ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Machine Learning for
Computer Vision

IMPROVEMENTS TO KNOWLEDGE DISTILLATION ON DEEP NEURAL NETWORKS

CANDIDATE

Giacomo D'Amicantonio

SUPERVISOR:

Prof. Samuele Salti

CO-SUPERVISOR:

Prof. Lugi Di Stefano

Academic Year 2020/21

2nd Session

SUMMARY

INTRODUCTION

One of the main problems in the field of Artificial Intelligence is the efficiency of neural networks models. For a while, in the past few years, it seemed that most tasks involving such models could simply be solved by designing larger, deeper models and training them on larger datasets for longer time. This approach requires better performing and therefore expensive and energy consuming hardware and will have an increasingly significant environmental impact when those models are deployed at scale. Last year OpenAI developed GPT-3, a Transformer-based model capable of performing multiple Natural Language Processing tasks. Even though it is a surprisingly effective model, it has the big drawback of having 175 billion parameters and training sets of about 570GB of data.

In the field of Computer Vision, we have the same kind of problems on a smaller scale. Even if it is true that language models are often much bigger than the models used in Image Classification or Object Detection, we still need heavy models which require expensive and energy consuming hardware to obtain good results on some datasets or tasks. Ideally, we would like to achieve the same level of performances by training smaller models in less time and using only a fraction of the computational resources required by these cumbersome models. Unfortunately, small models often lack the capability to learn complex relationships and knowledge in the training data as shown by Caruana and its collaborators in Model Compression[1]. Therefore, they show that we cannot avoid training cumbersome models for certain tasks and applications, but we can then transfer the knowledge they extracted into smaller models and deploy those. By doing so we can obtain small models with a reduced gap in performance with respect to the cumbersome ones compared to the same models trained in the standard way. Such models would also be much smaller than the cumbersome ones, allowing us to deploy them in contexts that do not have enough computational power at their disposal.

In 2015 G. Hinton, J. Dean and O. Vinyals presented Knowledge Distillation[2] (KD), a technique that leveraged the logits produced by a big, cumbersome model to guide the training of a smaller model. The two networks were called “Teacher” and “Student” given the analogy between the big model with large knowledge and the small model which has yet to learn everything. They proved that it is possible to extract useful knowledge from the teacher logits and use it to obtain a better performing student when compared with the same model

that learned all by itself. In the past few years, a lot of contributions from different researchers build on top of this basic framework, proposing new types of knowledge that can be used and improving on the knowledge transferring. Such intense research is summarized in a recent survey[3].

This thesis provides an overview of the current state-of-the-art in the field of Knowledge Distillation, analyses some of the most interesting approaches, and builds on them to exploit very confident logits in a more effective way. Furthermore, it provides experimental evidence on the importance of using also smaller logit entries and correcting mistaken predictions from the teacher in the distillation process.

The structure of the thesis is as follows. In Chapter 1, we will review the current state-of-the-art regarding Knowledge Distillation, explaining in detail the types of knowledge that can be extracted from a teacher and how they can be transferred to the student. In Chapter 2, we will present three new proposals to optimise and improve the transfer of Knowledge that, to the best of our knowledge, have not been investigated. In the last Chapter we will present the results of the experiments we ran to test the effectiveness of the above-mentioned proposals, before concluding with a brief recap of our work.

CHAPTER 1 – KNOWLEDGE DISTILLATION

1.1 - VANILLA KNOWELDGE DISTILLATION

Knowledge Distillation leverages the knowledge learned by the teacher model to improve the training of a student by transferring such knowledge to the student model. In the original paper, the authors noted that a lot of what the model learns during its training about the training data is lost in tasks such as image classification because it requires to condensate the acquired knowledge to a single class to make a prediction. Therefore, they proposed to use the logits produced by the teacher instead of the final output. The logits, or pseudolikelihood, are the values that compose the distribution over the classes in the last layer of the teacher, before the SoftMax function is applied.

Such a distribution is subject to the first of the two hyperparameters that Knowledge Distillation employs, the *Temperature*. In fact, the logits distribution produced by a network will have a higher peak in correspondence to the predicted class and smaller values for the other classes. The *Temperature* is used to smoothen such a distribution and to highlight inter-class relations that the model learned. This kind of knowledge was called "Dark Knowledge" by Hinton et al.

Formally, this means that the SoftMax output becomes:

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)} \quad \textbf{(1)}$$

This equation is equivalent to the standard SoftMax for $T = 1$, while for higher values of T the distribution becomes smoother, from which the name "soft targets". The data labels, one-hot encoded in a vector, are called "hard targets".

Even though the soft targets embed significant knowledge, they are not enough to train a network correctly on a labelled dataset by themselves. The authors proposed to mix the proportion of the loss function dependent on soft and hard targets according to another hyperparameter, A*lpha.*

The loss function therefore becomes:

$$\mathcal{L}(x, W) = \alpha * CE\big(y, \sigma(z_s, T = 1)\big) + (1 - \alpha) * KLD\big(\sigma(z_t, T = \tau)\,, \sigma(z_s, T = \tau)\big) * \tau^2 \quad \textbf{(2)}$$

The first term of the sum is the Cross-Entropy loss between the hard targets (y) and the labels predicted by the student ($\sigma(z_s, T = 1)$ is the SoftMax function applied to the logits of the student z_s), while the second term is the Kullback-Leibler Divergence[4], which is a measure of how much information is lost if the student logits distribution, smoothened by T, is used to approximate the smoothened teacher logits distribution.

Algorithm 1: Vanilla Distillation

Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α

For each epoch:

for x_i in x:

teacher logits = teacher(x_i)

student logits = student(x_i)

student pred = SoftMax(student logits)

teacher logits = teacher logits / τ

student logits = student logits / τ

teacher pred = SoftMax(teacher logits)

smoothen student pred = SoftMax(student logits)

loss = $\alpha * KLD$(*teacher pred, smoothen student pred*) + (1-α)*CE(*student pred*, y_i)

It is also important to notice how the soft targets term is multiplied by the squared value of the temperature. This is necessary since smoothing the distributions results in a smoothing of the gradient produced by the soft targets of a factor $\frac{1}{\tau^2}$, which needs to be corrected to avoid an unbalanced contribution despite the alpha factor. The algorithm is summarized in algorithm box 1.

In the same paper, the authors report how a student trained in this setting achieves two relevant objectives:

- Its accuracy on the test sets improves sensibly compared to when the model is trained only with hard targets in the standard way
- The model is capable of avoid overfitting, showing how soft targets also have a regularizing effect on the training

1.1.1 - SUBSEQUENT WORKS

In the following years, multiple new research projects have been focused on this topic. As mentioned earlier, it is undoubtedly very useful to get a small model to perform as well as a cumbersome one. According to a recent survey, the field of Knowledge Distillation is currently split in two main branches of techniques that focus on two different aspects of Knowledge Distillation. The first branch focuses on what kind of knowledge can be distilled, identified in three types of knowledge that can be extracted from a teacher. The second branch focuses on how to distil such a knowledge.

In the following sections, I will describe them providing an overview of a few state-of-the-art approaches for each one.

1.2 - TYPES OF KNOWLEDGE

1.2.1 – RESPONSE-BASED KNOWLEDGE

Response Based Distillation is the first branch of Knowledge Distillation approaches that followed directly from the Vanilla Distillation. In this batch of approaches, the main idea is to have the student imitate the logits of the last layer of the teacher. This general idea is formally expressed as:

$$\mathcal{L}_{ResD}(z_t, z_s) = \mathcal{L}_R(z_t, z_s) \quad \textbf{(3)}$$

Where $\mathcal{L}_{ResD}$ is the Response Based Distillation Loss and z_t and z_s are the logits of the teacher and the student. $\mathcal{L}_R(\cdot)$ is the divergence loss between logits. In general, such a setting can be thought as Fig. 1

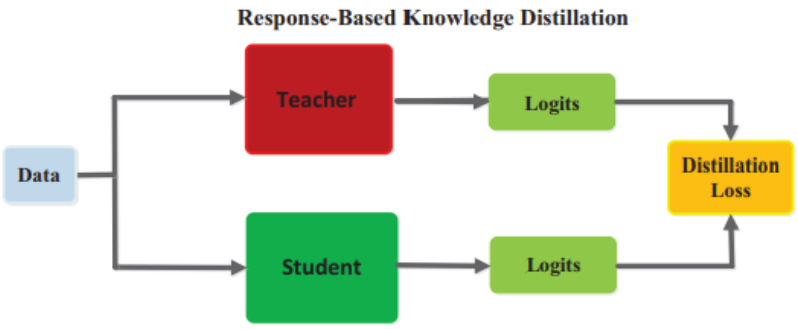

FIGURE 1 GENERAL FRAMEWORK FOR RESPONSE-BASED KNOWLEDGE DISTILLATION

As mentioned above, this setting has been proved to bring regularizing effects to the training similarly to what is done by label smoothing and other regularizes. Various approaches have been developed to exploit this effect of Knowledge Distillation. For example, in Transferring Knowledge to Smaller Network With Class-Distance Loss[5] they used an *l2* norm as a training objective for the student to minimize the distance between the feature vectors that are fed to the last dense layer of the teacher and the student. The training of the teacher is also slightly modified, adding an additional component to the Cross Entropy loss. Here, a multiplier is used to enhance the l2 norm of the vector produced for the current class and the mean vector of the other classes minus a minimal threshold finetuned for the current class.

Response Based approaches have been applied to a variety of tasks, from object detection to semantic localization in Computer Vision. It has been proved however that it has one significant drawback: it completely ignores the knowledge acquired by the teacher in its earlier layers, as showed in Fitnets: Hints for thin deep nets[6].

1.2.2 – FEATURE-BASED KNOWLEDGE

The methods that exploit the knowledge in those layers have been grouped in the Feature Based group. Here, the focus is on the intermediate representations learned by a network in its earlier layers. In Fitnets, the layers from which this knowledge is extracted have been called “Hint Layers”, a pun on the verb “to hint” and the name of the main author of Knowledge Distillation, Geoffrey Hinton. A similar layer is chosen in the student network and is called “guided layer”. The choice of the layers can heavily influence the training given the regularizing effect it has. Choosing deeper layers results in a loss of flexibility in how the student will learn in shallower layers, while shallower layers may not have feature maps rich enough in knowledge. For this reason, the layers are usually chosen in the middle of the networks.

The outputs of these intermediate layers are called feature maps. Given that usually the teacher network is bigger than the student, its feature maps will be bigger than the feature maps produce by the guided layer. The original proposal used a regressor to make the guided layer match the dimensions of the hint layer, while more recent approaches focused on more sophisticated ways face the task.

In Paying more Attention to Attention[7], the feature map of the hint layer is flattened to obtain a 2D tensor defined over the spatial dimensions. The absolute values of the elements of that tensor are used to compute an attention map across the channel dimension. The authors proposed to do this with three different functions:

- Sum of absolute values: $F_{sum}(A) = \sum_{i=1}^{C}|A_i|$
- Sum of absolute values raised to the power of p: $F_{sum}^{p}(A) = \sum_{i=1}^{C}|A_i|$
- Max of absolute values raised to the power of p: $F_{max}^{p}(A) = max_{i=1,C}|A_i|^p$

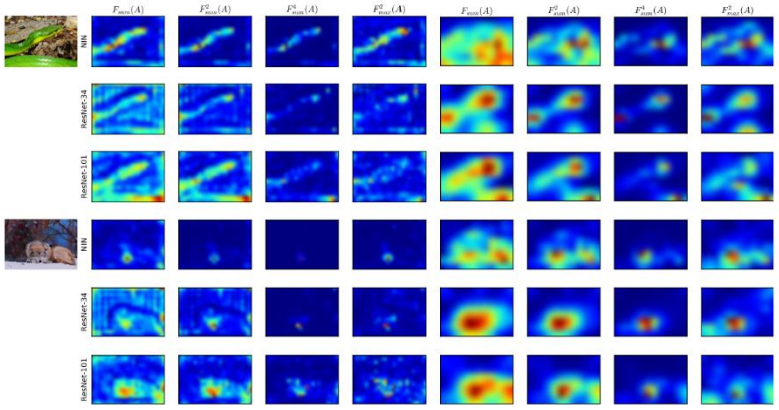

FIGURE 2 ACTIVATION ATTENTION MAPS FOR VARIOUS NETWORKS: NETWORK-IN-NETWORK, RESNET34, RESNET-101.

