Literature review
Distillation techniques for deep neural networks

Moritz Sprenger, 950222-6799, DIT892, gussprmo@student.gu.se

September 25, 2023

Supervisors
Yinan Yu, academic supervisor, yinan@chalmers.se
Charlotte von Numers, industry supervisor, charlotte.vonnumers@astrazeneca.com
Luis Arevalo, industry supervisor, imarevost@gmail.com

Introduction

In recent years deep neural networks (DNNs) have become the state-of-the-art in many application domains such as computer vision, natural language processing or game playing using reinforcement learning. These advances can partly be attributed to the availability of more specialized computing power enabling larger models with millions or, especially in the case of large language models, billions of parameters. Training or using these models therefore requires the usage of cloud services or large amounts of storage and computing resources.

With the widespread use of mobile and edge devices in every day life and commercial contexts it is natural to leverage the abilities of DNNs on these devices for a multitude of use cases such as language translation or image classification. With some applications having strict latency, privacy or connectivity requirements, preventing the usage of cloud-based services, it becomes infeasible to use such large models because of memory, computational and energy consumption limitations. This makes it critical to develop methods that are aimed at reducing size and/or inference time of models while keeping a similar performance.

This work gives an overview over the three most used compression techniques for DNNs: pruning, quantization and knowledge distillation. Due to the limited scope and length of this work, the presented approaches only represent a small fraction of the published literature and should mainly serve as an introduction to the field, stressing the main concepts for different distillation techniques.

Distillation techniques

This work refers to distillation techniques as methods employed mainly to compress and/or speed-up deep neural networks while having a maximally minimal negative or even a positive effect on network performance. Three categories of widely used distillation techniques are identified. Quantization, which "compresses the original network by reducing the number of bits required to represent each weight" [9]. Pruning identifies and prunes non-informative weights, which do not have significant effect on model performance in a trained neural network [11]. Knowledge distillation makes use of a much smaller student network that learns to mimic the generalization capabilities of the original large teacher model [9].

Quantization

The idea of quantization is to compress the network by reducing the number of bits, and thereby the number of different values, used to represent network parameters and activations. Commonly this is done by using a quantization function that maps real valued inputs in floating points, such as weights and biases in a neural network, to a finite set of values.

Normally weights as well as biases and their computations are performed with 32-bit floating point precision. Reducing this precision to half-precision (16 bit) or even integer representations significantly reduces the memory requirements and can speed up arithmetic computation on hardware that supports low-precision arithmetic [22] [34].

A popular choice for a quantization function is given in equation 1.

$$Q(r) = Int(r/S) - Z \tag{1}$$

Here Q is the quantization function, r is a real valued input, S is a scaling factor, Z an integer zero point and Int a rounding operation to the nearest integer or clipping [22]. This quantization function

leads to uniformly spaced output values and is referred to as uniform quantization. The choice of S in the quantization function is essential, since it controls the partitioning of the inputs r.

$$S = \frac{\beta - \alpha}{2^b - 1} \tag{2}$$

Equation 2 shows a choice for S, where $[\alpha, \beta]$ describes the clipping range of inputs and b is the resulting quantization bit width. To define the scaling factor it is therefore necessary to decide on a clipping range, which is referred to as calibration.

Depending on the choice of α, β the quantization can be symmetric, when $-\alpha = \beta$, or asymmetric, which leads to tighter bounds but a non-zero Z. A popular choice for the clipping range are the min/max values for the inputs. However, this is sensitive to outliers and alternative approaches like using quantiles [52], minimizing the Kullback-Leibner divergence between quantized and real values [53] or using the Mean Squared Error (MSE) between quantized and real values [69], [72] have been proposed. Additionally some works like LQNets [87] or LSQ+ [4] learn the clipping range jointly with the weights during the training of a DNN.

