COMPRESSION TECHNIQUES FOR DEEP NEURAL NETWORKS

Naveen Lalwani [1] **Rangeesh Muthaiyan** [1]

ABSTRACT

Neural Networks today find their application in variety of fields, from Computer Vision to Natural Language processing, from Speech Recognition to classification. But they are computationally expensive, memory intensive and difficult to deploy on embedded systems. To attempt to solve this problem, the project will present the survey of degree of compression and the performance of various Deep Neural Network(DNN) compression techniques like Quantization, Pruning and Knowledge Distillation(KD). The motivation for the proposed project is to leverage some/all of these DNN acceleration approaches to reduce the size of DNN models with little to no loss of accuracy to make the baseline models as fast and compressed as possible to enable the use of energy-efficient and real-time processing DNNs on embedded systems. The compressed models will be benchmarked on two different CPU and GPU configurations using MNIST and CIFAR-10 datasets.

1 INTRODUCTION

Deep Neural networks have become ubiquitous today being employed in a wide variety of applications but they involve a lot of computations and require a lot of memory storage. With modern DNNs having more than a billion parameters, the amount of memory required to store them is significantly high. This makes them difficult to deploy them on embedded systems with limited hardware and processor resources. For example, iPhones do not allow installation of apps having memory requirements over 100 MBs on the cellular networks. Also, the energy consumption aspect of the DNNs pose another hindrance to their use in smartphones and embedded technologies. Under 45nm CMOS technology, a 32 bit floating point add consumes 0.9pJ, a 32 bit SRAM cache access takes 5pJ, while a 32 bit DRAM memory access takes 640pJ (Han et al., 2016). The amount of energy that will be consumed in computing a billion parameters will quickly drain the device and is beyond the power envelope of a typical embedded device. Also, since the network won't be able to fit in on-chip storage, it would require costly DRAM accesses which makes the real time processing a challenge on the smartphones.

Thus, there is a need to compress these neural network models by reducing the number of insignificant parameters & weights and storing them with sufficient precision to make the DNN storage and energy efficient while at the same time maintaining the accuracy of the original model. The reduced precision and parameters will make the model faster due to quick arithmetic operations between small-bits parameters and also, due to reduced number of parameters. The smaller models can be used in the mobile systems for processing rather than using cloud to process the data.

The project proposes the use of compression techniques like Quantization, Weights and Channel Pruning and Knowledge Distillation to reduce the size of the DNNs and investigate the compatibility of these techniques in a single pipeline to attain further compression while preserving the accuracy and reducing loss of energy to ultimately achieve faster acceleration.

2 RELATED WORK

The early work of LeCun et al.(LeCun et al., 1990) , proposed a primitive way of reducing the parameters with small "saliency" without affecting the training error. They suggested pruning of weights by computing the second order derivatives and saliency of each parameter to delete unimportant weights which would allow reduction of the size of the neural network by the factor of six. But a major flaw of this approach was that the second order derivatives required additional computation.
Since then, there have been several other methods proposed to reduce the network complexity either by making the network sparser by pruning or by changing the way weights are stored and computed by quantization techniques. (Han et al., 2015) presented a method in which after an initial training phase, all the connections whose weights were lower than a

[*]Equal contribution [1]Electrical and Computer Engineering, Carnegie Mellon University, USA. Correspondence to: Naveen Lalwani <naveenl@andrew.cmu.edu>, Rangeesh Muthaiyan <rmuthaiy@andrew.cmu.edu>.

Proceedings of the 2^{nd} SysML Conference, Palo Alto, CA, USA, 2019. Copyright 2019 by the author(s).

set threshold were removed, thereby, converting a fully connected layer into a sparse layer. A critical aspect of the work was to retrain the pruned network to compensate for the lost connections to maintain the same accuracy. The work resulted in the reduction of the network size significantly, by a factor of 9x to 13x on modern state of the art neural networks. Continuing their work (Han et al., 2016) proposed on applying quantization and coding techniques to achieve more compression. Their method implemented weight sharing to reduce the memory storage of weights along with pruning techniques which resulted in the reduction of the network size further by a factor of 25x to 30x. They also proposed Huffman coding for further lossless compression of the weights and were able to reduce the network size even more by a factor of 35x to 49x with a minimal loss of accuracy.