The following year it was proposed to use the so called "factors" to transport the knowledge between networks using an interesting concept: features maps from the teacher are fed to a small stack of convolutional layers called "paraphraser", whose job is to simplify the feature extracted from the teacher's hint layer into a *factor* to make it comprehensible for the student, which in turn uses a similar convolutional stack as a "translator" to fully understand the *factor*. The paraphraser maintains the feature map's spatial dimension but resizes the channels by an hyperparameter *k*. It is trained in an unsupervised way:

$$L_{rec} = \|x - P(x)\|^2 \quad \textbf{(4)}$$

The translator is trained jointly with the student, so the student's loss is the sum of the classification task and the translator's losses:

$$L_s = CE(\hat{y},\ y) + \beta \left\| \frac{F_T}{\|F_T\|_2} - \frac{F_S}{\|F_S\|_2} \right\|_1 \quad \textbf{(5)}$$

One of the most recent approaches is called Semantic Calibration (SemCKD)[8] and focused on giving guidance to the student guided layers from multiple hint layers in a different capacity, according to an attention allocation process. It first constructs similarity matrices for multiple layers across student and teacher:

$$A^S_{S_l} = R\left(F^S_{S_l}\right) \cdot R\left(F^S_{S_l}\right)^{\mathrm{T}} \qquad A^T_{T_l} = R(F^T_{Tl}) \cdot R\left(F^T_{T_l}\right)^{\mathrm{T}} \quad \textbf{(6)}$$

Where $R(\cdot) : \mathbb{R}^{b\times c\times h\times w} \mapsto \mathbb{R}^{b\times chw}$ is a reshaping operation that makes A^S_{Sl} and A^T_{Tl} two $b \times b$ matrices.

It then uses a Multi-Layer Perceptron to create the queries and keys of an attention mechanism:

$$Q_{S_l}[i] = MLP_Q(A^S_{S_l}[i]) \qquad K_{T_l}[i] = MLP_K(A^T_{T_l}[i]) \quad \textbf{(7)}$$

From here, it computes the weight associated with each pair of hint-guided layers:

$$\alpha^i_{(S_l,T_l)} = \frac{e^{Q_{S_l}[i]^{\mathrm{T}} K_{T_l}[i]}}{\sum_j e^{Q_{S_l}[i]^{\mathrm{T}} K_{T_j}[i]}} \quad \textbf{(8)}$$

In the end, the teacher's feature maps are projected into the same space of the student's feature maps to match the dimensions and guide the training for every pair of layers, such that the loss term associated with SemCKD becomes:

$$\mathcal{L}_{SemCK} = \beta \sum_{S_l=1}^{S_L} \sum_{T_l=1}^{T_L} \sum_{i=1}^{b} \alpha^i_{(S_l,T_l)} MSE(F^T_{T_l}[i], Proj(F^S_{S_l}[i], T_l)) \quad \textbf{(9)}$$

$$\mathcal{L}_{Total} = \sum_{i=1}^{b} \mathcal{L}_{KDi} + \mathcal{L}_{SemCK} \quad \textbf{(10)}$$

However, the most effective method developed in this context is called "Rocket-Launching" [9] and consists in a simple idea: if the student is a smaller version of the same network as the

teacher (i.e., a ResNet18 and a ResNet34) we could unify their backbones, with small adjustments to their dimensionalities, and train them at the same time.

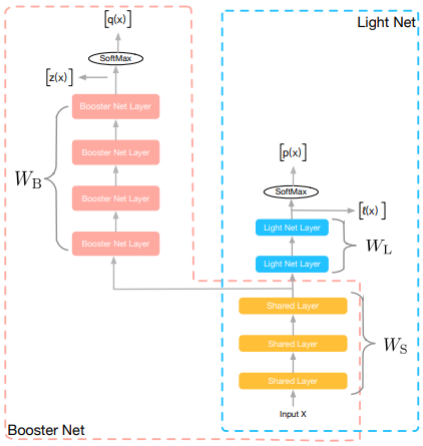

FIGURE 3 ROCKET LAUNCHER FRAMEWORK, THE LIGHT NET IS THE STUDENT AND THE BOOSTER NET IS THE TEACHER

In particular, the loss function that updates the shared backbone will have three components: the hard targets loss for the student, the hard-targets loss for the teacher and the distillation loss. Formally:

$$\mathcal{L}_{Total} = CE_T\big(y, q(x)\big) + CE_S\big(y, p(x)\big) + \mathcal{L}_{Mimic}(l(x), z(x)) \quad \textbf{(11)}$$

$\mathcal{L}_{Mimic}$ is the SNN-MIMIC loss formulated to workaround the vanishing gradient problem that can undermine other distances, like MSE. It is defined as:

$$\mathcal{L}_{Mimic} = \frac{1}{2T}\sum_{i} \| \sigma(z_s, T = 1) - z_s \|_2^2 \quad \textbf{(12)}$$

Where $l(x)$ is the logits distribution that produces $p(x)$ through SoftMax, same for $z(x)$ and $q(x)$. N and T are respectively the number of samples and the temperature.

1.2.3 – RELATION-BASED KNOWLEDGE

While both Response-Based and Feature-Based models focused on the outputs of specific layers in the model, the Relation-Based methods explore the relationships between data samples or different layers and try to exploit those to improve the student's training.

Generally, the framework of Relation-Based Distillation with feature maps is:

$$\mathcal{L}_{RelD}(f_t, f_s) = \mathcal{L}_{R^1}\left(\psi_t(\hat{f}_t, \check{f}_t), \psi_s(\hat{f}_s, \check{f}_s)\right) \quad \textbf{(13)}$$

While if we are considering the instances relations:

$$\mathcal{L}_{RelD}(f_t, f_s) = \mathcal{L}_{R^2}\left(\psi_t(t_i, t_j), \psi_s(s_i, s_j)\right) \quad \textbf{(14)}$$

In A Gift from KD[10] the authors showed how making a small network try to imitate the output of a bigger network can be a hard constraint on its training since there are multiple ways to obtain a specific output from an input. They proposed that student should learn the method to obtain the solution instead of trying to imitate completely the feature maps produced by the teacher. They proposed to consider two layers in both the teacher and the student and to compute their Flow Solution Process (FSP) matrix as:

$$G_{i,j}(x; W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x; W) \times F^2_{s,t,j}(x; W)}{h \times w} \quad \textbf{(15)}$$

Where $F^1_{s,t,i}$ and $F^2_{s,t,i}$ are the feature maps generated by the two layers in the teacher or the student, x is the input and W are the weights of the layer. The loss function for the distillation is then defined as:

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n} \lambda_i \times \left\| G^T_i(x, W_t) - G^S_i(x, W_s) \right\|^2_2 \quad \textbf{(16)}$$

It is clear in the equation that each data sample has a different weight to counterbalance the possibly unbalanced datasets.

The relationships between feature maps produced by different layers have been a very interesting topic for a lot of researchers. One of the most effective and interesting techniques developed for the task is called Graph-Based KD by Multi-Head Attention Network[11]. As the authors mention in their work, their idea was inspired by the fact that Graph Neural Networks can learn relation between vectors by embedding them in their own space and the Attention Network is the most widespread GNN. The most glaring case of this behaviour is given by the Attention Mechanism, in which a query vector and a key vector are embedded, through several layers of Attention Heads, into a graph of their relations.

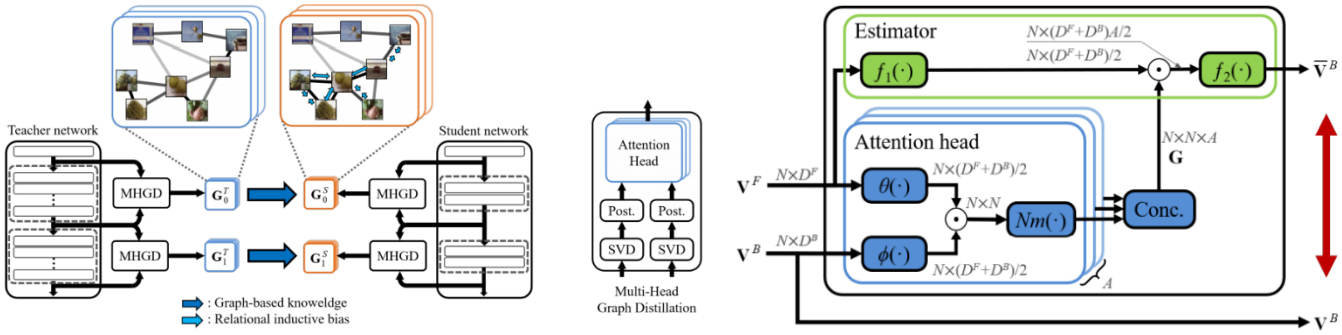

FIGURE 4 THE MULTI-HEAD GRAPH DISTILLATION (LEFT) AND THE DETAIL OF THE ATTENTION HEADS AND ESTIMATOR (RIGHT)

As in the FSP method, we choose two layers of each network from which we extract queries (the shallower layer's feature map, or Frontend Feature Vector set V^F), and keys (the deeper layer's feature map, or Backend Feature Vector set V^B):

$$V^B = \{v_i^B | 1 \le i \le N\} \quad V^F = \{v_j^F | 1 \le j \le N\} \tag{17}$$

The two vector sets have different dimensions that need to be matched and then embedded in a single set:

$$S = [\theta(v_i^B) \cdot \phi(v_j^F)]_{1 \le i \le N, 1 \le j \le N} \tag{18}$$

Such an embedding is performed, in practice, with a simple Fully Connected-Batch Normalization combination.

Attention G is obtained by normalizing with SoftMax every set obtained:

$$G = [Nm(S_a)]_{1 \le a \le A} \quad \text{where} \quad Nm(S) = [\frac{\exp(S_{i,j})}{\sum_k \exp(S_{i,k})}]_{1 \le i \le N, 1 \le j \le N} \tag{19}$$

At this point, the estimator shown in fig. 4 tries to estimate V^B from V^F and G:

$$\bar{V}^B = f_2(G \cdot f_1(V^F)) \tag{20}$$

$$f_1(V^F) = \max(0, BN(W_1 V^F)) \quad \text{and} \quad f_2(G \cdot f_1(V^F)) = \frac{W_2 G \cdot f_1(V^F) + b_2}{\|W_2 G \cdot f_1(V^F) + b_2\|_2} \tag{21}$$

Finally, the loss function for the training of the Attention Network is:

$$\mathcal{L}_{MHAN} = \sum_{m=1}^{M} \frac{1}{N} V_m^B \bar{V}_m^B \tag{22}$$

The Attention Mechanism pays attention to V^F to estimate V^B so in G we can find two kinds of information:

1. Feature Transform, which is the relation representing the FSP.
2. Intra-Data Relation, because the Attention Mechanism works throughout the mini batch, effectively embedding knowledge about the dataset in the graph

The knowledge is the transferred to the student by virtue of the Attention Networks shown in fig. 4:

$$\mathcal{L}_{transfer} = KLD(G^S_{m,i,j,a}, G^T_{m,i,j,a}) = \sum_{m,i,j,a} G^S_{m,i,j,a}(\log(G^S_{m,i,j,a}) - \log(G^T_{m,i,j,a})) \quad \textbf{(23)}$$

1.3 - TYPES OF DISTILLATION

1.3.1 - OFFLINE DISTILLATION

The Offline Distillation scheme is the classical scheme outlined in the Vanilla Distillation by Hinton. It consists in a pre-trained teacher that already contains all the knowledge we could distil and a student that needs to be trained using one or more of the knowledges described above. The training of the teacher is usually not discussed as part of the Knowledge Distillation process, so the offline methods are developed using any kind of dataset and architectures with no regard to what was used to train the teacher.

The main advantage of these methods is that they are very easy to implement, but it comes with some drawbacks. The fact that the teacher is fixed and usually has a much larger capacity than the student means that it still needs a lot of resources and time to be trained and often represent a hard constraint on a small student that cannot match its cap.

