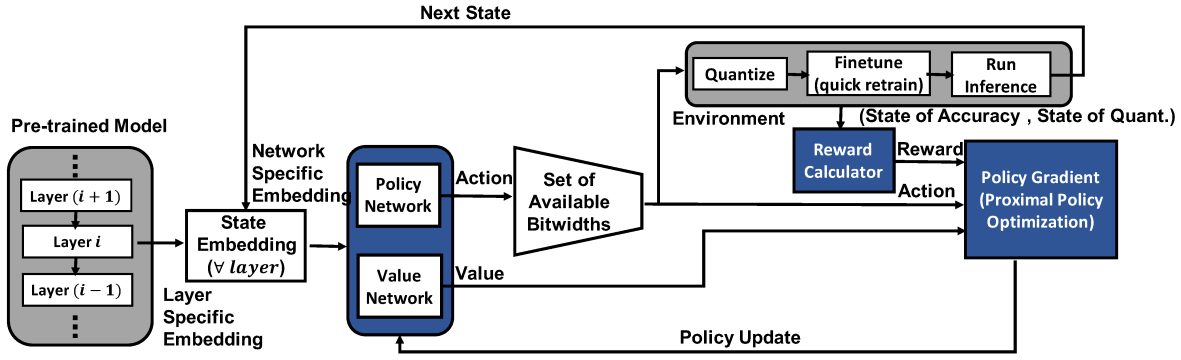

Figure 3. Overview of ReLeQ, which starts from a pre-trained network and delivers its corresponding deeply quantized network.

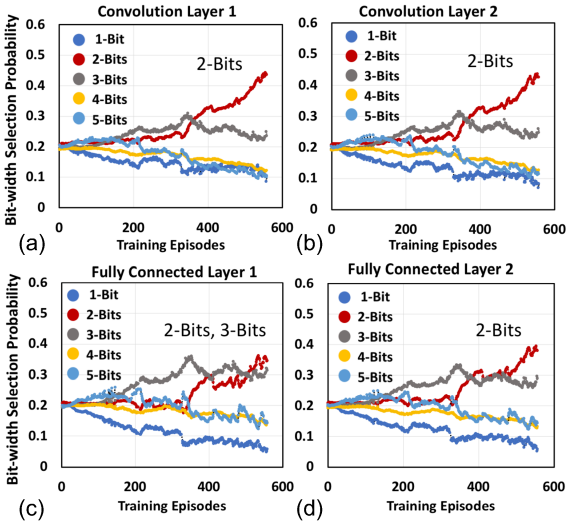

Figure 4. Action (Bitwidths selection) probability evolution over training episodes for LeNet.

To get around this issue, we reward the agent with an estimated validation accuracy after retraining for a shortened amount of epochs. For deeper networks, however, due to the longer retraining time, we perform this phase after all the bitwidths are selected for the layers and then we do a short retraining. Dynamic network specific parameters listed in Table 1 are updated based on the validation accuracy and current quantization levels of the entire network before stepping on to the next layer. In this context, we define an epsiode as a single pass through the entire neural network and the end of every episode, we use Proximal Policy Optimization [26] to update the Policy and Value networks of the ReLeQ agent. After the learning process is complete and the agent has converged to a quantization level for each layer of the network, for example 2 bits for second layer, 3 bits for fourth layer and so on, we perform a long retraining step using the quantized bitwidths predicted by the agent and then obtain the final accuracy for the quantized version of the network.

