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

Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh

Abstract—*Deep Quantization (below eight bits)* can significantly reduce DNN computation and storage by decreasing the bitwidth of network encodings. However, without arduous manual effort, this deep quantization can lead to significant accuracy loss, leaving it in a position of questionable utility. We propose a systematic approach to tackle this problem, by automating the process of discovering the bitwidths through an end-to-end deep reinforcement learning framework (**ReLeQ**). This framework utilizes the sample efficiency of Proximal Policy Optimization (PPO) to explore the exponentially large space of possible assignment of the bitwidths to the layers. We show how **ReLeQ** can balance speed and quality, and provide a heterogeneous bitwidth assignment for quantization of a large variety of deep networks with minimal accuracy loss ($\leq$ 0.3% loss) while minimizing the computation and storage costs. With these DNNs, **ReLeQ** enables conventional hardware and custom DNN accelerator to achieve $2.2\times$ speedup over 8-bit execution.

Index Terms—Neural networks, quantization, autoML.

1 Introduction

Deep Neural Networks (DNNs) have made waves across a variety of domains [1], however their compute efficiency has become a major constraint in unlocking further applications and capabilities. To this end, quantization of neural networks provides a path forward as it reduces the bitwidth of operations and memory footprint. For instant, in many scenarios, the bottleneck of running DNNs is in transferring the weights and data between main memory and compute cores. Using 8-bit integer rather than 32-bit, we instantly speed up the memory transfer by $4\times$.

Albeit alluring, quantization can lead to significant accuracy loss if not employed with diligence. To that end, two fundamental problems need to be addressed. (1) Developing learning techniques that can perform quantized training of DNNs. (2) Designing algorithms that identify appropriate bitwidth per-layer while preserving accuracy. This paper takes on the second challenge as there are inspiring efforts that have developed techniques for quantized training [2], [3].

However, this possibility (discovering bitwidths) is manually laborious as to preserve accuracy, the bitwidth varies across individual layers and different DNNs [2], [3]. Each layer has a different role and unique properties in terms of weight distribution; hence, displays different sensitivity towards quantization. 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 bitwidths. This exponentially large hyper-parameter space grows with the number of layers making it impractical to exhaustively assess and determine the bitwidth for each layer.

We develop an end-to-end framework, dubbed **ReLeQ**, that exploits the sample efficiency of the Proximal Policy Optimization [4] 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 bitwidth of each layer, determining its bitwidth while keeping classification accuracy almost intact. Observing that the quantization bitwidth for a given layer affects the accuracy of subsequent layers, our framework implements a Long short-term memory (LSTM)-based RL framework which enables selecting bitwidths with the context of previous layers' bitwidths. Rigorous evaluations with a variety of networks (AlexNet, CIFAR, LeNet, SVHN, VGG-11, ResNet-20, and MobileNet) show that **ReLeQ** can effectively find heterogenous bitwidths with minimal accuracy loss ($\leq$0.3% loss) while minimizing the computation and storage cost. The results (Table 1) show that there is a high variance in bitwidths across the layers of these networks. With the seven benchmark DNNs, **ReLeQ** enables conventional hardware [5] as well as a custom DNN accelerator [6] to achieve $2.2{-}2.7\times$ speedup over 8-bit execution. These results suggest that **ReLeQ** takes an effective first step towards automating the deep quantization of neural networks.

- *Ahmed T. Elthakeb is with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA, 92093. E-mail: a1yousse@eng.ucsd.edu*
- *Prannoy Pilligundla, Fatemehsadat Mireshghallah, and Hadi Esmaeilzadeh are with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA, 92093. E-mail: {ppilligu, fmireshg, hadi}@eng.ucsd.edu*
- *Amir Yazdanbakhsh is with Google Brain, Mountain View, CA. E-mail: ayazdan@google.com*

Manuscript received February 24, 2020; revised Month XX, 2020.

2 RL FOR DEEP QUANTIZATION OF DNNs

2.1 Method Overview

ReLeQ trains an RL agent that determines the bitwidth for each layer of the network. **ReLeQ** explores the search space of the bitwidths, layer by layer. The underlying optimization problem is multi-objective (higher accuracy, lower compute, and reduced memory); however, preserving the accuracy is the primary objective. With this formulation of the RL problem, **ReLeQ** employs the state-of-the-art Proximal Policy Optimization (PPO) [4] to train its policy and value networks. This section details the components and the research path we have examined to design them.

