The Knowledge Within: Methods for Data-Free Model Compression

Matan Haroush, Itay Hubara[1,2], Elad Hoffer[1], and Daniel Soudry[2]

[1]Habana Labs Research, Caesarea, Israel

[2]Department of Electrical Engineering, Technion, Haifa , Israel,
{mharoush,ihubara,ehoffer}@habana.ai, daniel.soudry@gmail.com

Abstract

Background: *Recently, an extensive amount of research has been focused on compressing and accelerating Deep Neural Networks (DNN). So far, high compression rate algorithms require part of the training dataset for a low precision calibration, or a fine-tuning process. However, this requirement is unacceptable when the data is unavailable or contains sensitive information, as in medical and biometric use-cases.*

Contributions: *We present three methods for generating synthetic samples from trained models. Then, we demonstrate how these samples can be used to calibrate and fine-tune quantized models without using any real data in the process. Our best performing method has a negligible accuracy degradation compared to the original training set. This method, which leverages intrinsic batch normalization layers' statistics of the trained model, can be used to evaluate data similarity. Our approach opens a path towards genuine data-free model compression, alleviating the need for training data during model deployment.*

1. Introduction

Quantization is a prevalent, accelerator friendly, compression method [13, 15] employed prior to DNNs deployment within real-world applications. However, high compression rates typically demand additional information to minimize quality loss. For example, when using low precision arithmetic, it is often beneficial to trade-off the numerical representation range with its resolution by clipping the dynamic range based on its real expected values. Thus, it is a common practice to gather per tensor statistics [15, 1] from a subset of samples drawn from the training set (Calibration). Furthermore, in cases of very high compression rates, model parameters often require additional adjustment to recover from catastrophic error accumulation, where access to the entire training data is needed. This may lead to an undesired coupling between deployment and training phases of a model through data. Especially in cases where the training data is sensitive or simply unavailable at the time of deployment. Therefore, it is appealing to investigate new methods to alleviate the need for real data for deployment purposes, for example, via substitution with synthetic data.

Generating high-quality samples requires capturing the prior distribution of the data which is often hard. A large body of work dedicated to generative models has shown that it is possible to learn such priors and generate high-resolution synthetic images [16, 6, 27, 9, 4]. Unfortunately, these techniques come at the cost of training a dedicated generative model. This training requires access to the real training data which we want to avoid.

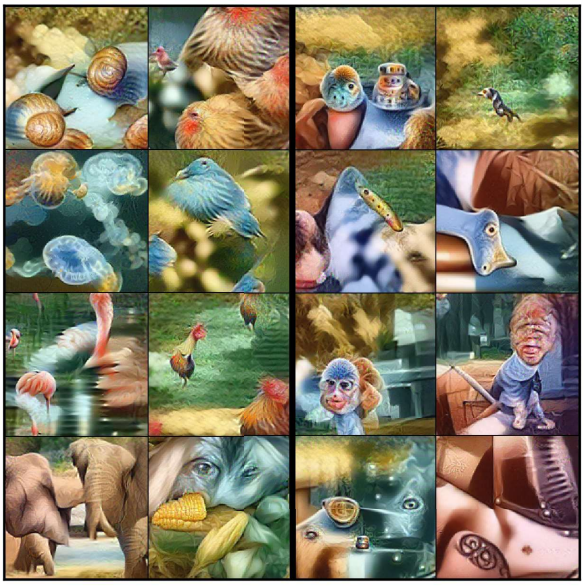


Figure 1: Synthetic samples generated from ResNet-18 trained on Imagenet, using our methods: BNS (right;3.3) and the class-conditional $BNS + \mathcal{I}$ (left;3.4). Additional samples are available in appendix-C.