One of the approaches developed for this kind of knowledge that inspired this work is called Spherical Knowledge Distillation. When a teacher makes a prediction, it often has a high confidence in it, resulting in a high logit for the predicted class and lower logits for the rest. This confidence comes from the knowledge it acquired during the training phase, the same knowledge we would like to transfer to the student.

In Spherical Knowledge Distillation, the authors noted how the confidence of the final prediction is highly determined by the magnitude of logits. Thus, compared with logits,

normalized logits are less affected by the teacher's confidence while retaining relevant knowledge to transfer.

They decomposed the teachers and students' logits as:

$$f^T(x) = \left\|f^T(x)\right\|_2^2 * \frac{\hat{f}_i^T(x)}{\left\|f^T(x)\right\|_2^2} \quad \textbf{(24)}$$

$$f^S(x) = \left\|f^S(x)\right\|_2^2 * \frac{\hat{f}_i^S(x)}{\left\|f^S(x)\right\|_2^2} \quad \textbf{(25)}$$

The experiments they conducted showed that using the norm of logits in Vanilla Knowledge Distillation is detrimental to the final accuracy of the student, while using normalized logits improves it.

From a gradient perspective, they noted that:

$$\frac{\partial \mathcal{L}_{\text{KD}}}{\partial\, \hat{f}_i^S(x)} = \frac{\partial \left\|l^S(x)\hat{f}^S(x) - l^T(x)\hat{f}^T(x)\right\|_2}{\partial \hat{f}_i^S(x)} = 2l^S(x)(l^S(x)\hat{f}^S(x) - l^T(x)\hat{f}_i^T(x)) \quad \textbf{(26)}$$

During the distillation process, while the student tries to learn $\hat{f}_i^T(x)$, the student's gradient of $\hat{f}^S(x)$ changes all the time and therefore will difficulty converge to $\hat{f}_i^T(x)$, which is stable. They avoid this problem by normalizing both logits by the average of the teacher's l2 norms before smoothening with the temperature. The distribution becomes more uniform, shrinking the distance between the higher logits and the lower.

$$\hat{p}_i = \frac{\exp\left(\hat{f}_i(x) * \frac{l_{avg}}{\tau}\right)}{\sum_j \exp\left(\hat{f}_j(x) * \frac{l_{avg}}{\tau}\right)} \quad \textbf{(27)}$$

The distillation loss becomes:

$$\mathcal{L} = \alpha \sum_i \hat{p}_i^T \log\left(\hat{p}_i^S\right) + (1-\alpha) \sum_i y \log\left(\hat{p}_i^S\right) \quad \textbf{(28)}$$

Algorithm 2: Spherical Distillation
Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α = alpha For each epoch: for x_i in x: *teacher logits = teacher(x_i)* *student logits = student(x_i)* *student pred = SoftMax(student logits)* *teacher pred = SoftMax(teacher logits* $* \frac{l_{avg}}{\tau}$*)* *smoothened student pred = SoftMax(student logits* $* \frac{l_{avg}}{\tau}$*)* **loss** = $\alpha * KLD$(*teacher pred, smoothen student pred*) + (1-α) * *CE*(*student pred,* y_i)

This approach highlighted that there is the possibility to exploit the confidence of the teacher to improve the distillation framework, which we will talk about in the chapter regarding Relative Knowledge Distillation.

1.3.2 - ONLINE DISTILLATION

Often the need for a pre-trained teacher poses a relevant obstacle for some applications. To avoid it, a variety of Online methods that train the student and the teacher at the same time have been devised. The Rocket-Launcher method outlined earlier is one of such examples, in which the logits predicted from the teacher at training time are included in the student's loss function.

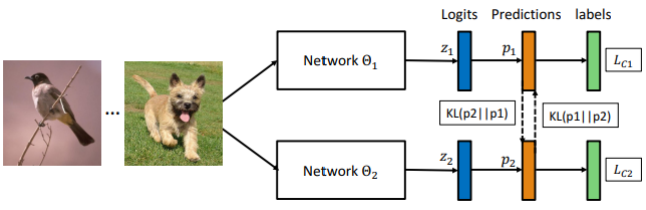

FIGURE 5 UMBRELLA DISTILLATION

The main contributions to this topic fall under the umbrella of Deep Mutual Learning[1312]. Here, two networks θ_1 and θ_2 are trained at the same time with similar loss functions:

$$\mathcal{L}_{\theta_1} = CE_{\theta_1} + KLD(p_2||p_1) \text{ and } \mathcal{L}_{\theta_2} = CE_{\theta_2} + KLD(p_1||p_2) \quad \textbf{(29)}$$

The two CE components are the Cross Entropy loss on the hard targets and the KLD is the Kullback-Leibler Divergence given by using the one network's prediction to approximate the others. This framework is easily extendable to situations in which there are more than two networks:

$$\mathcal{L}_{\theta_k} = CE_{\theta_k} + KLD(p_{avg}||p_k) \text{ where } p_{avg} = \frac{1}{K-1}\sum_{l=1,l\neq k}^{K} p_l \quad \textbf{(30)}$$

The topic of *n* networks trained together is further explored in Knowledge Distillation for Collaborative Learning (KDCL). Here, each network is fed the same image augmented in a different way to improve the overall generalization capability of the networks on the datasets.

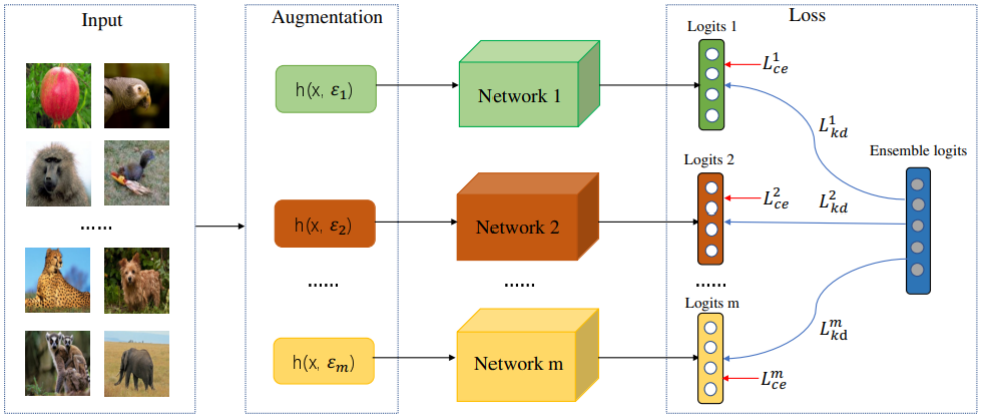

FIGURE 6 KNOWLEDGE DISTILLATION FOR COLLABORATIVE LEARNING

The loss function is:

$$\mathcal{L} = \sum_{i=1}^{m} L_{CE}^i + \lambda L_{KLD}^i \quad \textbf{(31)}$$

One idea to select a teacher between all the networks could be to choose the one with the lower loss with respect to the Cross-Entropy loss, so the KLD is calculated between each student and the selected teacher. This selection could be more effectively treated as an optimization problem in which all network's logits represent a column of a matrix Z and it boils down to solving the optimization problem:

$$min_{\alpha\in\mathcal{R}^m} L_{CE}(\alpha^T Z, y) \text{ given that } \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0 \quad \textbf{(32)}$$

Another idea is to test the performances of the networks on a validation subset after every training step, in which case we will be much surer to select the teacher with the highest

generalization ability. To do so, the authors proposed to compute the generalization error on an input of the ensemble of networks as:

$$E = \sum_{i=1}^{m}\sum_{j=1}^{m} \omega_i \omega_j C_{ij} \tag{33}$$

Where:

$$C_{ij} = \int (f_i(x) - t)\big(f_j(x) - t\big)p(x)dx \approx \frac{1}{N}\sum_{k=1}^{N} (f_i(x_k) - t)(f_j(x_k) - t) \tag{34}$$

Each ω_k is a weight such as $\omega_k = \frac{\sum_{j=1}^{m} C_{kj}^{-1}}{\sum_{i=1}^{m}\sum_{j=1}^{m} C_{kj}^{-1}}$ and $f_i(x)$ is the probability distribution of the i-th network.

Even more interesting was the use of multi-level distillation in Online Knowledge Distillation with Diverse Peers (OKDDip)[13]. The soft targets are derived from the group of peers by aggregating the predictions of all peers with different weights:

$$t_a = \sum_{b=1}^{m-1} \alpha_{ab} \cdot q'_b \tag{35}$$

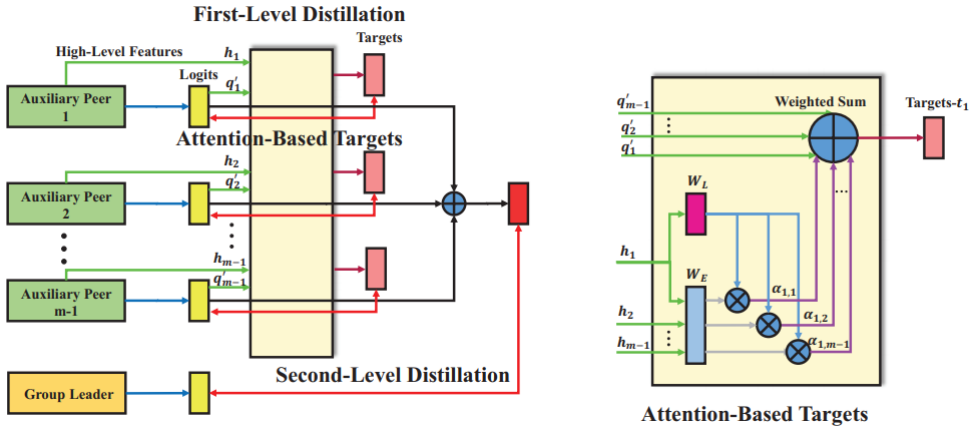

FIGURE 7 ONLINE KNOWLEDGE DISTILLATION WITH DIVERSE PEERS

The overall framework is trained by:

$$\mathcal{L}_{OKDDip} = \sum_{a=1}^{m} \mathcal{L}_{gt}(a) + T^2\mathcal{L}_{dis1}\big(t_a, q'_a\big) + T^2\mathcal{L}_{dis2}\,(t_m, q'_m) \tag{36}$$

$\mathcal{L}_{dis1}$ is the usual KLD between a peer and the aggregated soft targets of the others while $\mathcal{L}_{dis2}$ is the KLD between each peer and a group leader.

The use of Attention-Based Targets allows each peer to better attend the others by calculating the weights as an Embedded Gaussian distance with normalization:

$$\alpha_{ab} = \frac{e^{L(h_a)^T E(h_b)}}{\sum_{f=1}^{m-1} e^{L(h_a)^T E(h_f)}} \quad \textbf{(37)}$$

$$L(h_a) = W_L^T h_a \qquad W(h_a) = W_E^T h_a \quad \textbf{(38)}$$

h_a are the extracted features from a peer and W_E^T, W_L^T are learned projections matrices that embed respectively the features extracted by all peers and by the current peer. The use of both matrices allows the framework to keep two key properties:

- Asymmetry, which allows to suppress negative effect of having peers with different level of optimization without stopping the positive guidance that comes from multiple peers.
- Dynamicity, to give more flexibility to the ensemble by calibrating the use of peers according to how optimized they have become at each training step.

Some of the most recent works in the context of Online Distillation fall under the Generative Adversarial Distillation umbrella. Generative Adversarial Networks has had great success in a lot of fields. It consists of two competing networks, a generator and a discriminator, whose jobs are to generate a new data sample and to distinguish it from a real data sample. Such an approach has been proposed for knowledge distillation too by different authors.

