ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

An Empirical Study of Iterative Knowledge Distillation for Neural Network Compression

Sharan Yalburgi[1], Tirtharaj Dash[1], Ramya Hebbalaguppe[2], Srinidhi Hegde[2], Ashwin Srinivasan[1]

1- BITS Pilani, K.K. Birla Goa Campus, India

2- TCS Research, India

Abstract. In this paper we introduce Iterative Knowledge Distillation (IKD), the process of successively minimizing models based on the Knowledge Distillation (KD) approach in [1]. We study two variations of IKD, called *parental-* and *ancestral-* training. Both use a single-teacher, and result in a single-student model: the differences arise from which model is used as a teacher. Our results provide support for the utility of the IKD procedure, in the form of increased model compression, without significant losses in predictive accuracy. An important task in IKD is choosing the right model(s) to act as a teacher for a subsequent iteration. Across the variations of IKD studied, our results suggest that the most recent model constructed (parental-training) is the best single teacher for the model in the next iteration. This result suggests that training in IKD can proceed without requiring us to keep all models in the sequence.

1 Introduction

Deep Neural Networks have achieved remarkable results in various fields including medical diagnosis, finance, drug discovery, speech recognition and space exploration to name a few. However, they often have hundreds of millions of parameters leading to large model size. This is a burden while training and eventually deploying the model on real-time systems where predictions are required to be almost instantaneous and accurate in a resource constrained environment. As we entertain larger neural networks to model our data, it becomes imperative to consider its storage, computational and power requirements. Our study in this paper addresses the first two aspects only as the third aspect can be evaluated when the model is deployed in a hardware [2].

This paper is concerned with *model compression*, which refers to techniques for simplifying a large neural network to one that requires less resources—usually storage and computational—with no significant loss in performance. A simple form of model compression results, for example, by progressively dropping edges (synaptic connections) with low weights, as long as there is no significant change in estimates of model performance [3]. However, this form of weight-based dropout of edges (and nodes which do not have any incoming edges as a result) is not necessarily the best way to compress a model. In this paper, we propose a variant of a well-known model compression method called *knowledge distillation* (KD). In KD [1], a complex model transfers knowledge to a less complex model [4]. The complex model is often referred to as the *teacher* and the simpler model as the *student*.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

2 Iterative Knowledge Distillation (IKD)

Iterative knowledge distillation–as the name suggests–is an iterative application of knowledge distillation. The procedure is in Algorithm 1. We note that any

Algorithm 1: IKD used for experiments. KD is an abstraction of a function that performs a knowledge-transfer from a "teacher" model to a "student" model. In this formulation, the teacher model on an iteration i can be any of the models $M_0, \ldots, M_{i-1}$.

Given: Model M_0, number of iterations k, data D, loss function L;
$i := 0$;
while $i < k$ **do**
  increment i;
  $j := \underset{t=0,\ldots,(i-1)}{\arg\min}\ L(M_t|D)$ where $L(M_t|D) = L(KD(M_t, i-t)|D)$;
  $M_i = KD(M_j, (j-i))$;
return M_k;

model M is in fact a pair (σ, π), consisting of a structure σ and a set of parameters π. With this in mind, our implementation of the knowledge-distillation function KD is a generalisation of the description in [1]. In that paper, given $M_i = (\sigma_i, \pi_i)$ and σ_j, KD is a function that finds π_j s.t. $KD((\sigma_i, \pi_i), \cdot) = (\sigma_j, \pi_j)$. Here, we assume a *refinement operator* ρ and given $M_i = (\sigma_i, \pi_i)$ and $k > 0$, $KD(M_i, k)$ $= (\sigma_j, \pi_j)$ where $\sigma_j = (\rho^k(\sigma_i))$. Here $\rho^n(x) = \overset{n}{\underset{1}{\circ}} \rho(x)$.

Given a structure, an application of the refinement operator ρ results in a new (usually simpler) structure. In this paper, we use a ρ that implements *block reduction*. By a block, we mean the basic computational unit, repetitions of which make up the complete architecture (an example is the filter in a convolutional layer); block reducing the number of features within each block. In our implementation, each application of ρ reduces block size by 0.5 by reducing the number of filters by half.

3 Empirical Evaluation

3.1 Aims

Our aims in this section are two-fold. Using Algorithm 1:

1. We intend to examine the trade-off between decreased performance and increased compression for increasing iterations of IKD; and
2. We intend to investigate whether a student model benefits from being taught by models other than the model obtained most recently.

For simplicity, we will refer to teaching using the most recent model as *parental-training* and the more general case of using any of the models from previous

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

iterations as *ancestral-training*. In all experiments, the predictive performance of a model will mean accuracy on a test-set and compression will mean the reduction in parameters of the student model over the initial model. In addition, it is well known that pruning can result in substantial model compression [5]. An immediate practical question of interest is whether the use of pruning after IKD results in greater compression than pruning without IKD. We also present empirical results relevant to this.

