Improving Low Precision Network Accuracy with Teacher Assistant Model Distillation

Brian Liao, Aakash Parikh, Jesus Navarro

CS282A Spring 2020

Code Repository:
https://github.com/njesus15/TA-Knowledge-Distillation-for-Low-Precision-Neural-Networks

Problem Statement

Current neural networks are computationally expensive and require high amounts of power to run. One method to improve the performance of neural networks is quantization, where weights and activations are quantized to lower precision values, such as float16, int8, or even binary [0] and ternary [1] weights. However, lower precision weights and activations degrade performance, often underperforming full precision networks. Another technique from compressing large deep neural networks to a shallower, smaller neural network, called model distillation [2], has been shown to improve the performance of low precision networks. Model distillation attempts to encode the learned structure from a large neural network into a smaller one. Likewise, the learned embeddings of a full precision network can be encoded into a network of lower precision. Model distillation can be improved through Teacher Assistants [3], medium-sized networks that can learn the structure from the large network, and distill it in a way that can be learned by a smaller network. This is analogous to a graduate student instructor teaching material from the professor in a way that can be better understood by students.

While Teacher Assistants have been shown to improve the accuracy of smaller model distilled networks, it has not been shown to improve the accuracy of model distilled low precision networks. We wish to investigate if using Teacher Assistants with Model Distillation can improve the accuracy of low precision neural networks. Improving model accuracy of low precision neural networks improves performance and enables them to be run on low power and edge devices.

Background

Due to the large memory and storage usage of state-of-the-art neural network architectures, the deployment of these models on resource-constrained devices remains a challenging task. Reducing the model size to accommodate for the memory and storage limitations severely degrades the performance of neural networks and reduces their overall learning capacity [15, 14]. Several model compression methods exist to address this problem which can be categorized as the following: pruning, low-rank factorization, quantization, and knowledge distillation. For this work, we will focus only on knowledge distillation and quantization in the background section.

Knowledge Distillation

Analogous to a student learning from a teacher, knowledge distillation (KD) trains a smaller model (student network) by mimicking the behavior of a larger, usually trained, model (teacher network) [6]. The key idea is to use soft targets, or logits, from the teacher network to supervise the student network. Soft targets provide the student model more information about the classes that the teacher network found to be most similar to the predicted class. Soft logits, defined as $p_i = exp(z_i/T)/sum(exp(z_i/T))$, are used where T is the temperature hyperparameter which encourages 'softer' logits as T becomes larger. The objective function in the KD framework is a weighted sum of two cross entropies. One term is the cross entropy of the labels and regular softmax student logits and the other is the cross entropy between the student's and teacher's soft logits. Knowledge distillation has been widely adopted for a large range of learning tasks, including model compression.

There are many variants of the original KD framework [6, 7, 8, 9, 10] which, in summary, focus on using other components of the student and teacher networks throughout the learning process. Although KD is a very successful approach, it was empirically shown that KD becomes less effective as the teacher-student gap increases. To address this, Teacher Assistant Knowledge Distillation (TAKD) was proposed [3] to improve the distillation process of a student-teacher network pair with a large gap in model size.

Teacher Assistant Knowledge Distillation (TAKD)

In Teacher Assistant Knowledge Distillation (TAKD), intermediate-sized networks relative to the student and teacher networks are used to compensate for the large student-teacher gap and improve the distillation performance. The intuitive reasoning behind this approach is that teacher networks become so complex that the student network does not have the capacity to mimic the behavior of the teacher network. Thus, this approach consists of an intermediate distillation step between the teacher and teacher assistant network before distilling to the target student network. Theoretical analysis shows that the upper bound of the VC classification error is decreased when adding intermediate networks and empirical analysis shows that the loss landscape is smoothed when intermediate networks are added. Furthermore, it was shown that using multiple teacher assistant networks further improves the distillation process compared to using a single teacher assistant. Our work extends the TAKD approach where the size of the network remains constant and we consider the gap between bit width precision.

