Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion

Hongxu Yin[1,2†*], Pavlo Molchanov[1*], Jose M. Alvarez[1], Zhizhong Li[1,3†],
Arun Mallya[1], Derek Hoiem[3], Niraj K. Jha[2], and Jan Kautz[1]

[1]NVIDIA, [2]Princeton University, [3]University of Illinois at Urbana-Champaign

{hongxuy, jha}@princeton.edu, {zli115, dhoiem}@illinois.edu,
{pmolchanov, josea, amallya, jkautz}@nvidia.com

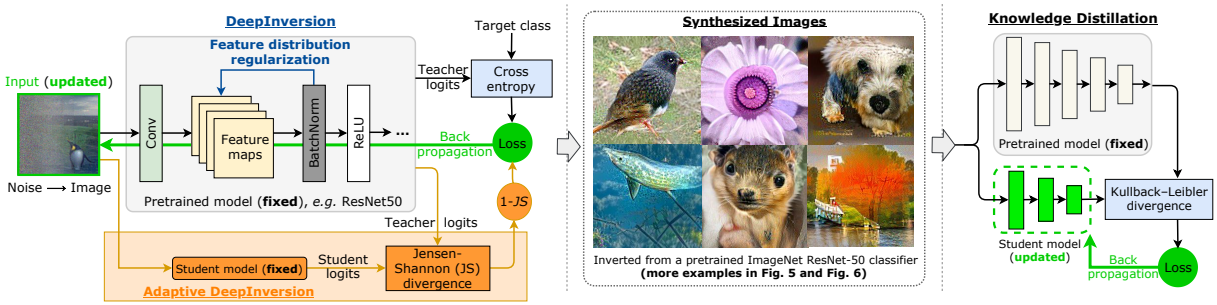


Figure 1: We introduce DeepInversion, a method that optimizes random noise into high-fidelity class-conditional images given just a pretrained CNN (teacher), in Sec. 3.2. Further, we introduce Adaptive DeepInversion (Sec. 3.3), which utilizes both the teacher and application-dependent student network to improve image diversity. Using the synthesized images, we enable data-free pruning (Sec. 4.3), introduce and address data-free knowledge transfer (Sec. 4.4), and improve upon data-free continual learning (Sec. 4.5).

Abstract

We introduce DeepInversion, a new method for synthesizing images from the image distribution used to train a deep neural network. We "invert" a trained network (teacher) to synthesize class-conditional input images starting from random noise, without using any additional information on the training dataset. Keeping the teacher fixed, our method optimizes the input while regularizing the distribution of intermediate feature maps using information stored in the batch normalization layers of the teacher. Further, we improve the diversity of synthesized images using Adaptive DeepInversion, which maximizes the Jensen-Shannon divergence between the teacher and student network logits. The resulting synthesized images from networks trained on the CIFAR-10 and ImageNet datasets demonstrate high fidelity and degree of realism, and help enable a new breed of data-free applications – ones that do not require any real images or labeled data. We demonstrate the applicability of our proposed method to three tasks of immense practical importance – (i) data-free network pruning, (ii) data-free knowledge transfer, and (iii) data-free continual learning. Code is available at `https://github.com/NVlabs/DeepInversion`.

1. 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. For example, one might want to use a more resource-efficient architecture for deployment on edge inference devices [44, 66, 76], or to adapt the network to the inference hardware [9, 63, 71], or for continually learning to classify new image classes [29, 34], *etc*. Most current solutions for such knowledge transfer tasks are based on the concept of knowledge distillation [20], wherein the new network (student) is trained to match its outputs to that of a previously trained network (teacher). However, all such methods have a significant constraint – they assume that either the previously used training dataset is available [8, 29, 45, 57], or some real images representative of the prior training dataset distribution are available [25, 26, 34, 56]. Even methods not based on distillation [27, 50, 74] assume that some additional statistics about prior training is made available by the pretrained model provider.

The requirement for prior training information can be very restrictive in practice. For example, suppose a very deep network such as ResNet-152 [18] was trained on datasets with millions [10] or even billions of images [36], and we wish to distill its knowledge to a lower-latency model such as ResNet-18. In this case, we would need access to these datasets, which are not only large but difficult to store, trans-

*Equal contribution. † Work done during an internship at NVIDIA.
Work supported in part by ONR MURI N00014-16-1-2007.

fer, and manage. Further, another emerging concern is that of data privacy. While entities might want to share their trained models, sharing the training data might not be desirable due to user privacy, security, proprietary concerns, or competitive disadvantage.

In the absence of prior data or metadata, an interesting question arises – can we somehow recover training data from the already trained model and use it for knowledge transfer? A few methods have attempted to visualize what a trained deep network expects to see in an image [3, 37, 46, 49]. The most popular and simple-to-use method is DeepDream [46]. It synthesizes or transforms an input image to yield high output responses for chosen classes in the output layer of a given classification model. This method optimizes the input (random noise or a natural image), possibly with some regularizers, while keeping the selected output activations fixed, but leaves intermediate representations constraint-free. The resulting "dreamed" images lack natural image statistics and can be quite easily identified as unnatural. These images are also not very useful for the purposes of transferring knowledge, as our extensive experiments in Section 4 show.

In this work, we make an important observation about deep networks that are widely used in practice – they all implicitly encode very rich information about prior training data. Almost all high-performing convolutional neural networks (CNNs), such as ResNets [18], DenseNets [22], or their variants, use the batch normalization layer [24]. These layers store running means and variances of the activations at multiple layers. In essence, they store the history of previously seen data, at multiple levels of representation. By assuming that these intermediate activations follow a Gaussian distribution with mean and variance equal to the running statistics, we show that we can obtain "dreamed" images that are realistic and much closer to the distribution of the training dataset as compared to prior work in this area.

