Advanced Deep Learning Reproducibility Project

Olivier Nicolini
oliviern@kth.se

Matteo Tadiello
tadiello@kth.se

Simone Zamboni
zamboni@kth.se

KTH Royal Institute of Technology
Department of EECS
Course DD2412
Stockholm 26-10-2019

1 Introduction

Latest neural networks are increasingly deep and wide and this, of course, makes them heavy and cumbersome. Their size allows them to learn from huge datasets and to be more accurate in many fields such as computer vision or natural language processing. The problem with this is the fact that these cumbersome models need very powerful hardware which not always is available and they often cannot be used in IoT applications due to their size. To avoid this problem, model compression techniques have been developed in the deep learning community, such as network pruning, knowledge distillation, and quantization. Knowledge distillation (Hinton et al.) is one of the most investigated ones, where large models are used to "distill" knowledge into a smaller network that tries to mimic the large one. This training settings is also referred to as "teacher-student".

According to the authors, the problem with compression techniques is due to the fact that to perform compression data is needed, an assumption that is not always true. It often happens that trained models are released without training data since that they could contain confidential information or they are very big in size.

For this reason, the authors introduce zero-shot knowledge transfer, a novel method designed to distil knowledge from a large model into a smaller one without the need of data by using a generator that produces pseudo-data used to compare teacher and student. They also define a measure of belief match between two networks in the vicinity of one's decision boundaries and demonstrate that the zero-shot model closely matches its teacher

In this work we try to reproduce the paper of Micaelli and Storkey, Zero-Shot Knowledge Transfer via Adversarial Belief Matching [7]. We test their results using the released code and then we analyze the results obtained after reimplementing their code from scratch. We also performed experiments changing the hyperparameters and the dataset to understand better how the proposed method behaves. Our code can be found at: https://github.com/SZamboni/advanceddeep

2 Background

2.1 Knowledge Distillation

In the latest years, an increasing interest in knowledge transfer techniques has risen. These techniques allow using the knowledge of a model gained in some application and use it as a starting point in another context. Among these, knowledge distillation is one of the most interesting ones. This method was first proposed by Hinton [1]. This technique compresses a large model (or an ensemble of models) into a smaller one. The smaller network, also called student, is trained to mimic the large one, called teacher. This is done by training the student using the outputs of the teacher as soft targets. Soft targets allow assigning probabilities also to all incorrect answers which can be used to understand how the teacher is able to generalize. As Hinton explains "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". This generalization capacity can be then transferred to the student by training it using the soft targets of the teacher. An interesting aspect of distillation is that when the soft targets have high entropy, they provide much more information than hard targets and have less variance in the gradient. For this reason, the student can be trained on less data than the teacher and with a higher learning rate.

After the introduction of knowledge distillation in the deep learning community, some attempts to ameliorate the method have been proposed. Among the latest models, Zagoruyko [12] proposed a modified distillation framework that also performs attention transfer, which the authors call kwonledge distillation + attention transfer (KD+AT). This allows to increase the performance of distilled networks by telling the student where the teacher is paying attention when it comes to classifying an image. More formally, attention is transferred from the teacher to the student by minimizing the L2 distance between the attention maps of the two networks at different layers and adding this/these term(s) to the classification loss.

2.2 Zero-Shot Transfer Learning

Micaelli and Storkey further developed Zagoruyko model by introducing a generator that is used to create fake (or pseudo) images. These images are then used to compare the results of the output of the student with the ones of the teacher. The generator is used in an adversarial fashion by having the opposite objective than the student, since the former wants to maximize student-teacher mismatch, measured using Kullback-Leibler Distance, and the latter wants to minimize it. For n_G times, the generator creates pseudo-images from a noise vector and then its weights are updated to maximize the KL Divergence between the student and teacher outputs from the generated images. For n_S times (with $n_S > n_G$) the student is trained on the generator images updated in order both to minimize the KL divergence and the attention transfer term. This attention transfer term is the same introduced by Zagoruyko, therefore the loss of the student is the following:

$$\mathcal{L}_S = D_{KL}(T(\mathbf{x}_p) \mid\mid S(\mathbf{x}_p)) + \beta \sum_{l}^{N_L} \left\| \frac{\mathbf{f}(A_l^{(t)})}{\left\|\mathbf{f}(A_l^{(t)})\right\|_2} - \frac{\mathbf{f}(A_l^{(s)})}{\left\|\mathbf{f}(A_l^{(s)})\right\|_2} \right\|_2 \tag{1}$$

$$\boldsymbol{f}(A_l) = (1/N_{A_l}) \sum_c \boldsymbol{a}_{lc}^2 \tag{2}$$

The model was tested by the authors on two datasets: SVHN [9] and CIFAR-10 [3]. For both datasets the authors used WideResNet (WRN) [13] architectures for both students and teachers, which are the same used by Zagoruyko in his work. After training the student with a teacher trained on those datasets, they compared their results with the KD+AT model by Zagoruyko and to the same models in the case they were trained from scratch on the whole dataset.

Our work, as part of the reproducibility challenge, will contribute in clarifying the implementation details of the zero-shot model, evaluate the results, and discuss a few experiments based on the insights obtained during the reproducibility challenge.

Algorithm 1: Zero-shot KT

```
pretrain: $T(\cdot)$
initialize: $G(\cdot;\phi)$
initialize: $S(\cdot;\theta)$
for $1, 2, ..., N$ do
    $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    for $1, 2, ..., n_G$ do
        $\boldsymbol{x}_p \leftarrow G(\boldsymbol{z};\phi)$
        $\mathcal{L}_G \leftarrow -D_{KL}(T(\boldsymbol{x}_p) \mid\mid S(\boldsymbol{x}_p))$
        $\phi \leftarrow \phi - \eta \frac{\partial L_G}{\partial \phi}$
    end
    for $1, 2, ..., n_S$ do
        $\mathcal{L}_S \leftarrow D_{KL}(T(\boldsymbol{x}_p) \mid\mid S(\boldsymbol{x}_p))$
        $\theta \leftarrow \theta - \eta \frac{\partial L_S}{\partial \theta}$
    end
    decay $\eta$
end
```

Algorithm 2: Transition curves between networks A and B, when stepping across decision boundaries of network A

```
pretrain: $net_A$
pretrain: $net_B$
for $\boldsymbol{x} \in \boldsymbol{X}_{test}$ do
    $i_A \equiv$ class of $\boldsymbol{x}$ according to $net_A$
    $i_B \equiv$ class of $\boldsymbol{x}$ according to $net_B$
    if $i_A = i_B = i$ then
        $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
        for $j \neq i$ do
            $\boldsymbol{x}_{adv} \leftarrow \boldsymbol{x}_0$
            for $1, 2, ..., K$ do
                $\boldsymbol{y}_A, \boldsymbol{y}_B \leftarrow$
                  $net_A(\boldsymbol{x}_{adv}), net_B(\boldsymbol{x}_{adv})$
                $\boldsymbol{x}_{adv} \leftarrow \boldsymbol{x}_{adv} - \xi \frac{\partial \mathcal{L}_{CE}(\boldsymbol{y}_A, j)}{\partial \boldsymbol{x}_{adv}}$
                save: $\boldsymbol{y}_A, \boldsymbol{y}_B$
            end
        end
    end
end
```

3 Original Implementation

The authors have made their code public, with one repository for the zero-shot [8] and one for the KD+AT [6]. We were able to run their code with some minor modifications. The results from KD+AT(as Table 1 shows) are very similar to the ones reported in the paper, and the results we obtained using their zero-shot code on CIFAR-10 (as Table 2 shows) are in five instances out of six better than the average results reported in the paper. The fact that our results are on average better than theirs could be explained by a lucky initialization if we take into account the reported standard deviations.

Therefore we can conclude that using their code is possible to reproduce the results in the paper.