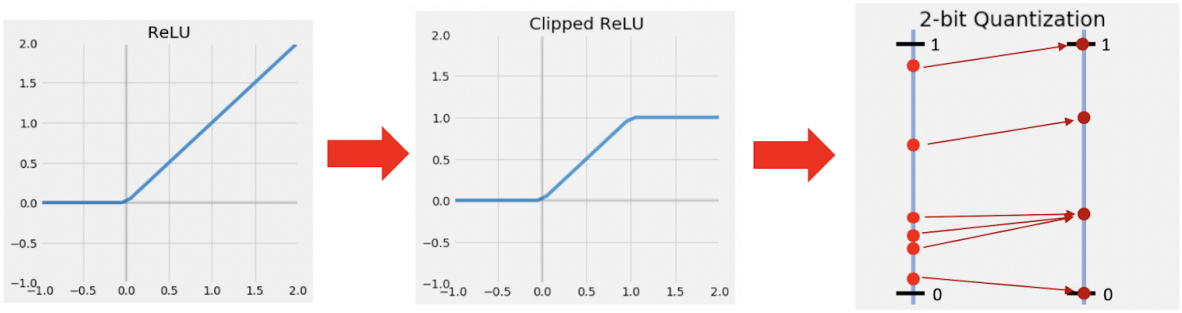
DoReFa Quantization

To perform quantization, we use DoReFa [19], a low bitwidth quantization method which allows for k-bit quantization of weights and activations. Other variants include WPRN and PACT [12, 13] that use different methods to clip the ranges of the activations. K-bit quantization is the

process of quantizing floating point numbers into values that can be represented by k-bits. The quantization function used in DoReFa is defined as

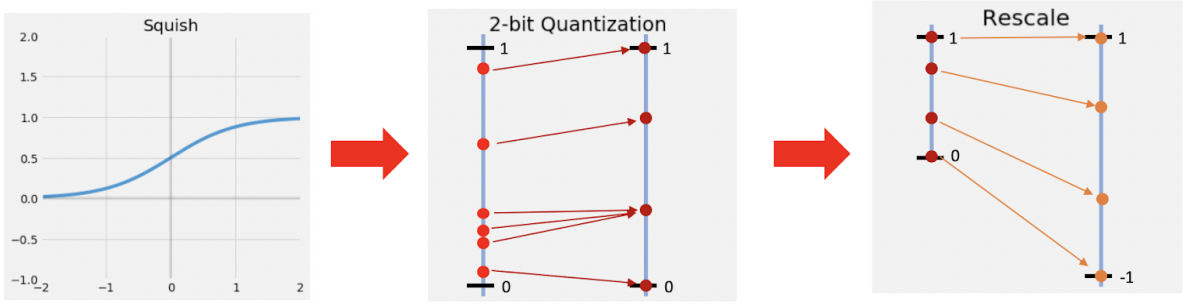
$x_q = quantize_k(x_f) = \frac{1}{2^k-1} * round((2^k - 1) * x_f)$ where $x_f \in [0,\ 1]$ and $x_q \in \{\frac{0}{2^k-1}, \frac{1}{2^k-1}, \ldots \frac{2^k-1}{2^k-1}\}$, a discrete quantized value between 0 and 1. For example, to quantize .21 in 8 bit precision, we have $quantize_8(.21) = \frac{1}{2^8-1} * round((2^8 - 1) * .21) = \frac{4389}{255} = \frac{54}{255}$.

Activations are quantized by first truncating to 0 to 1, then quantizing to k-bits.



Example of 2-bit Activation Quantization

Weights are quantized by first a squashing function $x = squish(w) = \frac{tanh(w)}{2max(|tanh(w)|)} + \frac{1}{2}$ that can be thought of as a sigmoid bounded by the maximum weight value. The final quantized weight is then calculated as $w_q = 2 * quantize_k(squish(w)) - 1$.



