FitNets: Hints For Thin Deep Nets

Introduction:

This paper extends the idea of Knowledge Distillation to allow the training of a student that is deeper and thinner than the teacher, using not only the softmax outputs but also the intermediate representations learned by the teacher as hints to improve the training process and final performance of the student. Because the student intermediate hidden layer will generally be smaller than the teacher's intermediate hidden layer, additional parameters are introduced to map the student hidden layer to the prediction of the teacher hidden layer.

Review of Knowledge Distillation:

The Knowledge Distillation compression framework trains a student network from the softened output of a teacher network. The idea is to allow the student network to capture not only the information provided by the true labels, but also the finer structure learned by the teacher network. Let T be a teacher network with an output softmax $P_T = softmax(a_T)$ where a_T is the vector of teacher pre-softmax activations. Let S be a student network with parameters W_S and output probability $P_S = softmax(a_S)$ where a_S is the student's pre-softmax output. The student network will be trained such that its output P_S is similar to the teacher's output P_T, as well as to the true labels y_{true}. Since P_T might be very close to the one hot code representation of the sample's true label, a relaxation $\tau > 1$ is introduced to soften the signal arising from the output of the teacher network, and thus, provide more information during training.

$$P_T^\tau = softmax(\frac{a_T}{\tau}), \quad P_S^\tau = softmax(\frac{a_S}{\tau})$$

The student network is then trained to optimize the following loss function:

$$\mathcal{L}_{KD}(W_S) = \mathcal{H}(y_{true}, P_S) + \lambda \mathcal{H}(P_T^\tau, P_S^\tau) \text{... Eq. (2)}$$

where $\mathcal{H}$ refers to cross-entropy loss and λ is a tunable parameter to balance both cross-entropies.

Hint-based training:

In order to help the training of deep FitNets (deeper than their teacher), the authors introduce hints from the teacher network. A hint is defined as the output of a teacher's hidden layer responsible for guiding the student's learning process. Analogously, a hidden layer of the FitNet is chosen as the guided layer, to learn from the teacher's hint layer. The goal is, the guided layer should be able to predict the output of the hint layer.

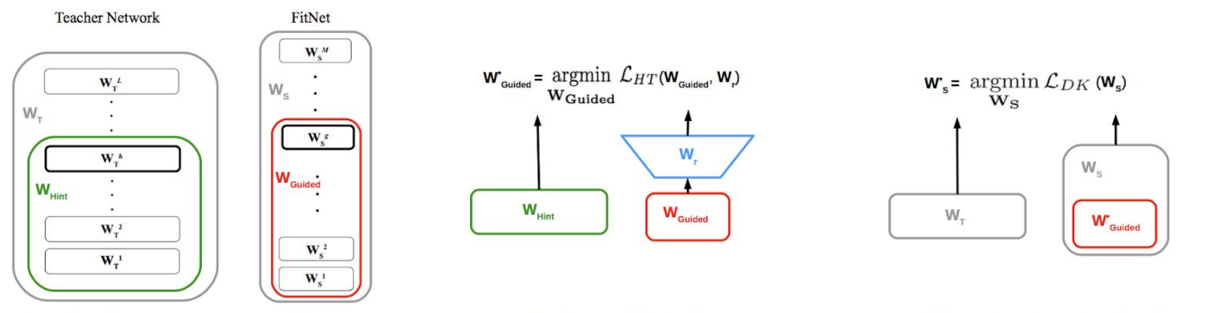
Given that the teacher network will usually be wider than the FitNet, the selected hint layer may have more outputs than the guided layer. For that reason, a regressor is added to the guided layer, whose output matches the size of the hint layer. Then, the FitNet parameters are trained from the first layer up to the guided layer as well as the regressor parameters by minimizing the following loss function:

$$\mathcal{L}_{HT}(W_{Guided}, W_r) = \frac{1}{2} \left\| u_h(x; W_{Hint}) - r(v_g(x, W_{Guided}); W_r) \right\|^2 \text{... Eq. (3)}$$

where u_h and v_g are the teacher/student deep nested functions up to their respective hint/guided layers with parameters W_{Hint} and W_{Guided}, r is the regressor function on top of the guided layer with parameters W_r.

Nevertheless, using a fully connected regressor increases the number of parameters and the memory consumption dramatically. Let $N_{h,1} \times N_{h,2}$ and O_h be the teacher hint's spatial size and number of channels, respectively. Similarly, let $N_{g,1} \times N_{g,2}$ and O_g be the FitNet guided layer's spatial size and number of channels. Then, the number of parameters in the weight matrix of a fully connected regressor is $N_{h,1} \times N_{h,2} \times O_h \times N_{g,1} \times N_{g,2} \times O_g$. To mitigate this limitation, the authors use a convolutional regressor instead. The number of parameters in the weight matrix of a convolutional regressor is $k_1 \times k_2 \times O_h \times O_g$ where $k_i = N_{h,i} + 1 - N_{g,i}$.

FitNet Stage-Wise Training:
The proposed FitNet model is trained in a stage-wise fashion following the student/teacher paradigm:
Step 1: Start with a trained teacher network and a randomly initialized FitNet, add a regressor parameterized by W_r on top of the FitNet guided layer and train the FitNet parameters W_{Guided} up to guided layer to minimise Eq. (3)
Step 2: Finally, from the pre-trained parameters, the parameters of whole FitNet W_S are trained to minimize Eq. (2)



(a) Teacher and Student Networks (b) Hints Training (c) Knowledge Distillation