Our approach, visualized in Fig. 1, called *DeepInversion*, introduces a regularization term for intermediate layer activations of dreamed images based on just the two layer-wise statistics: mean and variance, which are directly available with trained models. As a result, we do not require any training data or metadata to perform training image synthesis. Our method is able to generate images with high fidelity and realism at a high resolution, as can be seen in the middle section of Fig. 1, and more samples in Fig. 5 and Fig. 6.

We also introduce an application-specific extension of *DeepInversion*, called *Adaptive DeepInversion*, which can enhance the diversity of the generated images. More specifically, it exploits disagreements between the pretrained teacher and the in-training student network to expand the coverage of the training set by the synthesized images. It does so by maximizing the Jensen-Shannon divergence between the responses of the two networks.

In order to show that our dataset synthesis method is useful in practice, we demonstrate its effectiveness on three different use cases. First, we show that the generated images support knowledge transfer between two networks using distillation, even with different architectures, with a minimal accuracy loss on the simple CIFAR-10 as well as the large and complex ImageNet dataset. Second, we show that we can prune the teacher network using the synthesized images to obtain a smaller student on the ImageNet dataset. Finally, we apply DeepInversion to continual learning that enables the addition of new classes to a pretrained CNN without the need for any original data. Using our DeepInversion technique, we empower a new class of "data-free" applications of immense practical importance, which need neither any natural image nor labeled data.

Our main contributions are as follows:

- We introduce DeepInversion, a new method for synthesizing class-conditional images from a CNN trained for image classification (Sec. 3.2). Further, we improve synthesis diversity by exploiting student-teacher disagreements via Adaptive DeepInversion (Sec. 3.3).
- We enable data-free and hardware-aware pruning that achieves performance comparable to the state-of-the-art (SOTA) methods that rely on the training dataset (Sec. 4.3).
- We introduce and address the task of data-free knowledge transfer between a teacher and a randomly initialized student network (Sec. 4.4).
- We improve prior work on data-free continual (a.k.a. incremental) learning, and achieve results comparable to oracle methods given the original data (Sec. 4.5).

2. Related Work

Knowledge distillation. Transfer of knowledge from one model to another was first introduced by Breiman and Shang when they learned a single decision tree to approximate the outputs of multiple decision trees [4]. Similar ideas are explored in neural networks by Bucilua *et al.* [6], Ba and Caruana [2], and Hinton *et al.* [20]. Hinton *et al*. formulate the problem as "knowledge distillation," where a compact student mimics the output distributions of expert teacher models [20]. These methods and improved variants [1, 53, 57, 67, 73] enable teaching students with goals such as quantization [42, 55], compact neural network architecture design [57], semantic segmentation [31], self-distillation [14], and un-/semi-supervised learning [34, 54, 70]. All these methods still rely on images from the original or proxy datasets. More recent research has explored data-free knowledge distillation. Lopes *et al*. [33] synthesize inputs based on pre-stored auxiliary layer-wise statistics of the teacher network. Chen *et al.* [7] train a new generator network for image generation while treating the teacher network as a fixed discriminator. Despite remarkable

insights, scaling to tasks such as ImageNet classification, remains difficult for these methods.

Image synthesis. Generative adversarial networks (GANs) [16, 43, 47, 75] have been at the forefront of generative image modeling, yielding high-fidelity images, *e.g.*, using BigGAN [5]. Though adept at capturing the image distribution, training a GAN's generator requires access to the original data.

An alternative line of work in security focuses on image synthesis from a single CNN. Fredrikson *et al*. [13] propose the *model inversion* attack to obtain class images from a network through a gradient descent on the input. Follow-up works have improved or expanded the approach to new threat scenarios [19, 64, 68]. These methods have only been demonstrated on shallow networks, or require extra information (*e.g.*, intermediate features).

In vision, researchers visualize neural networks to understand their properties. Mahendran *et al*. explore inversion, activation maximization, and caricaturization to synthesize "natural pre-images" from a trained network [37, 38]. Nguyen *et al*. use a trained GAN's generator as a prior to invert trained CNNs [48] to images, and its followup Plug & Play [47] further improves image diversity and quality via latent code prior. Bhardwaj *et al*. use the training data cluster centroids to improve inversion [3]. These methods still rely on auxiliary dataset information or additional pretrained networks. Of particular relevance to this work is DeepDream [46] by Mordvintsev *et al*., which has enabled the "dreaming" of new object features onto natural images given a single pretrained CNN. Despite notable progress, synthesizing high-fidelity and high-resolution natural images from a deep network remains challenging.

3. Method

Our new data-free knowledge distillation framework consists of two steps: (i) model inversion, and (ii) application-specific knowledge distillation. In this section, we briefly discuss the background and notation, and then introduce our *DeepInversion* and *Adaptive DeepInversion* methods.

3.1. Background

Knowledge distillation. Distillation [20] is a popular technique for knowledge transfer between two models. In its simplest form, first, the teacher, a large model or ensemble of models, is trained. Second, a smaller model, the student, is trained to mimic the behavior of the teacher by matching the temperature-scaled soft target distribution produced by the teacher on training images (or on other images from the same domain). Given a trained model p_T and a dataset $\mathcal{X}$, the parameters of the student model, $\mathbf{W}_S$, can be learned by

$$\min_{\mathbf{W}_S} \sum_{x \in \mathcal{X}} \text{KL}(p_T(x), p_S(x)), \tag{1}$$

where $\text{KL}(\cdot)$ refers to the Kullback-Leibler divergence and $p_T(x) = p(x, \mathbf{W}_T)$ and $p_S(x) = p(x, \mathbf{W}_S)$ are the output distributions produced by the teacher and student model, respectively, typically obtained using a high temperature on the softmax inputs [20].