Since DNNs consist of multiple layers that can differ heavily in their weight distribution, it can make sense to adapt the ranges for quantization to individual layers or even more precisely to structures inside these layers. The simplest granularity would be to use the same range for the whole network. This often leads to significant losses in performance since the range of weight values can vary a lot between layers. The next granularity is layerwise quantization, which considers the statistics of weights for each layer to determine a range . Although more granular than the previous approach, this also results in sub-optimal performance especially for convolutional networks (CNNs) [42], since ranges between convolutional filters can also vary significantly [22]. More granular approaches as in [67] apply groupwise quantization, where multiple filters are grouped together, or use channelwise quantization to calculate a range for each individual filter as in [87], [36].

In addition to the introduced uniform quantization, non-uniform quantization can achieve better results by focusing on more important value ranges and thereby capturing the value distribution more closely. Rule-based methods often use a logarithmic distribution to increase quantization steps exponentially [56]. Other methods train the quantization steps similar to the quantization range jointly with the network [83] [87]. Clustering can also be used to quantize models by replacing model parameters with the centroid of the corresponding cluster, which can be binary encoded [10]. Wu et al.[80] use k-means clustering to determine quantization steps and Choi et al. [10] extends these approaches with Hessian-weighted k-means clustering to minimize the loss in performance.

Independent from the introduced concepts, quantization can be differentiated by the time it is used in the training process.
Quantization-Aware Training (QAT) (re-)trains the model with quantized parameters. Here the forward and backward pass are executed in floating point but the model parameters are quantized after each gradient update [23]. Due to the non-differentiable nature of common quantization functions, their gradient is often approximated with the identity function [17]. This approach is referred to as Straight Through Estimator (STE) [3]. Non-STE methods try to use regularization terms to avoid the usage of non-differentiable quantization functions [23].
A faster family of methods that don't require (re-)training but often have greater performance loss are Post-Training Quantization methods (PTQ). These methods only use a fraction of the training or test data to calibrate the quantization. To mitigate the performance loss, [2] analyze biases in quantized parameters and perform bias correction. Other approaches quantize model weights one-by-one, while adjusting the remaining weights [21]. Other works propose adaptive rounding methods that lead to better results than the round-to-nearest approach such as AdaRound [58] and the extension AdaQuant [37].

When no training or test data is available, due to the size or privacy of datasets, so called Zero-shot Quantization methods are used. These methods often use synthetic or other openly available data that theoretically resembles the training data distribution to calibrate the quantization methods [23].

To balance the advantages of using lower precision quantization with performance degradation, some works employ mixed-precision quantization, which quantizes layers with different bit precisions. Finding the precision configuration for different layers has been approached with reinforcement learning [77], analyzing the sensitivity of a layer to quantization [16] and Integer linear programming [85].
Extreme Quantization refers to quantization that lowers precision under 8 bit, with binary (1 bit) and ternary (2 bit) quantization as the extremes [94], [47].

Recently, advances to quantization methods have been facilitated by the availability and success of Large Language Models (LLMs). These models often have more than a billion parameters and distillation techniques have to be employed to make them usable on single devices such as PCs or laptops. Due to the size of LLMs and training data sets, QAT methods are infeasible and PTQ methods need to be used. Recent works applied to these networks are [59], [15], [84] and [20], which extends [21] to fast Extreme Quantization of LLMs.

Pruning

DNNs are usually overparameterized and the information stored in some model parameters is redundant [14]. Pruning aims to make use of this observation by removing parameters that have a small impact on the performance, and thereby decreasing the size of the model while minimizing the impact on performance [48].

An influential work for the research in pruning is the Lottery Ticket Hypothesis [18], which states that a trained DNN contains a subnetwork, that is capable of achieving the same performance using no more training than the full network, because it was initialized with suitable weights. Following works contest the claim of the importance of the initialization [50], [19] but the Lottery Ticket Hypothesis is an important contribution for pruning-based neural architecture search and motivates the effectiveness of pruning.

In general pruning methods can be differentiated by the criterion used to decide which parameters or structures to remove, the granularity of pruning, and the way finetuning is performed during the pruning process [46].

