

Distilling the Knowledge in a Neural Network

• [[📄 1503.02531v1.pdf]]

• Basic Idea

- Can we distill the information learned by a large gigantic model (or an ensemble of models) into a small model?
 - We don't know what is the correct way to generalise the knowledge. 🟡 **P2** When we are distilling the knowledge from a large model into a small one, however, we can train the small model to generalize in the same way as the large model. If the cumbersome model generalizes well because, for example, it is the average of a large ensemble of different models, a small model trained to generalize in the same way will typically do much better on test data than a small model that is trained in the normal way on the same training set as was used to train the ensemble.
 - This can be by training the smaller network to predict the soft targets which are the probability distributions predicted/generated by the large network. In the case of ensemble of models, arithmetic or geometric mean of the individual prediction distribution could be used as a soft target.
 - This soft target can also act as a regularizer. 🟡 **P2** When the soft targets have high entropy, they provide much more information per training case than hard targets and much less variance in the gradient between training cases, so the small model can often be trained on much less data than the original cumbersome model and using a much higher learning rate.
 - Advantage of soft targets over hard targets (which are used to train the original large network). 🟡 **P2** The relative probabilities of incorrect answers tell us a lot about how the cumbersome 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.
- *Problem of using probabilities for transferring knowledge:*
 - The probabilities assigned to the incorrect answers can become very small and therefore, may not contribute anything to the loss. For example, in MNIST classification, a good model may assign some version of digit 2 to digit 3 with probability 10^{-6} and to digit 7 with probability 10^{-9} . These probabilities though small, carry information about which version of 2 are similar to the digit 3 and 7.
 - To circumvent this, Caruana et al. used l_2 loss between the logits rather than probabilities.
 - Author propose to use temperature dependent softmax activation for generating softer probability distribution over classes.
- 🟡 **P2** For this transfer stage, we could use the same training set or a separate "transfer" set.
- 🟡 **P2** The transfer set that is used to train the small model could consist entirely of unlabeled data [1] or we could use the original training set.
- During transfer, adding a small term to enforce the smaller model to predict the true target (hard target) along with the soft target works well.

• Distillation

- *Notation:*
 - T : Temperature
 - z_i : Logits
 - q_i : Probabilities produced by applying softmax over z_i i.e.,

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

- Usually $T = 1$ in a usual softmax setting. However, 🟡 **P3** Using a higher value for T produces a softer probability distribution over classes. To see this consider the limit when $T \rightarrow \infty$, then $q_i = \frac{1}{N}$ where N is the number of classes, which is the softest distribution possible over all the classes.
- While training a cumbersome model, a high value of temperature is used to produce softer distribution. Then, 🟡 **P3** The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1.
- *Loss function for training distilled model includes:*
 - Cross-entropy with soft target. Softmax temperature is same between the cumbersome and distilled model.
 - Cross-entropy with correct labels. Computed using logits of distilled model. Softmax temperature set to 1.
 - The second term with correct labels is weighed significantly less.
 - 🟡 **P3** Since the magnitudes of the gradients produced by the soft targets scale as $1/T^2$ it is important to multiply them by T^2 when using both hard and soft targets. This ensures

that the relative contributions of the hard and soft targets remain roughly unchanged if the temperature used for distillation is changed while experimenting with meta-parameters.

- Matching logits is a special case of distillation

- **Notation:**

- C : Cross-entropy loss with soft target
- z_i : Logits of the distilled model
- v_i : Logits of the cumbersome model
- p_i : Soft target probabilities produced by the cumbersome model
- q_i : Probabilities produced by the distilled model by applying softmax function over the logits z_i
- T : Temperature at which transfer training is done

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T} \left(\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} - \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)} \right)$$

- **Proof:**

- Note that $C = -\sum_j p_j \log q_j$

$$\begin{aligned} \frac{\partial C}{\partial z_i} &= - \left[\sum_{j \neq i} p_j \frac{\partial \log q_j}{\partial z_i} + p_i \frac{\partial \log q_i}{\partial z_i} \right] \\ &= - \left[\sum_{j \neq i} p_j \frac{\partial \log q_j}{\partial q_j} \cdot \frac{\partial q_j}{\partial z_i} + p_i \frac{\partial \log q_i}{\partial q_i} \cdot \frac{\partial q_i}{\partial z_i} \right] \\ &= - \left[\sum_{j \neq i} \frac{p_j}{q_j} \frac{\partial q_j}{\partial z_i} + \frac{p_i}{q_i} \frac{\partial q_i}{\partial z_i} \right] \end{aligned}$$