3.2 Materials

We conduct all our experiments on two popular image datasets: CIFAR-10[1] and MNIST[2]. We compress three different deep models: DenseNet-121, VGGNet-19 and ResNet-152. All three of these models are trained with Adam optimiser [6]. The training hyper-parameters for the algorithm and the models are as follows: the batch size is set to 64, the learning rate is 10^{-4}, momentum factor is $\beta = (0.9, 0.999)$. All the experiments are conducted in Python environment in a machine with 64GB main memory, 16-core Intel processor and 8GB NVIDIA P4000 graphics processor.

3.3 Method

Our method is straightforward:

For each dataset:

1. Split the data into a training set D_{train} and a test set D_{test}
2. Construct a model M_0 using D_{train}.
3. For $k = 1, 2, \ldots, n$
   (a) Construct a model M_k using the IKD procedure with iterative bound k and training data D_{train}
4. For $k = 0, \ldots, n$
   (a) Let E_k be the error of M_k on D_{test} and S_k be the size of M_k
   (b) Let $M'_k = Prune(M_k)$
   (c) Let E'_k be the error of M'_k on D_{test} and let S'_k be the size of M'_k
5. For $k = 1, \ldots, n$
   (a) Compare (E_0, S_0) against (E_k, S_k)
   (b) Compare (E'_0, S'_0) against (E'_k, S'_k)

The following details are relevant:

- We repeat the experimental procedure above for three separate structures: DenseNet-121, VGGNet-19 and ResNet-152.
- The loss for the student model M_s given the training data D is defined as:

$$L(M_s|D) = \frac{1}{m}\sum_{j=1}^{m}\left\{2T^2\alpha D_{KL}(P^{(j)}, Q^{(j)}) - (1-\alpha)\sum_{i=1}^{c} y_i^{(j)} \log(1-\hat{y}_i^{(j)})\right\} \quad (1)$$

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`
[2] `http://yann.lecun.com/exdb/mnist/`

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

where, D_{KL} denotes the KL-divergence, m denotes the batch size, c denotes the number of classes, T is the temperature to soften probability distributions; α is the relative importance of teacher's guidance w.r.t. hard targets from data [1]. $P^{(j)}$ and $Q^{(j)}$ are the T-softened class probability distributions from the student and teacher model respectively for the j^{th} data instance; $Y = \{y_1, \ldots, y_c\}$ is the one-hot encoded ground truth, and $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_c\}$ is the prediction from the student model. In our work, we set $T = 20$ and $\alpha = 0.7$.

- In experiments reported here, n (the maximum number of iterations) is 4.
- Each model M_i is a pair (σ_i, π_i). We take model size $S_i = |\pi_i|$.
- When comparing model errors (E's) and sizes (S's), we say the model M_i is an acceptable compression of model M_j iff $E_i \approx E_j$ and $S_i < S_j$. We take the errors to be approximately equal if they are within some pre-defined tolerance (0.1 in our case). If M_i is an acceptable compression of M_j, then the *compression* of M_i is S_j/S_i.

3.4 Results

Results from experiments are tabulated in Table 1. The principal findings in the tabulation are these: (a) For both datasets (CIFAR-10 and MNIST) and the three architectures (VGGNet, DenseNet, and ResNet), increasing the number of iterations in IKD results in an increase in model-error and a decrease in model-size. These results are as expected; (b) It is possible to achieve acceptable compression with IKD (that is, decrease in size, with error being approximately the same). The amount of compression are both architecture- and data-dependent. For example, while increasing iterations of IKD result in acceptable compressions on both datasets with DenseNets, this is not the case with VGGNet on CIFAR, where error increases significantly after the third iteration; (c) For both datasets and all three architectures, the best teacher is always the immediately preceding model. Thus, it would appear that the more general ancestral-training approach does not add any value over parental-training.

Why is compression architecture-dependent? This is due to the granularity of the refinement operator for the architectures. VGGNet is a Convolutional Neural Network, which uses shared parameters (filter blocks). Therefore, the a block-reduction based refinement of structures is able to eliminate more parameters in VGGNet than it is for a DenseNet.

What about pruning? The IKD procedure is not intended to be alternative to pruning. In practice, we would expect to act as a pre-processing step to pruning. Table 2 tabulates results when it is used in this capacity.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

		DenseNet-121						VGGNet-19						ResNet-152					
		CIFAR-10			MNIST			CIFAR-10			MNIST			CIFAR-10			MNIST		
Model	S	T	E	C	T	E	C	T	E	C	T	E	C	T	E	C	T	E	C
Base	M_0	–	0.16	1x	–	0.02	1x	–	0.12	1x	–	0.01	1x	–	0.24	1x	–	0.02	1x
IKD	M_1	M_0	0.18	2x	M_0	0.02	2x	M_0	0.17	4x	M_0	0.01	4x	M_0	0.20	2x	M_0	0.02	2x
	M_2	M_1	0.17	2x	M_1	0.01	2x	M_1	0.20	16x	M_1	0.01	16x	M_1	0.20	3x	M_1	0.02	3x
	M_3	M_2	0.19	5x	M_2	0.02	5x	M_2	0.28	64x	M_2	0.01	64x	M_2	0.20	4x	M_2	0.02	4x
	M_4	M_3	0.19	9x	M_3	0.02	9x	M_3	0.44	254x	M_3	0.01	254x	M_3	0.19	7x	M_3	0.01	7x

Table 1: Results of using IKD without pruning. The "S" refers to student and the "T" refers to chosen teacher. The baseline model for comparisons is the initial model M_0. IKD with An entry of $S = M_j$ and $T = M_i$ for IKD denotes that the procedure returned the student model M_j after selecting M_i as the teacher from $M_0, \ldots, M_{i-1}$. "E" denotes the error of the student model on test data. C is the model compression (that is the ratio of the size of M_0 to the size of the student model).