Most commonly pruning methods are differentiated as structured or unstructured pruning methods. In unstructured pruning methods individual parameters are pruned leading to sparse networks with lower parameter counts. However, the lower parameter count doesn't necessarily result in a speedup in training or inference times because specialized hardware is necessary to take advantage of the sparsity [73].
Early works used second order derivatives as the decision criterion to prune weights [43], [29]. These methods didn't require any finetuning but calculating the derivatives was computationally expensive. Other works pruned based on the value similarity of weights [70]. Other authors followed a more iterative approach, where weights with a L2 norm below a certain threshold are masked out and the model is retrained. This procedure is repeated until model performance degrades significantly [28]. Extensions of this work enable the recovery of pruned weights that can become important again given the removal of other weights [27].
Structured pruning prunes entire network structures such as layers and weights or more specialized structures such as channels/filters in convolutional or heads in transformer [74] architectures. The pruning of these structures leads directly to a decrease in model size and speedups, which can be exploited directly without requiring dedicated hard- and software [5]. Authors use similar pruning criteria as in unstructured pruning for their structured approaches. [41] uses the L2 norm to group

and prune output channels below a certain threshold in CNNs. [79] extends this approach to multiple structure levels such as filter count, filter shape and layer width. The scaling factor of the Batch Normalization layer with slight modifications is also used as the pruning criteria [91]. Other approaches identify the redundancy in filters through the geometric mean and prune those filters without significant information loss [31]. With structured pruning being inherently tied to the network architecture, there is a significance body of work dedicated to different network structures such as CNNs [32].

Next to the structural nature of the pruning, pruning strategies also differ in the necessity of data in the pruning process. Some of the presented works are data free, do not need any retraining and rely on the static parameter properties [70, 43] , whereas other works are dependent on training data [28, 79].

Similarly to quantization, recent advances in pruning have been driven by transformer architectures either for computer vision [93, 86] or LLMs [78, 51, 71].

Knowledge distillation

Knowledge distillation (KD) denotes a method where a smaller student model is supervised by a larger teacher model and was popularized by [33]. The idea behind KD is that the student model mimics the teacher model and reaches similar or even greater performance with less capacity [24]. Figure 1 shows the general setup for knowledge distillation methods. Both models are trained with the same data and the student network is supported by knowledge transferred from the teacher model.

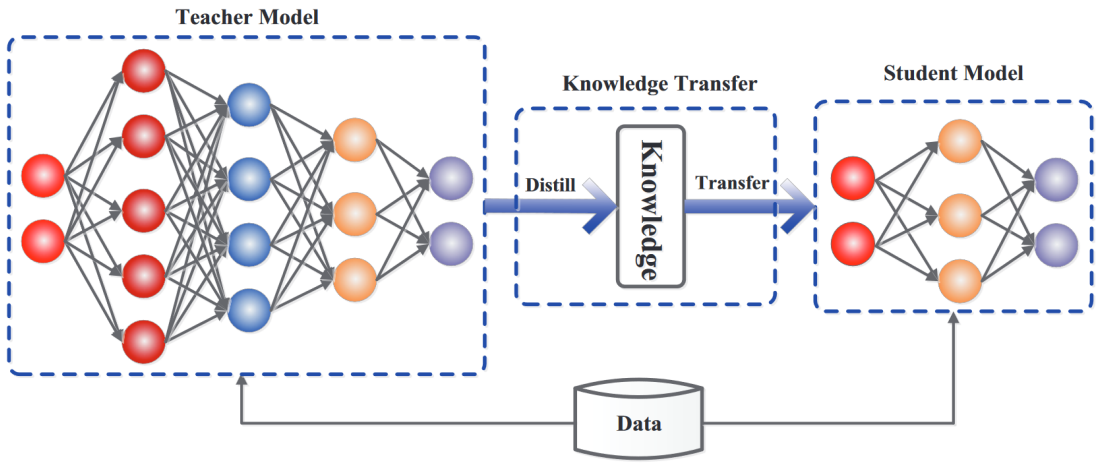

Figure 1: Basic framework for knowledge distillation [24]

Knowledge distillation system can generally be differentiated by their three main components: the type of knowledge, the distillation algorithm and the teacher-student architecture [24].