- Now, $q_k = \frac{\exp(z_k/T)}{\sum_j \exp(z_j/T)}$, therefore,

$$\frac{\partial q_k}{\partial z_k} = \begin{cases} \frac{1}{T}(q_k - q_k^2) & \text{if } k = i \\ -\frac{q_k q_i}{T} & \text{if } k \neq i \end{cases}$$

- Substituting $\frac{\partial q_k}{\partial z_k}$, we get

$$\begin{aligned} \frac{\partial C}{\partial z_i} &= \sum_{j \neq i} \frac{p_j}{q_j} \frac{q_j q_i}{T} - \frac{p_i}{q_i} \frac{(q_i - q_i^2)}{T} \\ &= \frac{1}{T} \left[q_i \sum_{j \neq i} p_j + p_i q_i - p_i \right] \\ &= \frac{1}{T} (q_i (1 - p_i) + p_i q_i - p_i) \\ &= \frac{1}{T} (q_i - p_i) \end{aligned}$$

- If the temperature is high compared with the magnitude of logits, we can approximate:



$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T} \left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T} \right)$$

This is true because the Taylor's expansion for $e^x = 1 + x + x^2 + \dots$ and if x is very small then $e^x \approx 1 + x$.

- Now, if the logits have been zero-meaned for each transfer case (each sample in the transfer set) individually, i.e., $\sum_j z_j = \sum_j v_j = 0$, then

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2} (z_i - v_i)$$

- Therefore, in the high temperature limit, distillation is equivalent to minimizing $\frac{1}{2}(z_i - v_i)^2$ (this is just the integral of the $\frac{\partial C}{\partial z_i}$), provided the logits are zero-meaned separately for each transfer case.

-  **P3** At lower temperatures, distillation pays much less attention to matching logits that are much more negative than the average. This is potentially advantageous because these logits are almost completely unconstrained by the cost function used for training the cumbersome model so they could be very noisy. On the other hand, the very negative logits may convey useful information about the knowledge acquired by the cumbersome model.  Which of these effects dominates is an empirical question. We show that when the distilled model is much too small to capture all of the knowledge in the cumbersome model,

intermediate temperatures work best which strongly suggests that ignoring the large negative logits can be helpful.

- **Using Soft-targets to train ensembles of specialists for very big datasets**



- *Motivation*

- Large ensembles have high computational requirements during test times which can be circumvented through distillation.
 - However, if each model in the ensemble is a neural network trained on a humungous dataset, training each ensemble can be too costly.
 - *Idea*: Train a "generalist" model on all training data and multiple "specialist" models on a subset of classes which are highly confusable.
 - *Problem*: Specialists with focus on fine-grained distinction can overfit easily because the dataset for each fine-grained class becomes very small.
 - This problem can be prevented using soft targets (More on this later).




- *Notation*:

- S^m : Set of classes which are predicted by the specialist model m .

- *Tricks of the trade/training details*:

-  **P6** The softmax of this type of specialist can be made much smaller by combining all of the classes it does not care about into a single dustbin class.
 - To reduce overfitting the weights of each of the specialists model is initialised with the weights of the generalist model.
 - *Training Data for Specialist model*: 50% comes from the classes belonging to the specialist set and 50% is randomly sampled from rest of the training set.  **P6** After training, we can correct for the biased training set by incrementing the logit of the dustbin class by the log of the proportion by which the specialist class is oversampled.
 - *Identifying subsets of classes for training specialist*
 - Highly confusable classes can easily be identified using confusion matrix. However, it has following con:
 - It requires true labels to identify such clusters. Since, the number of samples in the fine-grained classes may not be significant we can't be highly be sure of the identified clusters. Therefore, if we have a lot of unlabelled data for the fine-grained categories we can utilise the following approach.
 - Compute the co-variance matrix of the predictions of the generalist model. Then apply clustering algorithm over the columns of the matrix. A prediction from the generalist model will be a vector of size N where N is the number of classes to be predicted by the generalist network. Therefore, co-variance matrix will tell us if there is any correlation between scores of different classes. Once, we have this co-variance matrix, we can think of each column as describing a class using the correlation of that class with other classes.

