KD-Lib: A PyTorch library for Knowledge Distillation, Pruning and Quantization

Het Shah,[1] **Avishree Khare,**[2*] **Neelay Shah,**[3*] **Khizir Siddiqui** [4*]
Birla Institute of Technology and Science Pilani, K. K. Birla Goa Campus, India
{f20170093[1], f20170112[2], f20180400[3], f20180439[4]}@goa.bits-pilani.ac.in

Abstract

In recent years, the growing size of neural networks has led to a vast amount of research concerning compression techniques to mitigate the drawbacks of such large sizes. Most of these research works can be categorized into three broad families : Knowledge Distillation, Pruning, and Quantization. While there has been steady research in this domain, adoption and commercial usage of the proposed techniques has not quite progressed at the rate. We present KD-Lib, an open-source PyTorch based library, which contains state-of-the-art modular implementations of algorithms from the three families on top of multiple abstraction layers. KD-Lib is model and algorithm-agnostic, with extended support for hyperparameter tuning using Optuna and Tensorboard for logging and monitoring. The library can be found at - https://github.com/SforAiDl/KD_Lib

Introduction

Deep neural networks (DNNs) have gained widespread popularity in recent years, finding use in several domains including computer vision, natural language processing, human computer interaction and more. These networks have achieved remarkable results on several tasks, often even surpassing human-level performance.

The number of parameters of such DNNs often increase multi-fold with an increase in their representation capacity, limiting the deployment capabilities and hence, the commercial feasibility of these networks. This limitation warrants the need for efficient compression techniques that can shrink the networks in size while ensuring that the drop in performance is minimal. In this paper, we restrict our focus to three widely-used compression techniques: Knowledge Distillation, Network Pruning and Quantization.

Knowledge Distillation (Hinton, Vinyals, and Dean 2015) is a compression paradigm that leverages the capability of large neural networks (called teacher networks) to transfer knowledge to smaller networks (called student networks). While large models (such as very deep neural networks or ensembles of many models) have higher knowledge capacity than small models, this capacity might not be fully utilized. It can be computationally just as expensive to evaluate a model even if it utilizes little of its knowledge capacity. Knowledge distillation aims to transfers knowledge from a large model to a smaller model without loss of validity. Several advancements have been witnessed in the development of richer knowledge distillation algorithms, attempting to reduce the difference in test accuracies of the teacher and the student. These algorithms are model-agnostic and hence can be used for a wide variety of network architectures.

While knowledge distillation attempts to train an equally-competent smaller network, network pruning (LeCun et al. 1990) attempts to reduce the size of the existing network by removing unimportant weights. Different pruning techniques differ in the choice of weights to eliminate and the methods used to do the same. Pruning can help in reducing the size of the network up to 90% with minimal loss in performance. Some approaches have also been empirically shown to result in faster training of the pruned network along with a higher test accuracy (Frankle and Carbin 2018).

Quantization is another way to compress neural networks by reducing the number of bits used to store the weights. As the weights of a network are usually stored as 32-bit floating values (FP32), reducing the precision to 8-bit integer values (INT8) will reduce the size of the network by 4 times. Several approaches have been developed to quantize networks with minimal loss in performance.

These compression techniques have become extremely popular in recent years and are actively being researched. New algorithms proposed in research papers can be difficult to understand and implement, especially for potential users in a non-academic setting, thereby limiting their commercial usage. To the best of our knowledge, there does not exist an umbrella framework containing implementations of state-of-the-art algorithms in Knowledge Distillation, Pruning and Quantization. In this paper, we present KD-Lib, a comprehensive PyTorch based library for model compression. KD-Lib aims to bridge the gap between research and widespread use of model compression techniques. We envision that such a framework would be helpful to researchers as well, providing them a tool to build upon existing algorithms and helping them in going from idea to implementation faster.

Related work

We compare KD-Lib with several openly available frameworks and libraries. In our comparison, we do not include

[*]Equal contribution

Library	Knowledge Distillation	Pruning	Quantization
KD-Lib (Ours)	Present	Present	Present
Distiller(Zmora et al. 2019)	Present (only 1 algorithm)	Present	Present
AIMET[3]	-	Present	Present
AquVitae[1]	Present	-	-
Distiller[1]	Present	-	-

Table 1: Comparision of various libraries with KD-Lib

libraries that support less than two algorithms.