Three main categories of knowledge are used to describe KD systems: response-based knowledge, feature-based knowledge and relation-based knowledge [24].
Response-based knowledge refers to the outputs of the last layer in a DNN. This is the simplest type of knowledge transferred in KD systems [24]. The learning objective of the student network is to mimic the outputs of the teacher network. With labeled data available, this objective can be combined with the loss regarding the ground truth labels as a weighted sum in the loss function [33]. Using this type of knowledge is applicable in many domains. The most popular use is for (image) classification tasks, where the softmax outputs are used [33, 1]. Further tasks are object detection [7], pose estimation

[88] or LLMs [26], which extend the distillation loss with domain specific outputs such as offsets or heatmaps.
Since DNNs learn through building more and more useful representations in their intermediate layers, feature-based knowledge used for distillation refers to outputs based on the intermediate layers of networks. The distillation loss is then computed by matching the activations or proxies in the teacher model with those in the student model [24]. Some work match the feature activations directly [6, 63], while others use proxies such as activation statistics [61] or other derived information [44]. For feature-based knowledge two main questions have to be explored, which intermediate layers to choose for distillation, since the number of layers in the teacher network might be larger, and how to match feature representations, when spatial dimensions differ between teacher and student [24]. For the former, some works use attention to guide the choice of layers [6, 63]. The latter can for example be tackled by the usage of singular value decomposition [44].
While the discussed knowledge types consider singular specific layers of the teacher network, relation-based knowledge tries to capture information about the relationships between layers or data samples to guide the student network [24]. Works that focus on layer relationships use correlations between pairs of feature maps as knowledge [44] or try to model and match the information flow between teacher and student model [62]. The relations between data samples/instances is also used for knowledge distillation. Methods considering instance relations consider the spatial relations between multiple instances and try to match these structural relations between student and teacher models [60, 8].

Distillation algorithms can be grouped into three main categories: offline distillation, online distillation and self-distillation [24].
Most of the presented works use offline distillation, where the teacher model is trained first or assumed to be already trained. The student model is then trained in a second step under the guidance of the teacher model, which makes training more efficient [24].
In online distillation algorithms both teacher and student model are trained jointly in an end-to-end fashion, this can further improve the student performance and be necessary when no pretrained teacher model exists [24]. In deep mutual learning, multiple networks train jointly and each model functions as teacher and as a student model during the training process by transferring their response-based knowledge between them [90]. This approach is extended by [38], who fuses intermediate features used by a classifier that distributes its' knowledge back to the student networks.
In self-distillation the knowledge transfer is not between different networks but inside the same network. Some works distill knowledge from one section of a network to another, for example knowledge deeper layers of the network to shallower ones [89] or between different output layers in multi-exit architectures [64]. In other works the knowledge transfer happens between different times in the training process, where knowledge from earlier epochs is transferred to later epochs [82].

Next to the type of transferred knowledge and the distillation algorithm, the architecture of the teacher and student model also plays a role in knowledge distillation.
Due to the compression intention behind the KD approach most student networks are less complex than the teacher networks. This can either be achieved by student models being a simplified version of the teachers with less layers and/or channels [76, 45], a quantized network with the same structure [68] or other size optimized network architectures [35, 25]. In online distillation settings, teacher models can have the same architecture as the student or be an ensemble of them [90]. Due to the capacity gap between larger teacher and smaller student networks, the student networks might not be able to replicate the performance of teachers [54]. Some works tackle this problem by reducing the student capacity in a controlled manner [25] or by iteratively introducing smaller and smaller teacher assistants that are first acting as the student and then as the teacher for the next teacher assistant until the desired size of a final student model is reached.

Combinations of techniques

Some works do not solely use one of the presented techniques but combine multiple approaches to achieve more compression or better performance. It seems the most popular combination approach is mixing quantization with knowledge distillation, where the student model is either a quantized version of the teacher network [55, 81, 68] or a quatized network with a different architecture [65]. Pruning and quantization is combined by [21], who present a unified framework to prune and quantize networks.

