Knowledge Distillation for Efficiency

Team Member: Guanqi Zeng

November 2021

Abstract

In reality, a practical problem arises when we train a model with high capacity source but fail to compress it to smaller devices due the cumbersome structure. In such situations, we want to distill knowledge to keep most of the model's performance by training a smaller model. In other words, we transfer knowledge from a teacher network to a student network. One way to do this is that, instead of predicting the hard labels (real targets), in the student network, we predict the soft targets, which are the probabilities of the hard labels in the teacher network. In this project, we distill knowledge to learn a classifier in two scenarios: the training data of the teacher network is available and the training data of the teacher network is not available.

Keywords: teacher network, student network, hard labels, soft labels, temperature

1 Introduction

Knowledge distillation has been developed for a few years and applied to allow training much simpler models with similar performance as the complex model. The key is to transfer knowledge learned by the complex model to a much simpler model. We call the complex model, teacher model, and the simple model, student model. To transfer knowledge from the teacher model to the student model, we train the student model to predict both the hard label, the true target of the data set, and the soft label, the probabilities from the softened logits.

An even more practical consideration for model training is that, the training set for the teacher model sometimes may not be accessible. For example, people may concern about the data privacy, especially in healthcare industry. Another example might just to avoid information leakage to company's competitors. In this situation, we want to train the student model without any access to the training. To maintain the performance, we apply knowledge distillation. This type of knowledge distillation is called "Zero-Shot" or "Data-Free" knowledge distillation. The key idea is to generate training set for the student model, which is also called a transfer set, from the distribution of the training.

In remaining contents of this report explain the algorithms of these two types of knowledge distillation based on two paper, describe the the implementation of knowledge distillation and the generation of transfer set, and discuss the experimental results.

2 Related works

The project is originally based on two papers: *Distilling the Knowledge in a Neural Network* by Geoffrey Hinton, Oriol Vinyals, and Jeff Dean and *Zero-Shot Knowledge Distillation in Deep Networks* by Gaurav K. Nayak, Konda R. Mopuri, Vaisakh Shaj, R. Venkatesh Babu, and Anirban Chakraborty. The first paper discusses the situation that the training data of the teacher network is available; the second paper discusses the situation that the training data of the teacher network is not available. We explain the algorithms in each paper.

2.1 Case 1: Training Data of Teacher network is available

First, we explain how "temperature" is used to softened the probabilities. Below is the softmax function that produces the probabilities for each label in the data set.

$$p_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

p_i denotes the probability of label i. z_i and z_j denotes the logits. By bringing the temperature, T, the softened probability becomes,

$$q_i = \frac{exp(z_i)/T}{\sum_j exp(z_j)/T}$$

Compared to p_i, q_i tells us more about how correct or how incorrect the prediction is by lowering the highest predicted probability and increasing the predicted probabilities of those labels that are resemble to the highest one. For example, in MNIST, the softened predicted probabilities will show that for predicting the target of "7", "1" and "9" will have higher probabilities than "6".

Such quantification of correctness is what needs to be transferred to the student model. Thus, when training the student model, we want to minimize the loss that is actually a weighted loss of hard labels and soft labels. That is, the loss calculated between the softened predicted probabilities from the student model and the hard labels of the training set , and the loss calculated between the softened predicted probabilities from the student model and the predicted probabilities from the teacher model.

Something to notice is the choice of temperature. The value of the temperature does affect how we quantify the correctness and incorrectness of the prediction. A simple way to see this is by comparing the regular softmax function. The logits in the regular softmax function can be seen to be divided by $T = 1$. With this value of temperature, the probability on the correct label is

almost 1, whereas the others are similarly low. We will show more details in section 4.

2.2 Case 2: Training Data of Teacher network is not available

The key idea for this part is the generation of Data Impression, which is the generated transfer set for training the student network.

To generate Data Impressions, we need the distribution of the training set's teacher model softmax outputs and the class similarity matrix. The distribution of the softmax outputs follows a Dirichlet(K, $\alpha^{\mathbf{k}}$) distribution, as they sum up to one. K denotes the number of classes, and $\alpha^{\mathbf{k}}$ denotes the concentration. This concentration can be seen as how "concentrated" the probability mass is likely to be. A low value of α indicates that the probability mass will be concentrated in only a few components while the rest are almost zeros. A high value of α indicates that the probability mass will be more dispersed among the components. This concentration parameter is what carries the knowledge from the training.