We aim to understand the potential and limitations of synthetic samples for model compression tasks, specifically for reduced precision deployment i.e., calibration and fine-tune via Knowledge Distillation (KD [10]). In the context of this work, the full precision model (or parts of it) serves as the teacher, while the student is the low precision counterpart. We will focus our attention on DNNs for classification tasks (e.g. CIFAR and ImageNet [18, 26]). In the process, we revisit a feature visualization process

dubbed "Inceptionism" [22] which is based solely on the trained model. This process typically starts with an arbitrary input, which is iteratively adjusted to maximize the response of a set of target features via back-propagation. Furthermore, the optimization process is typically constrained using some prior knowledge about the input, such as a high correlation between nearby pixels within an image, to avoid over-fitting the generated sample. A similar approach was recently adapted for several use-cases including for data-free distillation [19, 3, 24] with limited success.

Our contribution is twofold: First, we offer novel methods for generating and leveraging synthetic samples for the use of knowledge distillation. These samples are created under a realistic data-free regime by exploiting the encapsulated knowledge within the provided model. We then empirically evaluate the usefulness of these samples for model compression, yielding comparable results to the original training dataset with minimal accuracy degradation. Second, we propose a novel approach for evaluating the similarity between a reference dataset and a set of arbitrary samples. In a nutshell, we suggest measuring the mean divergence of the second-order statistics drawn from a set of intermediate layers of a given model that was trained on the reference dataset (reference model). This can be done without relying on access to any real data by leveraging intrinsic measurements provided by Batch Normalization (BN) layers [14].

2. Related work

The notion of "data-free" quantization was recently introduced by Nagel et al., [23] who proposed using the pre-determined measurements of batch-normalization statistics [14], that are gathered during training to determine the proper dynamic range for each layer. However, this method requires either access to all of the layers' statistics or relying on a closed-form analytical solution for layers for which measurements are not available, based on their input distribution. Furthermore, this method is unable to handle cases where calibration leads to extreme degradation, so fine-tuning is required.

Previous work explored using data-free knowledge distillation for model compression by collecting metadata related to the statistics of a trained model output [19, 3]. In both cases, the generation scheme consists of sampling a set of random target tensors and minimizing the mean square error between the output and the sampled targets. Lopes et al.,[19] proposed collecting means and co-variance for a set of layers after training, while optimization targets are sampled per layer independently under multivariate Gaussian assumption. The authors mention this method fails to capture inter-layer relations and proposed an alternative approach to capture inter-layer relations via graph spectral analysis with compelling results for models trained on the MNIST dataset. However, this is impractical for large models or models with large input size, due to its computational cost.

Bhardwaj et al., [3] suggested it is sufficient to collect metadata from the layer before the linear classifier. Metadata collection is done by processing a small portion of the training set, clustering the high dimensional outputs and applying Principle Component Analysis (PCA) per cluster. The optimization target is based on the collected centroids and a random noise projected in the direction of the primary principle components. Bhardwaj et al., [3] presented a small set of experiments on CIFAR10 dataset, with relatively large degradation between real data to the generated samples.

Recently, Nayak et al., [24] proposed similar data-free method dubbed as zero-shot distillation, which relies solely on the final layer weights to compute a class similarity matrix. Then, sample soft targets via Dirichlet distribution for generating synthetic images. However, KD results on synthetic samples are only comparable to real data on easily separable datasets such as MNIST, while performing poorly on CIFAR10.

In this work, we aim to generate samples that mimic the training data distribution by some measure. This can be achieved with naive noise sampling according to the low-order statistics of original data, or by directly optimizing a similarity measure of the internal activations' statistics in a trained model. In contrast to previous attempts [19, 3] which required sampling activation generated by real data, our method relies on low order statistics captured by existing BN layers (i.e., channel-wise mean and standard deviation). We then track the statistics induced by the synthetic samples and directly optimize a divergence score with respect to the reference set, while relying on the model structure to naturally maintain inter-layer statistics' relations.

3. Methods for data-free distillation

We are interested in a data-free regime, wherein a model is given without its corresponding dataset used for training. This regime reflects a realistic scenario, as training data is often confidential or private. Therefore, we offer three methods for generating useful synthetic data for distillation and calibration:

- **Gaussian scheme:** samples are randomly drawn from a Gaussian distribution.
- **Inception scheme:** samples are generated via logit maximization (i.e., a special case of the Inceptionism scheme).
- **BN-Statistics scheme:** samples are generated by optimizing a novel internal statistics' divergence measure.

