Data-free model integration

Guy Shapira, Gal Sidi

October 2020

1 Abstract

Data accessibility is a critical need when it comes to solving problems using Knowledge Transfer approaches. Often, this assumption of an available dataset is not reasonable and does not hold. In many fields, the training dataset can contain personal or classified information, and thus accessibility to data over long periods of time might be an issue that can prevent the usage of knowledge transfer. Another restriction is in terms of memory when it comes to large and complex models. In [6] a method for data-free knowledge transfer has been presented. In this work we use this method of data generation from a pretrained model in order to create a dataset for knowledge transfer from multiple sources. Our work has shown that data-free knowledge transfer is possible even from more than one source. We extended prior work and generated images from several pretrained models in order to ingrate them into a single network model without using the original data. Our work also shown that the ability of decreasing model size is not harmed and combining several models into one can reduce the memory consumption over the split task.[1]

2 Introduction

The ability to transfer learned knowledge from a trained neural network to a new one with properties desirable for the task at hand has many appealing applications [6].

In the past, the field of knowledge distillation has been heavily studied, and many data driven solutions were suggested [1]. Those solutions has an obvious drawback, since it depended on the availability of the data post training. This can be very problematic both in terms of memory (networks like ResNet are often trained of millions of images [2]) but also in terms of legal availability to data (i.e. in the field of medicine, often data is only accessible for a very limited time due to privacy issues).

The concept of data free knowledge transfer was first introduced in [6], where they managed to generate "Deep Inverted" (as referenced in their paper) images

[1]implementation in https://github.com/Guy-Shapira/DeepInversion

from a pretrained model. They were able to produce syntactic images similar to the real data the pretrained model was trained on. This new concept is free from the mention drawback of the data based methods. In [6] it was shown this generated data is suitable for knowledge transfer. In the absence of prior data or metadata, an interesting question arises – can we achieve more then replicating the original model? For example, could it be used in order to merge few pretrained models (with distinct training tasks) in to one model.

In this work, we used the Adaptive DeepInversion method suggested in [6] in order to create a fully "deep inverted" dataset, from multiple pretrained models and introduced a new method for data-free model integration based on mentioned dataset. In order to show that our method can be used in practice, we demonstrate the integration of three diversion groups. Our main work is as follows:

1. We review an existing method for generation of deep inverted images (Sec 3.1).
2. We present a new method for data- free model integration , and discuss possible drawback for the method (Sec 3.2)
3. We address the task of data-free knowledge transfer between multiple teachers and a randomly initialized student network (Sec 4.1.2).
4. We suggest experiments that present our method's performance over variations in the number of teachers and the student architecture (Sec 4.2).

3 Methods

The main innovation presented in [6], is the ability to create syntactic data from a pretrained model, this syntactic data preserves the image distribution used to train the deep neural network (the pretrained model).
In the paper, this ability was used for three tasks:

1. Data free knowledge transfer.
2. Data free network pruning.
3. Data free continual learning.

(we would only focus on data free knowledge transfer in this work).
In their original approach, the authors used the deep inverted data (the generated syntactic data) in order to create a new model that mimic the pretrained model (without ever using the original data).

3.1 Generating Deep Inverted Images

In order to generated deep inverted images, the authors of [6] used the meta-data stored in the BatchNorm layers of a pretrained deep learning model. Usually a

BatchNorm layer is used in order to normalize the feature maps during training. But it also implicitly captures the channel wise means and variances of the data during training.
They suggested two similar methods (DeepInversion and Adaptive DeepInversion) in order to generate deep inverted images. Both of those methods are extensions of the method suggested in [4] (Google's "DeepDream") which was the first method suitable for optimizing a network that invert noise into images. Those methods are able to generate diverse and realistic images from noise using the BatchNorm meta-data. Their method include a regularization term that tries to minimize the distance between the feature map statistics of the generated data and BatchNorm meta-data for every BatchNorm layer in the model. They also prove that this feature distribution regularization substantially improves the quality of the generated images.
In our work we used the Adaptive DeepInversion method in order to generate data, this method has a GAN-like structure (i.e. two models competing with each other). Their method contains a student and a teacher, and the main idea is to encourage the synthesized images to cause student-teacher disagreement, and thus causes constant improvement of image creation.