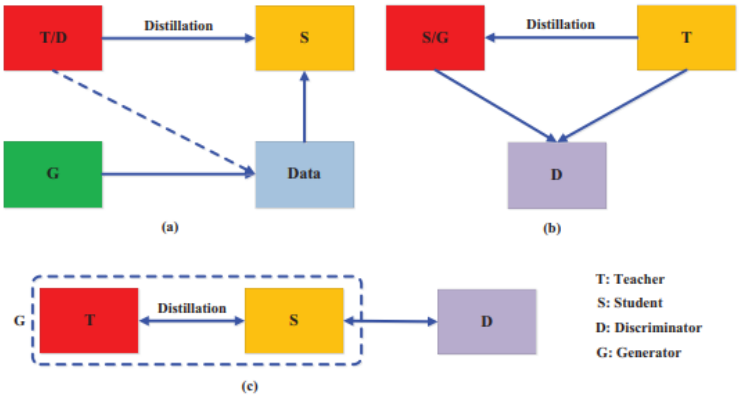

FIGURE 8 GENERAL FRAMEWORK FOR GENERATIVE ADVERSARIAL DISTILLATION

There are three main categories of adversarial distillation. In the first category's methods, the generator creates synthetic data that is then used as a training dataset or to augment the existing dataset. For example, in Zero-shot Knowledge Transfer[14], they propose to compute *n* gradient steps on the generator and then produce a batch of synthetic data to maximize the Kullback-Leibler Divergence between the teacher and the student. Only after that, they take *m* gradient steps on the student to make it match the teacher's distribution on the synthetic samples. However, it is theoretically possible that the generator starts exploring a larger portion of the data space in which the teacher has not been trained, producing unrealistic images. This happens because there is no constraint on the kinds of images it can generate. The experiments carried out to prove it did show that, if the number of classes is limited (i.e., Cifar10, MNIST) random noisy images are clustered in a single class.

The second category's methods use the discriminator to distinguish the student's logits or feature from those produced by the teacher while the last category's approaches use the teacher and the student as a generator and trains them jointly, much like it would happen in online distillation.

The main contribution for both these categories was given by Feature-map level Online Adversarial Knowledge Distillation[15]

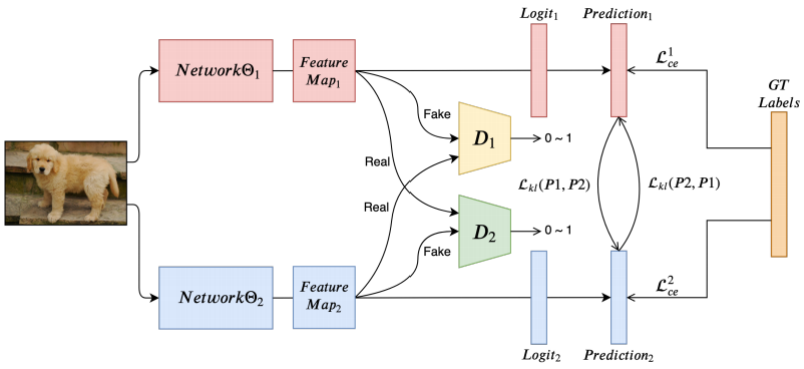

FIGURE 9 FEATURE-MAP LEVEL ONLINE ADVERSARIAL KNOWLEDGE DISTILLATION

As shown in Fig. 10, the two networks are not student and teacher because the distillation here happens in an online setting. Each network has its own discriminator that considers the other network's feature map as the real feature map, performing a simple binary classification to distinguish them. Each network can be decomposed in two parts:

- the feature extractor, which is roughly equivalent to a classical GAN generator, that will have loss function:

$$\mathcal{L}_{G_k} = [1 - D_k(G_k(x))]^2 \quad \textbf{(39)}$$

- the discriminator is trained by itself, its loss does not influence the networks losses:

$$\mathcal{L}_{D_k} = [1 - D_k(G_n(x))]^2 + D_k(G_k(x))^2 \quad \textbf{(40)}$$

The distillation then happens by computing the Kullback-Leibler Divergence between the logits produced by the two network, the discriminator's loss, and the Cross-Entropy loss:

$$\mathcal{L}_1 = CE + T^2 \times \mathcal{L}_{KLD}\left(\frac{z_1}{T}, \frac{z_2}{T}\right) + \mathcal{L}_{G_1} \quad \textbf{(41)}$$

$$\mathcal{L}_2 = CE + T^2 \times \mathcal{L}_{KLD}\left(\frac{z_2}{T}, \frac{z_1}{T}\right) + \mathcal{L}_{G_2} \quad \textbf{(42)}$$

1.3.3 - SELF DISTILLATION

Self-Distillation is a special case of Online-Distillation in which the same network is both the student and the teacher. The basic implementation of this intuition came from Be Your Own Teacher[16], in which was proposed to use deeper layers of the network to directly train shallower layers. An example of this concept can be easily imagined by thinking about a ResNet architecture such as Fig. 5

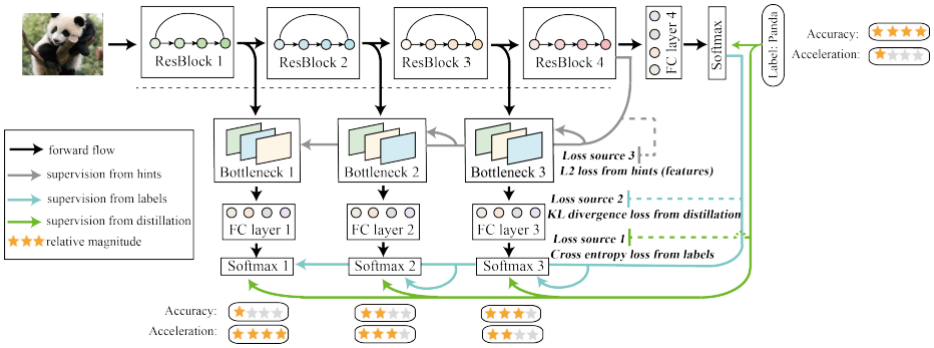

FIGURE 10 THE GENERAL FRAMEWORK OF KNOWLEDGE DISTILLATION ACCORDING TO BE YOUR OWN TEACHER

Each residual block is part of the overall network, but at training time a classifier head composed of a Residual Bottleneck layer and a Fully Connected layer with SoftMax activation is added on top of it. With this addition, each block produces its own prediction that can be used to distil knowledge from the deeper block, which will be more accurate and richer in information given what the previous block had already learned about the data space.

$$\mathcal{L} = \sum_{i}^{C}((1-\alpha)\cdot\mathcal{L}_{CE}(q^i, y) + \alpha\cdot\mathcal{L}_{KLD}(q^i, q^C) + \lambda\cdot\|F_i - F_C\|_2^2) \quad \textbf{(43)}$$

For this work the authors did not use the logits produced by each classifier but the output of the SoftMax q^i and the feature maps F_i.

Almost at the same time, Snapshot Distillation[17] was proposed. The authors noticed that even if the accuracy improvements brought by Knowledge Distillation are significant, often the process is computationally expensive.

They proposed to train the student with guidance provided by previous iterations of the same student instead of using different networks or layers as a teacher. In other words, we consider an intermediate state of the same model as teacher, store it, and use it to obtain a distillation loss to add to the Cross Entropy at the last iteration of a minibatch.

So, if a training process using a minibatch B_l consists of L epochs, the gradients are updated using:

$$\mathcal{L}(\mathcal{B}_l, \theta_{l-1}) = -\frac{1}{|B_l|} \sum_{(x_n, y_n) \in \mathcal{B}_l} \{\lambda_l^S \cdot y_n^T \ln f(x_n, \theta_{l-1}) + \lambda_l^T \cdot KL[f(x_n, \theta_{cl}) || f(x_n, \theta_{l-1})]\} \quad \textbf{(44)}$$

λ_l^S and λ_l^T are the weights that balance soft and hard targets. However, it is evident that being the teacher the same architecture as the student just a few passes earlier, their prediction may be very close, leading the second term of the loss to degenerate to zero.

It is worth noticing that in this approach the student did not use a temperature hyperparameter to smoothen the distribution of logits. They explained that using the same model as both teacher and student does not raise difficulties for what concerns the capacity of the student with respect to the teacher, while in classical settings such a gap needs to be accounted for by smoothening the logits distribution.

CHAPTER 2 – PROPOSED IMPROVEMENTS

In this chapter, we present some ways to improve the performance of Vanilla Knowledge Distillation. The methods we propose can be categorized as Response-Based, Offline-Distillation and focus on two different aspects of Knowledge Distillation that we found were not investigated enough in literature.

The first aspect concerns what to do with incorrect predictions by the teacher. In general, such predictions are not very useful, sometimes they may even be detrimental to the training of a distilled student, because they pull the gradients in the wrong direction with respect to the Cross-Entropy component of the loss, so the knowledge that the student would acquire may be lost in the process.

The second aspect is based on the observation that a too confident teacher is not really that useful. In fact, if the teacher predicts the correct class all the time with high confidence, then the SoftMax outputs something very similar to a one-hot encoded vector, no matter the temperature's smoothing that was applied.

Starting from these considerations, we propose three new approaches to Knowledge Distillation.

2.1 - EARTH MOVER DISTILLATION

The main idea behind Earth Mover Distillation is to obtain relevant knowledge from samples that were incorrectly classified by the teacher. To do so, we used Earth Mover Distance on the teacher's SoftMax predictions.

Earth Mover Distance, or Wasserstein metric, is a measure of the distance between two probability distributions interpreted as the minimum cost of turning one distribution into the other one. Informally, the name comes from the amount of dirt to move to turn one pile of dirt into another one.

Algorithm 3: Earth Mover Distillation
Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α For each epoch: for x_i in x: if y_{pred} != y_i: *teacher logits = teacher(x_i)* *teacher predictions = SoftMax(teacher logits / τ)* *sorted* = sort(*teacher predictions, by=ascending*) for *value* in *sorted*: *teacher predictions*[y_i] = *teacher predictions*[y_i] + *value* if *teacher predictions*[y_i] = = max(*teacher predictions*): **break** Vanilla Distillation(*teacher logits, student logits*, τ, α)

In practice, when the teacher's prediction is wrong, we smoothen the logits using the temperature hyperparameter as usual and apply the SoftMax. We then sum the smaller probabilities of the incorrect classes to the correct class, giving the student more samples to obtain knowledge from the teacher.

2.2 - TWO TEMPERATURE DISTILLATION

With "Two Temperature Distillation" we tried another way to improve the effectiveness of Knowledge Distillation when the teacher's predictions were wrong. Instead of correcting them, which is an artificial way to alter the output produced by the teacher, the idea behind it is that a higher temperature, and therefore a stronger smoothening of the logits distribution, makes the SoftMax output on the correct class closer to the output of the predicted class. When the teacher makes correct predictions we used a smaller temperature, to keep the correct class prediction higher than the rest.

This method is identical to Vanilla Distillation with the exception that distinguishes the predictions between correct and incorrect ones, giving each its own temperature.

Algorithm 4: Two Temperature Distillation
Input: pre-trained teacher, student, x = data samples, y = hard targets, τ_{right} = temperature for correct predictions, τ_{wrong} = temperature for incorrect predictions and α For each epoch: for x_i in x: *teacher logits = teacher(x_i)* *student logits = student(x_i)* *teacher pred = SoftMax(teacher logits)* if *max(teacher pred) == y_i:* Vanilla Distillation(*teacher logits, student logits,* τ_{right}, α) else: Vanilla Distillation(*teacher logits, student logits,* τ_{wrong}, α)

2.3 - RELATIVE KNOWLEDGE DISTILLATION

As outlined in Spherical Distillation, the confidence of a teacher can diminish the amount of knowledge we are able to transfer to the student. But a high confidence in a class also means that most of the other classes will have very low logits. This would convey the information that those classes have nothing in common with the correct class and, therefore, are not very useful for the distillation process.

In case the prediction is incorrect it would not be useful to the distillation process because it would steer the gradients in a different direction than the Cross Entropy loss, so we would like to adjust it similarly to what is described in Earth Mover Distillation.

We propose four different implementations of "Relative Knowledge Distillation", or RKD, to verify if these two assumptions are valid.