3.1. Gaussian Scheme

As a naive approximation of the original dataset, it is natural to consider a simple Gaussian generator (denoted as $\mathcal{G}$) with first and second moments defined to match the original input data. We suggest this alone may be sufficient for model calibration tasks required for low numerical precision inference, under mild compression demands. Such a generation scheme is appealing since an indefinite number of samples can be generated at will with minimal compute and storage requirements. However, as one would expect, under extreme compression demands, when the model parameters need to be adjusted (e.g., fine-tune using distillation), this method may prove to be insufficient.

Since this method does not preserve the original input's structure, the internal activation's statistics may differ significantly from the original statistic induced by the training data. This change can harm the model accuracy especially if it contains Batch-Normalization (BN) layers [14]. BN layers are commonly used in DNNs workloads, as they have been shown to improve both accuracy and speed of convergence by normalizing the input to the layer to have zero mean and unit variance before applying

its operation. During training, each BN layer keeps a running estimate of the empirical mean and standard-deviation per-channel of its inputs which latter applied to normalized the input data during inference.

Adjusting the model's parameters by using distillation over the randomly generated samples, will irretrievably alter the existing BN layer parameter to accommodate for the observed statistical properties, resulting in an imminent failure when turning back to evaluate real data. Thus, we suggest forcing all BN layers in the model to maintain their original running estimates for evaluation. As shown in section 4 this may negate this obstacle to some extent, enabling the use of such samples for the task of distillation.

3.2. Inception Scheme

Inceptionism [22] related generation schemes, typically impose constraints solely on the input and output of the model. We shall focus on a special case, which we term the Inception scheme (denoted as $\mathcal{I}$), where only a single neuron from the final model output is maximized under an input smoothness requirement. We define the optimization objective is the sum of a Domain Prior term and an Inception loss term. As a domain prior, we use a Gaussian smoothing kernel to produce a smoothed version of the provided input. The loss term is then computed as the mean squared error between the input image and the smoothed variant, encouraging nearby pixels to have similar values. The Inception Loss term is derived by choosing an arbitrary label and perform gradient descent on the negative exponent of the appropriate class logit drawn from the Fully Connected (FC) layer output, i.e. $e^{-logit/scale}$. The Inception Loss injects the desired class information, where the exponent and temperature-scale control the impact of the logit magnitude on the loss, preventing the model from producing inputs that cause the output to explode by exponentially decaying the logit contribution to the general loss as its magnitude increases. This scheme often results in high confidence outputs, however, it is highly sensitive to the hyper-parameters (see appendix-A).

3.3. BN-Statistics Scheme

Based on our Gaussian and Inception scheme experiments we hypothesize that the lack of regularization on the internal statistic during the sample generation process may result in significant internal statistics' divergence from the observed statistics on real data. Indicating such samples are not drawn from a distribution similar to the original data (as we show later, in Figure 2a). This, in turn, may impede their use for model compression and KD. Thus, our proposed approach is to directly minimize the internal statistics' divergence, by optimizing a novel measure which we term "BN-Stats" (denoted as BNS). BNS only assumes access to the predetermined empirical measurements that occur for each BN layer. We will use these estimates as targets and compare new data samples by the similarity between their measured statistics.

More formally, given the running estimates $\hat{\mu}$ and $\hat{\sigma}$ of a given batch-norm layer, we wish to measure $\tilde{\mu} = \mu(D)$ and $\tilde{\sigma} = \sigma(D)$ of our new data activations D and evaluate their similarity to the reference statistics. BNS define this similarity using the Kullback-Leibler (KL) divergence under a simplistic isotropic Gaussian assumption such that

$$\begin{aligned} BNS(D, \hat{\mu}, \hat{\sigma}) &= KL\left(\mathcal{N}(\hat{\mu}, \hat{\sigma}^2) || \mathcal{N}(\tilde{\mu}, \tilde{\sigma}^2)\right) \\ &= \log \frac{\tilde{\sigma}}{\hat{\sigma}} - \frac{1}{2}\left(1 - \frac{\hat{\sigma}^2 + (\hat{\mu} - \tilde{\mu})^2}{\tilde{\sigma}^2}\right) \end{aligned} \tag{1}$$

