Literature review
Knowledge distillation of deep neural networks

Andris Freimanis, 880317-3650, DIT892, gusandrifr@student.gu.se

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

1. Introduction

Deep learning has revolutionized the field of artificial intelligence (AI), achieving remarkable and record-breaking results in various tasks such as image recognition (Krizhevsky et al., 2017) and natural language processing (Devlin et al., 2019). However, this revolution has also brought about significant challenges, including exorbitant computational and memory requirements (Angelov & Sperduti, 2016). State-of-the-art deep learning models often consist of billions of parameters (Brown et al., 2020) and are computationally expensive to train and deploy. These restrictions can be impractical for real-world applications, particularly in environments with limited resources such as smartphones or in scenarios where real-time predictions are necessary, such as self-driving cars.

This is where model compression comes into play. It aims to mitigate the challenges of deep neural networks by reducing their size, memory usage, and computational complexity while preserving their performance. With model compression, these models can become more accessible and cost-effective, allowing them to be deployed in a wide range of environments.

2. Model Compression Techniques

There are a myriad of model compression techniques, including pruning, quantization, weight sharing, and knowledge distillation. Knowledge distillation (KD) will be described in-depth in the following section, while the other techniques are briefly described here.

Pruning involves reducing the size and complexity of a neural network model by selectively removing certain neurons or connections (and thus weights). The goal is to eliminate redundant or unimportant neurons or connections, resulting in a more compact and computationally efficient model.

Quantization involves reducing the precision of parameters or activations by converting them to a lower bit datatype. While this reduces the memory requirements of a model and can speed up inference, it can also potentially reduce the model's performance.

Weight sharing involves reusing the same weights or set of weights in multiple parts of the network (connections or neurons). This technique is often used in Convolutional Neural Networks (CNNs) to reduce memory requirements and computational complexity (Gou et al., 2021).

3. Knowledge Distillation

Introduction

KD is a model compression technique used in machine learning where a smaller, simpler model (known as the student) is trained to mimic a larger, more complex model (known as the teacher). It was first introduced by (Hinton et al. 2015) as a method to compress the knowledge of an ensemble of neural networks into a single neural network. The primary goal of KD is to compress the knowledge of the teacher model into the student model while maintaining similar

performance. This is particularly beneficial when deploying such models on devices with limited computational resources, such as smartphones. Moreover, student models often generalize better and are less prone to overfitting compared to training them on the original dataset.
Typically, a student model learns from both the teacher model and a dataset. It receives hard targets from the dataset and soft targets from the teacher model. Hard targets are the original labels in the dataset, usually one-hot encoded. Soft targets are the outputs of the teacher model. These outputs could be the activations of the last layer before a sigmoid or a softmax function (known as logits) or the probabilities output by the teacher model. Soft targets are often more informative for the student model because they show how confident the teacher model is about its output. Typically, the student model learns from both soft and hard targets simultaneously using a weighted combination of two loss functions (Gou et al., 2021).

Hyperparameters

KD introduces a few hyperparameters to control the training process of a student model. Two main hyperparameters are the temperature and the teacher/student model architecture.
The temperature is used in the softmax function to control the importance of each probability. Higher temperatures smooth out the probability distribution across various classes, which can help the student model by putting more emphasis on the relative likelihoods of other classes. A temperature that is too low can make the soft targets resemble hard targets (assuming the teacher model's prediction was correct), while a high temperature can make the probability distribution uniform, thus losing any information from the teacher model.
The architectures of the teacher and student models have also been identified as impacting the success of KD. The teacher model needs to be complex enough to learn representations in the data, while the student model needs to be simple enough to be deployable in limited resource environments. The difference in complexity between both models is known as the capacity gap. If it's too large, then the student model may struggle in learning to mimic the teacher and result in a large drop in performance. If the gap is too small, then KD might provide little to no benefit. It's also useful to remember that the goal of KD is often to obtain a smaller model for deployment while still retaining similar performance. While choosing teacher and student model architectures is recognized as an important aspect of KD, there is little research in this field and no consensus on what these architectures should be (Gou et al., 2021).

Knowledge types

There are multiple types of knowledge that can be distilled from a teacher model to a student model. Each type of knowledge provides the student model with a different perspective to learn from. These knowledge types can be combined or used individually, and in this section, we'll describe the three main types (Gou et al., 2021).