2.3.1 - RELATIVE KNOWLEDGE DISTILLATION V1

There are two main ideas behind the formulation of Relative Knowledge Distillation. First, one could think that a good teacher should be the most accurate possible on the task at hand. This is however an incorrect assumption: a very accurate teacher often has a lot of confidence in its predictions, resulting in the output of its last SoftMax layer being very similar to the one-hot encoding of the targets. In the Vanilla Distillation framework, the loss function is composed of two terms, the Cross Entropy loss between the student's predictions and the one-hot encoding of the labels, and the Kullback-Leibler Divergence between the teacher's and the student's predictions.

$$\mathcal{L} = \alpha\, CE(y,\, S) + (1-\alpha)\, KLD(T, S) \tag{45}$$

Where S and T are the SoftMax outputs of the student and the teacher, respectively. The KLD is defined as the measure of how one probability distribution can be used to approximate another one:

$$KLD(T||S) = \sum_i T(i) log_2\left(\frac{T(i)}{S(i)}\right) \tag{46}$$

It is evident that it relates closely to Cross Entropy. In fact:

$$KLD(T||S) = \sum_i T(i) log_2\left(\frac{T(i)}{S(i)}\right) \tag{47}$$

$$= \sum_i T(i) log_2\big(T(i)\big) - \sum_i T(i) log_2\big(S(i)\big)$$

$$= \sum_i T(i) log_2(T(i)) + CE(T(i), S(i))$$

Therefore, the final loss for Vanilla Distillation becomes:

$$\mathcal{L} = \alpha\, CE(y,\, S) + (1-\alpha)\, KLD(T, S) \tag{48}$$

$$= \alpha\, CE(y,\, S) + (1-\alpha) \sum_i T(i) log_2\big(T(i)\big) + CE\big(T(i), S(i)\big)$$

$$= \alpha\, CE(y,\, S) + (1-\alpha)\, CE(T,\, S) + (1-\alpha) \sum_i T(i) log_2\big(T(i)\big)$$

If $y \approx T$, meaning if the teacher is very confident and its predictions resemble the one-hot encoded vector of the targets, the two Cross Entropy terms are redundant. The other term is the entropy of the distribution *T:*

$$(1-\alpha)\sum_i T(i)log_2\big(T(i)\big)$$

The distribution T is simply the distribution that the teacher model learned to fit all data points in the space of the training set, therefore it is constant given that the training set does not change. The student will try to mimic the distribution of the teacher but their capacity gap, meaning the difference in depth and/or number of parameters, does not allow for an accurate mimicking, making the teacher an unreliable one for the distillation process.

The second idea comes from considering that not every class has something in common with the others. From a human point of view, we instinctively know that a dog has nothing in common with a house while it has a few things in common with a cat such as eyes, mouth, ears etc. In Vanilla Knowledge Distillation any class is treated equally, smoothened with the same temperature no matter the image that is being fed in input.

One could say that this approach overestimates the knowledge contained in the smallest classes and that it would be much more valuable to highlight the difference between two similar classes of images than the difference between very different classes.

In the first version of RKD, the absolute value of the lower, incorrect logits is subtracted from the higher logit when the prediction is correct. This operation can also be thought of as a form of penalty given to the teacher for not being totally confident that its decision was the correct one.

When the prediction is incorrect, all its logits are set to zero. In this way, the SoftMax prediction is uniform and therefore the predicted label is randomly chosen. Such an approach to handle incorrect predictions may seem, and in fact is, extreme. With this version of RKD we want to understand if we can extract useful knowledge even from the mistakes of the teacher.

Lowering the confidence has the effect of bringing closer the first few classes that are deemed more probable for the current data sample. When these adjusted logits are smoothened by the temperature parameter, the resulting distribution presents peaks that are much closer than they would be if the temperature was applied directly to the original logits.

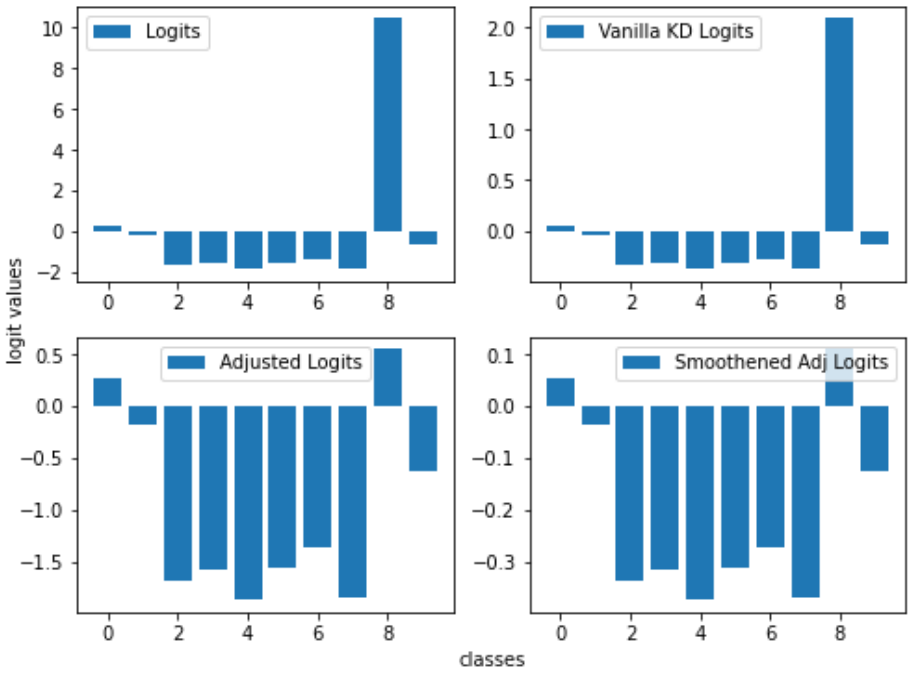

FIGURE 11 VANILLA LOGITS AND RELATIVE KD V1 LOGITS OF A CORRECT PREDICTION.

THE VALUE OF THE PREDICTED CLASS (8) IS LOWERED USING THE ABSOLUTE

VALUES OF THE SMALLER LOGITS AS A PENALTY BUT REMAINS THE HIGHEST ONE.

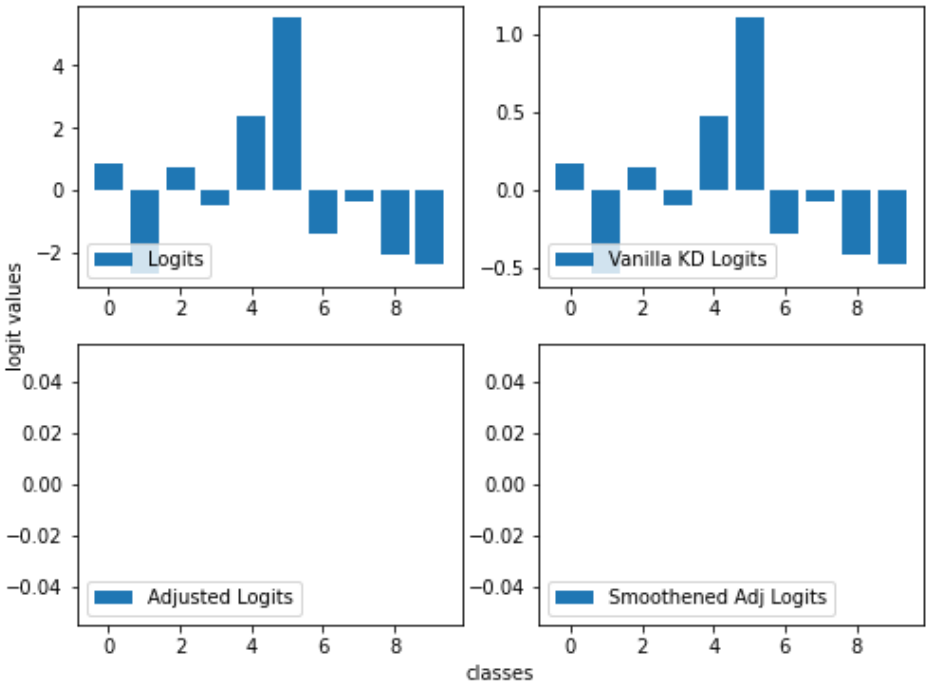

FIGURE 122 VANILLA LOGITS AND RELATIVE KD V1 LOGITS OF AN INCORRECT PREDICTION.

THE VALUE OF EVERY CLASS IS SET TO 0 SO THE OUTPUT OF THE SOFTMAX WILL BE RANDOMIC.

After adjusting the logits as described above, the classical Vanilla Distillation equation is applied. The algorithm works as follows.

Algorithm 4: Relative Knowledge Distillation V1

Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α

For each epoch:
  for x_i in x:
    teacher logits = teacher(x_i)
    student logits = student(x_i)
    teacher pred = SoftMax(teacher logits)
    if *max(teacher pred) == y_i*:
      sorted logits = sort(*teacher logits*)
      for *logit* in *sorted logits*:
        teacher logits[y_i] = teacher logits[y_i] - $|logit|$
        if *teacher logits[y_i] < sorted logits[-2]*:
          teacher logits[y_i]= teacher logits[y_i] + $|logit|$
          break
    else:
      teacher logits = [0, …, 0]
  Vanilla Distillation(*teacher logits*, *student logits*, τ, α)

2.3.2 - RELATIVE KNOWLEDGE DISTILLATION V2

In the first version, the incorrect predictions were completely discharged and substituted with a random prediction. The second version tries to improve on this aspect by using Earth Mover distance to correct the logits distribution when the teacher is wrong. It follows from the considerations expressed in the previous section on the confidence of the teacher that, if we accept that a too confident teacher gives a redundant term to the distillation loss function, and that most knowledge is embedded in the second highest logit produced, it is natural to assume that the logits produced by an image capable of tricking the teacher into making a mistake could contain a lot of interesting and useful knowledge.

The behaviour for correct predictions is the same as in RKD V1, meaning that the highest logit is lowered but remains higher than the second highest logit.

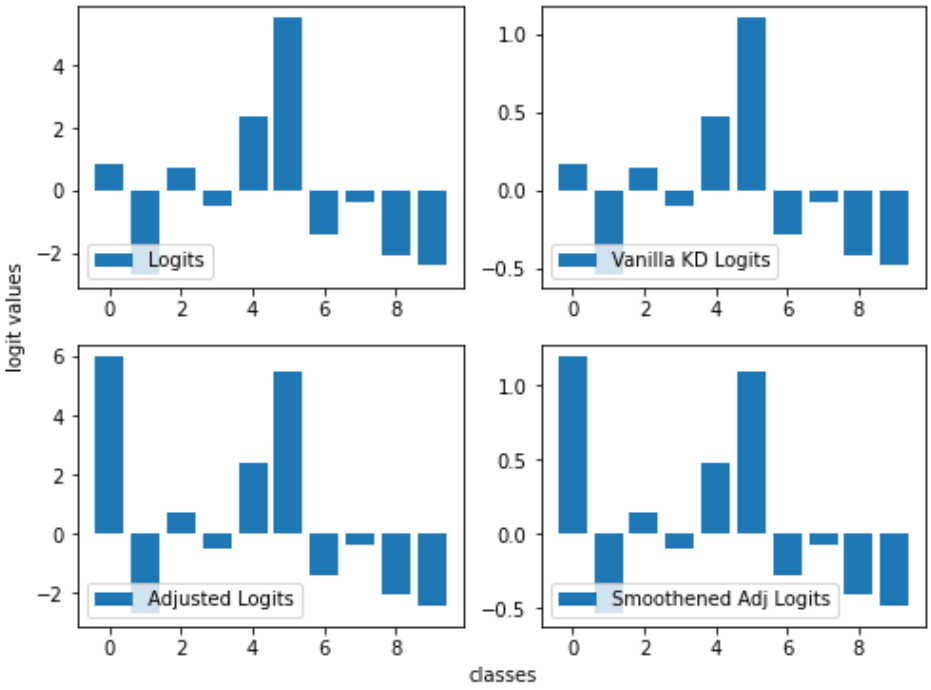

FIGURE 133 VANILLA KD AND RELATIVE KD V2 LOGITS FOR INCORRECT PREDICTIONS. CLASS 5 (INCORRECT PREDICTION) REMAINS THE SAME WHILE CLASS 0 (CORRECT CLASS) BECOMES THE HIGHEST ONE BUT STAYS CLOSE TO CLASS 5

Algorithm 5: Relative Knowledge Distillation V2

Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α

For each epoch:

for x_i in x:

teacher logits = teacher(x_i)

student logits = student(x_i)

teacher pred = SoftMax(teacher logits)

if *max(teacher pred) == y_i:*

sorted logits = sort(*teacher logits, by = ascending*)

for *logit* in *sorted logits*:

teacher logits[y_i] = teacher logits[y_i] - $|logit|$

if *teacher logits[y_i] < sorted logits[-2]*:

teacher logits[y_i] = teacher logits[y_i] + $|logit|$

break

else:

sorted logits = sort(*teacher logits*)

for $logit$ in *sorted logits*:

teacher logits[y_i] = teacher logits[y_i] + $|logit|$

if *teacher logits[y_i] < sorted logits[-2]*:

teacher logits[y_i] = teacher logits[y_i] + $|logit|$

break

Vanilla Distillation(*teacher logits*, *student logits*, τ, α)

2.3.3 - RELATIVE KNOWLEDGE DISTILLATION V3

Both methods described above try to extract the most knowledge possible from the higher logits. We try to understand if and how much the smaller logits impact the distillation process.