2.2 State Space Embedding

Interplay between layers. We design the state space to consider sensitivities and interplay between layers by including the knowledge about the bitwidth of previous layers, the index of the layer-under-quantization, layer size, and weights statistics (e.g. standard deviation).

However, this information is incomplete without knowing the accuracy of the network given a set of bitwidths and state of quantization for the entire network. As such, the parameters used to embed the state space of **ReLeQ** agent are categorized across two different axes. (1) “Layer-Specific” parameters which are unique to the layer (Layer index, Layer Dimensions, Weight Statistics) vs. “Network-Specific” parameters that characterize the entire network as the agent steps forward during training process (state of quantization and relative accuracy). (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 such as state of quantization and relative accuracy.

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 and memory costs 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 $120\times$.

It is intuitive to consider that the sum of memory and compute costs linearly scale 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 normalized sum over all layers (L) that accounts for the total memory and compute costs of the entire network.

$$State_{Quantization} = \frac{\sum_{l=0}^{L}[(n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}) \times n_l^{bits}]}{\sum_{l=0}^{L}[n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}] \times n_{max}^{bits}} \quad (1)$$

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}).

$$State_{Accuracy} = \frac{Acc_{Curr}}{Acc_{FullP}} \quad (2)$$

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

2.3 Flexible Actions Space

The **ReLeQ** agent steps through each layer sequentially and chooses the bitwidth of a layer from a discrete set of bitwidths which are provided as possible choices.

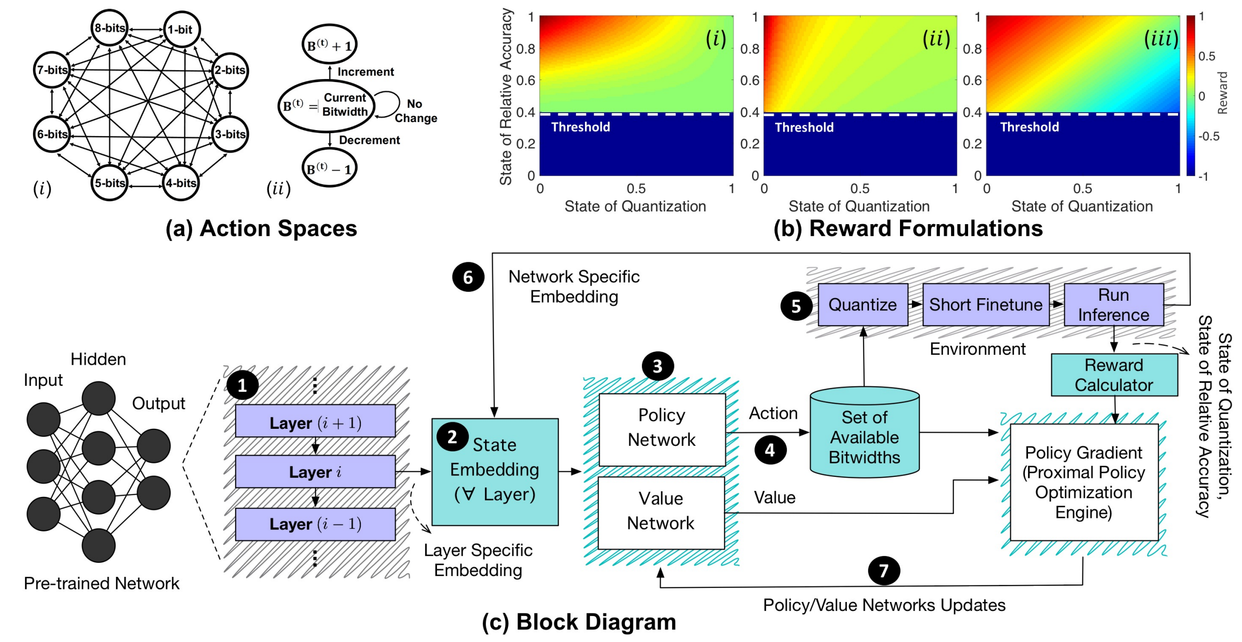
Figure 1(a)(i) shows the representation of action space in which the set of bitwidths is $\{1, 2, 3, 4, 5, 6, 7, 8\}$. As depicted, the agent can flexibly choose to change the bitwidth of a given layer from any bitwidth to any other bitwidth. An alternative (Figure 1(a)(ii)) that we experimented with was to only allow **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, as it encourages more exploration.