3.2 Our extension

Our extension of the paper was to use the Adaptive DeepInversion method in order to achieve data-free model integration. We used said method in order to produce deep inverted data from multiple (distinct) pretrained deep learning models (which from now on will be called teachers) and combined them in to a single student network.
This integration of multiple models that preform classification task over different classes in to one model has an advantage in memory usage: using knowledge transfer methods we manged to lower the overall number of trainable parameters in the system without significantly harming the accuracy. And also allows simplifying a complex system that consists of multiple models in to a simpler architecture, and thus enabling a more user friendly interaction.
Moreover we believe our approach can be used for task integration (in to a single model) in fields where data privacy is a major issue. Often in those fields only the the trained model can be shared but not the original training data, thus there might not be an alternatives for using deep inverted images (or similarly synthetic images).
Finally, another significant advantage for our method is our ability to combat imbalanced data. In this method we have ability to generate seemingly infinite deep inverted images, this allows as to integrate models that were trained on datasetes that vastly differs in size (although since we are working without any information on the data, there is still an unsolved drawback in this area that will present in the next section).

3.2.1 Drawbacks

Though our extension provide a reliable method for data-free model integration (as can be seen in the Results section), the success of our method heavily relies on the following factors:

1. Unbalanced teachers - our method tries to mimic the output of several teachers (this will be explained in section 4.1), thus difference in the teachers accuracy or task complexity could lead to unbalanced student which only manage to mimic some of it's teachers.

2. Distinct tasks- our method relies on the fact that each input could only be classified as an instance of only one class, if classes are shared between teachers, the student would need to classify it's instances for several different classes, which results in low accuracy.

3. Wrong data distribution- since we are only given trained classifiers, and are not able to access their training data, our method cannot predict what was the original data distribution (how many instances were given for each class), and therefore image creation can be imbalance (better for some classes then for the rest), and our Synthetic data set may not preserve the original's distribution (this can be seen both as advantage and a disadvantage, based on if the original was balanced). In our work we decided to generate data from uniform distribution over all classes which may this could impair the training process, but without any other assumption on the data, we could not find an alternative.

4 Implementation and experiments

4.1 Implementation

Our implementation is constructed of the following parts:

1. Training models as teachers, each on a different part of the data. (In real use, this part should be replaced by using provided pretrained models).

2. Adjusting the image generation implementation of the paper in order to support a different dataset.

3. Training an augmented image generator based on the pretrained teachers.

4. Using the deep inverted images and the pretrained models in order to train a student model which combines the tasks of all it's teachers.

We used most of the original paper implementation for the deep inverted image generator (with slight modifications in order to support our desired dataset).

The image creation process would be fully describes in sub-section 4.1.1, and the knowledge-transfer method would also be explained in details in sub-section 4.1.2. The full implementation is based on PyTorch library and has a dependency on NVIDIA hardware (must be compatible to NVIDIA's Apex library).

4.1.1 Augmented images creation

The main innovation of [6] was the ability to generate data from a pretrained model that matches the original training dataset distribution. In order to achieve data-free training for the student model, the transfer-learning training process was based on augmented dreamed-images creation, simulating the data that the teacher model was trained on. Most of the code of the dreamed-images generation part was taken from the original paper implementation, which was originally suitable ImageNet[2]. The process of creating dreamed-images (as can be shown in Figure 1, where it shows the process over multiple, distinct teachers) is done by sampling random noise and a class label and synthesizing an image that would optimize both classification loss and simulation of the original dataset feature distribution of the pretrained-model. The class labels to generate were sampled uniformly (as explained in section 3).

The image's features distribution is inferred from the pretrained model BatchNorm layers, since those layers hold the channel-wise mean and variance of the original data in that layer representation.