To do that, we developed RKD V3, which is the same as RKD V2 in every aspect except for a detail: every time a smaller logit is added or subtracted to the higher ones, it is also set to zero. Intuitively, one could argue that it is not a great idea to voluntarily lose part of the knowledge that could be extracted. The final goal of this version, and the experiments we ran for it, was to prove that there is knowledge there, but it is not as relevant or as impactful as the rest of the logits.

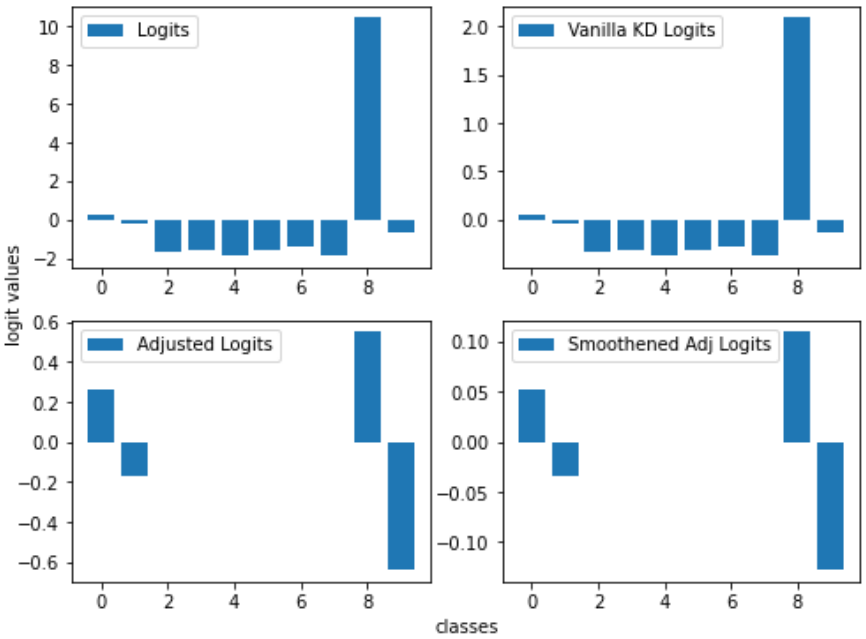

FIGURE 14 VANILLA KD AND RELATIVE KD V3 LOGITS FOR CORRECT PREDICTIONS.

THE VALUES SUBTRACTED TO THE CORRECT CLASS ARE SET TO 0,

LEAVING ONLY THE FOUR HIGHER VALUES. THE SAME THING HAPPENS FOR INCORRECT PREDICTIONS.

Algorithm 6: Relative Knowledge Distillation V3

Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α

For each epoch:

for x_i in x:

teacher logits = teacher(x_i)

student logits = student(x_i)

teacher pred = SoftMax(teacher logits)

sorted indexes logits = argsort(teacher logits, by = ascending)

if *max(teacher pred) == y_i:*

for *idx* in *sorted indexes logits:*

teacher logits[y_i] = teacher logits[y_i] – $|teacher\ logits[idx]|$

plc = teacher logits[idx]

teacher logits[idx]= 0

if *teacher logits[y_i] < sorted logits[-2]:*

teacher logits[y_i] = teacher logits[y_i] + $|teacher\ logits[idx]|$

$teacher\ logits[idx]$ *= plc*

break

else:

for idx in *sorted indexes logits:*

teacher logits[y_i] = teacher logits[y_i] + $|teacher\ logits[idx]|$

plc = $teacher\ logits[idx]$

$teacher\ logits[idx]$ *= 0*

if *teacher logits[y_i] < sorted logits[-2]:*

teacher logits[y_i] = teacher logits[y_i] + $|teacher\ logits[idx]|$

teacher logits[idx] = plc

break

Vanilla Distillation(teacher logits, student logits, τ, α)

2.3.4 - RELATIVE KNOWLEDGE DISTILLATION V4

One could say that the two ideas behind RKD conflicts with each other. In fact, the first idea at the roots of RKD says that an overconfident teacher is often a threat to an effective distillation process. In other words, this simply means that the distance between the predicted logit and the second highest logit should not be too large. This can be achieved by increasing the latter or lowering the former. But closing the gap between the two higher logits by increasing the correct one means that the distance between it and the other, smaller logits increases as well. In every version of RKD we have seen, we always increased one logit and therefore we increased such gap.

The second idea of RKD is to avoid a redundancy of the Cross Entropy term in the distillation loss equation as showed in the previous sections, particularly in **(48)**. If the prediction is incorrect the Kullback-Leibler distance will steer the loss and the gradients in a different direction than Cross Entropy, but that difference in direction could represent an effective source of knowledge.

For these reasons, the last version of RKD we propose behaves as usual, lowering the confidence of the teacher, but does not make any difference between correct and incorrect predictions. When the teacher is mistaken, it is not corrected by increasing the logit corresponding to the correct class. Instead, the higher logit predicted is always lowered to be as close as possible to the second one. The algorithm is reported in the following page.

Algorithm 7: Relative Knowledge Distillation V4

Input: pre-trained teacher, student, x = data samples, y = hard targets, τ = temperature and α

For each epoch:

for x_i in x:

teacher logits = teacher(x_i)

student logits = student(x_i)

teacher pred = SoftMax(teacher logits)

sorted indexes logits = argsort(teacher logits)

for idx in *sorted indexes logits:*

teacher logits[y_i] = teacher logits[y_i] – $|teacher\ logits[idx]|$

if *teacher logits[y_i] < sorted logits[-2]:*

teacher logits[y_i] = teacher logits[y_i] + $|teacher\ logits[idx]|$

break

Vanilla Distillation(teacher logits, student logits, τ, α)

CHAPTER 3 – EXPERIMENTAL RESULTS AND DISCUSSION

3.1 - EXPERIMENTAL SETUP

We have carried out experiments for each method described above. Each model, except for the ResNet152/ResNet50[18] combination, has been implemented in PyTorch and trained in Google Colab for 200 epochs with Stochastic Gradient Descent. The learning rate was set at 0.1, momentum at 0.9 and weight decay at 0.0005. The learning rate was multiplied by a factor of 0.2 at epochs 60, 120 and 160. We used the CIFAR10 and CIFAR100 datasets, which are formed by 32x32 images classified in 10 and 100 classes respectively, following the classical 50k/10k training and testing split. The dataset has been augmented with random crops and random horizonal flips.

TEACHER	STUDENT
ResNet152	ResNet50
ResNet34	ResNet18
WideResNet 40-1	WideResNet 16-1
WideResNet 40-2	WideResNet 16-2

TABLE 1 COMBINATIONS OF ARCHITECTURES USED IN THE EXPERIMENTS

The ResNet152/ResNet50 combination has been trained with Stochastic Gradient Descent with learning rate set at 0.1 and halved every time the validation accuracy does not improve for two consecutive epochs. If there is no improvement for three epochs, the training is stopped. This resulted in experiments running for 15/20 epoch, about 80 minutes in total, which is much less than the PyTorch experiments. The CIFAR10 dataset was augmented as described previously, with an additional upscaling added in the beginning of the networks to bring the images at 224x224 resolution.

Given the presence of hyperparameters in every method, I ran the experiments with different combinations of *alpha* and *temperature*. In the case of Two Temperatures Distillation, the combinations include *alpha, right temperature,* and *wrong temperature.*

- Alpha = 0.1, 0.3 and 0.5
- Temperature = 2, 5, 10

- Right temperature = 2, 5
- Wrong temperature = 10, 20, 50

The initial tests to prove if there was any validity behind the idea of Relative Distillation were run with a ResNet152 and a ResNet50 in Keras but have not been repeated in PyTorch with the same setting as the others because the computational power needed to train such big network was much higher than the runtime available on Colab. However, these initial tests were very important for the subsequent work, and we will therefore briefly report and talk about them.

We experimented with every method described in the previous Chapter and compared them with the teacher's and student's baselines, meaning the two networks trained only with Cross Entropy. In addition, to give a more comprehensive outlook on the improvements that these methods bring to Knowledge Distillation, we compare the results obtained with the performances of Vanilla Distillation and Spherical Distillation distilled from the same teacher.

3.2 - EXPERIMENTS

3.2.1 - RESNET-152/RESNET-50

The classical Vanilla Distillation performed reasonably well, improving over the baseline teacher of 2.04%. On the other hand, Spherical Distillation performed badly in this setting, being detrimental to the training of the student.

Teacher	Student	Vanilla	SKD	RKD V1	RKD V2	Two Temp	Earth Mover
94.35%	91.59%	93.63%	85.76%	**94.86%**	94.26%	94.08%	93.23%

TABLE 2 RESNET-152/RESNET-50 ON CIFAR10

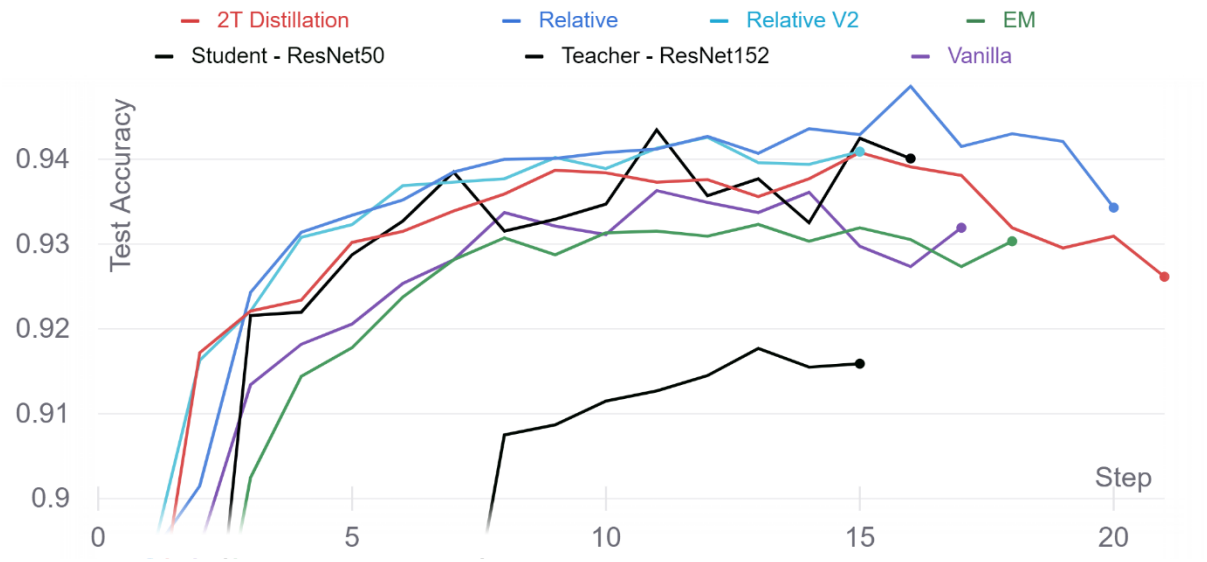

FIGURE 145 RESNET-152/RESNET-50 EXPERIMENTS ON CIFAR10

Two Temperature Distillation improved on the student's baseline by 2.49%, initially suggesting that there was a good intuition behind the idea of treating the incorrect predictions in a different way than the correct predictions. Earth Mover Distillation, even if it did not improve on the Vanilla Distillation performances, at least improved on the student's baseline by 1.64%.