Learning the Policy. Policy in terms of neural network quantization is to learn to choose the optimal bitwidth for each layer in the network. Since ReLeQ uses a Policy Gradient based approach, the objective is to optimize the policy directly, it's possible to visualize how policy for each layer evolves with respect to time i.e the number of episodes. Figure 4 shows the evolution of ReLeQ agent's bitwidth selection probabilities for all layers over time (number of episodes), which reveals how the agent's policy changes with respect to selecting a bitwidth per layer. As indicated on the graph, the end results suggest the following quantization patterns, 2,2,2,2 or 2,2,3,2 bits. For the first two convolution layers (Convolution Layer 1, Convolution Layer 2), the agent ends up assigning the highest probability for two bits and its confidence increases with increasing number of training episodes. For the third layer (Fully Connected Layer 1), the probabilities of two bits and three bits are very close. Lastly, for the fourth layer (Fully Connected Layer 2), the agent again tends to select two bits, however, with relatively smaller confidence compared to layers one and two. With these observations, we can infer that bitwidth probability profiles are not uniform across all layers and that the agent distinguishes between the layers, understands the sensitivity of the objective function to the different layers and accordingly chooses the bitwidths. Looking at the agent's selection for the third layer (Fully Connected Layer 1) and recalling the initial problem formulation of quantizing all layers while preserving the initial full precision accuracy, it is logical that the probabilities for two and three bits are very close. Going further down to two bits was beneficial in terms of quantization while staying at three bits was better for maintaining good accuracy which implies that third layer precision affects accuracy the most for this specific network architecture. *This points out the importance of tailoring the reward function and the role it plays in controlling optimization tradeoffs.*

5. Experimental Setup

Table 2. Benchmark DNNs and their deep quantization with ReLeQ.

Network	Dataset	Bitwidths	Accuracy Loss
AlexNet	ImageNet	{8,4,4,4,4,4,4,8}	0.08%
CIFAR-10	CIFAR-10	{5,5,5,5,5}	0.30%
LeNet	MNIST	{2,2,3,2}	0.00%
MobileNet	ImageNet	{8, 5, 6, 6,4, 4, 7, 8, 4, 6, 8, 5, 5, 8, 6, 7, 7, 7, 6, 8, 6, 8, 8, 6, 7, 5, 5, 7, 8, 8}	0.26%
ResNet-20	CIFAR-10	{8,2,2,3,2,2,2,3,2,3,3,3,2,2,2,3,2,2,2,2,2,8}	0.12%
SVHN	SVHN	{8,4,4,4,4,4,4,4,8}	0.00%
VGG-11	CIFAR-10	{8,5,8,5,6,6,6,6,8}	0.17%

Table 3. Hyper-Parameters of PPO used in ReLeQ.

Hyperparameter	Value
Adam Step Size	1 x 10^{-4}
Generalized Advantage Estimation Parameter	0.99
Number of Epochs	3
Clipping Parameter	0.1

Benchmarks. To assess the effectiveness of ReLeQ across a variety of DNNs, we use the following seven diverse networks that have been used in different real-world vision tasks: AlexNet, CIFAR-10 (simplenet), LeNet, MobileNet (Version 1), ResNet-20, SVHN and VGG-11. Of these seven networks, AlexNet and MobileNet were evaluated on the ImageNet (ILSVRC'12) dataset, ResNet-20, VGG-11 and CIFAR-10 on CIFAR-10, SVHN on SVHN and LeNet on the MNIST dataset.

Quantization technique. We use the technique proposed in WRPN [22] where weights are first scaled and clipped to the $(-1.0,1.0)$ range and quantized as per the following equation. The term k is the bitwidth used for quantization out of which $k-1$ bits are used for quantization and one bit is used for sign.

$$x_q = \frac{round((2^{k-1}-1)x_f)}{2^{k-1}-1}$$

Deep quantization with conventional hardware. ReLeQ's solution can be deployed on conventional hardware, such as general purpose CPUs to provide benefits and improvements. To manifest this, we have evaluated ReLeQ using TVM [5] on an Intel Core i7-4790 CPU. We use TVM since its compiler supports deeply quantized operations with bit-serial vector operations on conventional hardware. We compare our solution in terms of the inference execution time (since the TVM framework does not offer energy measurements) against 8-bit quantized network. The results can be seen in Figure 7 and will be further elaborated in the next section.

Deep quantization with custom hardware accelerators. To further demonstrate the energy and performance benefits of the solution found by ReLeQ, we evaluate it on Stripes [15], a custom accelerator designed for DNNs, which exploits bit-serial computation to support flexible bitwidths for DNN operations Stripes does not support or benefit from deep quantization of activations and it only leverages the quantization of weights. We compare our solution in terms of energy consumed and inference execution time against the 8-bit quantized network.

Comparison with prior work. We also compare against prior work [31], which proposes an iterative optimization procedure (dubbed ADMM) through which they find quantization bitwidths only for AlexNet and LeNet. Using Stripes [15] and TVM [5], we show that ReLeQ's solution provides higher performance and energy benefits compared to ADMM [31].