Note that ground truths are not required. Despite its efficacy, the process still relies on real images from the same domain. Below, we focus on methods to synthesize a large set of images $\hat{x} \in \hat{\mathcal{X}}$ from noise that could replace $x \in \mathcal{X}$.

DeepDream [46]. Originally formulated by Mordvintsev *et al*. to derive artistic effects on natural images, DeepDream is also suitable for optimizing noise into images. Given a randomly initialized input ($\hat{x} \in \mathcal{R}^{H \times W \times C}$, H, W, C being the height, width, and number of color channels) and an arbitrary target label y, the image is synthesized by optimizing

$$\min_{\hat{x}} \mathcal{L}(\hat{x}, y) + \mathcal{R}(\hat{x}), \tag{2}$$

where $\mathcal{L}(\cdot)$ is a classification loss (e.g., cross-entropy), and $\mathcal{R}(\cdot)$ is an image regularization term. DeepDream uses an image prior [11, 37, 49, 61] to steer $\hat{x}$ away from unrealistic images with no discernible visual information:

$$\mathcal{R}_{\text{prior}}(\hat{x}) = \alpha_{\text{tv}} \mathcal{R}_{\text{TV}}(\hat{x}) + \alpha_{\ell_2} \mathcal{R}_{\ell_2}(\hat{x}), \tag{3}$$

where R_{TV} and R_{ℓ_2} penalize the total variance and ℓ_2 norm of $\hat{x}$, respectively, with scaling factors α_{tv}, α_{ℓ_2}. As both prior work [37, 46, 49] and we empirically observe, image prior regularization provides more stable convergence to valid images. However, these images still have a distribution far different from natural (or original training) images and thus lead to unsatisfactory knowledge distillation results.

3.2. DeepInversion (DI)

We improve DeepDream's image quality by extending image regularization $\mathcal{R}(\hat{x})$ with a new feature distribution regularization term. The image prior term defined previously provides little guidance for obtaining a synthetic $\hat{x} \in \hat{\mathcal{X}}$ that contains similar low- and high-level features as $x \in \mathcal{X}$. To effectively enforce feature similarities at all levels, we propose to minimize the distance between feature map statistics for $\hat{x}$ and x. We assume that feature statistics follow the Gaussian distribution across batches and, therefore, can be defined by mean μ and variance σ^2. Then, the *feature distribution regularization* term can be formulated as:

$$\begin{aligned} \mathcal{R}_{\text{feature}}(\hat{x}) = &\sum_l || \, \mu_l(\hat{x}) - \mathbb{E}(\mu_l(x)|\mathcal{X}) \, ||_2 + \\ &\sum_l || \, \sigma_l^2(\hat{x}) - \mathbb{E}(\sigma_l^2(x)|\mathcal{X}) \, ||_2, \end{aligned} \tag{4}$$

where $\mu_l(\hat{x})$ and $\sigma_l^2(\hat{x})$ are the batch-wise mean and variance estimates of feature maps corresponding to the l^{th} convolutional layer. The $\mathbb{E}(\cdot)$ and $||\cdot||_2$ operators denote the expected value and ℓ_2 norm calculations, respectively.

It might seem as though a set of training images would be required to obtain $\mathbb{E}(\mu_l(x)|\mathcal{X})$ and $\mathbb{E}(\sigma_l^2(x)|\mathcal{X})$, but the running average statistics stored in the widely-used BatchNorm (BN) layers are more than sufficient. A BN layer normalizes the feature maps during training to alleviate covariate shifts [24]. It implicitly captures the channel-wise means and variances during training, hence allows for estimation of the expectations in Eq. 4 by:

$$\mathbb{E}\big(\mu_l(x)|\mathcal{X}\big) \simeq \text{BN}_l(\text{running\_mean}), \tag{5}$$

$$\mathbb{E}\big(\sigma_l^2(x)|\mathcal{X}\big) \simeq \text{BN}_l(\text{running\_variance}). \tag{6}$$

As we will show, this feature distribution regularization substantially improves the quality of the generated images. We refer to this model inversion method as *DeepInversion* – a generic approach that can be applied to any trained *deep* CNN classifier for the *inversion* of high-fidelity images. $R(\cdot)$ (corr. to Eq. 2) can thus be expressed as

$$\mathcal{R}_{\text{DI}}(\hat{x}) = \mathcal{R}_{\text{prior}}(\hat{x}) + \alpha_{\text{f}}\mathcal{R}_{\text{feature}}(\hat{x}). \tag{7}$$

3.3. Adaptive DeepInversion (ADI)

In addition to quality, diversity also plays a crucial role in avoiding repeated and redundant synthetic images. Prior work on GANs has proposed various techniques, such as min-max training competition [15] and the truncation trick [5]. These methods rely on the joint training of two networks over original data and therefore are not applicable to our problem. We propose *Adaptive DeepInversion*, an enhanced image generation scheme based on a novel iterative competition scheme between the image generation process and the student network. The main idea is to encourage the synthesized images to cause student-teacher disagreement. For this purpose, we introduce an additional loss $\mathcal{R}_{\text{compete}}$ for image generation based on the Jensen-Shannon divergence that penalizes output distribution similarities,

$$\mathcal{R}_{\text{compete}}(\hat{x}) \quad = \quad 1 - \text{JS}(p_T(\hat{x}), p_S(\hat{x})), \tag{8}$$

$$\text{JS}(p_T(\hat{x}), p_S(\hat{x})) = \frac{1}{2}\Bigg(\text{KL}(p_T(\hat{x}), M) + \text{KL}(p_S(\hat{x}), M)\Bigg),$$

where $M = \frac{1}{2}\cdot\big(p_T(\hat{x}) + p_S(\hat{x})\big)$ is the average of the teacher and student distributions.