Relative Distillation V1 and V2 performed the best out of all the methods, improving by 3.27% and 2.67% respectively. It is worth pointing out that RKD V1 produced a student that was even more accurate than its teacher. Even the worse performing combination of hyperparameters had an accuracy of 94.01%, which is higher than the best combination of every other method except Two Temperature.

Every method has a variance that depends on the specific combination of hyperparameters. Both RKD V1 and V2 proved to be more stable than Vanilla Distillation or Two Temperature. Earth Mover Distillation is the worst one from this point of view, oscillating by as much as 7% in this setting and even more in others.

3.2.2 - RESNET-34/RESNET-18

In the ResNet-34/18 experiments some patterns start to be recognizable from the plot of the different methods. For example, we can see how Relative Knowledge Distillation is always performing a little worse than the others, showing that a little bit of knowledge can be learned from the smaller logits. It is worth noting that RKD V1, V2, V4 and Vanilla perform

basically on par with each other. Given that the three RKD versions differ only by their handling of incorrect predictions, and that Vanilla does not distinguish between correct and incorrect predictions, it appears that when the teacher is wrong, it does not influence the distillation process. If correcting the predictions could be useful to the distillation, RKD V2 should perform sensibly better than RKD V1 and V4. On the same note, another recurring pattern are the Earth Mover's performances. EM distillation is often the worst performing method, actively worsening the student's baseline performances.

Teacher	Student	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
95.68	94.91	95.65	95.21	95.66	**95.68**	95.44	95.61	95.4	94.54

TABLE 3 RESNET-34/RESNET-18 ON CIFAR10

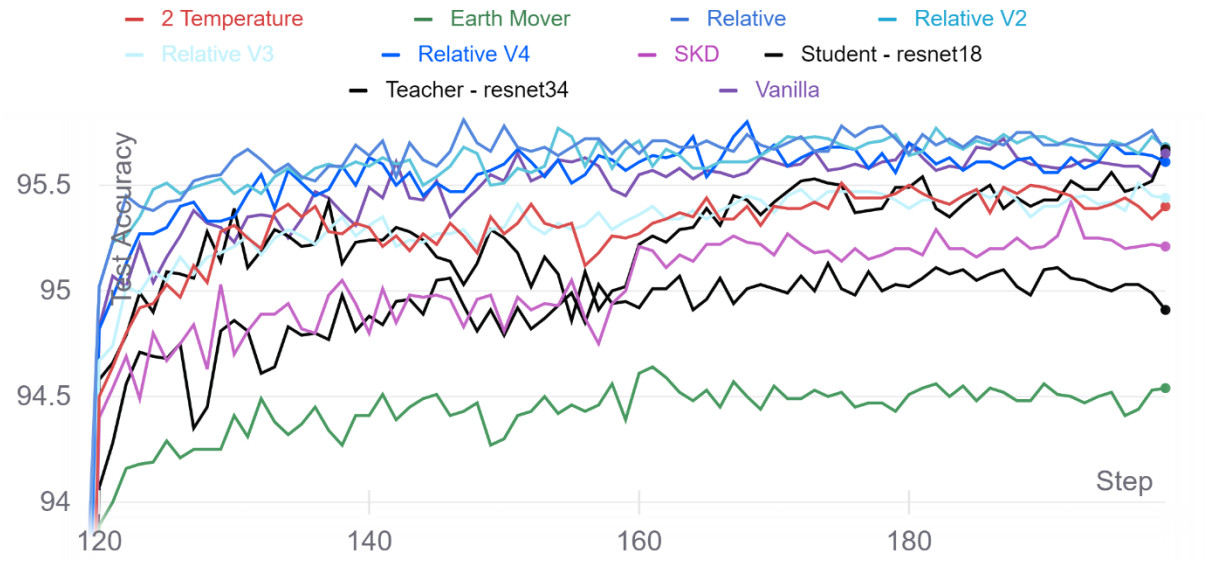

FIGURE 156 RESNET-34/RESNET-18 EXPERIMENTS ON CIFAR10

Two Temperature Distillation performs on par with RKD V3, improving on the student's baseline and RKD's main competitor, SKD, but not on Vanilla Distillation.

We ran the same experiments on Cifar100, expecting similar results. We expected the difference in the number of classes in the dataset to let RKD V3 perform much worse than the others or to dilute the knowledge of the smaller logits so much that it would not make a difference. As the graph and table below shows, the second interpretation seems to be true.

The best performing methods is still RKD V2, with V1 and V3 very close right after. It improves on Vanilla distillation by 0.25% and more importantly SKD by 0.80%. Two Temp

stays almost on par with SKD while Earth Mover distillation is still detrimental to the baseline student.

Teacher	Student	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
78.04	77.83	79.47	78.92	79.63	**79.72**	79.62	79.61	78.79	76.07

TABLE 4 RESNET-34/RESNET-18 ON CIFAR100

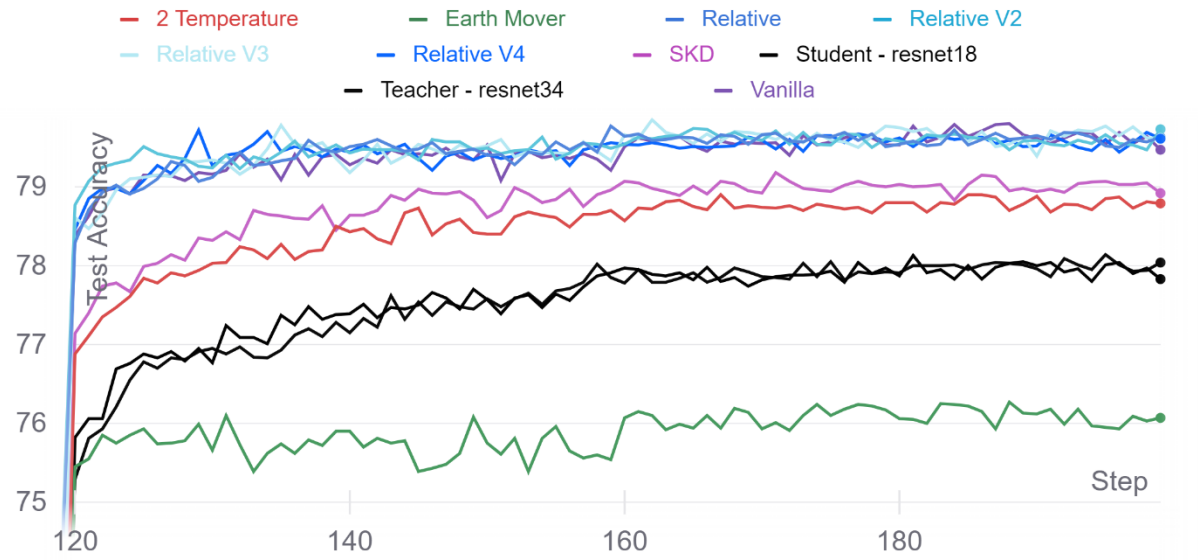

FIGURE 17 RESNET-34/RESNET-18 EXPERIMENTS ON CIFAR100

3.2.3 - WIDERESNET-40-1/WIDERESNET-16-1

WideResNet[19] is a version of ResNet modified to be shallower and therefore lighter, making up for the loss of layers and capacity by virtue of the widening factor, which is set to 1 in these experiments.

With this combination of architectures, Vanilla Distillation improves the baseline student by 1.05%. SKD performs very badly, with a loss of 0.80% in accuracy when even Earth Mover Distillation can slightly improve on the baseline. As in the other architectures, the best performing methods are RKD V1 and V2, improving 1.25% and 1.16% on baseline student.

It is however clear from the plot below that every method performs roughly the same in this case except for SKD and RKD V4.

Teacher	Student	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
93.78	91.10	92.16	90.29	**92.36**	92.27	90.77	91.14	92.1	92.05

TABLE 5 WIDERESNET-40-1/WIDERESNET-16-1 ON CIFAR10

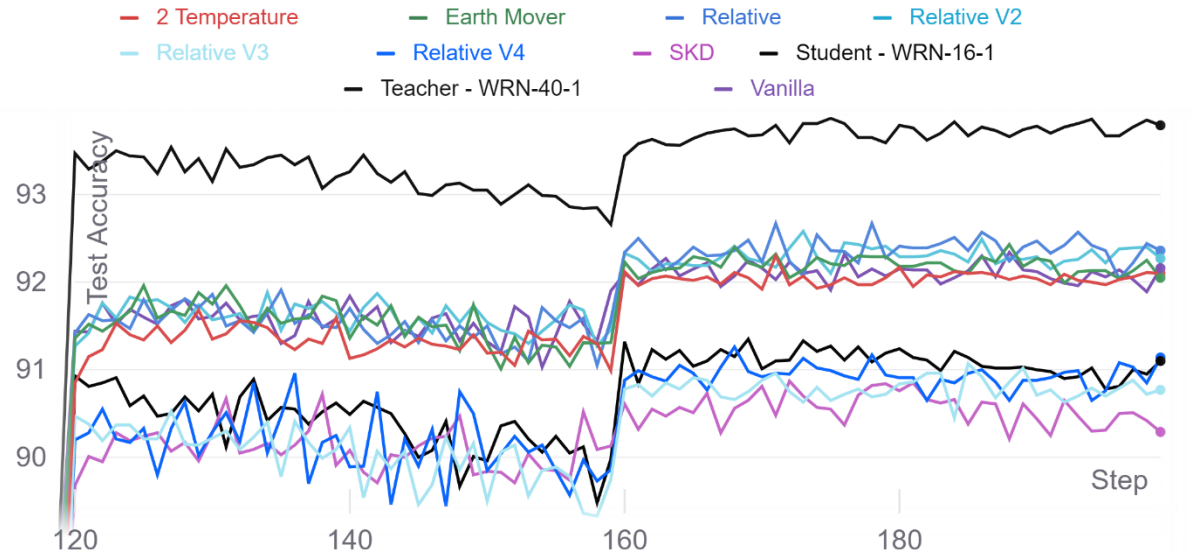

FIGURE 18 WIDERESNET-40-1/WIDERESNET-16-1 EXPERIMENTS ON CIFAR10

With Cifar100 the results are very different. There are two clear winners, RKD V1 and SKD, performing at 1.70% and 1.88% better than the baseline student and even 0.48% and 0.64% better than Vanilla. All the other versions of RKD performed on par or slightly worse than the baseline student.

The worst performing method is by far Earth Mover Distillation. In this case, it achieves an accuracy of 6.98%, showing that it clearly has big stability problems. We ran the experiments multiple times for this method and got the same result with every hyperparameters combinations.

WideResNet 40-1	WideResNet 16-1	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
69.61	66.39	67.63	68.1	**68.27**	65.64	63.52	66.16	66.23	6.98

TABLE 6 WIDERESNET-40-1/WIDERESNET-16-1 ON CIFAR100

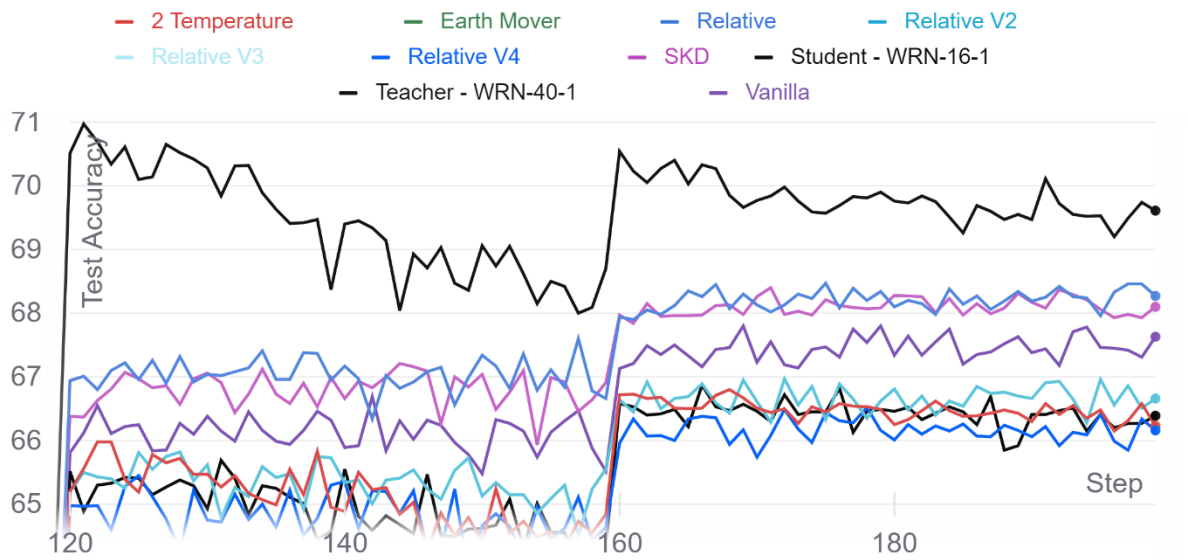

FIGURE 19 WIDERESNET-40-1/WIDERESNET-16-1 EXPERIMENTS ON CIFAR100

3.2.4 - WIDERESNET-40-2/WIDERESNET-16-2

The last combination of architectures that we tested is WideResNet-40-2 and WideResNet-16-2. For Cifar10, the results here are very similar to each other. RKD V4 and V2 are the only one to surpass the 94% accuracy threshold, together with the teacher. As usual, Earth Mover Distillation is detrimental to the student while all the other methods, including Vanilla Distillation and SKD, score between 93.80% and 93.98%.

