REPRODUCING KNOWLEDGE FLOW: IMPROVE UPON YOUR TEACHERS

Bartomiej Najdecki

ABSTRACT

Nowadays, we can find a lot of different neural nets for almost any given task. But what if there is no solution to our problem? Which network to use as initialization? The authors of the reproduced paper proposed to use *Knowledge Flow Framework*. The knowledge flow gives a possibility to pass knowledge from teachers nets to student net. Authors demonstrated their approach on a variety of supervised and reinforcement learning tasks, outperforming fine-tuning and other knowledge exchange methods. I have tried to reproduce only supervised part of the paper.

1 KNOWLEDGE FLOW FOR SUPERVISED LEARNING

Knowledge flow is a technique that is used to speed up learning and help achieve better results. For that purpose, we use teachers (deep neural net) that will transfer some part of knowledge to the student. At the end of teaching the student should be independent. The biggest advantage of this technique is that the teachers can be trained to a totally different task. For example, in the original paper, neural net from Seaquest (Atari game) was used as a teacher for training NN for games like Alien, Asterix and Boxing, which are totally different than Seaquest. Moreover, we can use more than one teacher.

1.1 HOW IT WORKS

In this section, I have briefly described the method. Firstly, we have to define our L_i lists - connection lists for every layer of student's net. For example, $L_1 = \{l_1^1, l_2^2, l_0^1\}$ means that we sum outputs of the first layer in the first net, the second layer in the second net, and the first layer in the student's net. This linear combination is the input for the second layer of student's net.
Then we start training for our parameters: p_j^i and Q_j^i.
p_j^i is the weight from the range $[0, 1]$ for an ith edge in L_j connection list. This parameter responds to transferring knowledge. p_0^i is the weight for the output of the ith layer of the student's net.
Q_j^i is the matrix that transforms the output of some layer to the input format of the $i + 1$ student's layer.
The input for $i + 1$ layer is equal to:

$$INPUT_{i+1} = \sigma(\sum_i p_j^i O_j^i Q_j^i) \tag{1}$$

In the equation (1) by O_j^i we denote the output of ith layer from L_j. More about the restriction for parameters you can find in section (1.3). The figure (1.1) gives some example of Knowledge Flow framework architecture.

1.2 CONNECTIONS BETWEEN LAYERS

The whole method doesn't limit the number of connection, but the authors of the paper claims, that the best results you can achieve when you connect every layer of a teacher with one or two layers of student. The paper doesn't describe how they connect layers in given examples.

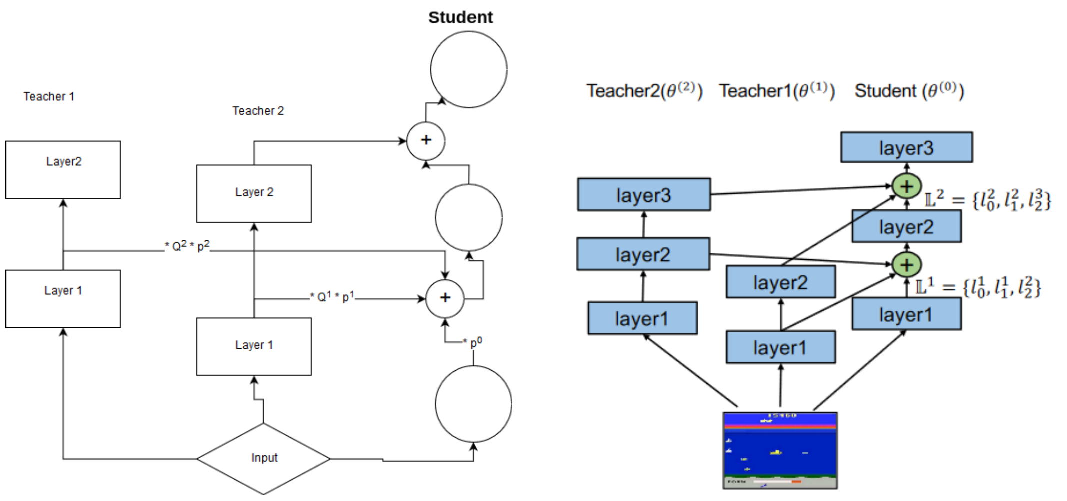

Figure 1: The knowledge flow examples. There is no restriction about connections between layers.

1.3 Deep net transformation

Knowledge flow enhances the student in learning by decreasing the p_j^i parameter. To enable learning process authors add invariant to the method:

$$\sum_{i=0}^{|L_j|} p_j^i = 0 \tag{2}$$

The equation (2) is true for every $1 \leq j \leq N$, where N denotes the number of student's layers. This invariant makes easy to define penalty function (marked in the paper as the teacher's influence metric):

$$-1/N \sum_{i=0}^{N} \log p_0^i \tag{3}$$

In equation (3) we denote the number of student's layer as N. Unfortunately authors don't specify how they combine this metric with some loss function.

2 Reproduction

At the moment of writing this paper, there was no code on github from the author of referenced paper.

2.1 Creating a framework

The first thing that needs to be done is the framework. I have created in python class name PassKnowledgeFramework. It takes a list of teachers and student and list of connections. At the beginning p_{i0} are initialized with 0.1, others p's are initialized with 1. Then the framework normalized them according to the previous section. In each forward call, framework calculates outputs of teacher's nets to use them for passing knowledge.