Example of 2-bit Weight Quantization

The range of the quantization function is a finite set with cardinality of 2^k, thus the gradient is nearly zero at every point. This is problematic when computing gradients with respect to the loss function during training. Therefore, we adopt the 'straight through estimator' [16, 17] which is an operator applied during the backward pass in the training phase. For the quantization function, $w_o = quantize_k(w_i)$, where w_o and w_i are the quantized and full precision weights respectively, we approximate the partial derivative of the loss with respect to w_i as the partial derivative of the

loss function with respect to w_o. Applying this STE results in a well-defined gradient approximator during the backward pass.

Approach

To evaluate our results, we used both the CIFAR-10 and CIFAR-100 datasets [4], both of which were used for evaluation in the Knowledge Distillation with Teacher Assistant paper. Both datasets include 60,000 color images at 32x32 with a total memory of 160MB, making it feasible for us to train. CIFAR-10 has 10 classes including airplanes, cats, and dogs, while CIFAR-100 has 100 classes including dolphins, spiders and buses.

For training and testing, we used standard pre-processing, including random cropping, random horizontal flipping, and normalization of training and testing images. We processed images with a batch size of 256. For each model, we used a temperature of T=5, a lambda of 0.05, which controls the trade-off between the knowledge distillation and student loss, stochastic gradient descent with 0.9 momentum, 5e-4 weight decay, a learning schedule with a learning rate of 0.1, and a decay of 0.1 every 45 epochs.

We performed all training on the ResNet-20 [5] model at different levels of precision: 32 bit, 16 bit, 8 bit, 4 bit, and 2 bit, and 1 bit, all using DoReFa quantization. Our baseline models were each individually trained quantized models with no knowledge distillation nor use of teacher assistants.

Since we are primarily interested in improving knowledge distillation of low-bitwidth precision neural nets using teacher assistants, weights and activations are only quantized in the forward pass. Therefore, the gradients are maintained at full precision while training. Prior work shows that performance is severely degraded when quantizing the first and final layer of the network, thus, those layers are also kept at full precision. Both the weights and activations are quantized at the same k-bit precision for each network.

Tools

We used PyTorch for training and evaluating our models. PyTorch is more pythonic than Tensorflow in its python idioms. Its environment provides us with a relatively simple and high-level approach to construct convolutional neural networks, while remaining a low-level framework. Additionally, we forked the Github code from the TA paper, which was also written in PyTorch [3] to use as started code. Instead of TA models training smaller network architectures, we modified the code to quantize weights and forward activations in student networks using DoReFa quantization.