- *Inference using an ensemble of specialists*:

- Generalist model can be used to determine which specialist models to use and also for those classes for which we don't have specialist models.
 -  **P6** Given an input image x , we do top-one classification in two steps:
 -  **P6** Step 1: For each test case, we find the n most probable classes according to the generalist model. Call this set of classes k . In our experiments, we used $n = 1$.
 -  **P6** Step 2: We then take all the specialist models, m , whose special subset of confusable classes, S^m , has a non-empty intersection with k and call this the active set of specialists A_k (note that this set may be empty). We then find the full probability distribution q over all the classes that minimizes:


 **P6**

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

where KL denotes the KL divergence,

\mathbf{p}^m probability distribution of the specialist model

\mathbf{p}^g probability distribution of the generalist model

- Note that \mathbf{q} is the full distribution i.e., the distribution over all the classes in the training set, where \mathbf{p}^m is the distribution over $S^m \cup \{\text{Dustbin-class}\}$. Therefore, for computing the $KL(\mathbf{p}^m, \mathbf{q})$, all the probabilities in $\mathbf{q} \notin S^m$ are summed up to get the probability of the dustbin class.
 - The above equation  **P7** does not have a general closed form solution, though when all the models produce a single probability for each class the solution is either the arithmetic or geometric mean, depending on whether we use $KL(\mathbf{p}, \mathbf{q})$ or $KL(\mathbf{q}, \mathbf{p})$.

- \mathbf{q} is parameterized using $\text{softmax}(\mathbf{z})$ (with $T = 1$) where \mathbf{z} is the parameter which needs to be determined. This can be done by minimising the above equation using gradient descent.
- Note that this optimization needs to be carried out for every image during inference time. Therefore, this step is separate from training.
- **Food for thought:**
 - How does this way minimizing KL divergence compares to a simple approach, where we just declare the final prediction as the prediction of the specialist model (in case the specialist model exists) else we can use the prediction of the generalist model?
 - I think the reason the simple approach cannot be applied is because [P7](#) the sets of classes for the specialists are not disjoint, we often had multiple specialists covering a particular image class. To obtain overlapping sets of classes, I think they would have used Fuzzy K-means algorithm.

- *Using soft targets to prevent specialists from overfitting:*

- Note that in the above training procedure, no soft targets are used. The specialists are trained on S^m and the dustbin classes with hard targets. However, this exposes the networks to the problem of overfitting.
- [P8](#) This problem cannot be solved by making the specialist a lot smaller because then we lose the very helpful transfer effects we get from modeling all of the non-specialist classes. (Transfer effects are due to initialising the weights of the specialists models with that of the generalist model.)
- *Idea:* Instead of collapsing all of the non-specialists classes into a dustbin class, we can train specialists networks with full softmax over all classes. [P8](#) Our experiment using 3% of the speech data strongly suggests that if a specialist is initialized with the weights of the generalist, we can make it retain nearly all of its knowledge about the non-special classes by training it with soft targets for the non-special classes in addition to training it with hard targets. The soft targets can be provided by the generalist. We are currently exploring this approach.

- Note: [P8](#) We have not yet shown that we can distill the knowledge in the specialists back into the single large net.

- **Relation to Mixture of Experts (MoE)**

- In MoE, a gating network learns to determine which specialist network (expert) should it assign a given sample. The specialist/expert network makes the final prediction.
- The training of MoE (i.e., gating and specialists networks) is done end-to-end. This means the samples on which a local expert is getting trained is adaptive (depends on the gating network as well as other experts) as well as based on the performance of the local expert networks the gating network has to learn which local expert it should assign the sample. Due to this, end-to-end training is difficult.
- On the other hand, the usage of the specialists networks proposed in this paper is parallelizable. The training of specialists and generalist model are independent and hence can be trained independently. Moreover, during inference based on the prediction of the generalist, we can decide which specialists to assign the sample to.
- Read section 7 of the paper for more details.