To the best of our knowledge, exploiting knowledge transfer (KT) to compress model was first proposed by (Buciluă et al., 2006). They trained a compressed/ensemble model of strong classifiers with pseudo-labeled data, and reproduced the output of the original larger network. But the work was limited to shallow models. Knowledge Distillation method was recently proposed by (Hinton et al., 2015) as an approach to neural network quality improvement. The paper proposes the use of the pre-trained modern state of the art deep and dense neural networks to transfer their generalizations to shallower or smaller neural network models. They introduced a KD compression framework, which eased the training of deep networks by following a student-teacher paradigm, in which the student was penalized according to a softened version of the teacher's output. (Mishra & Marr, 2017) and (Mishra et al., 2018) successfully implemented their method in the following form: a full-precision model was used as a model-teacher, and quantized neural network as a model-student, thus combining knowledge distillation with quantization. Such paradigm of learning gives not only a higher quality of the quantized network inference, but also allows reducing the bit capacity of quantized data and at the same time having an acceptable level of accuracy.

3 TECHNICAL DESCRIPTION

3.1 Quantization

Quantization refers to the process of reducing the number of bits that represent a number. In the context of deep learning, the predominant numerical format used for research and for deployment has so far been 32-bit floating point, or FP32. Thus, quantization is used to reduce the model precision from FP32 to n-bits integers (commonly used is INT8 or 8-bits integer). The parameters are mapped from a space of 32 bit floating point to 8 bit integers to accelerate inference on processors that support low precision math with reduced memory bandwidth. (Goncharenko et al., 2018)

3.2 Pruning

Pruning is a method to reduce the storage and computation required by neural networks by learning only the important connections, thereby, converting a fully connected layer into a sparse one. One of the methods proposed to prune the network follows a three step process. First, training of the network to learn which connections are important. Second step is to prune away the less significant weights, which could either be done by absolute value pruning or by a hessian loss function. The third and the final step is to retrain the pruned network to maintain accuracy. Second and third steps are repeated many times to perform iterative pruning which helps further to boost compression without loss of accuracy when compared to single aggressive pruning.

3.3 Knowledge Distillation

Knowledge Distillation is a model compression method in which a small model (generally, a shallow neural network) is trained to mimic a pre-trained, larger model (or ensemble of models). This training setting is sometimes referred to as "teacher-student", where the large model is the teacher and the smaller model is the student. The teacher model transfer generalizations to the student model rather than weights as in transfer learning. In Knowledge Distillation, the teacher models passes the "Dark Knowledge" (coined by (Hinton et al., 2015)) to the student model i.e. which classes the teacher found to be more similar to the predicted class. Thus, Dark knowledge is the information, that, given the input, which classes are more likely to be the output besides the predicted class and is therefore, information about the relation of the input to the each class.(ner, 2017)

4 METHODOLOGY

The DNN models selected for the study are LeNet-300-100 and LeNet-5. The training of the initial baseline models LeNet-300-100 and LeNet-5 has been done using Tensorflow and Keras framework on MNIST and CIFAR-10 datasets. The reason for selecting Tensorflow and Keras rather than earlier proposed PyTorch and Neural Network Distiller is that Tensorflow provides backend support for Keras and Scikit libraries with added advantage of support for storing INT8 values which is crucial for this project. The reason for selecting the above-mentioned DNN models is that while LeNet performs impressively on MNIST, they do not perform that well on CIFAR-10 and we want to observe the effect of compression techniques on models which are

[0]Project: https://github.com/Naveen-Lalwani/18663-Study-of-Neural-Network-Compression-Techniques/blob/master/ReadMe.md