Distiller (Zmora et al. 2019) is the most extensive framework we found, but it primarily focuses on quantization and pruning with only one knowledge distillation algorithm (Hinton, Vinyals, and Dean 2015). AquVitae[1] contains 4 distillation methods but no quantization and pruning algorithms. Similarly Distiller[2] has 11 knowledge distillation techniques but lacks pruning and quantization methods. AIMET[3] focuses mainly on quantization and some other relatively less popular model compression techniques such as tensor decomposition. In our survey, we found no library containing algorithms pertaining to all 3 of the popular compression paradigms - knowledge distillation, pruning and quantization. Table 1 shows concise comparison with different frameworks.

Features and Algorithms

KD-Lib houses several algorithms proposed in recent years for model compression. The following features have driven the design choices for the library:

- The main aim of KD-Lib is to make model compression algorithms accessible to a wide range of users, and hence the work is fully open-source.
- The library should act as a catalyst for further research in these fields. It should also be extendable to newer algorithms and other model compression fields. Hence, it is designed to be modular, allowing flexible modifications to essential components that can lead to novel algorithms or better extensions to existing algorithms.
- The interface should be easy to use. Hence, the core functionalities (distillation/pruning/quantization) are accessible in a few lines of code.
- As tuning the hyperparameters is essential for optimum performance, KD-Lib provides support for hyperparameter tuning via Optuna. Monitoring and logging support is also provided through Tensorboard.

A brief description of the implemented algorithms is as follows:

- **Knowledge Distillation :** The algorithms have been divided into two major task-types: Vision and Text. The Vision module currently supports 13 algorithms while the Text module supports distillation from BERT to LSTM-based networks(Tang et al. 2019).
- **Pruning :** The library currently supports pruning based on the Lottery ticket Hypothesis (Frankle and Carbin 2018).
- **Quantization :** Static Quantization, Dynamic Quantization and Quantization Aware Training (QAT) (Jacob et al. 2018) are currently supported by KD-Lib.

Code Structure

The structure of the library has been designed for efficient use with the following major principles kept in mind:

- The core function of an algorithm can be executed in one line of code. Hence, the classes contain a dedicated method for distillation/pruning/quantization.
- Each module allows extension to newer features and easy modifications. Hence, fluid components of algorithms (loss functions in distillation, for example) can be easily customized.
- Necessary statistics are available wherever needed. Hence, methods dedicated to these are also present ($get\_pruning\_statics$, for example).

```
Distiller
├── train_student
├── train_teacher
├── evaluate
└── calculate_kd_loss
```

Figure 1: Structure of a Distiller.

Knowledge Distillation algorithms can be accessed as Distiller objects (Figure 1), with at least the mentioned methods. The *train_student* method distills knowledge from a teacher network to a student network, where the teacher network could optionally be trained using the *train_teacher* method. The *evaluate* method can be invoked to test the performance of the student network. The *calculate_kd_loss* method can overridden to provide a custom loss function for distillation. This can also be leveraged by researchers to test novel Knowledge Distillation loss functions.

Pruning algorithms have been implemented as Pruner objects (Figure 2). Each Pruner object can access the *prune* method for pruning the network. Additionally, the

[1]https://github.com/aquvitae/aquvitae

[2]https://github.com/karanchahal/distiller

[3]https://github.com/quic/aimet

```
Pruner
├── prune
└── get_pruning_statistics
```

Figure 2: Structure of a Pruner.

```
Quantizer
├── quantize
├── get_performance_statistics
└── get_model_sizes
```

Figure 3: Structure of a Quantizer.

get_pruning_statistics method can be used to obtain information about the weights of the network after pruning (percentage of network pruned, for example).

Quantization algorithms can be accessed via Quantizer objects (Figure 3). The *quantize* method can be used for quantization (with differing implementations for different algorithms). Additionally, the *get_model_sizes* method can be used to compare sizes of the model before and after quantization and the *get_performance_statistics* method can be used to compare test-times and error metrics for the two networks.

The documentation for the library[4] has the description of all classes and selected tutorials with example code snippets.