Implementation and hyper-parameters of the Proximal Policy Optimization (PPO). As discussed, ReLeQ use PPO [26] as its RL engine, which we implemented in python where its policy and value networks use TensorFlow's Adam Optimizer with an initial learning rate of 10^{-4}. The setting of the other hyper-parameters of PPO is listed in Table 3.

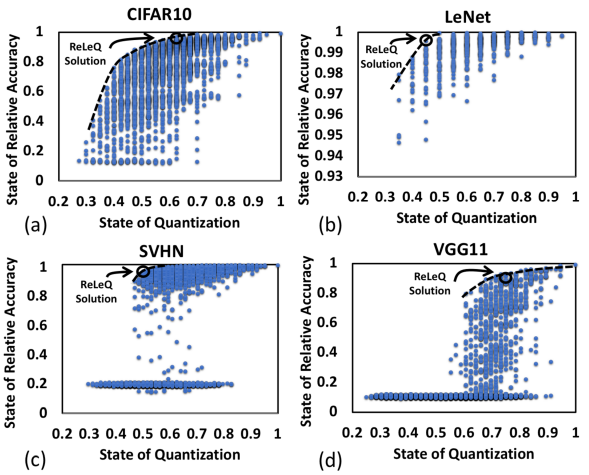

Figure 5. Quantization space and its Pareto frontier for (a) CIFAR-10, (b) LeNet, (c) SVHN, and (d) VGG-11.

6. Experimental Results

Quantization levels with ReLeQ. Table 2 provides a summary of the evaluated networks, datasets and shows the results with respect to layer-wise quantization levels (bitwidths) achieved by ReLeQ. Regarding the layer-wise quantization bitwidths, at the onset of the agent's exploration, all layers are initialized to 8-bits. As the agent learns the optimal policy, each layer converges with a high probability to a particular quantization bitwidth. As shown in the "Bitwidths" column of Table 2, ReLeQ quantization policies show a spectrum of varying bitwidth assignments to the layers. The bitwidth for MobileNet varies from 4 bits to 8 bits with an irregular pattern, which averages to 6.43. ResNet-20 achieves mostly 2 and 3 bits, again with an irregular interleaving that averages to 2.81. In many cases, there is significant heterogeneity and irregularity in the bitwidths and a uniform assignment of the bits is not always the desired choice to preserve accuracy. These results demonstrate that ReLeQ automatically distinguishes different layers and their varying importance with respect to accuracy while choosing their respective bitwidths. As shown in the "Accuracy Loss" column of Table 2, the deeply quantized networks with ReLeQ have less than 0.30% loss in classification accuracy. To assess the quality of these bitwidth assignments, we conduct a Pareto analysis on the DNNs for which we could populate the search space.

Validation: Pareto analysis. Figure 5 depicts the solutions space for four benchmarks (CIFAR10, LeNet, SVHN, and VGG11). Each point on these charts is a unique combination of bitwidths that are assigned to the layers of the network. The boundary of the solutions denotes the Pareto frontier and is highlighted by a dashed line. The solution found by ReLeQ is marked out using an arrow and lays on the desired section of the Pareto frontier where the accuracy loss can be recovered through