During optimization, this new term leads to new images the student cannot easily classify whereas the teacher can. As illustrated in Fig. 2, our proposal iteratively expands the distributional coverage of the image distribution during the learning process. With competition, regularization $R(\cdot)$ from Eq. 7 is updated with an additional loss scaled by α_c as

$$\mathcal{R}_{\text{ADI}}(\hat{x}) = \mathcal{R}_{\text{DI}}(\hat{x}) + \alpha_c\mathcal{R}_{\text{compete}}(\hat{x}). \tag{9}$$

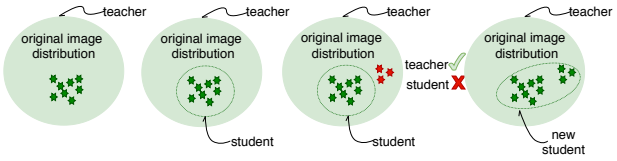


Figure 2: Illustration of the Adaptive DeepInversion competition scheme to improve image diversity. Given a set of generated images (shown as green stars), an intermediate student can learn to capture part of the original image distribution. Upon generating new images (shown as red stars), competition encourages new samples out of student's learned knowledge, improving distributional coverage and facilitating additional knowledge transfer. Best viewed in color.

3.4. DeepInversion vs. Adaptive DeepInversion

DeepInversion is a generic method that can be applied to any trained CNN classifier. For knowledge distillation, it enables a one-time synthesis of a large number of images given the teacher, to initiate knowledge transfer. Adaptive DeepInversion, on the other hand, needs a student in the loop to enhance image diversity. Its competitive and interactive nature favors constantly-evolving students, which gradually force new image features to emerge, and enables the augmentation of DeepInversion, as shown in our experiments.

4. Experiments

We demonstrate our inversion methods on datasets of increasing size and complexity. We perform a number of ablations to evaluate each component in our method on the simple CIFAR-10 dataset (32×32 pixels, 10 classes). Then, on the complex ImageNet dataset (224×224 pixels, 1000 classes), we show the success of our inversion methods on three different applications under the data-free setting: (a) pruning, (b) knowledge transfer, and (c) continual (class-incremental) learning. In all experiments, image pixels are initialized $i.i.d.$ from Gaussian noise of $\mu = 0$ and $\sigma = 1$.

4.1. Results on CIFAR-10

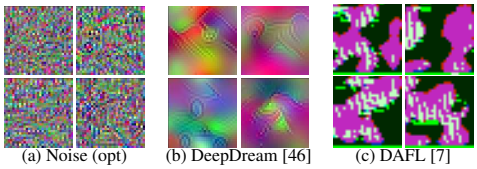
For validating our design choices, we consider the task of data-free knowledge distillation, where we teach a student network randomly initialized from scratch.

Implementation details. We use VGG-11-BN and ResNet-34 networks pretrained on CIFAR-10 as the teachers. For all image synthesis in this section, we use Adam for optimization (learning rate 0.05). We generate 32×32 images in batches of 256. Each image batch requires 2k gradient updates. After a simple grid search optimizing for student accuracy, we found $\alpha_{\text{tv}} = 2.5 \cdot 10^{-5}, \alpha_{\ell_2} = 3 \cdot 10^{-8}$, and $\alpha_f = \{1.0, 5.0, 10.0, 100.0\}$ work best for DeepInversion, and $\alpha_c = 10.0$ for Adaptive DeepInversion. See supplementary materials for more details.

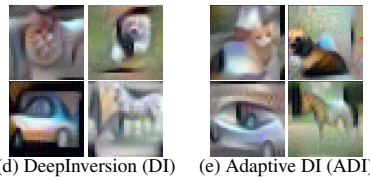
Baselines – Noise & DeepDream [46]. From Table 1, we observe that optimized noise, Noise ($\mathcal{L}$), does not provide any support for knowledge distillation – a drastic change

Teacher Network	VGG-11	VGG-11	ResNet-34
Student Network	VGG-11	ResNet-18	ResNet-18
Teacher accuracy	92.34%	92.34%	95.42%
Noise ($\mathcal{L}$)	13.55%	13.45%	13.61%
$+\mathcal{R}_{\text{prior}}$ (DeepDream [46])	36.59%	39.67%	29.98%
$+\mathcal{R}_{\text{feature}}$ (DeepInversion)	84.16%	83.82%	91.43%
$+\mathcal{R}_{\text{compete}}$ (ADI)	**90.78%**	**90.36%**	**93.26%**
DAFL [7]	–	–	92.22%

Table 1: Data-free knowledge transfer to various students on CIFAR-10. For ADI, we generate one new batch of images every 50 knowledge distillation iterations and merge the newly generated images into the existing set of generated images.



(a) Noise (opt) (b) DeepDream [46] (c) DAFL [7]



(d) DeepInversion (DI) (e) Adaptive DI (ADI)

Figure 3: 32×32 images generated by inverting a ResNet-34 trained on CIFAR-10 with different methods. All images are correctly classified by the network, clockwise: cat, dog, horse, car.

in input distribution disrupts the teacher and impacts the validity of the transferred knowledge. Adding R_{prior}, like in DeepDream, slightly improves the student's accuracy.

Effectiveness of DeepInversion (R_{feature}). Upon adding R_{feature}, we immediately find large improvements in accuracy of 40%-69% across all the teaching scenarios. DeepInversion images (Fig. 3(d)) are vastly superior in realism, as compared to the baselines (Fig. 3(a,b)).