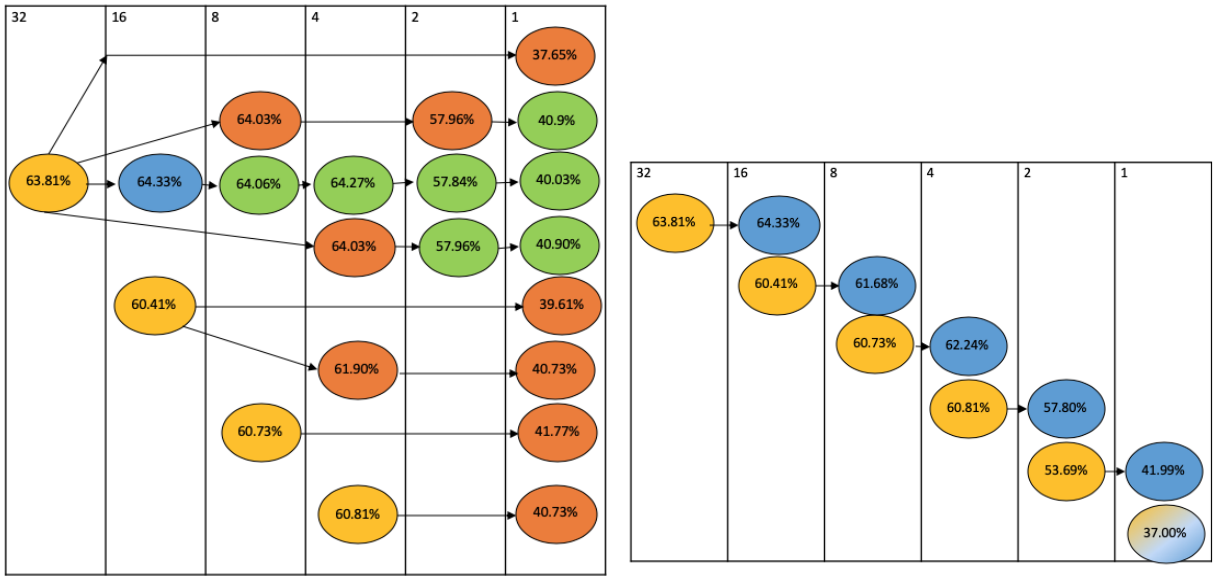
Results

We designed experiments to compare how low precision models performed on image classification on Cifar-10 and Cifar-100. A low precision network can be trained directly from

scratch or can be distilled from a pre-trained higher precision network. We classify these into four main training methods:

1. Individually trained networks. These networks are trained from scratch, the weights are randomly initialized, and we use a standard learning loss
2. Single level distillation. Given a pretrained network, we train a student network with half the precision of the original network. We use a combination of a standard learning loss and a knowledge distillation loss.
3. Skip level distillation. Given a pretrained network, we train a much smaller network, that is less than half of the precision of the original network. This also uses a combination of a standard learning loss and a knowledge distillation loss.
4. TA training. Given a pretrained network, we first use single or skip level distillation to train an intermediate level and then use a single level distillation to train the final lower precision network. Both the intermediate and the desired networks are trained using the combined standard learning loss and knowledge distillation loss.

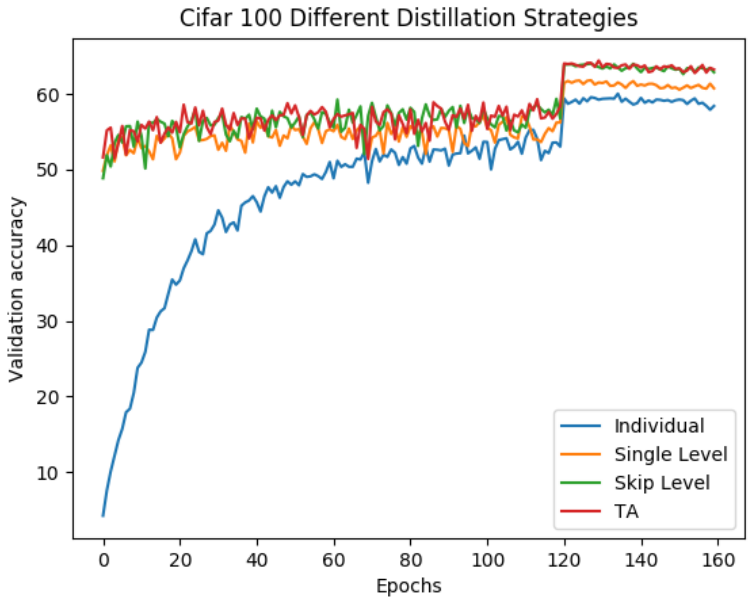
In order to assess these training methods, we train models across a variety of paths and record their accuracies for both Cifar-10 and Cifar-100 datasets.



Training path examples with validation set accuracy for Cifar-100

Validation accuracies for Cifar-10, Cifar-100 data sets for different training methods

Overall, individually trained models perform the worst compared to other training methods. Even higher precision (16 and 8 bit) models give lower accuracies on validation sets in comparison to lower precision models trained via other methods. Validation accuracies for the other three models are generally close in performance. However, out of these three, single level training has the lowest scores and skip level training performs slightly worse than TA training in most cases.



Training a 4-bit low precision model with various training methods

Additionally, using model distillation strategies results in faster training and a higher plateau over individually training a model. TA style training performs the best, having a better initial performance and a slightly higher final performance. Skip level training plateaus at a higher level than single level training.

Lessons Learned

Model Distillation with TA for Low Precision Neural Networks does have an improvement over individually training them and other model distillation strategies. Skip level training generally works better than a single level distillation and individually trained models perform the poorest.