Teacher	Student	Techer Scratch	Student Scratch	Their KD+AT	Paper KD+AT
WRN 16-2	WRN 16-1	94.09	91.09	84.54	84.54±0.21
WRN 40-1	WRN 16-1	93.13	91.09	82.57	81.71±0.25
WRN 40-2	WRN 16-1	93.13	91.09	81.74	81.25±0.67
WRN 40-1	WRN 16-2	94.72	94.09	86.65	85.74±0.47
WRN 40-2	WRN 16-2	94.72	94.09	85.95	86.39±0.33
WRN 40-2	WRN 40-1	94.72	93.13	87.26	87.35±0.12

Table 1: Experiment results using the authors' KD+AT code on CIFAR10. In the last column, the authors reported the mean and the standard deviation using three different seeds.

Teacher	Student	Teacher Scratch	Student Scratch	Zero-shot from the authors' code on GitHub	Paper zero-shot
WRN 16-2	WRN 16-1	94.09	91.09	83.54	82.44 ±0.21
WRN 40-1	WRN 16-1	93.13	91.09	81.65	70.93±1.11
WRN 40-2	WRN 16-1	93.13	91.09	84.29	83.69±0.58
WRN 40-1	WRN 16-2	94.72	94.09	88.15	86.60±0.56
WRN 40-2	WRN 16-2	94.72	94.09	89.22	89.71±0.10
WRN 40-2	WRN 40-1	94.72	93.13	88.18	86.60±1.79

Table 2: Experiment results using the authors' zero-shot code on CIFAR10. In the last column, the authors reported the mean and the standard deviation using three different seeds.

Teacher	Student	Teacher Scratch	Student Scratch	Zero-shot from the authors' code on github	Paper zero-shot
WRN 40-2	WRN 16-1	95.54	94.84	94.20	94.20 ±0.27

Table 3: Experiment results using the authors' code on SVHN.

4 Reimplementation details

To test the reproducibility of their experiments we tried to reimplement the proposed method and run the same experiments.

We first tried (and failed) to reimplement their algorithm in the Keras framework. Unfortunately, this framework was not flexible enough as we reached a point where it was not possible to train the student as they proposed. We found difficulties in getting the intermediate activations and use them in the loss, and in implementing an algorithm like the one previously described. Our repository contains also this failed attempt [10]. After this failure, we decided to switch to PyTorch where we were able to successfully reimplement the proposed method.

During the implementation, we found some missing information in the paper that we had to extract from their code. In particular, for the generator, it is stated: "We use a generic generator with only three convolutional layers, and our input noise z has 100 dimensions". This is a vague description of the generator and it is not sufficient for an exact reproduction of the results. Therefore, we had to look for the architecture of the generator in their code, which is composed of an initial fully connected layer, and then three blocks composed of an upsampling, a convolution with batch normalization and a Leaky ReLu.

Other missing information in the publication was the batch size, which in their code was set to 128. Moreover the learning rate is said to be 0.002 but in reality, that is true only for the student, while the generator has 0.001 in their code.

The loss of the student that uses the KL divergence with attention transfer was theoretically well explained but we had difficulties implementing it in practice. For this reason, we used the implementation of Zagoruyko (2016) that we found on GitHub[2]. We later found out that also the authors used the same implementation, therefore we are sure to have the same loss for the student as theirs.

The most difficult part to reproduce was how to train the generator. In particular, how to backpropagate from the generator loss to its parameters was not clear. We initially thought that it was something like a GAN, treating the student as a discriminator, but this approach did not work as we were not taking into account the teacher when backpropagating. The correct idea was to backpropagate through both the teacher and the student so that the generator could use the information contained in both architectures, but this was not clearly stated and no schema of the architecture was present.