Effectiveness of Adaptive DeepInversion (R_{compete}). Using competition-based inversion further improves accuracy by 1%-10%, bringing the student accuracy very close to that of the teacher trained on real images from the CIFAR-10 dataset (within 2%). The training curves from one of the runs are shown in Fig. 4. Encouraging teacher-student disagreements brings in additional "harder" images during training (shown in Fig. 3(e)) that remain correct for the teacher, but have a low student accuracy, as can be seen from Fig. 4 (left).

Comparison with DAFL [7]. We further compare our method with DAFL [7], which trains a new generator network to convert noise into images using a fixed teacher. As can be seen from Fig. 3(c), we notice that these images are "unrecognizable," reminiscent of "fooling images" [49]. Our method enables higher visual fidelity of images and eliminates the need for an additional generator network, while gaining higher student accuracy under the same setup.

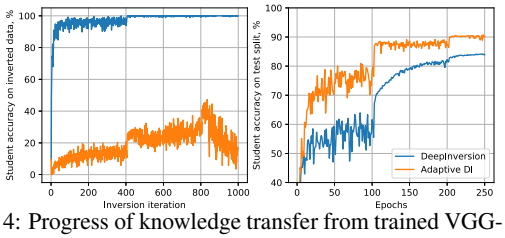


Figure 4: Progress of knowledge transfer from trained VGG-11-BN (92.34% acc.) to freshly initialized VGG-11-BN network (student) using inverted images. Plotted are accuracies on generated (left) and real (right) images. Final student accuracies shown in Table 1.

4.2. Results on ImageNet

After successfully demonstrating our method's abilities on the small CIFAR-10 dataset, we move on to examine its effectiveness on the large-scale ImageNet dataset [10]. We first run DeepInversion on networks trained on ImageNet, and perform quantitative and qualitative analyses. Then, we show the effectiveness of synthesized images on three different tasks of immense importance: data-free pruning, data-free knowledge transfer, and data-free continual learning.

Implementation details. For all experiments in this section, we use the publicly available pretrained ResNet-$\{18, 50\}$ from PyTorch as the fixed teacher network, with top-1 accuracy of $\{69.8\%, 76.1\%\}$. For image synthesis, we use Adam for optimization (learning rate 0.05). We set $\alpha_{\text{tv}} = 1 \cdot 10^{-4}, \alpha_{\ell_2} = \{0, 1 \cdot 10^{-2}\}, \alpha_f = 1 \cdot 10^{-2}$ for DeepInversion, and $\alpha_c = 0.2$ for Adaptive DeepInversion. We synthesize 224×224 images in batches of $1,216$ using 8 NVIDIA V100 GPUs and automatic-mixed precision (AMP) [41] acceleration. Each image batch receives 20k updates over 2h.

4.2.1 Analysis of DeepInversion Images

Fig. 5 shows images generated by DeepInversion from an ImageNet-pretrained ResNet-50. Remarkably, given just the model, we observe that DeepInversion is able to generate images with high fidelity and resolution. It also produces detailed image features and textures around the target object, *e.g.*, clouds surrounding the target balloon, water around a catamaran, forest below the volcano, *etc*.

Generalizability. In order to verify that the generated images do not overfit to just the inverted model, we obtain predictions using four other ImageNet networks. As can be seen from Table 2, images generated using a ResNet-50 generalize to a range of models and are correctly classified. Further, DeepInversion outperforms DeepDream by a large margin. This indicates robustness of our generated images while being transferred across networks.

Inception score (IS). We also compare the IS [58] of our generated images with other methods in Table 3. Again, DeepInversion substantially outperforms DeepDream with an improvement of 54.2. Without sophisticated training,

Figure 5: Class-conditional 224×224 samples obtained by DeepInversion, given only a ResNet-50 classifier trained on ImageNet and no additional information. Note that the images depict classes in contextually correct backgrounds, in realistic scenarios. Best viewed in color.

Model	DeepDream top-1 acc. (%)	DeepInversion top-1 acc. (%)
ResNet-50	100	100
ResNet-18	28.0	**94.4**
Inception-V3	27.6	**92.7**
MobileNet-V2	13.9	**90.9**
VGG-11	6.7	**80.1**

Table 2: Classification accuracy of ResNet-50 synthesized images by other ImageNet-trained CNNs.

Method	Resolution	GAN	Inception Score
BigGAN [5]	256	✓	178.0 / 202.6^{+}
DeepInversion (**Ours**)	224		60.6
SAGAN [75]	128	✓	52.5
SNGAN [43]	128	✓	35.3
WGAN-GP [16]	128	✓	11.6
DeepDream [46]*	224		6.2

Table 3: Inception Score (IS) obtained by images synthesized by various methods on ImageNet. SNGAN ImageNet score from [60]. *: our implementation. $^{+}$: BigGAN-deep.

DeepInversion even beats multiple GAN baselines that have limited scalability to high image resolutions.

4.3. Application I: Data-free Pruning

Pruning removes individual weights or entire filters (neurons) from a network such that the metric of interest (*e.g.*, accuracy, precision) does not drop significantly. Many techniques have been proposed to successfully compress neural networks [17, 28, 32, 35, 44, 45, 69, 72]. All these methods require images from the original dataset to perform pruning. We build upon the pruning method of Molchanov *et al*. [44], which uses the Taylor approximation of the pruning loss for a global ranking of filter importance over all the layers. We extend this method by applying the KL divergence loss, computed between the teacher and student output distributions. Filter importance is estimated from images inverted with DeepInversion and/or Adaptive DeepInversion, making pruning data-free. We follow the pruning and finetuning (30 epochs) setup from [44]. All experiments on pruning are performed with ResNet-50.