Our generation process starts with a batch of random input samples, which are then iterativly adjusted to minimize the mean statistical divergence across all layers in the reference set. Namely, for each optimization step over an input batch X, we extract per-BN-layer activation set $\{D_l\}_{l=1}^{N}$. We then define the optimization objective as the mean statistics' divergence over all $BNS(D_l, \hat{\mu}_l, \hat{\sigma}_l)$. Formally, we define the first and second moments sets as:

$$T, S := \{P_l \sim \mathcal{N}(\hat{\mu}_l, \hat{\sigma}_l)\}_{l=1}^{N}, \{Q_l \sim \mathcal{N}(\tilde{\mu}_l, \tilde{\sigma}_l)\}_{l=1}^{N}$$

while the optimization objective (i.e., BNS measure) is defined as the mean KL divergence on the set $\{T, S\}$, i.e.

$$\mathcal{J}_{KL}(X|T, S) := \frac{1}{N}\sum_{l=1}^{N} KL(P_l||Q_l) \tag{2}$$

$$= \frac{1}{N}\sum_{l=1}^{N} BNS(D_l, \hat{\mu}_l, \hat{\sigma}_l) \tag{3}$$

A small $\epsilon = 1e^{-8}$, is added to the measured variance of the induced synthetic distribution to accommodate for zero-variance channels. We note that alternative metrics can be used for BNS under the same underlying assumptions, such as Mean Square Error over the empirical moments which is symmetric and can handle zero variance channels. Finally, standard backpropagation is used to adjust X.

3.4. Combining $BNS + \mathcal{I}$

In addition to the three schemes described above, we experiment with a combination of BN-Statistics and Inception schemes objectives denoted as $BNS + \mathcal{I}$, see Algorithm 1. This method of generation attempts to harness the strength of each method. The class constraint imposed by $\mathcal{I}$ impacts the induced statistics depending on the batch composition, forcing the optimization process to compensate for the sampled classes. The drawback of this method is the additional loss scaling parameters which are now added to the original hyper-parameters of the Inception scheme. We didn't exhaustively investigate the best method to combine the two however, results of a simple aggregation provides an encouraging improvement in some cases.

4. Experiments

4.1. Generating data samples

We first describe several components used in our sample generation methods. Specifically, note that $\mathcal{I}$ & BNS can be seen as special cases of the $BNS + \mathcal{I}$ method, by scaling the appropriate loss components. $BNS + \mathcal{I}$ method is described in algorithm-1, the reader may refer to appendix-A for specific hyperparameters details.

Algorithm 1: Generating $BNS + \mathcal{I}$ samples

Input: a pre-trained model with N BN layers
Param: $input\_shape, batch\_size, budget$
Param: $class\_temp, \theta_{prior}, \alpha, \beta, \gamma$
Param: $optimizer(\theta), \#duplicates$
Output: a batch of synthetic samples — X_{budget}

Init: $X_0 \leftarrow randn(batch\_size, input\_shape)$
Init: $targets \leftarrow rand\_int(batch\_size, \#classes)$
Extract $T := \{\hat{\mu}_l, \hat{\sigma}_l\}_{l=1}^{N}$ // $P_l \sim \mathcal{N}(\hat{\mu}_l, \hat{\sigma}_l)$
while $i < budget$ **do**
  Compute: $X_i \leftarrow clamp(X_i, 0, 1)$
  Compute: $\hat{X}_i \leftarrow augment(X_i, \#duplicates)$
  Compute: $loss_p \leftarrow prior(\hat{X}_i|\theta_{prior})$ // e.g. $smooth(\hat{X}_i|k, \sigma)$
  Compute: $logits \leftarrow forward(\hat{X}_i)$
  Record: $S := \{\tilde{\mu}_l, \tilde{\sigma}_l\}_{l=1}^{N}$ // $Q_l \sim \mathcal{N}(\tilde{\mu}_l, \tilde{\sigma}_l)$
  Compute: $loss_i \leftarrow e^{-\frac{logits[targets]}{scale}}$
  Compute: $loss_s \leftarrow \mathcal{J}_{KL}(\hat{X}_i|T, S)$
  Compute: $loss \leftarrow \alpha * loss_s + \beta * loss_i + \gamma * loss_p$
  Compute: $G_{X_i} \leftarrow backward(loss, X_i)$ // i.e., $\frac{\partial loss}{\partial X_i}$
  Update: $X_{i+1} \leftarrow optimizer.step(G_{X_i}, \theta)$
  Update: $i \leftarrow i + 1$