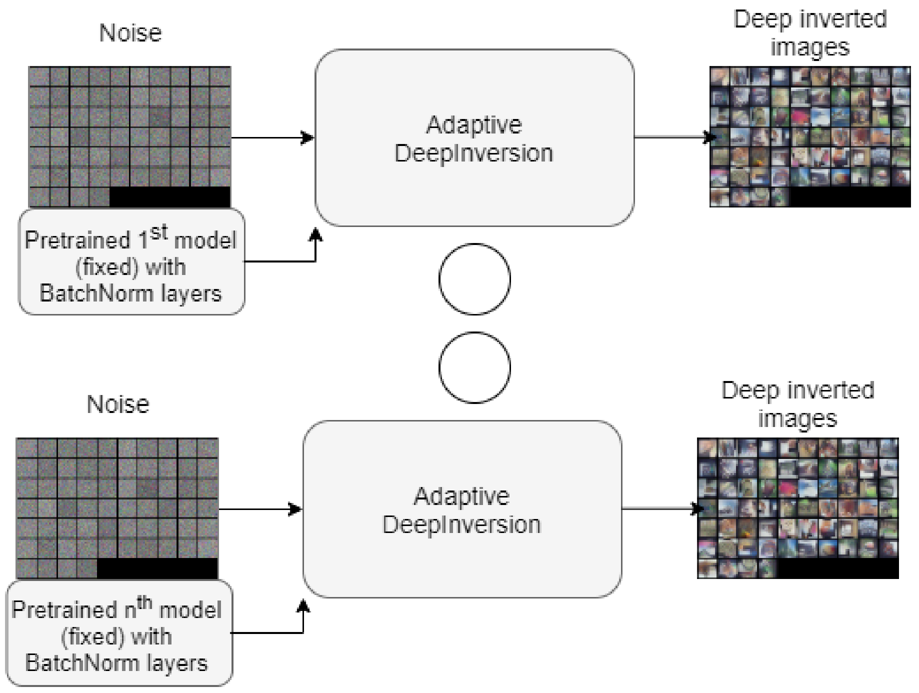
Noise
Deep inverted images
Adaptive DeepInversion
Pretrained 1st model (fixed) with BatchNorm layers
Noise
Deep inverted images
Adaptive DeepInversion
Pretrained nth model (fixed) with BatchNorm layers

Figure 1: Creation of the syntactic deep inverted data set from pretrained teachers

4.1.2 Training the student model

After the creation of the Deep Inverted the dataset we needed to adjust the adjusting knowledge transfer suggested in [6] in order to implement model integration (combining the knowledge from multiple teachers).

In order to mimic multiple teachers we constructed batches made of mini-batches. Each mini-batch is made of deep inverted images generated from a single teacher. In every batch exists one mini- batch from every one of the teachers. We calculated the loss over each of the different teachers mini-batches samples. After all the losses are calculated, we sum them in order to get the total loss over the entire batch containing all of the classes and only then make the back propagation step. That technique helped us to keep the balance between the different teachers in the learning process of the student. Since there must be a an overlap in the labels of the teachers (all start from 0) we had to shift some classes' designated label. For example, when training over a dataset that contains K classes and being split with N teachers, every single one of them trained over G = K/N classes, the classes learned from the teacher of index i in the student's classes will be the classes $[G \times i, G \times i + 1 - 1]$.
This part of the model integration process can be seen in figure 2.

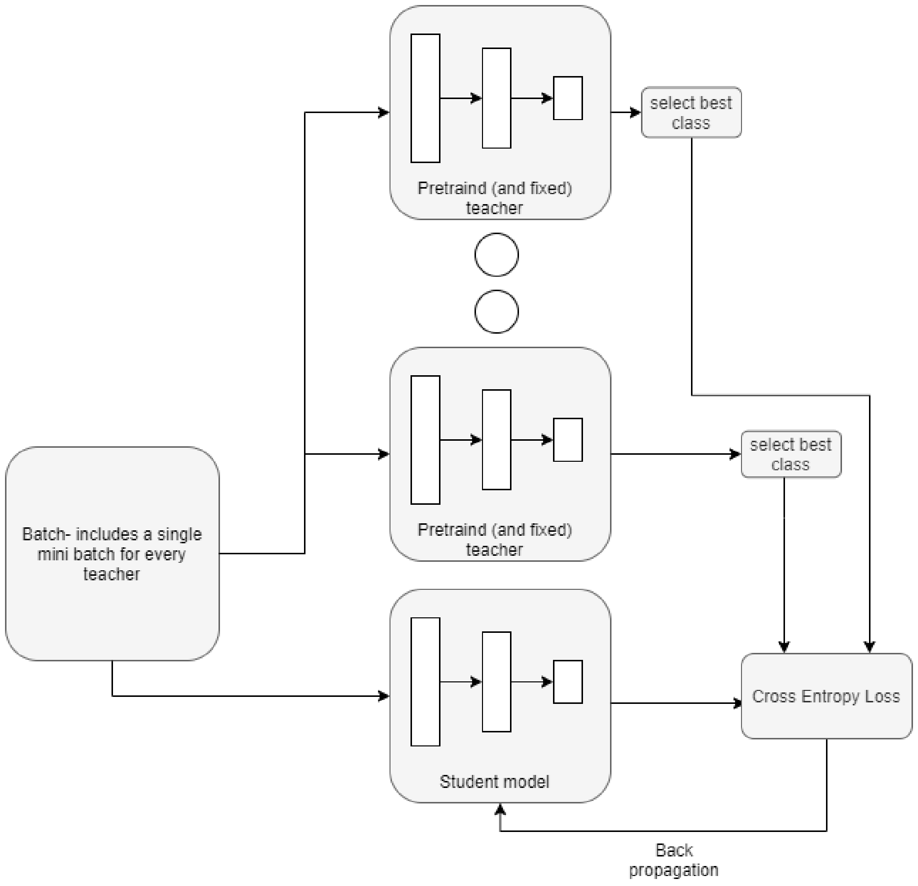
Pretraind (and fixed) teacher
select best class
Pretraind (and fixed) teacher
select best class
Batch- includes a single mini batch for every teacher
Student model
Cross Entropy Loss
Back propagation