4.1 Reproducibility Cost

Given that there was no mention in the paper about the required computational resources, we describe the computational cost required for running the experiments. Two full weeks were required to run all of our experiments. To do that, we set up 5 Virtual Machines on Google Cloud with a total of 7 Nvidia K80 GPUs, executing in parallel one experiment per each GPU available. When a teacher was a WRN 16-2 and the student was a WRN 16-1 the computation time of the experiment was around 36 hours, while in a setup teacher WRN 40-2 and student WRN 40-1 72 hours are required. The majority of our experiments are in a setup teacher WRN 40-2, student WRN 16-1, which requires around 44 hours of computing time on an Nvidia K80 GPU. Running all the experiments presented in this report, including the one in the appendix, required around 10000SEK of Google Cloud credits.

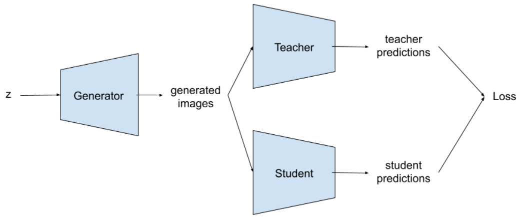

Figure 1: Proposed method architecture to train the generator.

4.2 Our reproducibility results

The first time we ran the same experiments as the authors on CIFAR-10 the obtained results were lower in accuracy compared to theirs. By checking their code we found out that we had a different learning rate: in their code the authors used an initial learning rate of 0.001 for the generator instead of 0.002 as stated. After correcting this problem we ran the experiment again, gaining around 2% on each experiment, with the results reported in Table 4, but our results are still lower than theirs.

Teacher	Student	Teacher Scratch	Student Scratch	Our zero-shot
WRN 16-2	WRN 16-1	92.69	90.50	72.47
WRN 40-1	WRN 16-1	93.24	90.50	69.75
WRN 40-2	WRN 16-1	94.29	90.50	77.35
WRN 40-1	WRN 16-2	93.24	92.69	81.86
WRN 40-2	WRN 16-2	94.29	92.69	83.39
WRN 40-2	WRN 40-1	94.29	92.69	84.20

Table 4: Our results on the dataset CIFAR-10

Our results are from 2% to 10% worse than those obtained using their code. Even after comparing our code to the authors' one we did not find any significant differences between the two.

In our implementation we used the same generator architecture as Micaelli's. The scheduler's learning rate is set to 0.001 with cosine annealing and the generator is trained using as loss the minus KL Divergence between the student and teacher predictions. Both the student and the teacher are WideResNets that output the final prediction and the intermediate activations. The student has 0.002 as learning rate and cosine annealing as learning rate scheduler, and is trained with the KL Divergence loss between the student and teacher predictions plus the attention transfer term. The batch size is set to 128 with 80k batches for CIFAR-10 and 50k for SVHN, n_G is set to 1 and n_S to 10. For the attention transfer β is set to 250 and the input dimension of the generator is 100. The teachers we used had very similar accuracy to those used in the original paper. Therefore, we can say we followed all the instructions present in the paper but we still weren't able to reproduce the results in the publication.

We suppose that there could be some untold implementation details that were not mentioned in the paper and that are not trivial to find inside their code that are needed to achieve higher accuracy.

Therefore, we conclude that the proposed method is able to train a student from a teacher without any data using a generator, but it is not possible to reproduce the exact results presented by the authors by only following the paper.

5 Beyond reproducibility

Even if we weren't able to exactly reproduce their results we were still intrigued by the proposed method and how it worked since the possibility to train a network in a zero-shot fashion is very interesting. Therefore, we did other experiments to have insights on how the method behaves.

5.1 Hyper-parameters testing

We first tested how the β parameter affected the network by using different betas from the baseline one (i.e. β=250). All results are reported in Table 5.1 and are obtained using a setup teacher WRN 40-2 and student WRN 16-1.

$\beta = 0$	$\beta = 100$	$\beta = 250$ (baseline)	$\beta = 1000$	$\beta = 2500$
73.47	75.81	77.35	75.16	72.23

Table 5: Different betas in a setup with teacher WRN 40-2 and student WRN 16-1.