[0]For the code files or models, please contact the authors to get the access to the private GitHub repository.

not the best architecture for that dataset. This will give us an insight if the compressed models perform similarly or better or worse and how much does the accuracy of the baseline model affects the compressed model.
The baseline models are trained after tuning the learning rate, number of epochs, optimization technique and batch size. A lot of time was spent in training the model to get the best test accuracy while avoiding overfitting and aiming for better generalization ability. The training and the inferences are taken on the following hardware setups:

- Intel Core i7-8750H, NVIDIA GeForce GTX 1050Ti Max-Q[1]
- Intel Core i7-8565U, NVIDIA GeForce MX150[2]

Inference time per image has been taken on both the setups and is measured over the entire test data of MNIST and CIFAR-10. Inference time per image is measured by employing the Time class of Python and time is measured at the beginning and at the end of the inference for the entire dataset and the time at the start is subtracted from the end time (popularly known as tic-toc method). The inference time per image is the average of the 10 observations taken so as to account for any noise in measurement due to the system.
Accuracy measured for comparison is the test accuracy and is measured against the baseline model which is compressed. It is presented individually due to baseline models being trained separately and not all compressed models are derived from the same baseline model. The training of baseline models yield different accuracy in some cases despite using same or similar parameters is because of the fact that the work was divided between the authors and different frameworks were employed by both of them (one employed Tensorflow while other employed Keras with Tensorflow as backend). Despite this, the results are not very different.

5 SCHEDULE

We achieved the following goals by Milestone 1 & 2:

1. **Milestone 1** (by April 1^{st})
   - Performed INT8 Quantization on LeNet300-100 and LeNet5 on MNIST and CIFAR-10.
   - Performed Weight Pruning on LeNet300-100 on MNIST and CIFAR-10.
   - Performed Filter Pruning on LeNet5 on MNIST and CIFAR-10 (suggested by Ruizhou Ding)
   - Performed Knowledge Distillation on LeNet300-100 and LeNet5 on MNIST and CIFAR-10.
   - Measure speedup, compression size and accuracy.

2. **Milestone 2** (by May 1^{st})
   - Performed Pruning and Quantization on LeNet300-100 on MNIST and CIFAR-10.
   - Performed Pruning and Quantization on LeNet5 on MNIST and CIFAR-10.
   - Performed Knowledge Distillation and Quantization on LeNet300-100 on MNIST and CIFAR-10.
   - Performed Knowledge Distillation and Quantization on LeNet5 on MNIST and CIFAR-10.
   - Recorded compression size and accuracy.
   - Measured speedup on both the hardware configurations.
   - Prepared poster, final report and presentation.

6 DIVISION OF LABOR

As Helen Keller had rightly said "Alone we can do so little; together we can do so much." We equally divided our work so that no one was under pressure and were able to learn a lot from the project and that is why we were able to reach our set goal for Milestone 1 and Milestone 2 without any hicupp. Naveen Lalwani handled the INT8 Quantization of both LeNet models on MNIST and CIFAR-10 while Rangeesh Muthaiyan handled pruning on both LeNet models on both the datasets. As it was suggested by TA Ruizhou Ding to look into Filter Pruning for better compression, Rangeesh took this task under his wing while Naveen took the implementation of Knowledge Distillation on both the models on MNIST and CIFAR-10 under himself. We kept each other in loop of our training parameters and about the results.
For the second milestone, Naveen handled the Knowledge Distillation followed by Quantization pipeline on all models and datasets while Rangeesh implemented the Pruning followed by Quantization on all models and datasets. The inference time per image on all the models was handled by both the authors on their respective hardware configurations. We also trained baseline models again to ensure we were getting similar accuracy as compared to original results and accordingly made updates. We tried to measure power using the NVIDIA System Management Interface (nvidia-smi) utility but our GPUs do not support nvidia-smi and thus, we were not able to do this part of our study. Also, constrained by time and other reasons, we weren't not able to implement the valuable suggestions that we had received during our poster presentation which we have now kept as goals for our future work.

