Improvements on Model Compression Based on Knowledge Distillation

Abstract

Despite the success of deep neural networks has been achieved in recent years, it remains a challenge to apply cumbersome DNN models to some resource-limited tasks. Knowledge distillation (KD) is a promising search field to enhance the performance of lightweight models by learning knowledge from larger models. The performance of KD models, however, suffers from 1) conflicts among multiple distillation objectives as well as 2) capacity mismatch between student and teacher models. To solve the first problem, I'll investigate the gradient flows of different distillation losses and remove the conflict parts of them using existing methods in multi-task learning and transfer learning. For the second problem, I'll loose the restriction of one student layer can only learn from one teacher layer. Specifically, I plan to apply an attention mechanism, letting each student layer can learn from multiple teacher layers and learning the attention weights of each teacher layer as well.

Introduction

In recent years, deep neural networks have been successful in both industry and academia, especially for computer vision tasks. The great success of deep learning models is mainly due to their scalability to encode large-scale data and billions of model parameters from deep neural networks. However, it is a challenge to deploy these large deep models on devices with limited resources, e.g., mobile phones and embedded devices, because of high computational complexity and large storage requirements. To this end, a variety of model compression and acceleration techniques have been developed. As a representative type of those techniques, KD aims to effectively transfer useful knowledge from one or several large teacher models to a small student model. A vanilla KD model would use the logits of a large deep model as the teacher knowledge, while features of intermediate layers, relations of those features and parameters from the teacher model can also provide rich information which student model may benefit from.

Two key steps of KD are 1)extracting knowledge from the teacher model and 2) using the knowledge extracted from the teacher model to guide the training of the student model. When constructing a KD model, most existing approaches leverage a combination of different kinds of knowledge, namely response-based, feature-based, and relation-based knowledge. Those knowledge have complex interactions with each other and sometimes even have contradicted influences on the same layer of the student model, reducing the quality of extracted knowledge. What's more, some recent works show that capacity mismatch between student and teacher models may lead to poor knowledge transferring effect, which calls for better design for

teacher-student architectures. To solve those problems, I will conduct research as follows: 1) investigate how different kinds of knowledge interact with each other and form a complementary framework utilizing existing techniques in multi-task learning and transfer learning. 2) make improvements to teacher-student architectures that help to bridge over the capacity mismatch between student and teacher models with an attention mechanism.

Research Plan

1. Investigate complementary framework of multi-knowledge for KD

   As mentioned before, most KD methods leverage a combination of different kinds of knowledge. A typical loss function of a KD model looks like this:

   $$\mathcal{L} = \mathcal{L}_{ce} + \lambda_1\mathcal{L}_{logits} + \lambda_2\mathcal{L}_{feature} + \lambda_3\mathcal{L}_{relation}$$

   where $\mathcal{L}_{ce}$ is the cross-entropy loss between the final prediction of the student model and ground truth labels, $\mathcal{L}_{logits}$ directly mimics the soft prediction of the teacher model, $\mathcal{L}_{feature}$ is used to mimic the intermediate process of the teacher model and $\mathcal{L}_{relation}$ is used to capture the relationships across different samples. Besides, $\lambda_1, \lambda_2, \lambda_3$ are weights for each KD loss respectively. The knowledge of different kinds, however, may have different influences on the training of the student model and it remains as a challenge to model different types of knowledge in a unified and complementary framework.

   As KD always involves with more than one optimizing goals, it can be viewed as a multi-task learning process. If optimization $\mathcal{L}_{ce}$ is regarded as the target task while others as source tasks, KD can also be modeled as a transfer learning task. Based on such insights, I plan to utilize techniques from the domain of multi-task and transfer learning to solve contradictory knowledge in KD.

   One possible technique is gradient decomposition. Previous studies have shown that two gradients having a negative cosine similarity might have a detrimental effect on the training process. So, in a KD process, if two gradients conflict, i.e., have a negative cosine similarity, I'll choose one gradient, apply orthogonal decomposition with respect to the other gradient and remove the projection vector. After applying the above procedure to every pair of gradients, there'll be no conflict gradients. Therefore, the problem of contradictory knowledge can be relieved.

2. Improve teacher-student architectures to bridge over the capacity mismatch between student and teacher models via attention mechanism.

   For the success of KD, some researchers have focused on the relation between students and teachers. To their surprise, they found out students distilled from a bigger teacher, one with more parameters and higher accuracy, can perform worse than the same students distilled from a smaller teacher. One possible explanation for this contradiction is that the student may fail to mimic the teacher because of the capacity mismatch in teacher-student architectures.

In KD, it's normal for a teacher model to have more layers than the student model, which makes it hard to decide which teacher layer a student layer should learn from when applying distillation to intermediate layers. Existing methods usually assign a mapping function between indices of student layers and teacher layers. For example, assuming that the student model has M intermediate layers and the teacher model has N intermediate layers (N>M). Then a function $n = g(m)$ is defined as the mapping function between indices from student layers to teacher layers, which means that the m-th layer of the student model learns the information from the g(m)-th layer of the teacher model. Such approach includes massive manual work when choosing the mapping function, and there's no guarantee that the information from the chosen teacher layer is sufficient or beneficial for the student layer to learn from, especially for cases where the teacher model has significantly more layers than the student model.

To solve this problem, I believe it's beneficial to introduce an attention mechanism. Instead of directly appointing a one-to-one mapping function, we can assign a range of teacher layers to one student layer and let the student layer itself to decide the weights of each teacher layer via attention mechanism. Specifically, for the $m - th$ layer of the student, we appoint k teacher layers instead of one, i.e., the $m_1 - th, m_2 - th, \ldots, m_k - th$ layer of teacher model, to provide information.

Assuming the output feature of the $m - th$ layer of the student is S_m and the features of k teacher layers are $T_{m_1}, T_{m_2}, \ldots, T_{m_k}$ respectively, the attention weights of teacher layers are defined as:

$$attn_m = softmax([a(T_{m_1}, S_m), a(T_{m_2}, S_m), \ldots, a(T_{m_k}, S_m)])$$

where $softmax()$ is the softmax function, $a()$ is some kind of similarity metric which returns a scalar.

Thus, the learning objective is :

$$\mathcal{L}_m = loss(\sum_{i=1}^{k} attn_m^i \cdot T_{m_i}, S_m)$$

where $attn_m^i$ means the i-th entry of the vector $attn_m$, and $loss()$ is some kind of loss function.