Performance

There are significant differences how performance is reported in the literature. Although, some datasets like CIFAR-10 [39] or ImageNet [13], or architectures like ResNet [30] or AlexNet [40] are more commonly used than others, there seems to be no agreed upon standard for performance analysis in this field. This observation extends to performance measures. Some authors report compression and speedup rates given in equations 3, 4 respectively, while others only report accuracy differences or technique specific measures like bit-width for quantization. In the equations, M and M^* denote the original and compressed model, while a,a^* denote the number of parameters and s,s^* the running times. For running times either the average training time per epoch is used or the average inference time [9].

$$\alpha(M, M^*) = \frac{a}{a^*} \tag{3}$$

$$\delta(M, M^*) = \frac{s}{s^*} \tag{4}$$

The limited scope of this work doesn't allow for a comprehensive comparative performance analysis of the different methods, this is exacerbated by the fact that performance comparisons are a challenge in this field, which will be further discussed in section . Interested readers are referred to more comprehensive survey articles like [11, 66, 24].

Challenges

The biggest challenge for distillation techniques is the lack of comprehensive comparable performance analysis. There are some model architectures and data sets that are more commonly used than other but the comparisons often are limited and rely on specific surveys. Furthermore different authors use different measures to compare their methods, sometimes only reporting task specific measures such as accuracy [5]. Although some works make an effort to introduce tools that helps this endeavor [5], the field lacks benchmarks or challenges, which are well established for task specific performance analysis [13, 75, 12]. An exception are LLM related benchmarks, which consider efficiency factors [49, 92]. A contributing factor is that achieved compression and speed up is intrinsically linked to the used hardware and sometimes requires specialized hardware to leverage improvements made by proposed methods, making finding a common standard challenging.
Another challenge is a limited amount of theoretical foundations method improvements are based on. Although there is a small body of work dedicated to furthering the theoretical understanding [21, 57], most advances are based on empirical evidence.

Appendix

References

[1] Lei Jimmy Ba and Rich Caruana. Do Deep Nets Really Need to be Deep?

[2] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation.

[4] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. LSQ+: Improving low-bit quantization through learnable offsets and better initialization.

[5] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the State of Neural Network Pruning?

[6] Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Yan Feng, and Chun Chen. Cross-Layer Distillation with Semantic Calibration.

[7] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 742–751. Curran Associates Inc.

[8] Hanting Chen, Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao. Learning Student Networks via Feature Embedding.

[9] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A Survey of Model Compression and Acceleration for Deep Neural Networks.

[10] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the Limit of Network Quantization.

[11] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. 53(7):5113–5155.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[14] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and prefix=de useprefix=true family=Freitas, given=Nando. Predicting Parameters in Deep Learning.

[15] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale.

[16] Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks.

[17] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. Training with Quantization Noise for Extreme Model Compression.

[18] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.

[19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Stabilizing the Lottery Ticket Hypothesis.

[20] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers.

[21] Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning.

[22] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference.

[23] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. 129(6):1789–1819.

[24] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. 129(6):1789–1819.

[25] Jindong Gu and Volker Tresp. Search for Better Students to Learn Distilled Knowledge.

[26] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge Distillation of Large Language Models.

[27] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic Network Surgery for Efficient DNNs.

[28] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both Weights and Connections for Efficient Neural Networks.

[29] B. Hassibi and D. Stork. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition.

[31] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration.

[32] Yang He and Lingao Xiao. Structured Pruning for Deep Convolutional Neural Networks: A survey.

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network.

[34] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14.

[35] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[36] Qijing Huang, Dequan Wang, Zhen Dong, Yizhao Gao, Yaohui Cai, Tian Li, Bichen Wu, Kurt Keutzer, and John Wawrzynek. CoDeNet: Efficient Deployment of Input-Adaptive Object Detection on Embedded FPGAs. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '21, pages 206–216. Association for Computing Machinery.

[37] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving Post Training Neural Quantization: Layer-wise Calibration and Integer Programming.

[38] Jangho Kim, Minsung Hyun, Inseop Chung, and Nojun Kwak. Feature Fusion for Online Mutual Knowledge Distillation.

[39] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105. Curran Associates Inc.

[41] Vadim Lebedev and Victor Lempitsky. Fast ConvNets Using Group-wise Brain Damage.