- **Inception Loss:** The Inception loss is defined as the exponent of the negative value of the appropriate class logit drawn from the Fully Connected (FC) layer of the model i.e., $e^{-logit/scale}$. A random set of labels are chosen as targets.
- **Image Prior:** Nearby pixels of an image often have similar values, common techniques attempt to encourage such relations. In practice we use a Gaussian smoothing prior, by applying a convolution with a Gaussian kernel on the input, creating a smoothed version that is then used to compute the mean squared error from the original input.
- **Statistics Loss:** we define $\mathcal{J}_{KL}$ as our statistics loss. Reference statistics are extracted from all BN layers in the model. Those are simply the running mean and variance gathered during training. Additionally, we treat the reported dataset normalization in the same manner.
- **Input Trimming:** At the start of each optimization step we clip the input values to $[0, 1]$ range to match real data values.
- **In-Batch Augmentations:** Since activation gradients are not aggregated across batch during back standard propagation, we use in-batch augmentations [11] technique to gain a gradient smoothing effect per sample; Specifically, each sample is duplicated N times with a set of random differential augmentations chosen from a set containing Random-N-Cutout, Crop-Resize and Flip.

Since the generation process must start with a previously trained model, we concentrate most of our experiments on ResNet meta-architecture [8, 30], as it is a popular example of an architecture in which training quality heavily relies on batch normalization layers.

4.2. Data-free model compression

General settings. Model compression for deployment usually requires some additional effort. The compression level correlates with the amount of information and work necessary to reduce accuracy degradation. Using low numerical precision as a model compression technique for inference typically requires a prior calibration step to determine the dynamic range (which sets the scale and zero-point values) [17] for each intermediate layer. This restriction sacrifices the dynamic range for an improved numerical resolution. When enforcing highly demanding compression rates on a model it is often required to retrain the model under similar compression constraints to recover from any significant accuracy loss. The process of adjusting the model parameters to the compression constrains is commonly called fine-tuning. Since each model responds differently to quantization, we choose numerical precision to ensure that calibration or fine-tuning is needed. We will consistently use the notation of $\#w\#a$, to denote the number of numerical precision bits used for representing the quantized activations and weights throughout the model.

In all our experiments the case of uniform quantization is considered, with per-channel scale for the weights and a per-tensor scale for the layer's activation values as detailed in [17]. A copy of the weights [13], as well as the biases and gradients, are kept in single-precision where the latter is derived using a straight-through estimator [2]. All results are reported for simulated quantization (i.e. discretizing the inputs and weights before applying float operators). Additionally, Batch-Normalization layers were kept in single precision.

Calibration details. Our experiments begin by calibrating each model using real and synthetic data, under increasing compression rates. During calibration, we use a smoothed absolute dynamic range measurement. That is, the dynamic range is measured via running estimates of the mean absolute min/max values within chunks containing 16 samples each. We argue that more advanced calibration methods [23, 1] may improve final accuracy as they are uncoupled to our proposed approach of using synthetic data. Unless otherwise specified, 200 calibration steps are used with batch size 256. Samples are drawn with replacement out of a balanced dataset with size limited to 1% of the training dataset size. Standard data augmentations (e.g., random crop and mirroring) are applied on each input batch to maximize the utilization of available data. Additionally, results are reported as mean and standard deviation with 5 different seeds.

