MLCV2017: Multi-Layer Knowledge Transfer for Neural Networks

Lennard Kiehl
lennard.kiehl@gmail.com

Roman Remme
roman.remme@gmx.de

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous CVPR abstracts to get a feel for style and length.

1. Introduction

The idea of transferring knowledge between different architectures of neural networks, specifically from bigger models to smaller models, has been introduced in [1]. Part of the motivation for this process called distilling is to create a smaller model, which is faster at runtime, with the same knowledge as the bigger model. Distilling works by introducing an additional term to the loss function that links the last layers of both networks by calculating the cross entropy between them. The softmax has an added temperature dependency

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{1}$$

with logits z_i, class probabilities p_i and temperature T, which is normally set to 1. Using a higher value for T produces a softer probability distribution. It was shown in [1] that this alone serves as a very good knowledge transfer tool. We want to extend on this idea and also add links between intermediate layers. The popular VGG-16 model introduced in [4] serves as the bigger model and the goal is to distill each group of convolutional layers into only one convolutional layer, for an overview of the architectures see Table 1.1. We investigate how training hyper-parameters influence the process of successfully distilling knowledge.

1.1. Models

For our experiments we use VGG-16 [4] as the big model. For the small model all stacks of convolutional layers have been replaced by one single convolutional layer (see Table 1.1) and the number of fully connected layers was reduced by one. The similarity between the models is by design and makes it possible to have a maximum of 6 separate links while doing the knowledge transfer. To get the knowledge we want to transfer in the first place, the big model is trained on CIFAR10 [2] with the hyper-parameters shown in Table 2. The convolutional layers of the big model are initialized with pre-trained weights on ImageNet [3] while the fully connected layers are initialized randomly. The small model had to be trained from scratch as it is an uncommon architecture. The accuracies in Table 2 serve as our baseline and we expect the accuracy of the small model after distilling to be somewhere between these two values.

1.2. Loss

The loss function for transferring knowledge into the small model is a weighted sum of three terms:

- "hard" loss: cross-entropy between output and correct label at temperature $T = 1$.
- "soft" loss: cross-entropy between output and prediction of big model at temperature T
- "intermediate" loss: sum of MSE between linked intermediate layers

INSERT FORMULAS HERE
The third term is the new part of our approach. A theoretical advantage is that training times should be reduced, as gradients do not have to be propagated through the whole network to reach the first layers. Typical factors in the weighted sum are

$$\text{loss} = 0.05 \cdot \text{hard} + 0.6 \cdot \text{soft} + 0.35 \cdot \text{intermediate}. \tag{2}$$

1.3. Dataset

We use CIFAR-10 [2] as the dataset to train both models on for all experiments. It consists of 50000 training and 10000 test RGB images of size 32×32 pixels. Each image belongs to one of ten classes. To use the standard VGG architecture with these low-resolution images, they are scaled

Network Configurations with linkable layers		
	VGG-16	VGG-7
	input (224×224 RGB image)	
link 1	conv3-64	conv3-64
	conv3-64	
	maxpool 2×2	
link 2	conv3-128	conv3-128
	conv3-128	
	maxpool 2×2	
link 3	conv3-256	conv3-256
	conv3-256	
	conv3-256	
	maxpool 2×2	
link 4	conv3-512	conv3-512
	conv3-512	
	conv3-512	
	maxpool 2×2	
link 5	conv3-512	conv3-512
	conv3-512	
	conv3-512	
	maxpool 2×2	
link 6	FC-4096	FC-4096
	FC-4096	
	FC-10	FC-10
	softmax	

Table 1. **Network configurations.** The convolutional layers are denoted as conv(*kernel size*)-(*number of channels)* and the fully connected layers as FC-(*number of output channels*). ReLu units are omitted for brevity. The leftmost column gives the links between both networks that are added to the loss function.

	small model	big model
batchsize	40	40
momentum	0.9	0.9
weight decay	0.00002	0.00002
init learning rate (LR)	0.004	0.004
epochs between LR decay	10	10
epochs	25	25
train accuracy	84.8%	84.8%
test accuracy	84.8%	84.8%

Table 2. **Baseline Training.** Both network architectures were trained with the given parameters to have a baseline to compare our transfer training to. The conv. layers of the big model had pretrained weights while the small model was trained from scratch.

up to 224×224 pixels. Each image is preprocessed by subtracting the mean RGB value, computed on the training set, from each pixel.

Temperature	Test set accuracy
0.6	76.3 %
1	76.5 %
1.5	77.0 %
2	**77.4** %
2.5	76.7 %
3	77.2 %
5	73.1 %
10	64.4 %
40	67.3 %

Table 3. **Last layer transfer results.** bla bla

Linked layers	β	Test set accuracy
1	10	78.3%
2	10	80.3%
3	10	83.0%
4	10	85.6%
5	10	**87.9**%
6	10	86.1%
3, 4	10	84.8%
2, 3, 4, 5	10	87.0%
2, 3, 4, 5	40	**87.9**%
2, 3, 4, 5, 6	10	78.5%
1, 2, 3, 4, 5, 6	10	81.2%

Table 4. **Intermediate Layers transfer results.** used last layer with temperature 2 and $\alpha = 10$. MEINE AUSWAHL SIEHT HIER ZIEMLICH DUMM AUS.. VIELLEICHT WAS WEGLASSEN

2. Experiments

We used stochastic gradient descent with momentum 0.9 (CITATION NEEDED?) as an optimizer. We started with a learning rate of 0.004 and let it decay by a factor of 10 every 10 epochs for 25 epochs. This way the accuracy stopped changing significantly prior to the drops in learning rate.

2.1. Temperature of soft loss

Before adding the ”intermediate” loss, we trained the model only using the transfer method from [1]. The relative weigth of the ”soft” loss was chosen to be ten times that of the ”hard” loss. We found the best temperature T to be approximately 2, see Table 3. This value was also used for all following experiments. The corresponding test set accuracy is 77.4%, improving the small model baseline only slightly, by X%.

2.2. Linking intermediate layers

the relative weight of the ”intermediate” loss was chosen to be identical to that of the ”soft” loss. When connecting multiple layers, the losses of the individual links was averaged. Table 4 shows an overview of the link configurations we used, and the corresponding accuracies. First, we used one link at a time. Link 5, the last link in the convolutional part of the network, gave the best test set accuracy of 87.9%. Using multiple intermediate layers yielded at best equally good results. But since far from all possibilities were explored, it is possible that further improvements are possible with the right choice of layers to link.

2.3. Evolution of loss contributions

CANT SAY ANYTHING ABOUT SOFT LOSS WITHOUT FURTHER EXPERIMENT.. ”hard” loss drops fast on train set, but not on test set (duh..) ”intermediate” loss: fast drop in first epochs, then converges to constant value. remarkable: almost identical on train and test set –¿ good regularization (would be interesting to try on very small train set..)

3. Discussion

This is the discussion. We are great!

- good results, intermediate much better than only last
- choice of hyperparameters partially arbitrary: relative weights of the losses, regularization missing, choice of layers to use for transfer
- longer training could lead to further improvements, intermediate loss did not stop declining (though quite slow $\rightarrow$ limit for us)
- decent results are achieved much faster (fewer epochs) compared to hard loss only. reason: ”shorter way” to first layers
- potential advantages on small training sets
- overall: good initial results, further investigation necessary (other datasets, architectures, better tuning of hyperparameters)

References

[1] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[2] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.