[42] Yann Lecun and Yoshua Bengio. Convolutional networks for images, speech, and time-series. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

[43] Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

[44] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised Knowledge Distillation Using Singular Value Decomposition.

[45] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few Sample Knowledge Distillation for Efficient Network Compression.

[46] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. 461:370–403.

[47] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated Binary Neural Network.

[48] Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. Pruning Algorithms to Accelerate Convolutional Neural Networks for Edge Applications: A Survey.

[49] Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. Towards Efficient NLP: A Standard Evaluation and A Strong Baseline.

[50] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the Value of Network Pruning.

[51] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the Structural Pruning of Large Language Models.

[52] Jeffrey L. McKinstry, Steven K. Esser, Rathinakumar Appuswamy, Deepika Bablani, John V. Arthur, Izzet B. Yildiz, and Dharmendra S. Modha. Discovering Low-Precision Networks Close to Full-Precision Networks for Efficient Embedded Inference.

[53] Szymon Migacz. Nvidia 8-bit inference with tensorrt.

[54] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved Knowledge Distillation via Teacher Assistant.

[55] Asit Mishra and Debbie Marr. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy.

[56] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional Neural Networks using Logarithmic Data Representation.

[57] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-Distillation Amplifies Regularization in Hilbert Space.

[58] Markus Nagel, Rana Ali Amjad, prefix=van useprefix=true family=Baalen, given=Mart, Christos Louizos, and Tijmen Blankevoort. Up or Down? Adaptive Rounding for Post-Training Quantization.

[59] Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. LUT-GEMM: Quantized Matrix Multiplication based on LUTs for Efficient Inference in Large-Scale Generative Language Models.

[60] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational Knowledge Distillation.

[61] Nikolaos Passalis and Anastasios Tefas. Learning Deep Representations with Probabilistic Knowledge Transfer.

[62] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Heterogeneous Knowledge Distillation using Information Flow Modeling.

[63] Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. ALP-KD: Attention-Based Layer Projection for Knowledge Distillation.

[64] Mary Phuong and Christoph Lampert. Distillation-Based Training for Multi-Exit Architectures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1355–1364.

[65] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization.

[66] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification.

[67] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT.

[68] Sungho Shin, Yoonho Boo, and Wonyong Sung. Knowledge distillation for optimization of quantized deep neural networks.

[69] Sungho Shin, Kyuyeon Hwang, and Wonyong Sung. Fixed-Point Performance Analysis of Recurrent Neural Networks. 32(4):158–158.

[70] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for Deep Neural Networks.

[71] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models.

[72] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of Deep Neural Networks under Quantization.

[73] Sunil Vadera and Salem Ameen. Methods for Pruning Deep Neural Networks.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need.

[75] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems.

[76] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2769–2775. AAAI Press.

[77] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-Aware Automated Quantization with Mixed Precision.

[78] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured Pruning of Large Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

[79] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning Structured Sparsity in Deep Neural Networks.

[80] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized Convolutional Neural Networks for Mobile Devices.

[81] Xiaoxia Wu, Zhewei Yao, Minjia Zhang, Conglong Li, and Yuxiong He. Extreme Compression for Pre-trained Transformers Made Simple and Efficient.

[82] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L. Yuille. Snapshot Distillation: Teacher-Student Optimization in One Generation.

[83] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xiansheng Hua. Quantization Networks.

[84] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers.

[85] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W. Mahoney, and Kurt Keutzer. HAWQV3: Dyadic Neural Network Quantization.

[86] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & Depth Pruning for Vision Transformers. 36(3):3143–3151.

[87] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks.

[88] Feng Zhang, Xiatian Zhu, and Mao Ye. Fast Human Pose Estimation.

[89] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation.

[90] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep Mutual Learning.

[91] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational Convolutional Neural Network Pruning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2775–2784.

[92] Xiyou Zhou, Zhiyu Chen, Xiaoyong Jin, and William Yang Wang. HULK: An Energy Efficiency Benchmark Platform for Responsible Natural Language Processing.

[93] Mingjian Zhu, Yehui Tang, and Kai Han. Vision Transformer Pruning.

[94] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation.