# KNOWLEDGE DISTILLATION
# FOR IMAGE CLASSIFICATION

Giacomo D'Amicantonio

In this report I explore the topic of Knowledge Distillation, briefly discussing the theoretical background and then talking about its implementation and some experiments I performed.

## DISTILLATION THEORY

The problem addressed by Knowledge Distillation can be inscribed in the Transfer Learning field. Specifically, the authors start from the observation that models are trained to optimize on train data when the actual objective should be to perform well on new data. In theory, to generalize well the model should know a lot of information that is often not available or unclear. But, distilling the knowledge of a big model into a smaller one makes it possible for a small model to generalize as well as or even better than the big one. At the same time a distilled model performs better than the same model trained in the classical way.

The key idea behind KD is that we could train the student model to imitate the teacher model. This is possible using two different loss functions to compute the final loss value of the student:

- Categorical Cross Entropy, which is the classical Cross Entropy function that computes the distance between the predicted label and the actual label.
- Kullback-Leibler Divergence, that measures the difference between the predictions of the smaller model and those of the bigger model

These two loss functions are combined using a parameter called *alpha* which decides how much importance to give to the two losses. In the original paper, the final loss is defined as:

$$loss = \alpha CE(true\ label, student\ prediction) + (1-\alpha)\ CE(teacher\ prediction, student\ prediction)$$

It is equivalent, however, to use KL divergence instead of CE to compute the distance between teacher predictions and student predictions. This is possible because KL divergence is CE plus a constant, which is given by the entropy of the dataset. The parameter $\alpha$ is used to balance the loss, specifying how much importance to give to imitate the teacher model compared to actually predicting the correct class.

An obvious way to transfer learning is to use the class probabilities produced by the big model as *soft target* for the smaller model. This distribution includes a lot of information that is lost once the softmax is applied. The best example is the MNIST dataset, where a 2 much more easily look like a 7 than a 3, which enriches the data structure and is otherwise lost if the model only cares to optimize for the correct label. Using a temperature value on the class probabilities allows us to soften the distribution. The smaller model cannot replicate exactly these results, so we use a small term of classical lost over *hard target* to steer it in the correct direction. This temperature is used only in training phase, while in the evaluation phase is set to 1. This concept is explained by:

$$q_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{Z_j}{T}\right)}$$

where $q_i$ is the probability of a class, computed by dividing the logit of that class $z_i$ by the sum of the probabilities of all classes. The temperature smoothens the distributions of the probabilities, lowering peaks that would be otherwise eclipse the contribute of other classes for a data point. In the paper, the authors claim that matching logits is a special case of distillation. This is true because each logit $z_i$ of the distilled model and each logit $v_i$ of the teacher model contributes to the CE gradient according to:

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

If the temperature is higher than the magnitude of the logits, we can approximate it as:

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

Assuming that for each case the logit has been zero-meaned (normalized) so that

$$\sum_j z_i = \sum_j v_i = 0$$

The equation becomes:

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

This means that the distillation becomes a minimization of the above equation.

At lower temperatures, distillation cares less about logits that are significantly lower than the average and this could be advantageous because these logits are probably ignored by the cost function used to train the teacher and are therefore very noisy, but they could also contain useful information about the dark knowledge acquired by the teacher model, so the temperature cannot be set a priori but must be tuned for each teacher/student system. In a similar way, the parameter $\alpha$ must be tuned for each case. It is relevant to notice that a model that performs very well is not necessarily a good teacher, meaning that it may not contain enough dark knowledge to improve the generalization performances of the student model when compared with the student trained from scratch.

## IMPLEMENTATION

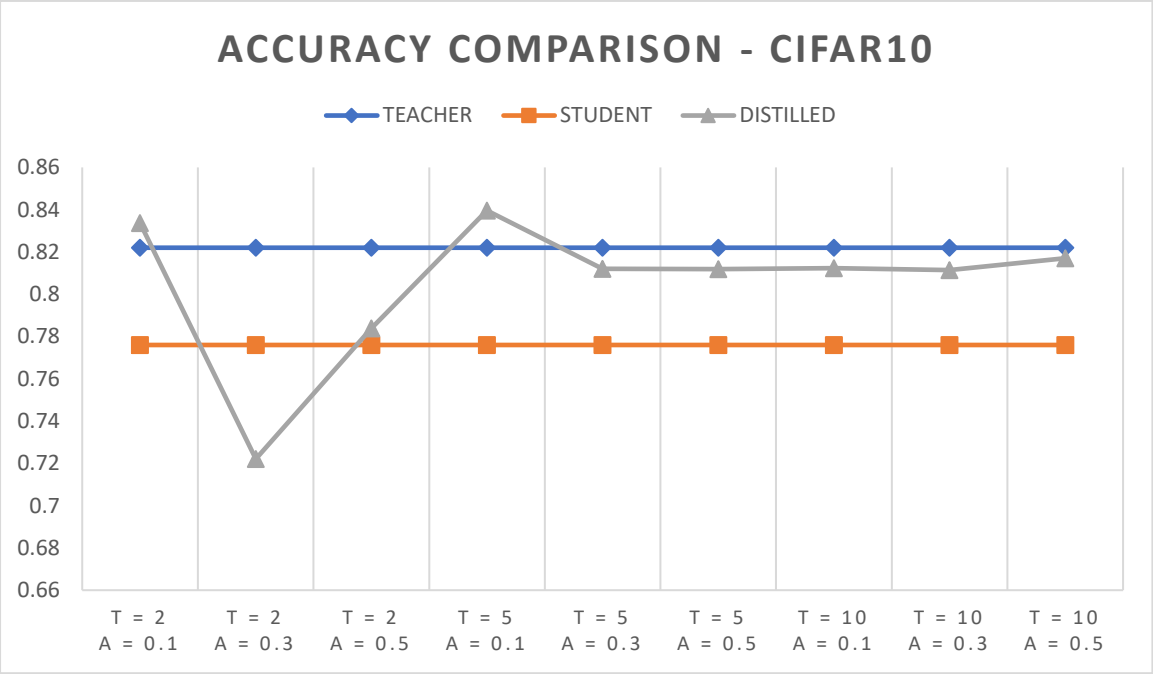
I wrote a Colab notebook to experiment with KD. I used the Keras implementations of ResNet-152V2 and ResNet-50V2 as teacher and student, respectively. I developed a class Distiller that extends Model to perform KD. The dataset I choose for the initial experiments is the CIFAR10, that consists in 50k images grouped in 10 labels. Every model used included an input layer that upscaled the dimensions of the input data from 32x32x3 to 64x64x3. I

found that this was the best compromise between performances in accuracy and in training time, significantly improving the score of the models from the original dimensions.

The teacher was pretrained on ImageNet and was finetuned on CIFAR10 for 10 epochs, with batch size of 128, Cross Entropy as loss function and Adam as optimizer. The teacher reached 82.2% accuracy on the test set.

I then defined two student models, one to be trained in the normal way and the other distilled from the teacher. The student trained normally is initialized in the same way of the teacher, with weights pretrained on ImageNet and is trained for the same epochs, loss function and batch size of the teacher. This student, used as a baseline to estimate the distilled student, reached an accuracy of 77.6%.

In the end, I have distilled a student from the teacher using different combinations of temperature and $\alpha$.

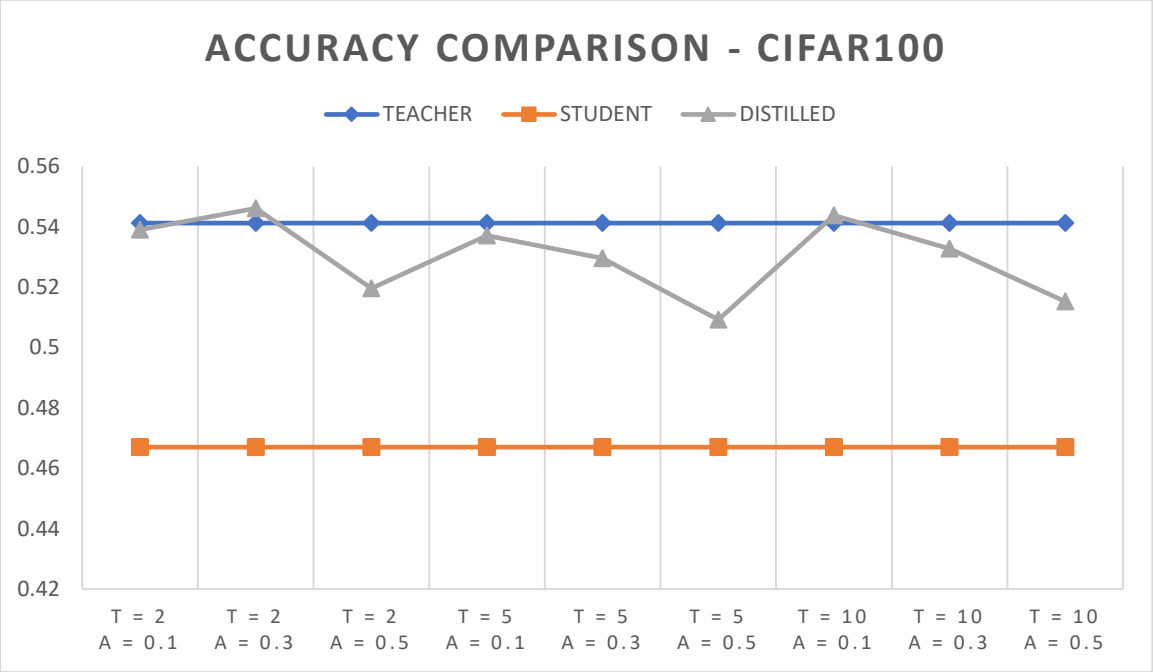



In the plot above are showed the results obtained from the same teacher using different combinations of temperature and $\alpha$. Every distilled model performed better on test data than the model trained from scratch except for one. I have run multiple time the simulation, with different seedings, and the combination of temperature = 2 and $\alpha = 0.3$ always performed worse than the others. On the other hand, the best combination was usually temperature = 2 and $\alpha = 0.1$. In some instance, it performed better with temperature = 5 but never with $\alpha > 0.1$.

This result seems to indicate that, at least for CIFAR10, the smoothness of the distribution of soft targets is not as important as the contribution of the teacher loss on the distillation loss and, in general, on the ability of the distilled model to generalize from training data.

I then tried to understand how the labels in the dataset can influence the distillation by using CIFAR100, the same dataset as CIFAR10 but with 100 label classes instead of 10. The teacher, the student baseline and the distilled student are trained with the same training strategy as for CIFAR10. The teacher achieved an accuracy of 54.12 % on test data while the

student trained from scratch achieved 46.7% of accuracy. In this case it was much more evident how increasing the $\alpha$ parameter results in worst performances in the distilled student. In any case, the distilled model still performed sensibly better than the student trained from scratch and it even performed as well as the teacher for low values of $\alpha$. The temperature did not influence the performance of the distilled model significantly.



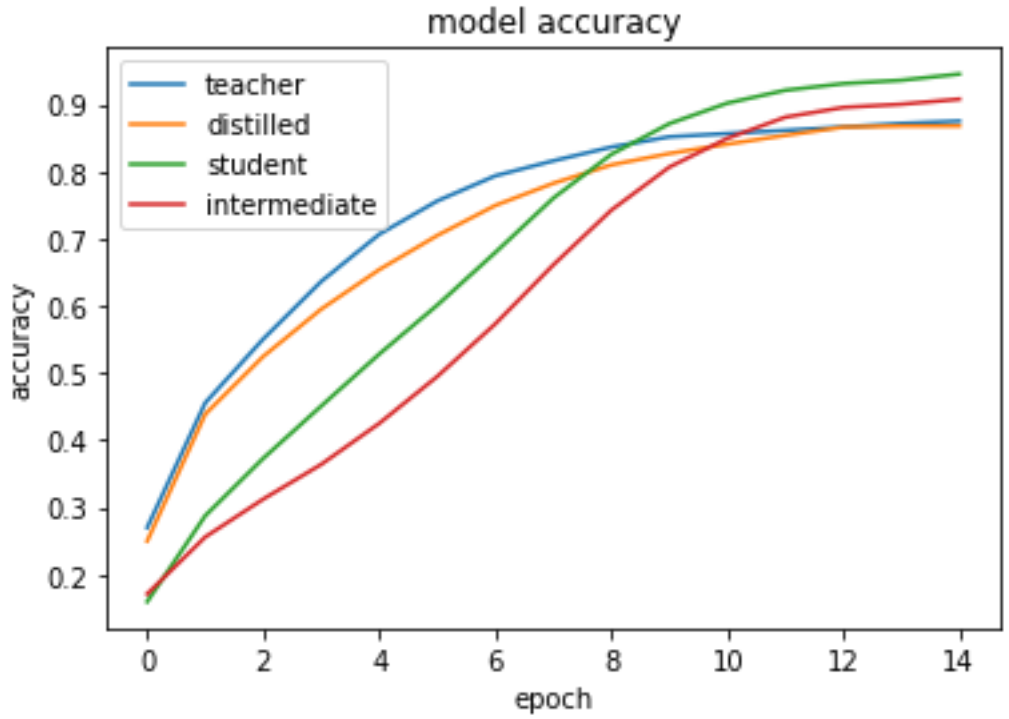

## STAGE-BY-STAGE DISTILLATION

Since every experiment I did showed that the soft labels were much more important to a good performing distilled model than hard labels, I wanted to try if it was possible to distil a small ResNet from a bigger one by training each of its stages to imitate the distribution produced by the same stage of the bigger ResNet. One way to frame this approach would be to consider each stage as a feature extractor. Of course, not having a ground truth for each stage's result is equivalent to performing KD with $\alpha = 0$.

Training each stage required about 70 epochs except for the first one, that only required 10 epochs. After a stage has been trained, its weights were locked so that every subsequent training would not modify them.

For the first two stages, the distilled model was able to imitate the teacher model with an accuracy of more than 95% on training data. The third stage resulted much more difficult to be trained, never achieving more than 80% of accuracy. The last stage would significantly improve the overall performances of the model, achieving more than 90% accuracy. In the end, the last stage of the ResNet was trained to perform the actual classification, never obtaining more than 30% of accuracy on test data.

It is interesting to notice that, on training data, this approach actually performs better than the classical distillation method and the teacher model, as shown in the plot below.




No combinations of hyperparameters provided better results than teacher or student on unseen data.

A similar approach was suggested in An Embarrassingly Simple Approach to Knowledge Distillation, Gao et al., 2018. In there, the authors found that this approach performed better than classical Distillation. The authors used Stochastic Gradient Descent instead of Adam to distil a ResNet-20 from a ResNet-56. I was ultimately unable to replicate their results.