[1]: Config1
[2]: Config2

7 RESULTS & DISCUSSION

7.1 Quantization

We employed Quantization-aware model training which ensured that the forward pass matches precision for both training and inference which had two aspects to this:

- Operator fusion at inference time was accurately modeled at training time.
- Quantization effects at inference were modeled at training time.

For LeNet-300-100, we measured accuracy, inference time per image, and size of parameters for different precision and recorded the following results.

Table 1. Effect of precision on LeNet-300-100 on MNIST

Precision	**Size (kB)**	**Accuracy (%)**
FLOAT64	2090	96.54
FLOAT32	1043	96.49
FLOAT16	521	96.43
INT8	266	96.42

For other models we confined our study to commonly used FP32 which we chose as reference and INT8 precision as compressed, and tabulated the results as follows:

Table 2. Quantization

Dataset	**Model**	**Precision**	**Size (kB)**	**Accuracy (%)**	**Inference Time / Image (sec)**	
					Config1	**Config2**
MNIST	LeNet-5	FP32	244	98.44	3.00E-04	4.44E-04
		INT8	67	98.42	2.33E-04	2.34E-04
CIFAR-10	LeNet-300-100	FP32	3724	43.57	1.05E-03	1.21E-03
		INT8	936	43.85	4.67E-04	5.24E-04
	LeNet-5	FP32	245	61.30	1.11E-03	1.16E-03
		INT8	68	61.09	9.03E-04	1.02E-03

7.2 Pruning

Two types of pruning were employed to accomodate the different model architectures, weight pruning and filter pruning. Weight pruning to remove connections from the fully connected dense layers and filter pruning to prune filters in a convolutional layer. Since LeNet-300-100 only has fully connected dense layer only weight pruning could be done. In case of LeNet-5 we could do weight pruning in the two fully connected layers and do filter pruning on the two convolutional layers. For lossless pruning of the model, we need to retrain the pruned model again to recover some of the accuracy. But pruning can be done to only a certain limit, because after a certain sparsity level there is a significant loss in weight information and the model accuracy takes a big hit.

We performed Pruning on LeNet-300-100 and LeNet-5 models trained on MNIST and CIFAR-10 datasets and recorded their accuracy, size and inference time per image along with the number of parameters per layer before and after pruning.

Table 3. Pruning LeNet-300-100 on MNIST

Sparsity	**Size (kB)**	**Accuracy (%)**	**Inference Time / image (sec)**	
			Config1	**Config2**
Original	1043	97.35	2.79E-04	2.88E-04
50%	493	98.09	1.42E-04	1.49E-04
60%	390	97.76	1.17E-04	1.24E-04
70%	290	97.74	8.82E-05	9.41E-05
80%	191	97.62	6.60E-05	6.33E-05
90%	95	87.56	3.57E-05	3.91E-05

Table 4. Total Parameters for LeNet-300-100 on MNIST

Sparsity	**Dense**	**Dense1**	**Dense2**	**Total Parameters**
Original	235200	30000	1000	266200
50%	117600	7500	500	125600
60%	94080	4800	400	99280
70%	70560	2700	300	73560
80%	47040	1200	200	48440
90%	23520	300	100	23920

Table 5. Pruning LeNet-5 on MNIST

Layer Pruned	**Size (kB)**	**Accuracy (%)**	**Inference Time / image (sec)**	
			Config1	**Config2**
Original	245	98.97	3.00E-04	4.40E-04
Dense	24	95.75	2.53E-04	4.08E-04
Dense + Conv	14	94.25	2.32E-04	3.38E-04

Table 6. Total Parameters for LeNet-5 on MNIST

Model	**Conv 1**	**Conv 2**	**Dense 1**	**Dense 2**	**Dense 3**	**Total Parameters**
Original	156	2416	30840	10164	850	44426
Pruned	156	604	1028	15	40	1843