Fine-tuning details. In our KD setting, the float model is the teacher while the student is its quantized variant. Specifically, we apply the most demanding configuration from the previous calibration experiment to create the student. Additionally, teacher predictions are used as targets without additional labels. Furthermore, optimization settings were fixed throughout each set of experiments to enable a fair comparison: all models are optimized with a fixed number of Stochastic Gradient Decent (SGD) iterations, the learning rate scheduler starts with a short warm-up phase followed by a cosine decay phase, while for each optimization step, a batch is drawn with replacement from the target dataset and augmented using standard augmentations methods. We also follow McKinstry et al., [20] and freeze the dynamic range of the student activations during the entire optimization process (we didn't find the suggestion of freezing the dynamic range of the weights necessary).

We also take advantage of the shared teacher and student networks' structure, and apply a simple tweak under the assumption that a good low precision proxy exists within the vicinity of the

reference model in the parameters space. Specifically, we use intermediate layer outputs to compute the smoothed-ℓ_1 distance between the teacher and the student features, we named this loss Intermediate Quantization (IQ) loss. We found IQ generally lead to a more stable training convergence under extreme quantization and improve KD results for both synthetic and real samples. Additionally, we apply in batch MixUp technique on the inputs similar to [31], without mixing the teacher's outputs. The full distillation process is illustrated by algorithm-2, along with an ablation experiment results (table-1).

Algorithm 2: Quantized Distillation with IQ

Input: T : the teacher model with parameters Θ_t
Input: $L := \{l^i_{aux}\}^N$: a set of auxiliary layers (e.g., pointers)
Input: $\mathcal{D}$: dataset
Param: $criterion_{IQ}, criterion_{KD}$
Param: $\alpha, \beta, \theta_{mix}$
Param: $quantizer(\theta_q), optimizer(\theta_{opt})$
Output: a fine-tuned quantized model

Init: $S \leftarrow quantizer(T, \mathcal{D}, \theta_q)$ `// quantize and calibrate the student`
for $X_i \in \mathcal{D}$ **do**
  Compute: $\hat{X}_i \leftarrow InputMix(X_i, \theta_{mix})$
  Compute: $logits_t, aux_t \leftarrow T.forward(\hat{X}_i, L)$
  Compute: $logits_s, aux_s \leftarrow S.forward(\hat{X}_i, L)$
  Compute: $loss_{IQ} \leftarrow \sum_{l_i \in L} criterion_{IQ}(aux_t^{l_i}, aux_s^{l_i})$ `// e.g., Smooth-L1 loss`
  Compute: $loss_{KD} \leftarrow criterion_{KD}(logits_t, logits_s)$ `// e.g., KL-Divergence loss`
  Compute: $loss \leftarrow \alpha * loss_{KD} + \beta * loss_{IQ}$
  Compute: $G_S \leftarrow backward(loss, \Theta_s^i)$ `//` $\frac{\partial loss}{\partial \Theta_s^i}$, *where* Θ_s^i *are the weights of* S
  Update: $\Theta_s^{i+1} \leftarrow optimizer.step(G_S, \theta_{opt})$

Table 1: KD ablation study on input-mixing and IQ. We fine-tune a quantized ResNet-44 (2w4a, first & final layers are in 4 bits) on CIFAR10 dataset with varying dataset size.

#samples	*KD*	*KD+IQ*	*KD+Mix*	*KD+IQ+Mix*
1	76.32	79.67	82.49	83.68
10	81.12	81.71	84.01	85.44
100	80.16	81.64	85.05	85.95
1000	84.03	84.66	87.2	87.28
2000	85.41	85.74	87.84	88.36

Small scale experimental details. We wish to evaluate the applicability of synthetic samples for model calibration and distillation compared to real data. For this experiment we use ResNet44 and Wide-Resnet28-10 [8, 30] on CIFAR10, CIFAR100 datasets [18] respectively.

We start the process by following each of the proposed generation schemes to produce synthetic samples. Next, we apply increasingly demanding quantization configurations to the models and calibrate each variant using samples drawn from the target dataset which is limited to 50 samples per class. After the initial calibration, we optimize the selected models for 16,000 iterations of SGD with a batch size of 512 and 256 for ResNet44 and Wide-ResNet28-10 respectively. IQ loss is computed as the mean of the smoothed-l1 over the outputs of blocks 2-5 from ResNet architecture and multiplied by a scale of 0.001.