Response-Based Knowledge

Response-based knowledge only uses output layers (either logits or probabilities with or without temperature) of the teacher model when training the student model. The aim is for the student to learn to mimic this output of the teacher. This is typically done using a divergence loss such as Kullback-Leibler loss. The most notable disadvantage is that response-based knowledge cannot supervise the learning of intermediate layers of the student model, which is important for learning representation in data (Gou et al., 2021).

Feature-Based Knowledge

Intermediate layers in neural networks represent abstract representations of data. Because of this, it might be prudent to make use of these abstractions when training a student model. The aim is for the student model to learn to mimic intermediate feature maps of the teacher model. Typically, a loss that compares feature maps is used. If the feature maps are in different shapes, then transformation functions are applied before the loss function calculation.
However, there is still no consensus on how to choose intermediate teacher layers (hint layers) and student layers (guided layers). Neither is there consensus on how to properly match feature representations of different shapes in these layers (Gou et al., 2021).

Relation-Based Knowledge

Correlations between nodes in consecutive layers or relations between instances of data provide another source of knowledge. This shows either how information flows through the teacher model's nodes and layers or how similar instances of data are represented by the teacher model's feature maps. The student can learn to either represent the instance information in the same way as the teacher or find correlations between instances in the same way as the teacher model does, or both. Typically, similarities between pairs of feature maps or feature maps for pairs of data instances are calculated for both models. Then a loss (such as correlation) between these similarities is calculated. Like with other knowledge types, there is no consensus on what the proper way to model relation information between layers or instances is (Gou et al., 2021).

Distillation strategies

There are also multiple strategies in which KD can work:

Offline Distillation

Most KD methods employ offline distillation, where the teacher model is fully pretrained and its weights are fixed during the training of the student. In offline distillation, the focus is placed on improving KD, i.e., the design of knowledge, choice of loss function, etc. This strategy is relatively simple and allows for the use of a well-optimized teacher model (Gou et al., 2021).

Online Distillation

In online distillation, both models - teacher and student; are trained at the same time on the same data stream. While the teacher is trained exclusively on the data, the student is trained to match the predictions and intermediate representations (feature maps) of the teacher model as well as on the data. This allows for a more dynamic interaction between the teacher and student models during training, which can lead to better performance. However, it's also more complex to implement than offline distillation (Gou et al., 2021).

Self-Distillation

In self-distillation, the teacher and student networks are the same network. The model can first be trained on data and then tuned on its own predictions. In another case, the data representations from a deeper layer can be transferred to shallower layers. This can be taken advantage of by introducing an early exit layer in the network that tries to mimic the predictions of a deeper exit layer. This strategy avoids the need for two separate models (Gou et al., 2021).

4. Conclusion

Deep learning has undeniably revolutionized the field of artificial intelligence. However, the computational and memory requirements of state-of-the-art deep learning models often make them impractical for real-world applications, particularly in resource-constrained environments.
Model compression techniques, including pruning, quantization, weight sharing, and knowledge distillation, have emerged as potential solutions to this challenge. These techniques aim to make these models more accessible and cost-effective by reducing their size and improving inference speed.
Knowledge distillation (KD) has shown great promise. By training a smaller, simpler student model to mimic the behavior of a larger, more complex teacher model, KD can improve the student model's performance and generalization ability in comparison to training it on data alone. Despite its benefits, KD also has challenges - the choice of teacher and student architectures, the capacity gap between these models, the type of knowledge used for distillation, distillation strategy, and the temperature parameter are all factors that can significantly impact the success of the distillation process (Gou et al., 2021).
The different knowledge types that can be distilled from a teacher model (response-based knowledge, feature-based knowledge, and relation-based knowledge) provide a different perspective for the student model to learn from. They can be used individually or combined depending on the specific requirements of your task. Furthermore, there are multiple strategies with which KD can work: offline distillation, online distillation and self-distillation (Gou et al., 2021).

References

Angelov, P., & Sperduti, A. (2016). Challenges in Deep Learning. *Computational Intelligence*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners* (arXiv:2005.14165). arXiv. http://arxiv.org/abs/2005.14165

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, *129*(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network* (arXiv:1503.02531). arXiv. http://arxiv.org/abs/1503.02531

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386