Hardware-aware loss. We further consider actual latency on the target hardware for a more efficient pruning. To achieve this goal, the importance ranking of filters needs to reflect not only accuracy but also latency, quantified by:

$$\mathcal{I}_{\mathcal{S}}(\mathbf{W}) = \mathcal{I}_{\mathcal{S},err}(\mathbf{W}) + \eta\, \mathcal{I}_{\mathcal{S},lat}(\mathbf{W}), \qquad (10)$$

where $\mathcal{I}_{\mathcal{S},err}$ and $\mathcal{I}_{\mathcal{S},lat}$, respectively, focus on absolute changes in network error and inference latency, specifically, when the filter group $s \in \mathcal{S}$ is zeroed out from the set of neural network parameters $\mathbf{W}$. η balances their importance. We approximate the latency reduction term, $\mathcal{I}_{\mathcal{S},lat}$, via precomputed hardware-aware look-up tables of operation costs (details in the Appendix).

Data-free pruning evaluation. For better insights, we study four image sources: **(i)** partial ImageNet with 0.1M original images; **(ii)** unlabeled images from the proxy dataset, MS COCO [30] (127k images), and PASCAL VOC [12] (9.9k images) datasets; **(iii)** 100k generated images from the BigGAN-deep [5] model, and **(iv)** a data-free setup with the proposed methods: we first generate 165k images via DeepInversion, and then add to the set an additional 24k/26k images through two competition rounds of Adaptive DeepInversion, given pruned students at 61.9%/73.0% top-1 acc.

Results of pruning and fine-tuning are summarized in Table 4. Our approach boosts the top-1 accuracy by more than 54% given inverted images. Adaptive DeepInversion performs relatively on par with BigGAN. Despite beating

Image Source	Top-1 acc. (%) −50% filters −71% FLOPs	Top-1 acc. (%) −20% filters −37% FLOPs
No finetune	1.9	16.6
Partial ImageNet		
0.1M images / 0 label	69.8	74.9
Proxy datasets		
MS COCO	66.0	73.8
PASCAL VOC	54.4	70.8
GAN		
Generator, BigGAN	63.0	73.7
Noise (**Ours**)		
DeepInversion (DI)	55.9	72.0
Adaptive DeepInversion (ADI)	60.7	73.3

Table 4: ImageNet ResNet-50 pruning results for the knowledge distillation setup, given different types of input images.

Method	ImageNet data	GFLOPs	Lat. (ms)	Top-1 acc. (%)
Base model	✓	4.1	4.90	76.1
Taylor [44]	✓	2.7 (1.5×)	4.38 (1.1×)	75.5
SSS [23]	✓	2.8 (1.5×)	-	74.2
ThiNet-70 [35]	✓	2.6 (1.6×)	-	72.0
NISP-50-A [72]	✓	3.0 (1.4×)	-	72.8
Ours				
Hardware-Aware (HA)	✓	3.1 (1.3×)	4.24 (1.2×)	76.1
ADI (Data-free)		2.7 (1.5×)	4.36 (1.1×)	73.3
ADI + HA		2.9 (1.4×)	4.22 (1.2×)	74.0

Table 5: ImageNet ResNet-50 pruning comparison with prior work.

VOC, we still observe a gap between synthesized images (Adaptive DeepInversion and BigGAN) and natural images (MS COCO and ImageNet).

Comparisons with SOTA. We compare data-free pruning against SOTA methods in Table 5 for the setting in which 20% of filters are pruned away globally. We evaluate three setups for our approach: (i) individually applying the hardware-aware technique (HA), (ii) data-free pruning with DeepInversion and Adaptive DeepInversion (ADI), and (iii) jointly applying both (ADI+HA). First, we evaluate the hardware-aware loss on the original dataset, and achieve a 16% faster inference with zero accuracy loss compared to the base model, we also observe improvements in inference speed and accuracy over the pruning baseline [44]. In a data-free setup, we achieve a similar post-pruned model performance compared to prior works (which use the original dataset), while completely removing the need for any images/labels. The data-free setup demonstrates a 2.8% loss in accuracy with respect to the base model. A final combination of both data-free and hardware-aware techniques (ADI+HA) closes this gap to only 2.1%, with faster inference.

4.4. Application II: Data-free Knowledge Transfer

In this section, we show that we can distill information from a teacher network to a student network without using any real images at all. We apply DeepInversion to a ResNet50v1.5 [52] trained on ImageNet to synthesize images. Using these images, we then train another randomly initialized ResNet50v1.5 from scratch. Below, we describe two practical considerations: a) image clipping, and b) multi-resolution synthesis, which we find can greatly help boost accuracy while reducing run-time. A set of images generated

Figure 6: Class-conditional 224×224 images obtained by DeepInversion given a ResNet50v1.5 classifier pretrained on ImageNet. Classes top to bottom: (left) daisy, volcano, quill, (right) cheeseburger, brown bear, trolleybus.

by DeepInversion on the pretrained ResNet50v1.5 is shown in Fig. 6. The images demonstrate high fidelity and diversity.

a) Image clipping. We find that enforcing the synthesized images to conform to the mean and variance of the data preprocessing step helps improve accuracy. Note that commonly released networks use means and variances computed on ImageNet. We clip synthesized images to be in the $[-m/s, m/s]$ range, with m representing the per channel mean, and s per channel standard deviation.

b) Multi-resolution synthesis. We find that we can speed up DeepInversion by employing a multi-resolution optimization scheme. We first optimize the input of resolution 112×112 for 2k iterations. Then, we up-sample the image via nearest neighbor interpolation to 224×224, and then optimize for an additional 1k iterations. This speeds up DeepInversion to 84 images per 6 minutes on an NVIDIA V100 GPU. Hyperparameters are given in the Appendix.