2.2 Teaching student

Author of paper used Kullback-Leibler regularizer to prevent from fast changing of p's at the beginning of learning phase - it takes to find good Q's transformations. I have simplified it a little bit - described more in examples of use. The used loss function is CrossEntropyLoss summed with $IndFunc$. After each step p's are normalized (2).

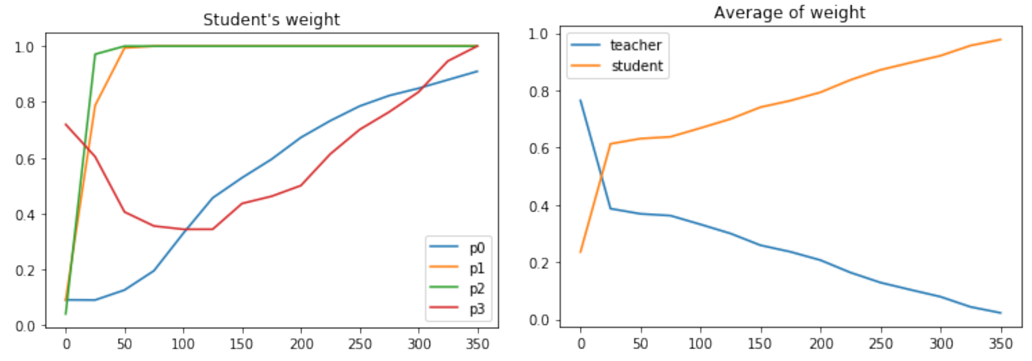

Figure 2: The plot of p-weight in student and teachers nets during cifar 10 to cifar 10 experiments. As we can see at the end of training the net was almost independent.

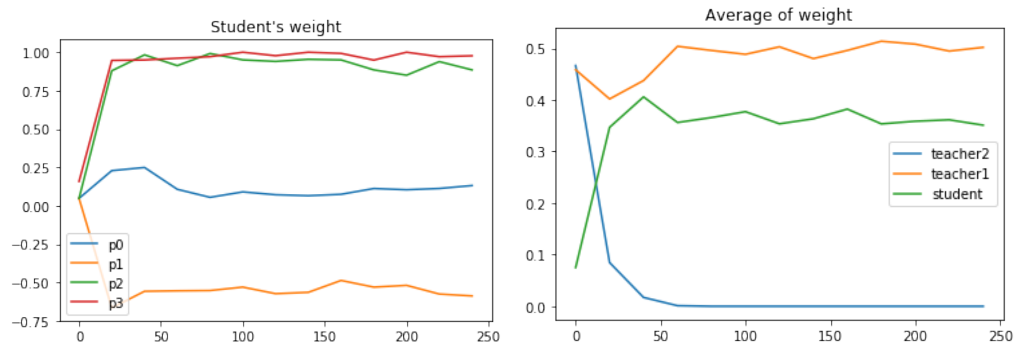

Figure 3: The plot for cifar 10 + cifar 100 to cifar 100 experiment. Negative values of p are rounded to 0.

2.3 Examples of use - cifar-10 to cifar-10

First thing was to use the framework to pass the knowledge from network trained for cifar-10 to the same not trained net. We don't have a problem with choosing connections of the network. As a teacher I have used some simple net which achieves something near 60% of accuracy.

As the only parameters in this example were p's and student net's parameters, training should be easy. On the first attempt (30 epochs) of reproducing only p's for the first and second layer went to 1 - all the others stayed at low values. Student net was untrained to solve independently.

After optimizing the whole framework, the same program was run for 350 epochs (interrupted after 16h). The results were quite interesting. After 350 epochs student become independent (weight for students were: 0.9097, 1.0000e+00, 1.0000e+00, 9.9999e-01) and the net was still training. The accuracy of a student with help of teacher's net is higher (59.5% vs 58%). The knowledge pass takes a lot of computing power (Q was constant here - in normal knowledge pass Q is a great bunch of parameters). The main reason for that difference between paper and my implementation might be other loss function - I am using simple sum as described above (the authors don't write anything about the function they were using). The weights changes are plotted on the figure (2.2).

2.4 Example of use - cifar-10 to cifar-100

The same algorithm was run for learning student to solve cifar-100 based on trained cifar-10 and cifar-100 net. Negative weights are considered as 0 in loss-function. After 240 epochs (10h) the net lost the accuracy and ran to some local minimum making student fully depend. This might be a problem with my implementation and not using the q parameter (it adds a lot of parameters). The weights changes are plotted on the figure (2.3).

3 Conclusion

Knowledge flow method has great cost - it isn't useful for speeding up training, but might increase student's accuracy. It isn't useful for learning simple nets - they might be trained much faster with almost the same accuracy. This method might be great for reinforcement learning which wasn't tested here.

References

[1] Anonymus author. *Knowledge Flow: Improve upon your teachers*, ICLR 2019.

[2] Bartomiej Najdecki, *Knowledge Flow Framework*, `https://colab.research.google.com/drive/1FQvQBpxps10bEcqUoxL6fzz4DVFLu_If`