Figure 2: Student's training process from multiple teachers

4.2 Experiments

We demonstrate our model integration methods on the same dataset (*CIFAR100*) in all our experiments. We measure the success of the integrated model (the student) on the same task, when it is always trained in a data-free method.
In the following experiments we test our method accuracy under the following settings: a) Varying the quantity of the teachers, b) Varying the student's architecture. For all image synthesis in this section we used *Adam* as an optimizer (with learning rate of 0.1). Data generation was adjusted to *CIFAR100* dataset. The generation model was adjusted with the following parameters: Batch size of 64, di_lr - lr for *deep inversion* was set to 0.085, di_var_scale - the TV L2 regularization coefficient was set to 0.001, di_l2_scale - the L2 regularization coefficient was set to 0, r_feature_weight - the weight for BatchNorm regularization statistic was set to 10, the iters_mi - number of iterations for model inversion was set to 2000.

In all of our experiments we decided to measure the *top-5* accuracy of both the student and it's teachers, instead of ordinary accuracy (top 1), since we believe this learning task is not equivalent to normal classification task over *CIFAR100*, but much more difficult, and thus can be measured using different methods.

4.2.1 Experiment I

In our first experiment we trained a student model as a classifier for *CIFAR100* [3]. In this experiment we wanted to find the optimal settings for our model integration method, both in terms of the quality and the quantity of the teachers. In order to to understand the effect the number of teachers (pretrained models) on the overall student accuracy we split the *CIFAR100* dataset in to $n \in \{ 2, 4, 10\}$ equal and distinct parts, and trained a *ResNet34* model on it (those models represent the teachers), generated the same amount of synthetic images from each teacher, and used the synthetic images for "data-free" knowledge transfer and model integration. In all cases, we generated the same amount of data, and used *ResNet34* as the student's architecture.
In this experiment we wanted to find the whether our method is more suitable when many and more accurate teachers were available (when the number of teachers was higher, each teacher was given less classes to classify over, resulting in better accuracy) or for the integration task of fewer teachers. On the one hand, learning from more teachers is more difficult to the student, since in our learning process, the student is trained to mimic all his teachers, and thus the training target is more complex. On the other hand, in our method, the teacher's accuracy is even more important then in regular knowledge transfer, since the synthetic images are generated from the teacher, less accurate teacher will lead to flawed and wrongly labeled data for our student, meaning that when the teacher's accuracy is low, the Deep Inverted images generated from it might not represent the original data at all.
Implementation details: All teachers (*ResNet34*) were trained with hyper-parameters suggested in [5]. The student (*ResNet34*) was trained with SGD optimizer (lr = 0.05, *momentum* = 0.9, *weight decay* = 0.0002, those values were chosen empirically after a grid search), and a batch size of 64. All results are after 125 epochs.
We used Cross Entropy Loss as our loss function (in a way that was explained in the beginning of this section).
In this experiment we slightly modified the *ResNet34* architecture, we added a dropout layer (with $p = 0.4$) in order to reduce over-fitting in early stages of the learning process,