As we can see from Table 5.1 the β used by the authors is the best in terms of accuracy, and this means that the authors searched extensively the right hyperparameter. Another experiment we tried was to completely eliminate the attention transfer by setting β=0. This model showed to be less accurate than the baseline by 5% but interestingly enough more accurate than the one using beta = 2500. This shows that attention in the student is not fundamental to make the algorithm converge but it can improve its performance.

We also tested how the algorithm performed by changing the number of times the generator was updated (n_G) from 1 for the whole training time to a more dynamic one where it changed by initially setting it to 1, then to 2 after 33% of the total number of batches and then to 4 after 66% of the batches. We found out that this way the student performed worse than in the case of having $n_G = 1$ the whole time (respectively 76.82 and 77.35). From our hyper-parameter experiments, we can conclude that the proposed architecture is very sensible to the parameter β and an extensive search for the best value is needed, while the parameter n_G is less influential.

5.2 Other teacher-student setups

Another experiment we tried was to use the same architecture for both student and teacher (WRN 16-2) to understand the theoretical limit of the proposed method and if the generalization error would decrease, but we found out that this combination leads to a worse accuracy for the student: from 92.69 to **87.58**.

We also tried to see the accuracy of a student training it using another student. For our experiment, we chose to go from a first 40-2 teacher to a 16-2 teacher/student to a final 16-1 student. The accuracy we got went from 83.39 of the 16-2 teacher/student model to **72.26** for the final 16-1 student. This value is interesting since it is very close to the one we obtained by training a 16-1 student from a 16-2 teacher trained from scratch on the whole CIFAR-10 dataset with an accuracy of 92.69.

5.3 Other datasets

We also tried to use other datasets to see how the algorithm performed. In particular, we tested on SVHN, which was the other dataset the authors tested their algorithm, MNIST [5], Fashion-MNIST [11], and CIFAR-100 [4]. The results can be seen in Table 5.3. The only dataset where we see a huge drop in performance was CIFAR-100 which is, of course, the hardest one since it has ten times more classes. We can confidently say that the result on CIFAR-100 could not improve with more batches because in our experience both on CIFAR-10 and on CIFAR-100 we gained only 2% in the last 40k batches. This shows that the method is not good enough for complex datasets with many classes and it needs some improvements.

Teacher	Student	Dataset	Techer Scratch	Student Scratch	Result	Number of Batches
WRN 40-2	WRN 16-1	SVHN	96.40	95.60	91.88	50k
WRN 40-2	WRN 16-1	MNIST	99.20	99.10	98.88	25k
WRN 40-2	WRN 16-1	Fashion-MNIST	94.01	93.13	87.38	35k
WRN 40-2	WRN 16-1	CIFAR-100	74.01	65.47	13.15	80k

Table 6: Our results on other datasets.

6 Transition Curves

In our experiments, we reimplemented the transition curves using only algorithm 2 defined in the paper. The algorithm was self-explanatory and we were able to correctly implement it.
We found some discrepancies between the publication and the provided code: the value K for the adversarial steps was 30 in the paper's images but in the code was set to 100, this is probably a choice for visualization.
We have used the same hyperparameters used in their code: $K = 100$, $Epsilon = 1$ and we have calculated the average of each step K over 1000 samples.

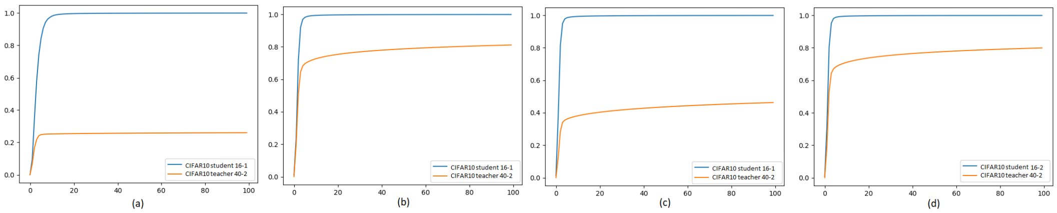

Figure 2: Transition Curves on CIFAR-10. (a) KD+AT using models trained on the authors' code, teacher 40-2, student 16-1. (b) zero-shot using models trained on the authors' code, teacher 40-2, student 16-1. (c) Our zero-shot, teacher 40-2, student 16-1. (d) Our zero-shot, teacher 40-2, student 16-2

