# Distilling the Knowledge in a Neural Network

## 1 Citation

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

`https://arxiv.org/pdf/1503.02531.pdf`

## 2 Abstract

Ensemble neural nets are expensive to deploy, so we distill the knowledge of an ensemble into a single model by training this model to match a combination of the true labels and the ensemble output. We also show how to train specialist models that focus on classes that a generalist model gets incorrect.

## 3 Introduction

During training, we usually train a large, cumbersome model that gets high accuracy. Such a model may be too compute-intensive to deploy to customers. Such a model produces a softmax probability distribution over classes that is quite informative. For example, when seeing a picture of a BMW, a model might give higher probability to ”garbage truck” than ”carrot”, indicating that a garbage truck is more similar to a BMW than a carrot is.

To distill the cumbersome model’s knowledge to the small model, we can use the former’s output as soft targets for the latter. Or even better, we can combine the soft target with the true hard target from the ground truth. For an ensemble, we can compute the geometric mean or arithmetic mean of their outputs to create the soft targets. We use a high temperature late in the training of the cumbersome model and during distillation to ensure that the targets are softened more (i.e. the probability distribution is pushed towards the uniform distribution).

## 4 Distillation

A neural net converts logit $z_i$ into probability $q_i$ with a softmax with temperature $T$ (higher means softer probabilities):

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

We train the cumbersome model with a low temperature initially and then with a high temprature later in training. This high temperature is used during the training of the small model as well. Our loss function is a weighted sum of cross entropy loss of ground truth label and cross entropy loss of the high temperature cumbersome model output. The former is given weight $T^2$ while the latter is given weight 1.

## 5 Preliminary Experiments on MNIST

We train a 2 hidden layer, ReLU activated neural net strongly regularized with dropout and weight constraints on MNIST. Images were jittered to augment the dataset. This model made 67 test errors. We trained a small model that made 146 test errors. Then, we further trained the small model with distillation, and it only made 74 test errors. We found that the smaller the small model, the lower the temperature should be. We also tried training the small model on a dataset that does not include the digit 3. Then, when we distilled, the small model does a good job at recognizing 3 even though it never saw it. It learned about 3 from the cumbersome model's output probability distribution.

## 6 Experiments on Speech Recognition

Speech recognition neural nets use 26 frames of 40 Mel-scaled filter bank coefficients with a 10ms advance per frame and predict a probability distribution over tied triphone states. This gets fed into an HMM that figures out the most likely tied triphone state sequence given the neural net outputs and a language model.

We make a neural net with 8 hidden layers and 14K tied triphone states. It was an old version of Android voice search. We train 10 models in this architecture with different random initializations (showing them different datasets did not help much). Then, we distilled this information into a small model found that 80% of the ensemble's knowledge was transferred to the small model.

## 7 Training Ensembles of Specialists on Very Big Datasets

Distilled models are almost as good as ensembles, but are much less compute-intensive. However, it might be too expensive to train an ensemble in the first place. To avoid this problem, we can use specialist models as described below. Our dataset is JFT, an internal Google dataset with 100M images. Training a model on this dataset (using asynchronous SGD with multiple machines) takes 6 months - training an ensemble would take years.

Thus, we train one generalist model. Then, we identify subsets of classes where the generalist struggles and train a specialist model (initialized with the weights of the generalist) for each one. The specialist receives training half its training examples from the subset classes and the other half are random training examples (we scale the logits to correct this oversampling at test time). It aims to classify images into the correct subset class or into a general "dustbin" class.

To identify the subsets of confusable classes, we simply apply k-means clustering on the columns of our covariance matrix.

Here's how we ensemble the generalist model and specialist model at test time. First, we compute the top $n = 1$ classes (call this set $k$) for an input image $\mathbf{x}$ using the generalist model. $k$ intersects with some of the specialist model class subsets (call this set of specialist models $A_k$). Then, denoting $\mathbf{p}^m$ and $\mathbf{p}^g$ as the output class distribution of model $m$ and the generalist model, respectively, we find the $\mathbf{q}$ that minimizes:

$$KL(\mathbf{p}^g, \mathbf{q}) + \sum_{m \in A_k} KL(\mathbf{p}^m, \mathbf{q}) \tag{2}$$

where the dustbin class is the sum of probabilities over all non-subset classes in the case of specialist models.

These specialists are trained in parallel and only require a few days to train. They provide an improvement in test error compared to the generalist model alone. We can also distill this generalist-specialist ensemble.

## 8 Soft Targets as Regularizers

Recall that we train the small model on the training set first before distillation. If we train it on just 3% of the training set, it severely overfits. However, if we train it on just 3% of the training set, but use the cumbersome model's output as a soft target, then it actually does quite well. The soft targets carry a lot more information than the ground truth label and act as a regularizer.

We can also replace the specialist "dustbin" class with all of the non-subset classes and use soft targets from the generalist to train the specialist. We are exploring this.

## 9 Relationship to Mixtures of Experts

A mixture of experts uses a gating network to figure out which expert should handle an example. During training, the gating network learns how to pick the best expert and the experts learn to classify in their area of expertise. Notice that training this model is not parallelizable. Our generalist-specialists ensemble, however, is parallelizable.

## 10 Discussion

Distilling can transfer's an ensemble's knowledge into a small, less compute-intensive, model. We also show how a generalist-specialists ensemble can be much less compute-intensive to train than a regular ensemble.