Knowledge transfer. We synthesize 140k images via DeepInversion on ResNet50v1.5 [52] to train a student network with the same architecture from scratch. Our teacher is a pretrained ResNet50v1.5 that achieves 77.26% top-1 accuracy. We apply knowledge distillation for 90/250 epochs, with temperature $\tau = 3$, initial learning rate of 1.024, batch size of 1024 split across eight V100 GPUs, and other settings the same as in the original setup [52]. Results are summarized in Table 6. The proposed method, given only the trained ResNet50v1.5 model, can teach a new model from scratch to achieve a 73.8% accuracy, which is only 3.46% below the accuracy of the teacher.

4.5. Application III: Data-free Continual Learning

Data-free continual learning is another scenario that benefits from the image generated from DeepInversion. The main idea of continual learning is to add new classification

Image source	Real images	Data amount	Top-1 acc.
Base model	✓	1.3M	77.26%
Knowledge Transfer, 90 epochs			
ImageNet	✓	215k	76.1%
MS COCO	✓	127k	72.3%
Generator, BigGAN		215k	64.0%
DeepInversion (DI)		140k	68.0%
Knowledge Transfer, 250 epochs, with mixup			
DeepInversion (DI)		140k	73.8%

Table 6: Knowledge transfer from the trained ResNet50v1.5 to the same network initialized from scratch.

Method	Top-1 acc. (%)			
	Combined	ImageNet	CUB	Flowers
ImageNet + CUB (1000 → 1200 outputs)				
LwF.MC [56]	47.64	53.98	41.30	–
DeepDream [46]	63.00	56.02	**69.97**	–
DeepInversion (**Ours**)	**67.61**	**65.54**	69.68	–
Oracle (distill)	69.12	68.09	70.16	–
Oracle (classify)	68.17	67.18	69.16	–
ImageNet + Flowers (1000 → 1102 outputs)				
LwF.MC [56]	67.23	55.62	–	78.84
DeepDream [46]	79.84	65.69	–	**94.00**
DeepInversion (**Ours**)	**80.85**	**68.03**	–	93.67
Oracle (distill)	80.71	68.73	–	92.70
Oracle (classify)	79.42	67.59	–	91.25
ImageNet + CUB + Flowers (1000 → 1200 → 1302 outputs)				
LwF.MC [56]	41.72	40.51	26.63	58.01
DeepInversion (**Ours**)	**74.61**	**64.10**	**66.57**	**93.17**
Oracle (distill)	76.18	67.16	69.57	91.82
Oracle (classify)	74.67	66.25	66.64	91.14

Table 7: Continual learning results that extend the network output space, adding new classes to ResNet-18. Accuracy over *combined* classes $\mathcal{C}_o \cup \mathcal{C}_k$ reported on individual datasets. Average over datasets also shown (datasets treated equally regardless of size, hence ImageNet samples have less weight than CUB or Flowers samples).

classes to a pretrained model without comprehensive access to its original training data. To the best of our knowledge, only LwF [29] and LwF.MC [56] achieve data-free continual learning. Other methods require information that can only be obtained from the original dataset, *e.g.*, a subset of data (iCaRL [56]), parameter importance estimations (in the form of Fisher matrix in EWC [27], contribution to loss change in SI [74], posterior of network weights in VCL [50]), or training data representation (encoder [62], GAN [21, 59]). Some methods rely on network modifications, *e.g.*, Packnet [40] and Piggyback [39]. In comparison, DeepInversion does not need network modifications or the original (meta-)data, as BN statistics are inherent to neural networks.

In the class-incremental setting, a network is initially trained on a dataset with classes $\mathcal{C}_o$, *e.g.*, ImageNet [10]. Given new class data (x_k, y_k), $y_k \in \mathcal{C}_k$, *e.g.*, from CUB [65], the pretrained model is now required to make predictions in a combined output space $\mathcal{C}_o \cup \mathcal{C}_k$. Similar to prior work, we take a trained network $\big($denoted $p_o(\cdot)$, effectively as a *teacher*$\big)$, make a copy $\big($denoted $p_k(\cdot)$, effectively as a *student*$\big)$, and then add new randomly initialized neurons to $p_k(\cdot)$'s final layer to output logits for the new classes. We train $p_k(\cdot)$ to classify simultaneously over all classes, old and new, while network $p_o(\cdot)$ remains fixed.

Continual learning loss. We formulate a new loss with DeepInversion images as follows. We use same-sized batches of DeepInversion data $(\hat{x}, p_o(\hat{x}))$ and new class real data (x_k, y_k) for each training iteration. For $\hat{x}$, we use the original model to compute its soft labels $p_o(\hat{x})$, *i.e.*, class probability among old classes, and then concatenate it with additional zeros as new class probabilities. We use a KL-divergence loss between predictions $p_o(\hat{x})$ and $p_k(\hat{x})$ on DeepInversion images for prior memory, and a cross-entropy (CE) loss between one-hot y_k and prediction $p_k(x_k)$ on new class images for emerging knowledge. Similar to prior work [29, 56], we also use a third KL-divergence term between the new class images' old class predictions $p_k(x_k|y \in \mathcal{C}_o)$ and their original model predictions $p_o(x_k)$. This forms the loss

$$\begin{aligned}\mathcal{L}_{\text{CL}} =& \text{KL}\big(p_o(\hat{x}), p_k(\hat{x})\big) + \mathcal{L}_{xent}\big(y_k, p_k(x_k)\big) \\ &+ \text{KL}\big(p_o(x_k|y \in \mathcal{C}_o), p_k(x_k|y \in \mathcal{C}_o)\big).\end{aligned} \tag{11}$$

Evaluation results. We add new classes from the CUB [65], Flowers [51], and both CUB and Flowers datasets to a ResNet-18 [18] classifier trained on ImageNet [10]. Prior to each step of addition of new classes, we generate 250 DeepInversion images per old category. We compare our results with prior class-incremental learning work LwF.MC [56] as opposed to the task-incremental LwF [29] that cannot distinguish between old and new classes. We further compare results with oracle methods that break the data-free constraint: we use the same number of real images from old datasets in place of $\hat{x}$, with either their ground truth for classification loss or their soft labels from $p_o(\cdot)$ for KL-divergence distillation loss. The third KL-divergence term in Eq. 11 is omitted in this case. Details are given in the Appendix.