Getting definitive conclusions from such close scores is very difficult. It could seem that both the smaller logits and the correction of incorrect predictions have a small influence on the distillation process.

WideResNet 40-2	WideResNet 16-2	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
94.69	93.4	93.85	93.95	93.98	94.02	93.81	**94.0.7**	93.86	91.93

TABLE 7 WIDERESNET-40-2/WIDERESNET-16-2 ON CIFAR10

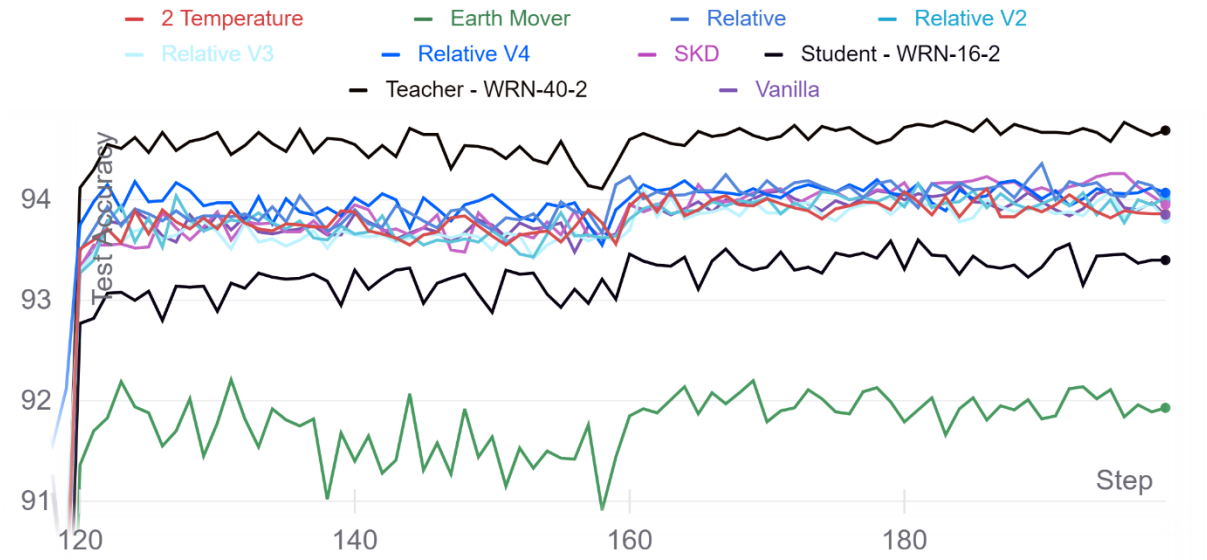

FIGURE 20 WIDERESNET-40-2/WIDERESNET-16-2 EXPERIMENTS ON CIFAR10

For what concerns Cifar100, there is a clear outlier in the plot below. One of the combinations of hyperparameters we tried for RKD V2 (alpha set at 0.5 and temperature at 2) produced an interesting student that would converge much faster than any other method to the final accuracy and surpass it by up to 4% 50 epochs before finishing training.

Here, Earth Mover Distillation is the worst performing method as usual and Two Temperature Distillation worsened the student.

WideResNet 40-2	WideResNet 16-2	Vanilla	SKD	RKD V1	RKD V2	RKD V3	RKD V4	Two Temp	Earth Mover
75.92	71.7	74.06	74.18	73.73	**74.63**	73.06	73.47	71.55	55.83

TABLE 8 WIDERESNET-40-2/WIDERESNET-16-2 ON CIFAR100

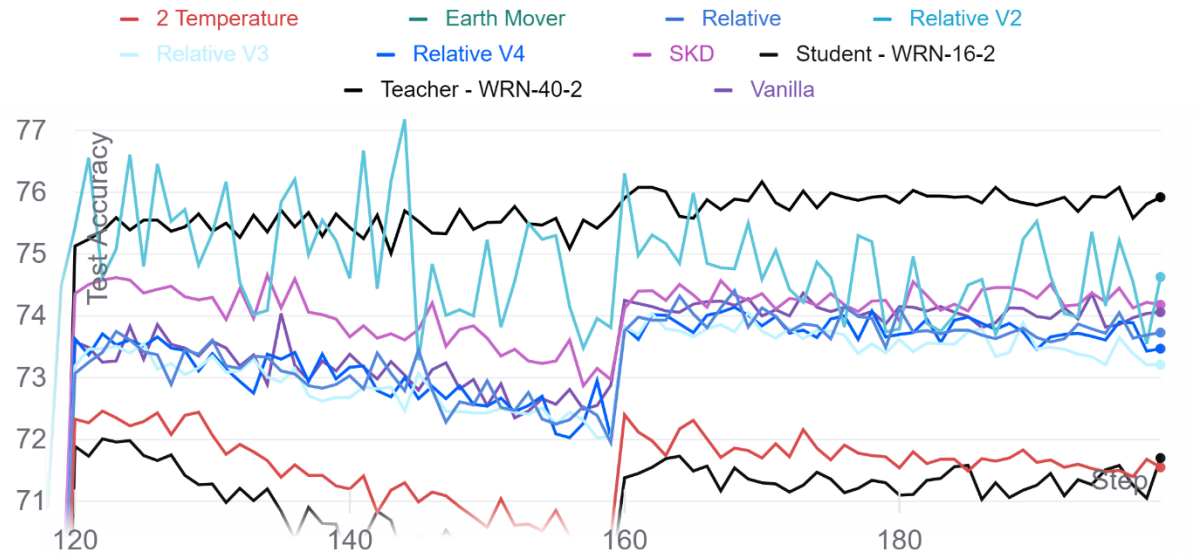

FIGURE 161 WIDERESNET-40-2/WIDERESNET-16-2 EXPERIMENTS ON CIFAR100

CONCLUSIONS

In this thesis we set out to improve Knowledge Distillation to produce better students. To do so, we proposed a few methods based on two main ideas: a) a too confident teacher generates a logits distribution that produces a Kullback-Leibler Divergence term redundant with respect to the Cross Entropy term of the loss function, b) the incorrect predictions pull the gradients in the wrong direction, which can be detrimental to the student's training.

We showed that reducing the confidence of the teacher in an Offline, Response Based Distillation setting with Relative Knowledge Distillation improves the results obtained by both Vanilla Distillation and Spherical Distillation. To do so, we proposed to reduce the distance between the higher logits in the teacher's distribution by considering the smaller logits as a kind of penalty due to the teacher's lack of confidence in its predictions. The experiments on RKD V3, in which the smaller logits are set to 0 after they have been subtracted or summed to the highest logit, showed that the smaller logits still convey a little amount of knowledge which can be useful in the distillation process, but the larger part of the knowledge distilled is embedded in the higher logits.

We also showed that correcting the mistaken predictions can convey a bit of knowledge useful in the distillation process. There are two ways to do this: after the SoftMax, but the distillation process becomes often detrimental to the student's training and is heavily dependent on the hyperparameters combinations, as it can be seen in the Earth Mover Distillation experiments, or in the logits. In the latter case, we showed with RKD V2 that it brings improvements to the distillation with respect to RKD V4, which is the same method without adjusting the logits. In Two Temperature Distillation we experimented with using two different temperature parameters to smoothen the logits when the teacher's predictions are correct or incorrect. The wrong prediction temperature was set to be much higher than the correct one to produce a much smoother distribution. It did not bring consistently significant improvements over Vanilla Distillation.

Finally, for what concerns the improvements on Knowledge Distillation, we showed that RKD V1 improves on Vanilla and Spherical Distillation by 1.23% on the ResNet152/ResNet50 combination, while in other cases does not bring any improvements. Most notably, RKD V2 brings improvements on almost every combination of architectures,

up to 0.8% over Vanilla and Spherical Distillation. This proved to be the most consistent of the methods we designed and experimented with.

RKD V3 and RKD V4 did not improve consistently the results of Vanilla and Spherical Distillation. RKD V3 often worsened the other two method's performances, most likely because of the loss of the knowledge in the smaller logits. RKD V4 proved to be the best method in just one case, in the other was often worse or on par with Vanilla Distillation.

In the future, it could be interesting to study how Relative Knowledge Distillation can be used in Feature Based Distillation frameworks, for example by adding classifiers head on top of feature maps and applying the RKD framework on each of them.

TABLE OF CONTENTS

FIGURES

TABLES

BIBLIOGRAPHY

[1] C. Buciluˇa, R. Caruana, and A. Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 535–541, New York, NY, USA, 2006. ACM

[2] Hinton, Geoffrey; Vinyals, Oriol; Dean, Jeff (2015). Distilling the knowledge in a neural network. arXiv:1503.02531

[3] Jianping Gou, Baosheng Yu, Stephen John Maybank, Dacheng Tao (2021). Knowledge Distillation: A Survey. arXiv:2006.05525

[4] S. Kullback e R.A. Leibler, *On Information and Sufficiency*, in *Annals of Mathematical Statistics*, vol. 22, n. 1, 1951, pp. 79–86, DOI:10.1214/aoms/1177729694

[5] Kim, S. W. & Kim, H. E. (2017). Transferring knowledge to smaller network with class-distance loss. In: ICLRW

[6] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In: ICLR.

[7] Zagoruyko, S. & Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: ICLR

[8] Chen, D., Mei, J. P., Zhang, Y., Wang, C., Wang, Z., Feng, Y., & Chen, C. (2021). Cross-Layer Distillation with Semantic Calibration. In: AAAI.

[9] Zhou, G., Fan, Y., Cui, R., Bian, W., Zhu, X. & Gai, K. (2018). Rocket launching: A universal and efficient framework for training well-performing light net. In: AAAI

[10] Yim, J., Joo, D., Bae, J. & Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR.

[11] Lee, S. & Song, B. (2019). Graph-based knowledge distillation by multi-head attention network. In: BMVC.

[12] Zhang, Y., Xiang, T., Hospedales, T. M. & Lu, H. (2018b). Deep mutual learning. In: CVPR.

[13] Chen, D., Mei, J. P., Wang, C., Feng, Y. & Chen, C. (2020a) Online knowledge distillation with diverse peers. In: AAAI.

[14] Micaelli, P. & Storkey, A. J. (2019). Zero-shot knowledge transfer via adversarial belief matching. In: NeurIPS.

[15] Chung, I., Park, S., Kim, J. & Kwak, N. (2020). Feature-map-level online adversarial knowledge distillation. In: ICML.

[16] Zhang, L., Song, J., Gao, A., Chen, J., Bao, C. & Ma, K. (2019b). Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: ICCV

[17] Yang, C., Xie, L., Su, C. & Yuille, A. L. (2019b). Snapshot distillation: Teacher-student optimization in one generation. In: CVPR.

[18] K. He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[19] Zagoruyko, S., Komodakis, N., Wide Residual Networks. arXiv:1605.07146v4