		DenseNet-121						VGGNet-19						ResNet-152					
		CIFAR-10			MNIST			CIFAR-10			MNIST			CIFAR-10			MNIST		
Model	S	T	E	C	T	E	C	T	E	C	T	E	C	T	E	C	T	E	C
Base	M'_0	–	0.19	1x	–	0.02	1x	–	0.16	1x	–	0.02	1x	–	0.23	1x	–	0.03	1x
IKD	M'_1	M_0	0.17	1x	M_0	0.04	2x	M_0	0.18	2x	M_0	0.09	2x	M_0	0.23	2x	M_0	0.03	2x
	M'_2	M_1	0.17	1x	M_1	0.05	2x	M_1	0.29	6x	M_1	0.02	7x	M_1	0.25	3x	M_1	0.05	3x
	M'_3	M_2	0.19	2x	M_2	0.04	5x	M_2	0.30	30x	M_2	0.02	32x	M_2	0.26	6x	M_2	0.05	6x
	M'_4	M_3	0.20	2x	M_3	0.05	9x	M_3	0.43	60x	M_3	0.09	180x	M_3	0.22	10x	M_3	0.03	11x

Table 2: IKD with pruning. Assuming a prune function $Prune$, here $M'_i = Prune(M_i)$. It suggests that pruning with IKD gives 2 to 180 fold increase in compression over pruning without IKD depending on the architecture and the dataset. This gain is equivalent to $|Prune(IKD(M_0))|/|Prune(M_0)|$.

When do we stop? The procedure in Algorithm 1 allows us to examine changes in error and compression with increasing number of iterations. However, in practice, we would expect to use a validation set to decide when to stop. That is, we would call the procedure with increasing values of depth, stopping when validation error increases beyond an acceptable amount.

The apparent sufficiency of parental-training allows us to simplify the procedure in Algorithm 1. Incorporating a validation-based check results in the procedure in Algorithm 2. The procedure terminates when performance of a student on the validation set drops beyond an acceptable tolerance.

Algorithm 2: Practical IKD. A student is taught only by the model obtained on the previous iteration (parental-training).

Given: Model M_0, an iteration bound k, validation data D_v, a performance measure A, a tolerance ϵ, and a prune function $Prune$;
$i := 0$;
$M^* := M_0$;
$done := false$;
while $i < k$ *and* $\neg done$ **do**
  increment i;
  $M_i := KD(M_{i-1}, 1)$;
  if $(|A(M_i|D_v) - A(M^*|D_v)| \leq \epsilon)$ **then**
    $M^* := M_i$;
  else
    $done := true$;
return $Prune(M^*)$;

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

4 Related Works

Koutini *et al.* [7] propose an iterative method for learning audio event detection. This approach applies KD between the same teacher and student model in each temporal instance of an input audio signal and, unlike our approach, does not iteratively reduce the model size. Wang *et al.* [8] propose an iterative blockwise pruning method in KD. Our empirical analysis is generic for such blockwise pruning operations and additionally provides a holistic comparison between *parental*- and *ancestral*- training.

5 Conclusion

In this paper, we introduced iterative variant of KD process described in [1] for neural network compression. The results with two benchmark datasets, and three different kinds of neural network architectures all provide evidence for the following: (a) iterating knowledge distillation can increase compression. The gains can be moderate to substantial, depending on the dataset and architecture; and (b) Iterative knowledge distillation only requires parental-training. The result in (b) is interesting in the context of knowledge-transfer, and useful computationally. In fact, additional experiments we have performed (and not reported here for reasons of space) suggest that parental-training is sufficient, even if all ancestral models were used to teach the student.

Acknowledgement

This work is supported by "The DataLab" agreement between BITS Pilani, K.K. Birla Goa Campus and TCS Research, India.

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[2] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019.

[3] Haipeng Jia, Xueshuang Xiang, Da Fan, Meiyu Huang, Changhao Sun, Qingliang Meng, Yang He, and Chen Chen. Droppruning for model compression. *arXiv preprint arXiv:1812.02035*, 2018.

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[5] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Khaled Koutini, Hamid Eghbal-zadeh, and Gerhard Widmer. Iterative knowledge distillation in r-cnns for weakly-labeled semi-supervised sound event detection. Technical report, Tech. Rep., DCASE2018 Challenge, 2018.

[8] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *IJCAI*, pages 2769–2775, 2018.