4.2.2 Experiment II

In our second experiment we trained a student model as a classifier for *CIFAR100* [3]. In this experiment we tested the influence of the student architecture on the quality of the knowledge transfer.
In order to understand the effect of the architecture of the student on it's accu-

racy we repeated the first experiment (*CIFAR100* dataset split in to $n \in \{$ 2, 4, 10$\}$ trained using *ResNet34* models as teachers) using *ResNet18*, *ResNet34* and *ResNet50* as student. In this experiment we wanted to find whether deeper and more complex models are more suitable for knowledge transfer from multiple sources.
We find this experiment to be very interesting, since usually one of the main goals of knowledge transfer is to reduce the complexity (size) of the model. Thus we wanted to understand if our method that can obtain that preserve this advantage. Since this is not an ordinary knowledge transfer task (we have multiple teachers and are reliant on syntactic data), this may not be possible. Our method faces a significantly more difficult task, and we may need more complex models in order to achieve the desired result.
We assume that a that in our case, the student's model will only be able to achieve the learning task when it has at least the same number of parameters (same architecture) as it's teachers.

Usually this assumption is not ambitious, but since we are combining models at once this hypothesis needs to be checked.
Implementation details: All teachers (*ResNet34*) were trained with hyperparameters suggested in [5]. The students were trained with SGD optimizer (*lr* = 0.05, *momentum* = 0.9, *weight decay* = 0.0002, we also used a multi step learning rate with a factor of 0.5, that was activated after every 25 epochs). We had batches of 64 images each. All results are after 125 epochs. We used Cross Entropy Loss as the student's loss function .

5 Results and discussion

For all experiments in this section, We synthesize 32 ˆ 32 in batches of 64, using NVIDIA's GeForce RTX 2080 Ti. Each image batch receives 2k updates.
The same GPU was also used for model training.

5.1 Experiment I

The results for Experiment I can be seen in table 1.

Number of teachers	Teacher's top 1	Teacher's top 5	Student's top 5
2	76%	92.1%	90.175%
4	79%	95.2%	80.123%
10	82%	98.9%	71.462%

Table 1: Classification accuracy of *ResNet34* over the synthesized images, with regrades to number of teachers

As can be seen in the table, our student achieved better results when fewer pretrained teachers were given. This means, that our method is more suitable for the task of integrating few teachers then to task of integrating many and more accurate teachers.

We concluded that when the number of teachers increases, the training becomes much more complex, since the student tend to bias towards a subgroup of the teachers. In all our tests over 10 teachers, the student always had significant bias towards a group of 2 teachers (different teachers each time), and achieved much better results on them compared to the other 8. When we used 4 teachers, this phenomenon occur, but to lesser magnitude.

5.2 Experiment II

The results for Experiment 2 can be seen in table 2.

Number of teachers	Teacher's top 5	Student's architecture	Student's top 5
2	92.1%	ResNet18	86.671%
		ResNet34	90.175 %
		ResNet50	85.25%
4	95.2%	ResNet18	73.972%
		ResNet34	80.123%
		ResNet50	73.34%
10	98.9%	ResNet18	70.189%
		ResNet34	71.462%
		ResNet50	67.368%

Table 2: Classification accuracy of student w.r.t it's architecture and the number of pre-trained teachers

As can be seen in the table, in all three scenario, our best result was achieved using *ResNet34* for the student, i.e. the exact same architecture as it's teachers.

We found that using *ResNet50* was counter productive since it both the most complex network and also gave the worst results. We believe the mediocre results were caused by severe overfitting as textitResNet50 appears to be too deep and complex to match our classification problem.

We would like to highlight, that even though our method was found to be more suitable when the student's architecture was the same as it's teachers (*ResNet34*), we did manged to reduce the overall complexity, since we managed to distill their knowledge into a single network.

6 Acknowledgment

We would like to thank Hongxu Yin and Pavlo Molchanov (both are authors of the paper we based this work on) for their help with the deep inverted image creation process and helpful discussion.
We would also like to thank Ilya Kolchinsky for supplying us the resources for this project.

References

[1] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. *Net2Net: Accelerating Learning via Knowledge Transfer*. 2016. arXiv: `1511.05641 [cs.LG]`.

[2] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[3] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[4] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going Deeper into Neural Networks*. 2015. URL: `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[5] *Pytorch-cifar100*. URL: `https://github.com/weiaicunzai/pytorch-cifar100`.

[6] Hongxu Yin et al. "Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion". In: *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*. June 2020.