Benchmarks

We summarize benchmark results on some of the algorithms implemented in KD-Lib in Tables 2, 3 and 4.

Algorithm	Accuracy
None	0.57
DML (Zhang et al. 2018)	0.62
Self Training (Yun et al. 2020)	0.61
Messy Collab (Arani, Sarfraz, and Zonooz 2019)	0.60
Noisy Teacher (Sau and Balasubramanian 2016)	0.59
TAKD (Mirzadeh et al. 2019)	0.59
RCO (Jin et al. 2019)	0.58
Probability Shift (Wen, Lai, and Qian 2019)	0.58

Table 2: The accuracies of networks trained by some of various knowledge distillation algorithms KD-Lib packages on the CIFAR10 dataset. All models were trained with the same hyperparameter set to ensure a fair comparison. We consider ResNet34 as the teacher network (with an accuracy of 0.63) and report accuracies for the student network (ResNet18). *None* refers to a ResNet18 model trained from scratch without any model compression algorithm. The compression ratio for all of the knowledge distillation algorithms is 50.7%

Conclusion and Future Work

In this paper, we present KD-Lib, an easy-to-use PyTorch-based library for Knowledge Distillation, Pruning and Quantization. KD-Lib is designed to facilitate the adoption of current model compression techniques and act as a catalyst for further research in this direction. We plan on actively maintaining the library and also expanding it to include more algorithms and desirable features (distributed training, for example) in the future. We further plan on extending this library to other domains relevant to the research community including but not limited to explainability and interpretability in knowledge distillation.

[4]https://kd-lib.readthedocs.io/

Pruning Epoch	% Model Pruned	Accuracy
1	0.0	0.9878
2	0.10	0.9891
3	0.19	0.9890

Table 3: Pruning percentage and accuracy of ResNet18 model on MNIST using Lottery Ticket Pruning (Frankle and Carbin 2018). Each pruning epoch consists of 5 training epochs. 'Model pruned' is the percentage of model pruned and 'Accuracy' is the corresponding accuracy at the end of the epoch.

Algorithm	% Size Change	BA	NA
Static	-0.75	0.72	0.70
QAT	-0.75	0.72	0.71
Dynamic	-0.19	0.70	0.70

Table 4: Comparison of various quantization algorithms. 'BA' (Base Accuracy) is the accuracy of the model before quantization, and 'NA' (New Accuracy) is the accuracy of the model after quantization. '% Size change' refers to the change in size after quantization. In Static Quantization and QAT, ResNet18 is tested on the CIFAR10 dataset. For Dynamic Quantization, LSTM is tested on IMDB dataset.

References

Arani, E.; Sarfraz, F.; and Zonooz, B. 2019. Improving Generalization and Robustness with Noisy Collaboration in Knowledge Distillation.

Frankle, J.; and Carbin, M. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Jin, X.; Peng, B.; Wu, Y.; Liu, Y.; Liu, J.; Liang, D.; Yan, J.; and Hu, X. 2019. Knowledge Distillation via Route Constrained Optimization.

LeCun, Y.; Denker, J. S.; ; and Solla, S. A. 1990. Optimal brain damage. *In Advances in Neural Information Processing Systems* .

Mirzadeh, S.-I.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; and Ghasemzadeh, H. 2019. Improved Knowledge Distillation via Teacher Assistant.

Sau, B. B.; and Balasubramanian, V. N. 2016. Deep Model Compression: Distilling Knowledge from Noisy Teachers.

Tang, R.; Lu, Y.; Liu, L.; Mou, L.; Vechtomova, O.; and Lin, J. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *CoRR* abs/1903.12136.

Wen, T.; Lai, S.; and Qian, X. 2019. Preparing Lessons: Improve Knowledge Distillation with Better Supervision.

Yun, S.; Park, J.; Lee, K.; and Shin, J. 2020. Regularizing Class-Wise Predictions via Self-Knowledge Distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2018. Deep Mutual Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zmora, N.; Jacob, G.; Zlotnik, L.; Elharar, B.; and Novik, G. 2019. Neural Network Distiller: A Python Package For DNN Compression Research URL https://arxiv.org/abs/1910.12232.