2.4 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 1(b)(i). This reward uses the same terms of $State_{Quantization}$ and $State_{Acc}$ from Section 2.2.

```
Reward Shaping:
$reward = 1.0 - (State_{Quantization})^a$
if ($State_{Acc} < th$) then
    $reward = -1.0$
else
    $Acc_{discount} = State_{Acc}^{(b/State_{Acc})}$
    $reward = reward \times Acc_{discount}$
end if
```

This used formulation (1) produces a smooth reward gradient as the agent approaches the optimum quantization combination. (2) The varying 2-dimensional gradient speeds up the agent's convergence time. In the reward formulation, th is threshold for relative accuracy below which the accuracy loss may not be recoverable and those bitwidths are completely unacceptable. After some trials, we observe that $a = 0.2, b = 0.4, th = 0.4$ provide reasonable convergence times and accuracy-quantization trade-off; thus, we fixed



1	**ReLeQ** agent steps through all layers one by one, determining the quantization level for the layer at each step
2	For every step, the state embedding for the current layer is fed as an input to the Policy and Value networks
3	Policy network outputs a probability distribution over the different possible bitwidths, whereas the Value network outputs a value of being in a certain state
4	**ReLeQ** agent then takes a stochastic action based on that probability distribution and chooses a quantization level for the current layer
5	Weights for this particular layer are quantized to the predicted bitwidth, and reward is calculated based on estimated validation accuracy after retraining for a shortened amount of epochs
6	Dynamic network-specific parameters are updated based on the validation accuracy and current quantization levels of the entire network before stepping on to the next layer
7	In this context, episode is defined as a single pass through the entire neural network (all layers) and at the end of every episode, we use Proximal Policy Optimization (PPO) to update the Policy and Value networks of the **ReLeQ** agent

Fig. 1: (a) Action spaces: (i) Flexible action space (used in ReLeQ), (ii) Alternative action space with restricted movement. (b) Reward shaping with three different formulations as functions of the optimization objectives: state of relative accuracy and state of quantization: (i) Proposed formulation, (ii) direct division, and (iii) direct subtraction. The color palette shows the intensity of the reward. (c) Overview of ReLeQ, which starts from a pre-trained network and delivers its corresponding deeply quantized network.

them throughout the experiments. While Figure 1 (b) (i) shows the aforementioned formulation, Figures 1 (b) (ii) and (iii) depict two other alternatives. Figure 1 (b) (ii) is based on $State_{Acc}/State_{Quantization}$ while Figure 1 (b) (iii) is based on $State_{Acc} - State_{Quantization}$. In summary, based on our experiments, the formulation for Figure 1(b) (i) offers faster convergence.

2.5 Network Architecture of Policy and Value Networks

Both Policy and Value are functions of state, so the state space, defined in Section 2.2, is encoded as a vector and fed as input to LSTM layer which 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.

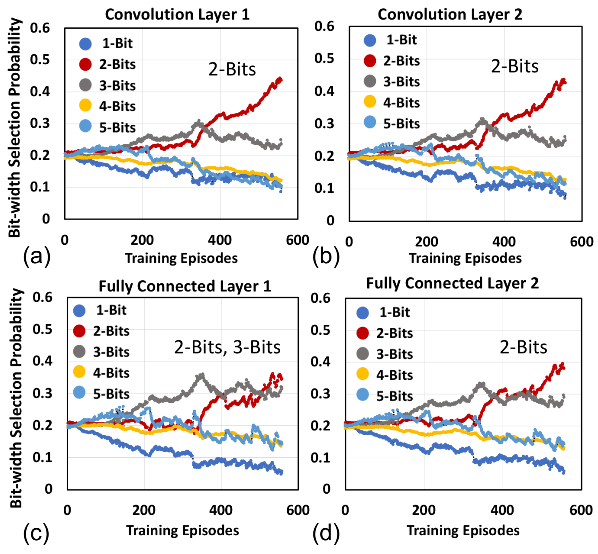


Fig. 2: Action (Bitwidths selection) probability evolution over training episodes for LeNet.