Results are shown in Table 7. Our method significantly outperforms LwF.MC in all cases and leads to consistent performance improvements over DeepDream in most scenarios. Our results are very close to the oracles (and occasionally outperform them), showing DeepInversion's efficacy in replacing ImageNet images for continual learning. We verify results on VGG-16 and discuss limitations in the Appendix.

Conclusions

We have proposed DeepInversion for synthesizing training images with high resolution and fidelity given just a trained CNN. Further, by using a student-in-the-loop, our Adaptive DeepInversion method improves image diversity. Through extensive experiments, we have shown that our methods are generalizable, effective, and empower a new class of "data-free" tasks of immense practical significance.

Acknowledgments

We would like to thank Arash Vahdat, Ming-Yu Liu, and Shalini De Mello for in-depth discussions and comments.

References

[1] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai. Variational information distillation for knowledge transfer. In *CVPR*, 2019.

[2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014.

[3] K. Bhardwaj, N. Suda, and R. Marculescu. Dream distillation: A data-independent model compression framework. In *ICML Workshop*, 2019.

[4] L. Breiman and N. Shang. Born again trees. Technical report, UC Berkeley, 1996.

[5] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[6] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *SIGKDD*, 2006.

[7] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, and Q. Tian. Data-free learning of student networks. In *ICCV*, 2019.

[8] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016.

[9] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, P. Vajda, M. Uyttendaele, and N. K. Jha. ChamNet: Towards efficient network design through platform-aware model adaptation. In *CVPR*, 2019.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[11] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010.

[13] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCCS*, 2015.

[14] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. In *ICML*, 2018.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *NeurIPS*, 2017.

[17] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *ICLR*, 2016.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] Z. He, T. Zhang, and R. B. Lee. Model inversion attacks against collaborative inference. In *ACSAC*, 2019.

[20] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[21] W. Hu, Z. Lin, B. Liu, C. Tao, J. Tao, J. Ma, D. Zhao, and R. Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2019.

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[23] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. In *ECCV*, 2018.

[24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] A. Kimura, Z. Ghahramani, K. Takeuchi, T. Iwata, and N. Ueda. Few-shot learning of neural networks from scratch by pseudo example optimization. In *BMVC*, 2018.

[26] A. Kimura, Z. Ghahramani, K. Takeuchi, T. Iwata, and N. Ueda. Few-shot learning of neural networks from scratch by pseudo example optimization. In *BMVC*, 2018.

[27] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2016.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient ConvNets. In *ICLR*, 2017.

[29] Z. Li and D. Hoiem. Learning without forgetting. *IEEE TPAMI*, 2017.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[31] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang. Structured knowledge distillation for semantic segmentation. In *CVPR*, 2019.

[32] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *CVPR*, 2017.

[33] R. G. Lopes, S. Fenu, and T. Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[34] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. In *ICLR*, 2016.

[35] J.-H. Luo, J. Wu, and W. Lin. ThiNet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[36] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[37] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[38] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *IJCV*, 2016.

[39] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

[40] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[41] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. In *ICLR*, 2018.

[42] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*, 2018.

[43] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[44] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *CVPR*, 2019.

[45] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *ICLR*, 2017.

[46] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`, 2015.

[47] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017.

[48] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NeurIPS*, 2016.

[49] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

[50] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *ICLR*, 2018.

[51] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICCVGIP*, 2008.

[52] NVIDIA. ResNet50v1.5 training. `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/ConvNets/resnet50v1.5`, 2019. [Online; accessed 10-Nov-2019].

[53] W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *CVPR*, 2019.

[54] A. Pilzer, S. Lathuiliere, N. Sebe, and E. Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *CVPR*, 2019.

[55] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. In *ICLR*, 2018.

[56] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017.

[57] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for thin deep nets. In *ICLR*, 2015.

[58] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

[59] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017.

[60] K. Shmelkov, C. Schmid, and K. Alahari. How good is my GAN? In *ECCV*, 2018.

[61] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[62] A. Triki Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. In *ICCV*, 2017.

[63] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. HAQ: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019.

[64] Y. Wang, C. Si, and X. Wu. Regression model fitting under differential privacy and model inversion attack. In *IJCAI*, 2015.

[65] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, Caltech, 2010.

[66] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *CVPR*, 2019.

[67] Z. Xu, Y.-C. Hsu, and J. Huang. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks. In *ICLR Workshop*, 2018.

[68] Z. Yang, E.-C. Chang, and Z. Liang. Adversarial neural network inversion via auxiliary knowledge alignment. *arXiv preprint arXiv:1902.08552*, 2019.

[69] J. Ye, X. Lu, Z. Lin, and J. Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018.

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

[71] H. Yin, G. Chen, Y. Li, S. Che, W. Zhang, and N. K. Jha. Hardware-guided symbiotic training for compact, accurate, yet execution-efficient LSTM. *arXiv preprint arXiv:1901.10997*, 2019.

[72] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. NISP: Pruning networks using neuron importance score propagation. In *CVPR*, 2018.

[73] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[74] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[75] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *ICML*, 2019.

[76] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. In *ICLR*, 2017.