Do Deep Nets Really Need to be Deep?

Lei Jimmy Ba, Rich Caruana, U of Toronto, Microsoft, NIPS 2014, (cited 53times)

1. Empirical work shows that it is difficult to train shallow nets to be as accurate as deep nets (for example, vision tasks, 150 layers)
2. We empirically demonstrate that shallow feed-forward nets can learn the complex functions previously learned by deep nets and achieve accuracies previously only achievable with deep models. We do this by first training a state-of-the-art deep model, and then training a shallow model to mimic the deep model (model compression).
3. We are not able to train these shallow nets to be as accurate as the deep nets when the shallow nets are trained directly on the original labeled training data.
4. If a shallow net with the same number of parameters as a deep net can learn to mimic a deep net with high fidelity, then it is clear that the function learned by that deep net does not really have to be deep.

Model compression works by passing unlabeled data through the large, accurate model to collect *the scores* produced by that model. This synthetically labeled data is then used to train the smaller mimic model. The mimic model *is not trained on the original labels*—it is trained to learn the function that was learned by the larger model. If the compressed model learns to mimic the large model perfectly it makes exactly the same predictions and mistakes as the complex model.

In the deep model, the output is given by the softmax layer: $p_k = e^{z_k} / \sum_j e^{z_j}$.
The shallow model is trained using logits, i.e. values z_k before the softmax activation. The learning objective is given as a regression problem:

$$\mathcal{L}(W, \beta) = \frac{1}{2T} \sum_t ||g(x^{(t)}; W, \beta) - z^{(t)}||_2^2,$$

where W is the weight matrix between input features x and hidden layer, β is the weights from hidden to output units, g(x(t);W,β) = βf(Wx(t)) is the model prediction on the t-th training data point and f (·) is the non-linear activation of the hidden units. The parameters W and β are updated using standard error back-propagation algorithm and stochastic gradient descent with momentum.

5. Model compression works best when the unlabeled set is very large, and when the unlabeled samples do not fall on train points where the deep model is likely to have overfit.

Speech recognition:

	Architecture	# Param.	# Hidden units	PER
SNN-8k	8k + dropout trained on original data	∼12M	∼8k	23.1%
SNN-50k	50k + dropout trained on original data	∼100M	∼50k	23.0%
SNN-400k	250L-400k + dropout trained on original data	∼180M	∼400k	23.6%
DNN	2k-2k-2k + dropout trained on original data	∼12M	∼6k	21.9%
CNN	c-p-2k-2k-2k + dropout trained on original data	∼13M	∼10k	**19.5%**
ECNN	ensemble of 9 CNNs	∼125M	∼90k	**18.5%**
SNN-MIMIC-8k	250L-8k no convolution or pooling layers	∼12M	∼8k	**21.6%**
SNN-MIMIC-400k	250L-400k no convolution or pooling layers	∼180M	∼400k	**20.0%**

Table 1: Comparison of shallow and deep models: phone error rate (PER) on TIMIT core test set.

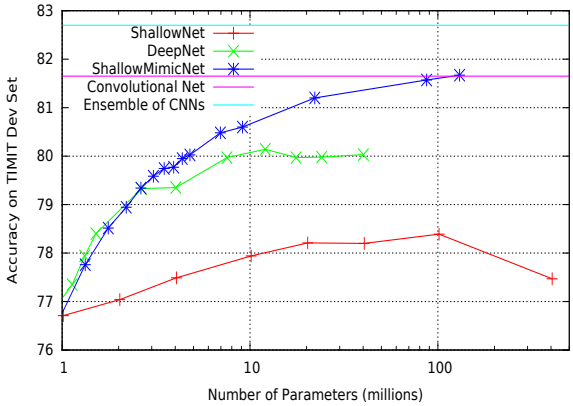

Image classification:

In preliminary experiments we observed that non-convolutional nets do not perform well on CIFAR-10, no matter what their depth. We allow our shallow models to benefit from convolution while keeping the models as shallow as possible.

Distilling the Knowledge in a Neural Network,
Geoffrey Hinton, Oriol Vinyals, Jeff Dean, Google Inc., arXiv2015 (cited 54times)

1. The relative probabilities of incorrect answers tell us a lot about how the complex model tends to generalize. An image of a BMW, for example, may only have a very small chance of being mistaken for a garbage truck, but that mistake is still many times more probable than mistaking it for a carrot.

2. It is generally accepted that the objective function used for training should reflect the true objective of the user as closely as possible. Despite this, models are usually trained to optimize performance on the training data when the real objective is to generalize well to new data. When we are distilling the knowledge from a large model into a small one, we can train the small model to generalize in the same way as the large model.

3. **Distillation idea** is to raise the temperature of the final softmax until the complex model produces a suitably soft set of targets. We then use the same high temperature when training the small model to match these soft targets, and after it has been trained it uses a temperature of 1.

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

   T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes. In practice, when the distilled model is much too small to capture all of the knowledge in the complex model, intermediate temperatures work best.

4. Matching logits is a special case of distillation.