3 Putting it All Together: ReLeQ in Action

As discussed in Section 2, 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. We use linear quantization as proposed in [7]. Figure 1(c) depicts the entire workflow for **ReLeQ** and this section gives an overview of how everything fits together in practice.

Learning the policy. Policy in terms of neural network quantization is to learn to choose the optimal bitwidth for each layer in the network. Figure 2 shows the evolution of **ReLeQ** agent's bitwidth selection probabilities for all layers of LeNet over training 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, the agent ends up assigning the highest probability for two bits. For the third layer (FC1), the probabilities of two bits and three bits are very close. Lastly, for the fourth layer (FC2), 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. As such, the agent distinguishes between the layers, understands the sensitivity of the objective function to different layers and accordingly chooses the bitwidths.

4 Experimental Results

4.1 Quantization Levels with ReLeQ

Table 1 provides a summary of results with respect to layer-wise quantization bitwidths achieved by **ReLeQ**. 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 bitwidth. As shown in the "Quantization Bitwidths" column of Table 1, **ReLeQ** quantization policies show a spectrum of varying bitwidth assignments to the layers. The bitwidth for MobileNet varies with an irregular pattern, which averages to 6.43. ResNet-20 achieves mostly 2 and 3-bit, again with an irregular interleaving that averages to 2.81. In many cases, there is significant heterogeneity 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 1, the deeply quantized networks with **ReLeQ** have less than 0.30% loss in accuracy. To assess the quality of these bitwidths assignments, we conduct a Pareto analysis on the DNNs for which we could populate the search space.

4.2 Validation: Pareto Analysis

Figure 3 (a) 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 fine-tuning, which demonstrates the quality of the obtained solutions. It is worth noting that as a result of the moderate size of these four networks, it was possible to enumerate the design space, obtain Pareto frontier and assess ReLeQ quantization policy for each network. However, such enumeration is infeasible for state-of-the-art deep networks (e.g., MobileNet, AlexNet) which further stresses the importance of automation and efficacy of **ReLeQ**.

4.3 Learning and Convergence Analysis

An appropriate evidence for the correctness of a formulated RL problem is the ability of the agent to consistently yield improved solutions. Figures 3 (b) shows (through different quantities (i-v)) that **ReLeQ** 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 other networks.

4.4 Execution Time and Energy Benefits with ReLeQ

Deep quantization with conventional hardware. **ReLeQ**'s solution can be deployed on conventional hardware, such as general purpose CPUs to provide improvements. To manifest this, we evaluated **ReLeQ** using TVM [5] on an Intel Core i7-4790 CPU. Figure 4 (a) shows the speedup for each benchmark using TVM compiler. The baseline is the 8-bit runtime for inference. **ReLeQ**'s solution offers, on average, $2.2\times$ speedup over the baseline as the result of merely quantizing the weights that reduces the amount of computation and data transfer during inference.

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 [6], a custom accelerator designed for DNNs, which exploits bit-serial computation to support flexible bitwidths for DNN operations. Figure 4 (b) shows the speedup and energy reduction benefits of **ReLeQ**'s solution on Stripes. Baseline is the 8-bit inference execution. **ReLeQ**'s solutions yield, on average, $2.0\times$ speedup and an additional $2.7\times$ energy reduction. MobileNet achieves $1.2\times$ speedup which is coupled with a similar degree of energy reduction. On the other end of the spectrum, ResNet-20 and LeNet achieve $3.0\times$ and $4.0\times$ benefits, respectively.

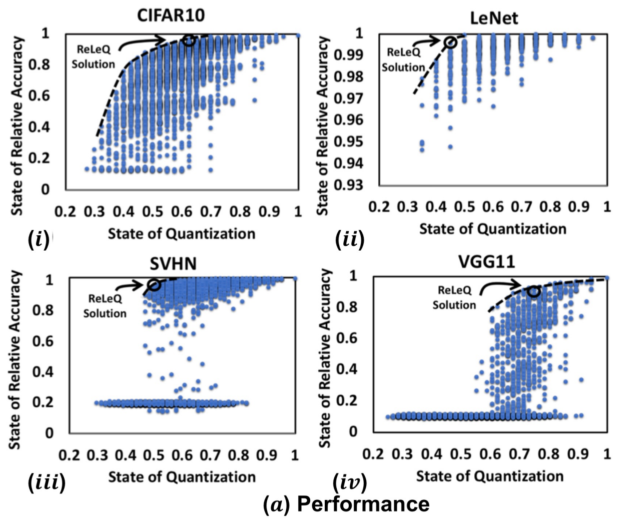
4.5 Speedup and Energy Reduction over ADMM