To obtain concentration for generating Data Impression, we use the weights from the last layer of the teacher model to construct a class similarity matrix to indicate the similarity between classes. The formula resembles correlation:

$$C(i,j) = \frac{\mathbf{w_i}^T \mathbf{w_j}}{||\mathbf{w_i}||||\mathbf{w_j}||}$$

where C demotes the matrix and $\mathbf{w}$ denots the weights. We also need to input scaling factors, β, which controls the variance of the distribution.

The algorithm is as follows. We obtain the number of classes, K, from the teacher model. For a trained teacher model, we obtain the class similarity matrix. For each class, k, we set the concentration α^k to equal to the vector in class similarity matrix component corresponding to this class. Then, for each scaling factor, β, we generate a sample from $Dir(K, \beta x \alpha^k)$. This is the generated probabilities. We then generate a sample of random noise that has the same dimension of a sample of the data, and optimize it by a loss function between the generated probabilities and the output from the teacher model with the random sample.

When we obtain these Data Impressions, we train the student by the knowledge distillation loss function between the output from the teacher model with the DIs and the output from the student model with the DIs.

3 Details of the project

There are two parts of the project. The first part demonstrate knowledge distillation when the training set of the teacher network is available. The second part discusses the situation when the training set of the teacher network is not available.

```
Teacher(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=9216, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
```

Figure 1: Part 1 Experiment 1 Model Architecture

3.1 Part 1

For the first part, we conduct three experiments. The first experiment uses MNIST. The purpose of this experiment is to show how the value of temperature can convey the correctness and incorrectness the model thinks about the prediction. The reason to use MNIST is that the similarity of shapes between numbers is obvious to eyeball. We train a neural network, extract the logits prediction of the first sample in the test set, and manually add a series values of temperatures to the logits. Then we send the logits to a softmax function and plot the probabilities for each label under each temperature.

The second experiment uses FashionMNIST. We use the same CNN that trains MNIST as the teacher network to train FashionMNIST. We also build a much simpler neural network, which only has two fully connected layers, as the student network. In this experiment, we train the student network without and without the temperatures. The purpose for this experiment is to compare and contrast the effect of temperature to the student model.

The third experiment uses CIFAR10. We use LeNet5 as the teacher model and the LeNet5-Half (which reduces the filter sizes of LeNet5 by half) as the student model. As CIFAR10 has 3 channels and is much different from FasionMNIST and MNIST, we want to know about the performance of knowledge distillation on this dataset. We also compare the accuracies of the teacher model and the student model.

3.2 Part 2

For the second part, we conduct one experiment on FashionMNIST. The purpose of this experiment is to implement the algorithm of generating Data Impressions from the training set. We also plot the similarity matrix to show how the knowledge transfer works in this scenario.

4 Experimental results

4.1 Part 1: Knowledge Distillation with Teacher's Training Set

4.1.1 Demonstration of the effect of temperature

The model's architecture is shown in Figure 1. The extracted sample is shown in Figure 2. The

By using a range of temperatures, $T = 1, 5, 10, 15, 20, 40, 60$, the probabilities for each label is plotted in Figure 3:

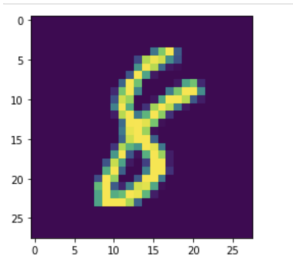

Figure 2: Part 1 Experiment Image (8)

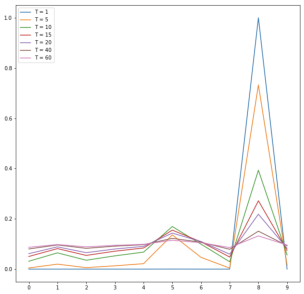

Figure 3: Impact of Temperature

We can see that, in Figure 3, a low temperature gives us sharp probability on the prediction the model thinks is correct, and the remaining probabilities are very low and similar. With temperatures of 5 and 10, we can see that the model thinks the prediction of a "5" is much more likely to be correct than a prediction of "0". The result resembles to the judgement of human eyes. When the temperature is too large, it's hard for the model to tell if "5" or "8" is correct.