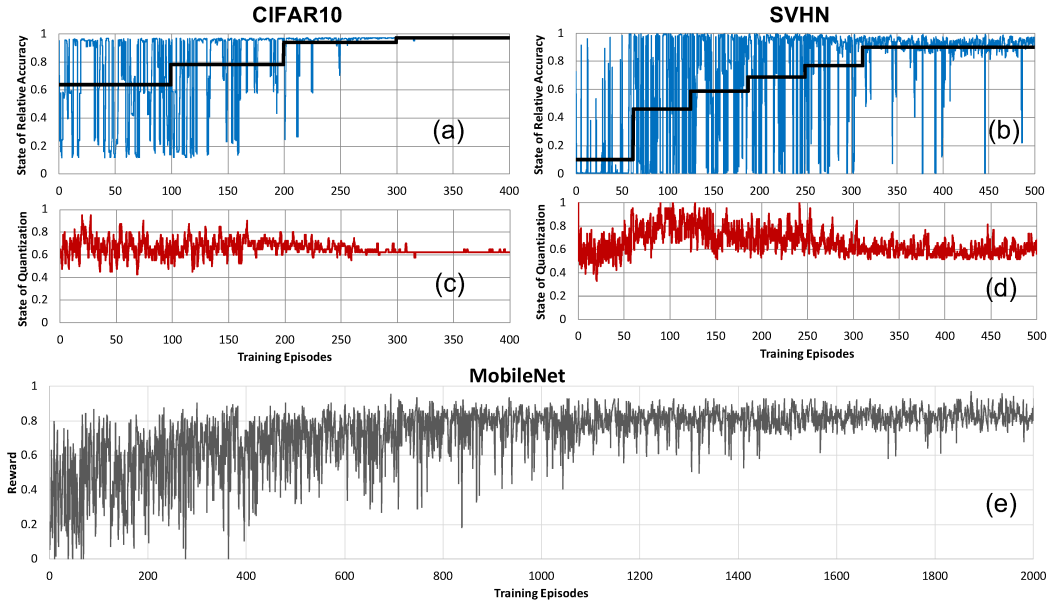

Figure 6. The evolution of reward and its basic elements, (a), (b) State of Relative Accuracy and (c), (d) State of Quantization) as the agent learns through the episodes. The last plot (e) shows an alternative view by depicting the evolution of reward for MobileNet. The trends are similar for the other networks.

fine-tuning. It is worth noting that as a result of the moderate size of the four networks presented in this subsection, it was possible to enumerate the design space, obtain Pareto frontier and assess ReLeQ quantization policy for each of the four networks. However, it is infeasible to do so for state-of-the-art deep networks (e.g., MobileNet and AlexNet) which further stresses the importance of automation and efficacy of ReLeQ.

Learning and convergence analysis. We further study the desired behavior of ReLeQ in the context of convergence. An appropriate evidence for the correctness of a formulated reinforcement learning problem is the ability of the agent to consistently yield improved solutions. The expectation is that the agent learns the correct underlying policy over the episodes and gradually transitions from the exploration to the exploitation phase. Figures 6(a) and (b) first show the State of Relative Accuracy for CIFAR10 and SVHN, respectively. We overlay the moving average of State of Relative Accuracy as episodes evolve, which is denoted by a black line in Figures 6(a) and (b). Similarly, Figures 6(c) and (d) depict the evolution of State of Quantization. As another indicative parameter of learning, Figure 6 plots the evolution of the reward, which combines the two States of Accuracy and Quantization (Section 3.3). As all the graphs show, the agent consistently yields solutions that increasingly preserve the accuracy (maximize rewards), while seeking to minimize the number of bits assigned to each layer (minimizing the state of quantization) and eventually converges to a rather stable solution. The trends are similar for the other networks.

Execution time and energy benefits with ReLeQ. Figure 7 shows the speedup for each benchmark network conventional hardware using TVM compiler. The baseline is the 8-bit run-

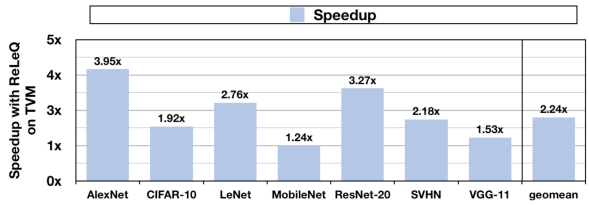

Figure 7. Speedup with ReLeQ for conventional hardware using TVM over the baseline run using 8 bits.

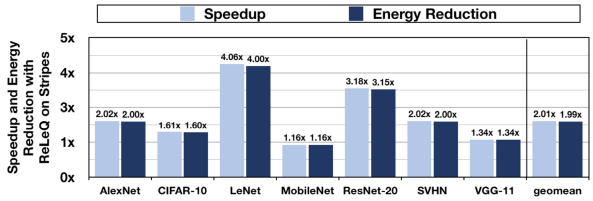

Figure 8. Energy reduction and speedup with ReLeQ for Stripes over the baseline execution when the accelerator is running 8-bit DNNs.

time for inference. ReLeQ's solution offers, on average, 2.2× speedup over the baseline as the result of merely quantizing the weights that reduces the amount of computation and data transfer during inference. Figure 8 shows the speedup and energy reduction benefits of ReLeQ's solution on Stripes custom accelerator. The baseline here is the time and energy consumption of 8-bit inference execution on the same accelerator.

ReLeQ's solutions yield, on average, 2.0× speedup and an additional 2.0× energy reduction. MobileNet achieves 1.2× speedup which is coupled with a similar degree of energy reduction. On the other end of the spectrum ResNet-20 and LeNet achieve 3.0× and 4.0× the benefits, respectively. As

Table 4. Speedup and energy reduction with ReLeQ over the benefits with ADMM [31] over the two networks that ADMM reports. The improvements with ReLeQ is reported with both TVM and Stripes.

Network	Dataset	Technique	Bitwidths	ReLeQ speedup on TVM	ReLeQ speedup on Stripes	Energy improvement of ReLeQ on Stripes
AlexNet	ImageNet	ReLeQ	{8,4,4,4,4,4,4,8}	1.20X	1.25X	1.25X
		ADMM	{8,5,5,5,5,3,3,8}			
LeNet	MNIST	ReLeQ	{2,2,3,2}	1.42X	1.86X	1.87X
		ADMM	{5,3,2,3}			

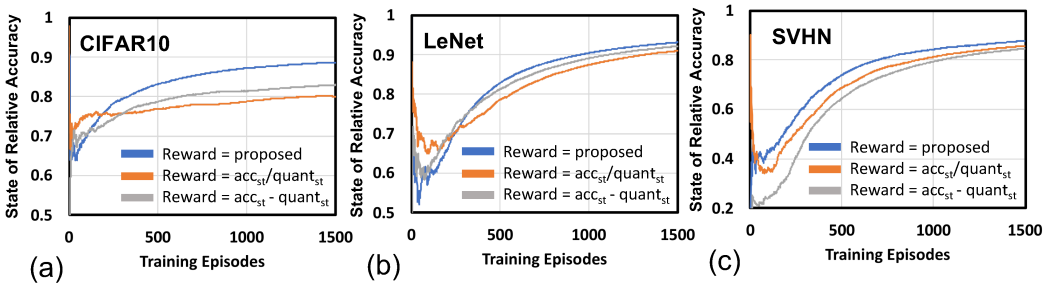

Figure 9. Three different reward functions and their impact on the state of relative accuracy over the training episodes for three different networks. (a) CIFAR10, (b) LeNet, and (c) SVHN.

shown in Table 2, MobileNet needs to be quantized to higher bitwidths to maintain accuracy, compared with other networks and that is why the benefits are smaller.

Speedup and energy reduction over ADMM. As mentioned in Section 5, we compare ReLeQ's solution in terms of speedup and energy reduction against ADMM [31], another procedure for finding quantization bitwidths. As shown in Table 4, ReLeQ's solution provides 1.25× energy reduction and 1.22× average speedup over ADMM with Stripes for AlexNet and the benefits are higher for LeNet. The benefits are similar for the conventional hardware using TVM as shown in Table 4. ADMM does not report other networks.

Sensitivity analysis: influence of reward function. The design of reward function is a crucial component of reinforcement learning as indicated in Section 3.3. There are many possible reward functions one could define for a particular application setting. However, different designs could lead to either different policies or different convergence behavior. In this paper, we incorporate reward engineering by proposing a special parametric reward formulation. To evaluate the effectiveness of the proposed reward formulation, we have compared three different reward formulations in Figure 9: (a) proposed in Section 3.3, (b) $R = State_{Accuracy}/State_{Quantization}$, (c) $R = State_{Accuracy} - State_{Quantization}$. As the blue line in all the charts shows, the proposed reward formulation consistently achieves higher State of Relative Accuracy during the training episodes. That is, our proposed reward formulation enables ReLeQ finds better solutions in shorter time.

Tuning: PPO objective clipping parameter. One of the unique features about PPO algorithm is its novel objective function with clipped probability ratios, which forms a lower-bound estimate of the change in policy. Such modification controls the variance of the new policy from the old one, hence improves the stability of the learning process. PPO uses a Clipped Surrogate Objective function, which uses the minimum of two probability ratios, one non-clipped and one clipped in a range between $[1-\varepsilon, 1+\varepsilon]$, where ε is a hyper-parameter that helps to define this clipping range. Table 5 provides a summary of tuning epsilon (commonly in the range of 0.1, 0.2, 0.3). Based on our performed experiments, $\varepsilon = 0.1$ often reports the highest average reward across different benchmarks.

Table 5. Sensitivity of reward to different clipping parameters.

PPO Clipping Parameter	Avg. Normalized Reward (Performance)		
	LeNet	CIFAR10-Net	SVHN-Net
$\epsilon = 0.1$	0.209	0.407	0.499
$\epsilon = 0.2$	0.165	0.411	0.477
$\epsilon = 0.3$	0.160	0.399	0.455

7. Conclusion

Quantization of neural networks offers significant promise in reducing their compute and storage cost. However, the utility of quantization hinges upon automating its process while preserving accuracy. This paper set out to define the automated discovery of quantization levels for the layers while complying to the constraint of maintaining the accuracy. As such, this work offered the RL framework that was able to effectively navigate the huge search space of quantization and automatically quantize a variety of networks leading to significant performance and energy benefits. The results suggests that a diligent design of our RL framework, which considers multiple concurrent objectives can automatically yield high-accuracy, yet deeply quantized, networks.

References

[1] Imagenet classification with deep convolutional neural. 2013.

[2] Anonymous. ———. *arXiv preprint arXiv:*—.—, November 5, 2018. 2

[3] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing Neural Network Architectures using Reinforcement Learning. 2017. 2

[4] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ASPLOS*, 2014.

[5] T. Chen, T. Moreau, Z. Jiang, H. Shen, E. Q. Yan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy. Tvm: End-to-end optimization stack for deep learning. *CoRR*, abs/1802.04799, 2017. 2, 6

[6] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018. 2

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011. 1

[8] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. E. Kozyrakis. TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory. In *ASPLOS*, 2017. 3

[9] J. Hauswald, M. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. N. Mudge, V. Petrucci, L. Tang, and J. Mars. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ASPLOS*, 2015. 1

[10] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *ECCV*, 2018. 2

[11] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 1

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized Neural Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*. 2016. 2

[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. *J. Mach. Learn. Res.*, 2017. 1, 2

[14] P. Judd, J. Albericio, T. H. Hetherington, T. M. Aamodt, and A. Moshovos. Stripes: Bit-serial deep neural network computing. *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016. 1

[15] P. Judd, J. Albericio, T. H. Hetherington, T. M. Aamodt, and A. Moshovos. Stripes: Bit-serial deep neural network computing. *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016. 2, 6

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60:84–90, 2012. 1

[17] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989. 1

[18] H. Lee, R. B. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 1

[19] F. Li and B. Liu. Ternary Weight Networks. *CoRR*, abs/1605.04711, 2016. 1, 2

[20] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017. 1

[21] A. K. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *CoRR*, abs/1711.05852, 2017. 1

[22] A. K. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr. WRPN: Wide Reduced-Precision Networks. In *ICLR*, 2018. 1, 2, 6

[23] Nvidia. Automatic mixed precision for nvidia tensor core architecture in tensorflow. https://devblogs.nvidia.com/nvidia-automatic-mixed-precision-tensorflow/. 2

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 2016. 2

[25] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015. 1

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 3, 4, 5, 6

[27] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 764–775, 2018. 1

[28] S. E. I. N. W. O. O. S. Tatistical, S. A. V. Ancouver, and P. M. Jones. Robust real-time object detection. 2001. 1

[29] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. HAQ: Hardware-Aware Automated Quantization. *arXiv preprint arXiv:1811.08886*, November 21, 2018. 2

[30] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *CoRR*, abs/1812.00090, 2018. 1

[31] S. Ye, T. Zhang, K. Zhang, J. Li, J. Xie, Y. Liang, S. Liu, X. Lin, and Y. Wang. A unified framework of dnn weight pruning and weight clustering/quantization using admm. *CoRR*, abs/1811.01907, 2018. 2, 6, 8

[32] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *CoRR*, 2016. 1, 2

[33] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained Ternary Quantization. In *ICLR*, 2017. 1, 2

[34] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. In *ICLR*, 2017. 2