Table 7. Pruning LeNet-300-100 on CIFAR-10

Sparsity	**Size (kB)**	**Accuracy (%)**	**Inference Time / image (sec)**	
			Config1	**Config2**
Original	3724	48.55	1.05E-03	1.21E-03
50%	1834	46.41	5.20E-04	6.53E-04
70%	1094	46.19	3.17E-04	3.39E-04
80%	728	44.41	2.17E-04	2.37E-04

Table 8. Total Parameters of LeNet-300-100 on CIFAR-10

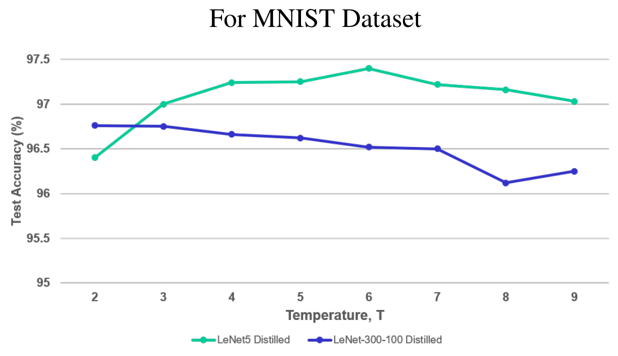
Sparsity	Dense1	Dense1	Dense2	Total Parameters
Original	921600	30000	1000	952600
50%	460800	7500	500	468800
70%	276489	2700	300	279489
80%	184320	1200	200	185720

Table 9. Pruning LeNet-5 on CIFAR-10

Layer Pruned	Size (kB)	Accuracy (%)	Inference Time / image (sec)	
			Config1	Config2
Original	245	53.73	1.11E-03	1.16E-03
Dense	117	48.91	1.11E-03	1.20E-03
Dense + Conv	64	47.89	1.02E-03	1.10E-03

Table 10. Total Parameters of LeNet-5 on CIFAR-10

Model	Conv 1	Conv 2	Dense 1	Dense 2	Dense 3	Total Parameters
Original	456	2416	48120	10164	850	62006
Pruned	456	1208	12462	1512	100	15738



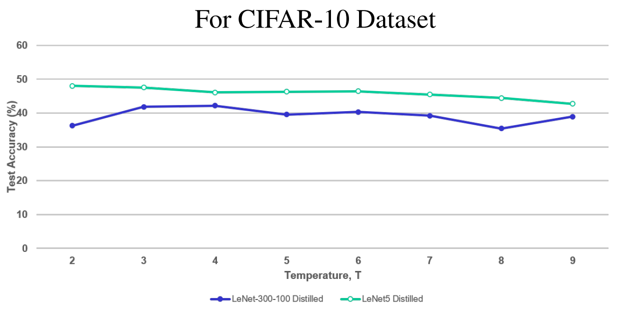


Figure 1. Effect of Temperature on Accuracy

7.3 Knowledge Distillation

For implementing knowledge distillation, we trained the LeNet models on MNIST and CIFAR-10 and transferred their dark knowledge to a shallow neural network. We chose a shallow neural network having 1 hidden layer with 50 hidden units. We trained this shallow model and tried to optimize it′s performance to the maximum by tuning the hyperparameters and compared it′s performance with the same model trained using the dark knowledge. We found the distilled model to be far better and also found that it required less number of epochs to converge. The distilled model was almost near the teacher model in performance for the MNIST dataset while for the CIFAR-10 dataset, it performed better than the shallow model but was worse in terms of accuracy as compared to the teacher model. We transferred the dark knowledge using the following equation (Hinton et al., 2015):

$$q_i = \frac{e^{\frac{Z_i}{T}}}{\sum_j e^{\frac{Z_j}{T}}} \qquad (1)$$

where,
q_i are the soft probability values
Z_i are the logits of teacher model
T is the temperature
We trained the model for various temperature values to find the best temperature that optimized the performance of the student model and selected the temperature that gave the best accuracy for same parameters for the student model.

We now report our results on the performance of the student model and compare it with the teacher models for both the datasets.