We compare **ReLeQ**'s solution in terms of speedup and energy reduction against ADMM [8], another procedure for finding quantization bitwidths. As shown in Table 2, **ReLeQ**'s solution provides $1.25\times$ energy reduction and $1.22\times$ average speedup over ADMM with Stripes for AlexNet and the benefits are higher for LeNet.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MM.2020.3009475, IEEE Micro

TABLE 1: Benchmark DNNs and their deep quantization with ReLeQ.

Network	Dataset	Quantization Bitwidths	Average Bitwidth	Accuracy Loss (%)
AlexNet	ImageNet	{8, 4, 4, 4, 4, 4, 4, 8}	5	**0.08**
SimpleNet	CIFAR10	{5, 5, 5, 5, 5}	5	**0.30**
LeNet	MNIST	{2, 2, 3, 2}	2.25	**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}	6.43	**0.26**
ResNet-20	CIFAR10	{8, 2, 2, 3, 2, 2, 2, 3, 2, 3, 3, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 8}	2.81	**0.12**
10-Layers	SVHN	{8, 4, 4, 4, 4, 4, 4, 4, 4, 8}	4.80	**0.00**
VGG-11	CIFAR10	{8, 5, 8, 5, 6, 6, 6, 6, 8}	6.44	**0.17**
VGG-16	CIFAR10	{8, 8, 8, 6, 8, 6, 8, 6, 8, 6, 8, 6, 8, 6, 8, 8}	7.25	**0.10**



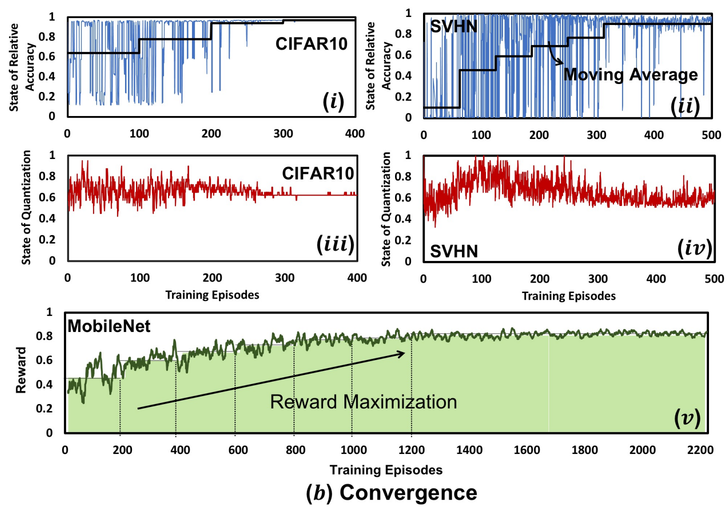


Fig. 3: (a) Performance: quantization space and its Pareto frontier for (i) CIFAR-10, (ii) LeNet, (iii) SVHN, and (iv) VGG-11. (b) Convergence: the evolution of reward and its basic elements: State of Relative Accuracy for (i) CIFAR-10, (ii) SVHN. State of Quantization for (iii) CIFAR-10, (iv) SVHN, as the agent learns through the episodes. The last plot (v) shows an alternative view by depicting the evolution of reward for MobileNet. The trends are similar for the other networks.

TABLE 2: Speedup and energy reduction with ReLeQ over ADMM [8].

Network	Dataset	Technique	Bitwidth	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	1.22X	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}			

5 Related Work

ReLeQ is the initial step in utilizing reinforcement learning to automatically find the bitwidth for the layers of DNNs such that their accuracy is preserved.

Reinforcement learning for automatic tuning. RL based methods have attracted much attention within neural architecture search (NAS) after obtaining the competitive performance on the CIFAR-10 dataset employing RL as the search strategy [9]. Different RL approaches differ in how they represent the agent's policy. [9] uses an RNN trained by policy gradient to sequentially sample a string that in turn encodes a neural architecture.

