Title: Model compression

URL: https://en.wikipedia.org/wiki/Model_compression

PageID: 78148111

Categories: Category:Deep learning, Category:Machine learning

---

Model compression is a machine learning technique for reducing the size of trained models. Large models can achieve high accuracy, but often at the cost of significant resource requirements. Compression techniques aim to compress models without significant performance reduction. Smaller models require less storage space, and consume less memory and compute during inference.

Compressed models enable deployment on resource-constrained devices such as smartphones , embedded systems , edge computing devices, and consumer electronics computers. Efficient inference is also valuable for large corporations that serve large model inference over an API, allowing them to reduce computational costs and improve response times for users.

Model compression is not to be confused with knowledge distillation , in which a separate , smaller "student" model is trained to imitate the input-output behavior of a larger "teacher" model.

Techniques

Several techniques are employed for model compression.

Pruning

Pruning sparsifies a large model by setting some parameters to exactly zero. This effectively reduces the number of parameters. This allows the use of sparse matrix operations , which are faster than dense matrix operations.

Pruning criteria can be based on magnitudes of parameters, the statistical pattern of neural activations , Hessian values , etc. [1] [2]

Quantization

Quantization reduces the numerical precision of weights and activations. For example, instead of storing weights as 32-bit floating-point numbers, they can be represented using 8-bit integers. Low-precision parameters take up less space, and takes less compute to perform arithmetic with.

It is also possible to quantize some parameters more aggressively than others, so for example, a less important parameter can have 8-bit precision while another, more important parameter, can have 16-bit precision. Inference with such models requires mixed-precision arithmetic . [3] [4]

Quantized models can also be used during training (rather than after training). PyTorch implements automatic mixed-precision (AMP), which performs autocasting, gradient scaling, and loss scaling. [5] [6]

Low-rank factorization

Weight matrices can be approximated by low- rank matrices. Let W {\displaystyle W} be a weight matrix of shape m × n {\displaystyle m\times n} . A low-rank approximation is W ≈ U V T {\displaystyle W\approx UV^{T}} , where U {\displaystyle U} and V {\displaystyle V} are matrices of shapes m × k , n × k {\displaystyle m\times k,n\times k} . When k {\displaystyle k} is small, this both reduces the number of parameters needed to represent W {\displaystyle W} approximately, and accelerates matrix multiplication by W {\displaystyle W} .

Low-rank approximations can be found by singular value decomposition (SVD). The choice of rank for each weight matrix is a hyperparameter, and jointly optimized as a mixed discrete-continuous optimization problem. [7] The rank of weight matrices may also be pruned after training, taking into account the effect of activation functions like ReLU on the implicit rank of the weight matrices. [8]

Training

Model compression may be decoupled from training, that is, a model is first trained without regard for how it might be compressed, then it is compressed. However, it may also be combined with training.

The "train big, then compress" method trains a large model for a small number of training steps (less than it would be if it were trained to convergence), then heavily compress the model. It is found that at the same compute budget, this method results in a better model than lightly compressed, small models. [9]

In Deep Compression, [10] the compression has three steps.

First loop (pruning): prune all weights lower than a threshold, then finetune the network, then prune again, etc.

Second loop (quantization): cluster weights, then enforce weight sharing among all weights in each cluster, then finetune the network, then cluster again, etc.

Third step: Use Huffman coding to losslessly compress the model.

The SqueezeNet paper reported that Deep Compression achieved a compression ratio of 35 on AlexNet, and a ratio of ~10 on SqueezeNets. [11]

References

Review papers Li, Zhuo; Li, Hengyi; Meng, Lin (March 12, 2023). "Model Compression for Deep Neural Networks: A Survey" . Computers . 12 (3). MDPI AG: 60. doi : 10.3390/computers12030060 . ISSN 2073-431X . Deng, By Lei; Li, Guoqi; Han, Song; Shi, Luping; Xie, Yuan (March 20, 2020). "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey". Proceedings of the IEEE . 108 (4): 485– 532. doi : 10.1109/JPROC.2020.2976475 . Cheng, Yu; Wang, Duo; Zhou, Pan; Zhang, Tao (October 23, 2017). "A Survey of Model Compression and Acceleration for Deep Neural Networks". arXiv : 1710.09282 [cs.LG]. Choudhary, Tejalal; Mishra, Vipul; Goswami, Anurag; Sarangapani, Jagannathan (February 8, 2020). "A comprehensive survey on model compression and acceleration". Artificial Intelligence Review . 53 (7). Springer Science and Business Media LLC: 5113– 5155. doi : 10.1007/s10462-020-09816-7 . ISSN 0269-2821 .

Li, Zhuo; Li, Hengyi; Meng, Lin (March 12, 2023). "Model Compression for Deep Neural Networks: A Survey" . Computers . 12 (3). MDPI AG: 60. doi : 10.3390/computers12030060 . ISSN 2073-431X .

Deng, By Lei; Li, Guoqi; Han, Song; Shi, Luping; Xie, Yuan (March 20, 2020). "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey". Proceedings of the IEEE . 108 (4): 485– 532. doi : 10.1109/JPROC.2020.2976475 .

Cheng, Yu; Wang, Duo; Zhou, Pan; Zhang, Tao (October 23, 2017). "A Survey of Model Compression and Acceleration for Deep Neural Networks". arXiv : 1710.09282 [cs.LG].

Choudhary, Tejalal; Mishra, Vipul; Goswami, Anurag; Sarangapani, Jagannathan (February 8, 2020). "A comprehensive survey on model compression and acceleration". Artificial Intelligence Review . 53 (7). Springer Science and Business Media LLC: 5113– 5155. doi : 10.1007/s10462-020-09816-7 . ISSN 0269-2821 .

v

t

e

Differentiable programming

Information geometry

Statistical manifold

Automatic differentiation

Neuromorphic computing

Pattern recognition

Ricci calculus

Computational learning theory

Inductive bias

IPU

TPU

VPU

Memristor

SpiNNaker

TensorFlow

PyTorch

Keras

scikit-learn

Theano

JAX

Flux.jl

MindSpore

Portals Computer programming Technology

Computer programming

Technology