4.1.2 Experiment on FashionMNIST

The test accuracy of the student model along without temperature is 98.15%. With $T = 10$, after the same number of epochs, the test accuracy of the student model is 91.07%, and the test accuracy of the teacher model is 90.78%. The test accuracy from the teacher model and the student model grow at relatively same pace.

The student model's architecture is shown in Figure 4.

```
Student(
  (fc1): Linear(in_features=784, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=10, bias=True)
)
```

Figure 4: Part 1 Experiment 2 Student

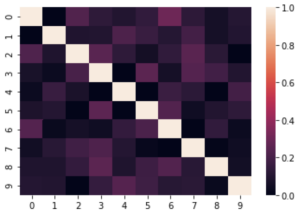

Figure 5: Part 2 Similarity Matrix

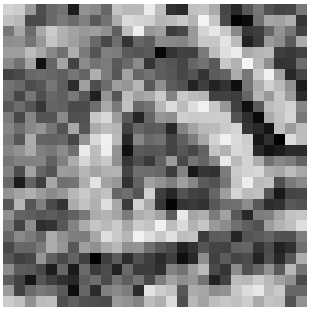

Figure 6: class 0

4.1.3 Experiment on CIFAR10

The test accuracy of the teacher and the student are 44.41% and 44.27% after 30 epochs. The accuracy in both model grow very slowly.

4.2 Part 2: Knowledge Distillation without Teacher's Training Set

In this experiment, we use LeNet5 as the Teacher model. The test accuracy is 98.91%. We train the model, and extract the weights of the last layer. Then, by using these weights, we produce the class similarity matrix. Figure 5 shows the result of the matrix.

We also generate the Data Impressions. Figure 6 7 shows two of them.

5 Concluding remarks

There are several things remarkable about knowledge distillation. We don't need to train a model as complex as the teacher to get similar test accuracy. Also, as long as the "knowledge" are transferred to the student, we don't even need the access to the training data.

There are also some downsides we see from the experiments. In Part 1, first experiment, we can see that the accuracy is actually lower with temperature

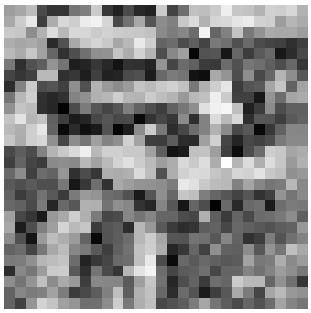

Figure 7: class 7

than without temperature. We may adjust the value of the temperature or the number of training epochs. However, the number of training epochs needed might be much larger for some data, as we can see in the third experiment of part 1. This can bring some computational issue.

A second downside might be that using Data Impression may not reach a high accuracy. The hyper-parameters, for example, scaling factors, are also hard to determine. A further study on DI generation process might be a good direction to explore.

Other things to think about can be the choice of loss function and the distillation weight between the loss to predict soft target and the loss to predict the hard target.

6 Discussion

6.1 Transfer learning and Knowledge Distillation

It might be confusing to distinguish between knowledge distillation and transfer learning. However, they fundamentally have very different objectives. For transfer learning, we're transferring the weights from a pre-trained network and building another one that has the exactly same architecture. For knowledge distillation, however, the student network is much lighter.

6.2 Theoretical Background Support

Another thing to be noticed about knowledge distillation is that, the method stems from practical experience. In other words, it's heuristic. Potential future topics might dig into the theoretical proofs on why distilling knowledge work. It might also be a good direction to study the features of this algorithm and the theoretical supports behind other variations of knowledge distillation.

7 Reference

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015).

Nayak, Gaurav Kumar, et al. "Zero-shot knowledge distillation in deep networks." International Conference on Machine Learning. PMLR, 2019.

https://towardsdatascience.com/knowledge-distillation-and-the-concept-of-dark-knowledge-8b7aed8014ac

https://github.com/yangze01/Distilling$_{t}he_{K}nowledge_{i}n_{a}_{N}eural_{N}etwork_{p}ytorch$

https://github.com/da2so/Zero-shot$_{K}nowledge_{D}istillation_{P}ytorch$

https://paperswithcode.com/paper/zero-shot-knowledge-distillation-in-deep/review/