Table 11. Knowledge Distillation Results on MNIST

Model	Size (kB)	Accuracy (%)	Inference Time / Image (sec)		Epochs
			Config1	Config2	
Student Model	156.15	86.57	5.35E-05	5.70E-05	300
LeNet-300-100	1043	97.59	2.79E-04	2.88E-04	N/A
LeNet-5	244	98.93	3.00E-04	4.40E-04	N/A
LeNet-300-100 Distilled	156.15	96.76	5.35E-05	5.70E-05	50
LeNet-5 Distilled	156.15	97.24	5.35E-05	5.70E-05	50

Table 12. Knowledge Distillation Results on CIFAR-10

Model	Size (kB)	Accuracy (%)	Inference Time / Image (sec)		Epochs
			Config1	Config2	
Student Model	604	16.01	1.65E-04	1.87E-04	4000
LeNet-300-100	3724	50.35	1.05E-03	1.21E-03	N/A
LeNet-5	245	62.40	1.11E-03	1.16E-03	N/A
LeNet-300-100 Distilled	604	42.18	1.65E-04	1.87E-04	50
LeNet-5 Distilled	604	48.08	1.65E-04	1.87E-04	50

7.4 Pruning + Quantization

We pruned the base model and then in the same pipeline, we quantized the pruned model and recorded the accuracy, inference time per image and the size of various quantized models of different sparsity along with their non-quantized counterparts. This has been done for both the models LeNet-300-100 and LeNet-5 on MNIST and CIFAR-10 datasets.

Table 13. Pruning & Quantization on LeNet-300-100 on MNIST

Sparsity	Precision	Size (kB)	Accuracy (%)	Inference Time / image (sec)	
				Config1	Config2
Original	FP32	1043	97.35	2.79E-04	2.88E-04
	INT8	266	97.36	1.34E-04	1.47E-04
50%	FP32	493	98.09	1.42E-04	1.49E-04
	INT8	127	98.05	7.71E-05	7.92E-05
60%	FP32	390	97.76	1.17E-04	1.24E-04
	INT8	101	97.76	6.63E-05	7.12E-05
70%	FP32	290	97.74	8.82E-05	9.42E-05
	INT8	75	97.77	5.75E-05	5.78E-05
80%	FP32	191	97.62	6.60E-05	6.33E-05
	INT8	50	97.56	4.56E-05	4.25E-05
90%	FP32	95	87.56	3.57E-05	3.91E-05
	INT8	26	87.57	3.22E-05	3.08E-05

Table 14. Pruning & Quantization on LeNet-5 on MNIST

Layer Pruned	Precision	Size (kB)	Accuracy (%)	Inference Time / Image (sec)	
				Config1	Config2
Original	FP32	245	98.97	3.00E-04	4.40E-04
	INT8	69	98.96	2.33E-04	2.34E-04
Dense	FP32	24	95.75	2.53E-04	4.08E-04
	INT8	10	95.68	2.26E-04	2.30E-04
Dense+ conv	FP32	14	94.25	2.32E-04	3.38E-04
	INT8	7	94.25	2.15E-04	2.17E-04

Table 15. Pruning & Quantization on LeNet-300-100 on CIFAR10

Sparsity	Precision	Size (kB)	Accuracy (%)	Inference Time / image (sec)	
				Config1	Config2
Original	FP32	3724	48.55	1.05E-03	1.21E-03
	INT8	937	48.54	4.67E-04	5.24E-04
50%	FP32	1834	46.41	5.20E-04	6.53E-04
	INT8	462	46.46	2.67E-04	3.45E-04
70%	FP32	1094	46.19	3.17E-04	3.39E-04
	INT8	276	46.13	1.94E-04	2.33E-04
80%	FP32	728	44.41	2.17E-04	2.37E-04
	INT8	184	44.03	1.55E-04	1.76E-04