The results demonstrate that the KD+AT student performs worst than any zero-shot student. We also noticed that the KD+AT curve stops to grow after 15-20 iteration, while the zero-shot curves grow slowly but continuously after that point. Moreover, the 16-1 student trained with our code follows the teacher less than the 16-1 student trained with their code. This is expected because our student performs worse than theirs. If we take a "better" student that performs similar to their, like our 16-2 (our 16-1 has 77.35 accuracy, our 16-2 has 83.89 accuracy, while their 16-1 has 84.29), we can see that its transition curve follows more closely the teacher. We also computed the MTE values (the code for that can be found online) and they are in line with what previously said and the values presented by the authors.

7 Conclusion

We have seen that the algorithm proposed by Micaelli and Storkey introduces a novel way to train a student without the need for any data but only using trained teacher. Their intuition of using a generator in an adversarial fashion has proven to work and, even if the reported results were not replicable, the proposed method opens up the possibility for future developments for other zero-shot approaches.
To conclude our reproducibility report we propose the following directions for future research and improvement of current results:

- Use another metric instead of the KL Divergence to compute the distance between the teacher and the student predictions, for example, the Wasserstein distance (that has shown to work well on GANs)

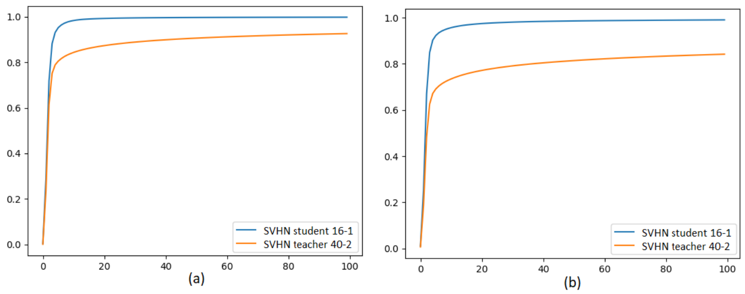

Figure 3: Transition Curves on SVHN. (a) zero-shot using models trained on the authors' code, teacher 40-2, student 16-1 with accuracy 94.20. (b) Our zero-shot, teacher 40-2, student 16-1 with accuracy 91.88. This little difference in accuracy could explain the little difference in the curves.

- Use other architectures for the student and teacher
- Improve the architecture of the generator, for example by trying to include the spectral normalization (that has shown to work well on GANs)
- Try to use as a teacher an ensemble of networks
- Study the behaviour of the method when applied to non-image datasets, for example on tasks that uses text or audio

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv e-prints*, page arXiv:1503.02531, Mar 2015.

[2] industky8.
wide-resnet.pytorch: https://github.com/indussky8/wide-resnet.pytorch, 2018.

[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[4] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[5] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[6] Paul Micaelli and Amos Storkey.
fewshotknowledgetransfer: https://github.com/polo5/fewshotknowledgetransfer, 2019.

[7] Paul Micaelli and Amos Storkey. Zero-shot Knowledge Transfer via Adversarial Belief Matching. *arXiv e-prints*, page arXiv:1905.09768, May 2019.

[8] Paul Micaelli and Amos Storkey.
zeroshotknowledgetransfer: https://github.com/polo5/zeroshotknowledgetransfer, 2019.

[9] Yuval Netzer, Tiejie Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[10] Olivier Nicolini, Matteo Tadiello, and Simone Zamboni.
advanceddeep: https://github.com/szamboni/advanceddeep, 2019.

[11] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, page arXiv:1708.07747, Aug 2017.

[12] Sergey Zagoruyko and Nikos Komodakis. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. *arXiv e-prints*, page arXiv:1612.03928, Dec 2016.

[13] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv e-prints*, page arXiv:1605.07146, May 2016.