Aside from NAS, [10] employs RL to prune existing architectures where a policy gradient method is used to automatically find the compression ratio for different layers of a network.

Techniques for selecting bitwidths. Recent work ADMM [8] runs a binary search to minimize the total square quantization error in order to decide the bitwidths for the layers. Then, they use an iterative optimization technique for fine-tuning. Other work [11] focused on binarized neural networks. There is a concurrent work HAQ [12] 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

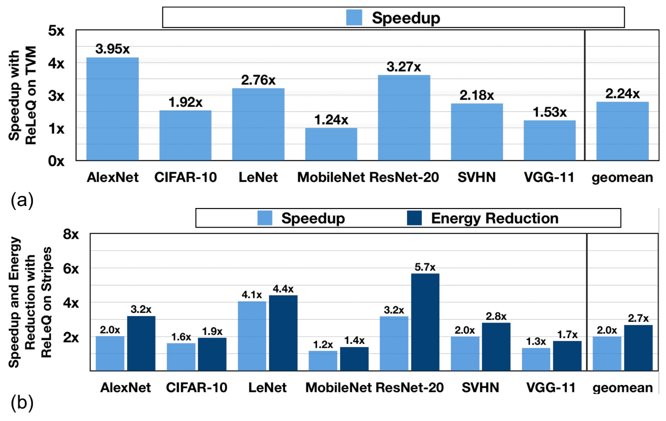


Fig. 4: (a) Speedup with ReLeQ for conventional hardware using TVM over the baseline run using 8 bits. (b) Energy reduction and speedup with ReLeQ for Stripes over the baseline execution when the accelerator is running 8-bit DNNs.

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 that can generalize. Additionally, we also provide a systemic study of different design decisions, and have significant performance gain across diverse benchmarks. The initial version of our work [13], predates HAQ, and it is the first to use RL for quantization. Later HAQ was published in CVPR, and we published initial version of **ReLeQ** in NeurIPS ML for Systems Workshop.

6 Conclusion

This paper sets out to define the automated discovery of bitwidths 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 suggest that a diligent design of our RL framework, which considers multiple concurrent objectives can automatically yield high-accuracy, yet deeply quantized, networks.

References

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] 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.
[3] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained Ternary Quantization," in *ICLR*, 2017.
[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[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*, vol. abs/1802.04799, 2017.
[6] 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)*, pp. 1–12, 2016.
[7] A. K. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide Reduced-Precision Networks," in *ICLR*, 2018.
[8] 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*, vol. abs/1811.01907, 2018.
[9] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," in *ICLR*, 2017.
[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.
[11] 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.
[12] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-Aware Automated Quantization," *arXiv preprint arXiv:1811.08886*, November 21, 2018.
[13] A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Yazdanbakhsh, and H. Esmaeilzadeh, "Releq: A reinforcement learning approach for deep quantization of neural networks," *CoRR*, vol. abs/1811.01704, November 5, 2018. [Online]. Available: http://arxiv.org/abs/1811.01704

Ahmed T. Elthakeb is pursuing his Ph.D. degree in the Alternative Computing Technologies (ACT) lab, University of California San Diego. His current research interests include developing cross-stack solutions to improve the performance and energy efficiency of machine learning algorithms.

Prannoy Pilligundla is pursuing Master's in Computer Science at University of California, San Diego. His research focus is on designing frameworks and cross platform solutions for accelerating machine learning applications.

Fatemehsadat Mireshghallah She is currently pursuing her Ph.D. in computer science (starting 2018) at the University of California, San Diego. Her research focuses on Deep Learning and Privacy.

Amir Yazdanbakhsh joined Google Brain as a Research Scientist in 2019 following a one year AI residency. He obtained his PhD in Computer Science from the Georgia Institute of Technology. His research interests include machine learning, computer architecture, and programming language for hardware design.

Hadi Esmaeilzadeh is currently the inaugural holder of Halicioglu Chair in Computer Architecture with the rank of associate professor in computer science and engineering at the University of California, San Diego, where he was awarded early tenure. His research interests lie at the intersection of architecture, machine intelligence, system design, and software engineering.