Table 16. Pruning & Quantization on LeNet-5 on CIFAR-10

Layer Pruned	Precision	Size (kB)	Accuracy (%)	Inference Time / Image (sec)	
				Config1	Config2
Original	FP32	245	53.73	1.11E-03	1.16E-03
	INT8	68	53.82	9.03E-04	1.02E-03
Dense	FP32	117	48.91	1.11E-03	1.20E-03
	INT8	33	48.67	8.98E-04	1.02E-03
Dense+ conv	FP32	64	47.89	1.02E-03	1.10E-03
	INT8	20	48.00	8.88E-04	9.81E-04

7.5 Knowledge Distillation + Quantization

We now consider another DNN compression pipeline where we distill the knowledge of the teacher model in the student model and quantize the student model to INT8 precision for both the datasets MNIST and CIFAR-10 for both LeNet-300-100 and LeNet-5 models. This allows us to leverage two orthogonal compression techniques in a single pipeline.

Table 17. KD + Quantization Results on MNIST

Model	Precision	Size (kB)	Accuracy (%)	Inference Time / Image (sec)	
				Config1	Config2
LeNet-300-100	FP32	1043	97.59	2.79E-04	2.88E-04
	INT8	266	97.53	1.31E-04	1.35E-04
LeNet-5	FP32	244	98.93	3.00E-04	4.40E-04
	INT8	67	98.92	2.33E-04	2.34E-04
LeNet-300-100 Distilled Model	FP32	156.15	96.76	5.25E-05	5.64E-05
	INT8	42	96.77	3.66E-05	3.95E-05
LeNet-5 Distilled Model	FP32	156.15	97.31	5.45E-05	5.74E-05
	INT8	42	97.29	3.78E-05	3.89E-05

Table 18. KD + Quantization Results on CIFAR-10

Model	Precision	Size (kB)	Accuracy (%)	Inference Time / Image (sec)	
				Config1	Config2
LeNet-300-100	FP32	3724	50.35	1.05E-03	1.21E-03
	INT8	936	50.29	4.76E-04	9.03E-04
LeNet-5	FP32	245	62.40	1.11E-03	1.16E-03
	INT8	68	62.23	5.24E-04	1.02E-03
LeNet-300-100 Distilled Model	FP32	604	42.18	1.65E-04	1.33E-04
	INT8	154	42.07	1.90E-04	1.46E-04
LeNet-5 Distilled Model	FP32	604	48.08	1.64E-04	1.35E-04
	INT8	154	48.01	1.84E-04	1.43E-04

8 CONCLUSIONS

From the results, we see that INT8 quantization leads to a 4 times reduction in size for all models with the accuracy between $\pm$ 0.3% of the baseline models, with even better accuracy in one of the cases. We can also see that the speedup per image is almost 1.4x to 2x more than the baseline models.

From the results of pruning, the best compression on LeNet-300-100 without loosing accuracy is 5x with sparsity level of 80% and for LeNet-5 the best possible compression obtained was about 18x with an increase in speedup of 4x and 1.2x respectively. The maximum loss in accuracy for the highest sparsity is of 6% while for most other pruned models, the accuracy loss is within 3%.

It has been verified that knowledge distillation improves the ability of shallow neural networks which are smaller in size to perform better and that too, with less training and efforts. This provides us a way to make use of already trained large neural networks architectures to train smaller models and deploy smaller models in the embedded systems. Though knowledge distillation doesn't always guarantee reduction in size, it does shows acceleration in inference time. We can see a increase in speedup per image to be 6x to 7x in all the cases for both the hardware configurations. It can also be seen that better the teacher model, better is the student model. Thus, we can safely say that the use of more specialized model will further improve the accuracy and performance of student models. We also notice that Temperature doesn't show a specific trend and thus, we consider it to be a hyperparameter that needs to be tuned to get the best performance for shallower models. The accuracy loss is not significant on MNIST dataset but it is quite significant on CIFAR-10 but we think that with more training and more tuning, even this deviation of accuracy from the baseline model can be decreased and with better teacher models, we can expect the shallow model to perform on par with the baseline models.