One surprising result was that skip level training performed better than a single level distillation in most cases. As a single level training uses a teacher model closer in precision to the target model, we had predicted that the carry over would be better. However, the better performance in the skip level training is possibly attributed to a better initial performance in the teacher model, which is at a higher precision than in single level training. However, when the difference in precision between two models is large (ie. 32 bit teacher, 1 bit student) the skip level training suffers, performing similarly to individually training a 1 bit student.

We also found that TA models benefited from knowledge accumulation, through many iterations of training, distilling and training again. However, due to the dual nature of our loss, which included a standard learning loss and a knowledge distillation loss, we believe that mistakes from a teacher model can also propagate over time.

One interesting future study could be made on looking at how this component of the loss function affects TA training. This component pushes the student network to look similar to the teacher network. Removing it could allow the network to move away from the local minima found by the teacher network into a possibly lower local minima.

Another interesting direction would be to look at modifying the training set to improve student networks in TA training. For example, we could modify our training set to give higher weights to examples that the teacher network or an individually trained low precision network could not classify in an attempt to have the student network learn these. This is a similar idea to AdaBoost [18], but for model distillation rather than for ensemble training.

Team Contributions

Brian worked on converting the Knowledge Distillation with TA starter code to have a pipeline for low precision TA Knowledge Distillation. Brian debugged and was able to combine Aakash and Jesus' work in adding low precision code into our training pipeline.

Aakash worked on transferring weights from teacher and TA networks to student networks. He also worked on determining training paths to run and analyzing the results.

Jesus worked on implementing the DoReFa quantization method on the ResNet architecture. He also ran Aakash's training path experiments on the Cifar-10 dataset.

All members worked on training networks, parallelizing the process. We all contributed equally to the project (33% each).

References

[0] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. arXiv preprint arXiv:1603.05279, 2016.

[1] C. Zhu, S. Han, H. Mao, and W. J. Dally, “Trained ternary quantization,” in Proc. ICLR, 2017.

[2] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[3] Mirzadeh, S., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. AAAI 2020, abs/1902.03393.

[4] A. Krizhevsky, V. Nair, and G. Hinton, “The cifar-10 dataset,”online:http://www. cs. toronto. edu/kriz/cifar. html, vol. 55, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[6] Hinton, Geoffrey & Vinyals, Oriol & Dean, Jeff. (2015). Distilling the Knowledge in a Neural Network.

[7] Romero, A.; Ballas, N.; Kahou, S.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

[8] Yu, R.; Li, A.; Morariu, V. I.; and Davis, L. 2017. Visual relationship detection with internal and exter- nal linguistic knowledge distillation. In ICCV.

[9] Czarnecki, W.; Osindero, S.; Jader- berg, M.; Swirszcz, G.; and Pascanu, R. 2017. Sobolev training for neural networks. In NIPS, 4278–4287.

[10] Tarvainen, A., and Valpola, H. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In NIPS.

[11] Zhang,Y. Xiang,T. Hospedales,T. and Lu, H. 2017. Deep mutual learning. CoRR abs/1706.00384.

[12] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085, 2018.

[13] A. Mishra, E. Nurvitadhi, J. Cook, and D. Marr. Wrpn: Wide reduced-precision networks. arXiv preprint arXiv:1709.01134, year=2017.

[14] Boris Hanin. Universal function approximation by deep neural nets with bounded width and ReLU activations. arXiv preprint arXiv:1708.02691, 2017.

[15] Boris Hanin and Mark Sellke. Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[16] Hinton, Geoffrey, Srivastava, Nitsh, and Swersky, Kevin. Neural networks for machine learning. *Coursera, video lectures*, 264, 2012b.

[17] Bengio, Yoshua, Léonard , Nicholas, and Courville, Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[18] Zhu, J., Rosset, S., Zou, H., and Hastie, T. Multi-class AdaBoost. Technical Report 430, Department of Statistics, University of Michigan, 2006.

[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, Jun. 2016, [online] Available: https://arxiv.org/abs/1606.06160.