We received great results on leveraging these compression techniques together in a single pipeline. We considered the two cases: Pruning followed by quantization of the pruned model and Distilling knowledge of the baseline model in the student model followed by quantization of the student model. For the former pipeline, we achieved about 2x to 8x speedup on LeNet-300-100 on MNIST and about 2x to 7x speed up on CIFAR-10. For LeNet-5 on MNIST we observe a speed up of 1.2x to 1.4x and similar speedups in CIFAR-10 and we observed a compression in size by 20x to 40x with almost minimal loss in accuracy.For the latter pipeline, for the student model selected, we observe 25x reduction in size for LeNet-300-100 models and there is 11x-13x speedup as compared to original teacher models.

We finally present the compression size and inference time per image on Config1 for our best models.

Table 19. Size in kB for the best models

Compression Technique	MNIST		CIFAR-10	
	LeNet-300-100	LeNet-5	LeNet-300-100	LeNet-5
Original	1043	244	3724	245
Quantization	266	67	936	68
Pruning	191	14	728	64
Knowledge Distillation	156.15	156.15	604	604
Pruning + Quantization	50	7	184	20
KD + Quantization	42	42	154	154

Table 20. Inference time per image(sec) of the best models

Compression Technique	MNIST		CIFAR-10	
	LeNet-300-100	LeNet-5	LeNet-300-100	LeNet-5
Original	1.31E-04	3.00E-04	1.05E-03	1.11E-03
Quantization	1.31E-04	2.33E-04	4.67E-04	9.03E-04
Pruning	6.60E-05	2.32E-04	2.17E-04	1.02E-03
Knowledge Distillation	5.35E-05	5.35E-05	1.65E-04	1.65E-04
Pruning + Quantization	4.56E-05	2.15E-04	1.55E-04	8.88E-04
KD + Quantization	3.66E-05	3.78E-05	1.90E-04	1.84E-04

9 FUTURE SCOPE OF WORK

We plan to run these models on a machine which supports nvidia-smi utility to record the power consumed by these models during inference. We received a few suggestions during poster presentation and we plan to implement them as well. One of the suggestions that stands out was given by TA Rudy Chin which was to distill the knowledge of the original model (teacher) to a pruned model of the original model (student) and then quantize this student model. Another suggestion that we received was to prune a distilled model and then quantize the pruned distilled model. We had thought of doing the latter suggestion after achieving milestone 2 but couldn't due to various other commitments.

10 ACKNOWLEDGEMENT

We sincerely would like to thank Prof. Diana Marculescu and the TAs for the course Ting-Wu Chin and Ruizhou Ding for their support, suggestions, guidance and help. We would also like to thank our classmates for their suggestions and ideas which helped us perform better.

REFERENCES

Compression scheduling - neural network distiller. https://nervanasystems.github.io/distiller/schedule/index.html, 2017. [Online; Accessed 2 Mar. 2019].

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 535–541, New York, NY, USA, 2006. ACM.

Goncharenko, A., Denisov, A., Alyamkin, S., and Terentev, E. Fast adjustable threshold for uniform neural network quantization. *CoRR*, abs/1812.07872, 2018. URL http://arxiv.org/abs/1812.07872.

Han, S., Pool, J., J.Tran, and Dally, W. J. Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1135–1143. Curran Associates, Inc., 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In Touretzky, D. S. (ed.), *Advances in Neural Information Processing Systems 2*, pp. 598–605. Morgan-Kaufmann, 1990. URL `http://papers.nips.cc/paper/250-optimal-brain-damage.pdf`.

Mishra, A., Nurvitadhi, E., Cook, J. J., and Marr, D. WRPN: Wide reduced-precision networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1ZvaaeAZ`.

Mishra, A. K. and Marr, D. Using knowledge distillation techniques to improve low-precision network accuracy. *CoRR*, abs/1711.05852, 2017. URL `http://arxiv.org/abs/1711.05852`.