Small scale results. Results on post-calibration and KD fine-tuning are shown in table-2. We find that synthetic samples which minimize BNS are superior to samples from other generation schemes, while achieving comparable accuracy to real data, as the fine-tuned models recover from extreme accuracy degradation. Additionally, we observe a surprising outcome regarding the usefulness of Gaussian samples for calibration and distillation — under mild compression requirements. We noticed that performing performing KD with Gaussian samples requires an additional step to freeze all batch normalization layers, to prevent corruption of the intermediates statistics of the original training data. We believe the full potential Gaussian samples are yet to be discovered and we encourage it as a future research direction. One simple path is truncating the model to disjoint intervals between BN layers then retrain each interval using KD over synthetic data generated by the applying Gaussian scheme on the statistics of the previous BN layer. We also note that Inception related schemes appear to be more sensitive to the hyper-parameters choice (see appendix-A).

Large scale experimental details. To demonstrate the applicability of our findings in a large-scale setting, we provide our results on ImageNet [26]. ImageNet has been noted to be a challenging generative task even when full data access is granted, due to the relatively large spatial size of the data and the number of classes [27]. Previous work [23] showed that 8-bit models can be calibrated without any data if BN layers exist. However, they did not investigate more challenging numerical precision levels and thus did not need to fine-tune the model. We conduct our experiment with a pre-trained models from *torchvision model-zoo* [25]. First, we generate 10K/100k samples (see examples in appendix-C), then perform KD to fine-tune the quantized student, following similar settings to ones described in the previous section, except for a longer regime of 44000 steps and a batch size of 256. Throughout our experiments, the quantization scheme is fixed to a widely used method described in [17]. Methods that improve this scheme (such as bit allocation, bias correction [1], and equalization [23, 21]) are orthogonal to this work and are expected to improve accuracy respectively.

Large scale results. Table-3 presents promising results for recovering lost model accuracy on ImageNet classification task. To the best of the authors' knowledge, this is the first time a compressed model was successfully fine-tuned without using any real data other than the trained model itself at this scale.

We observe an accuracy gap (relative 1.5% degradation) between standard Cross-Entropy (CE) objective and KD when fine-tuning the model with the full dataset is available. However, results are in favor of KD when using significantly fewer samples (a detailed comparison is available in appendix-B). We believe this can be attributed to a couple of factors. First, semi-supervised KD does not use the ground truth label information. Thus, final accuracy depends solely on the prediction quality of the teacher, whereas label information can be used to penalize the student when repeating similar mistakes made by the teacher. Additionally, we speculate that a given bias in the reference model's prediction towards certain classes may degrade the student accuracy

Table 2: CIFAR validation accuracy for low precision models using data from synthetic and real datasets. The *Settings* column refers to the number of bits used as well as the number of samples-per-class in the dataset. The *Reference* column presents the original training data results. Fine-tune (KD) and calibration with synthetic samples is on par compared to real data, while the results reflect the importance of using statistics preserving samples for demanding compression settings.

	Settings	*Reference*	BNS	$\mathcal{I}$	$BNS + \mathcal{I}$	$\mathcal{G}$
ResNet-44 - CIFAR10, fp32 accuracy 93.23						
Calibration	4w8a, 50[1]	92.18 (0.05)	92.21 (0.03)	92.37 (0.03)	92.25 (0.04)	92.24 (0.01)
	4w4a, 50	89.19 (0.15)	88.5 (0.13)	87.44 (0.13)	89.1 (0.09)	87.51 (0.18)
	2w4a, 50[2]	19.47 (0.16)	19.14 (0.05)	19.42 (0.2)	19.49 (0.1)	18.48 (0.2)
KD	2w4a, 50[2]	90.27	88.52	79.62	88.16	70.03
	2w4a, 4000[2]	91.24	89.6	79.59	88.51	71.02
Wide ResNet-28-10 - CIFAR100, fp32 accuracy 83.69						
Calibration	8w8a, 50	82.96 (0.03)	83.11 (0.04)	83.01 (0.04)	83.06 (0.02)	83.15 (0.04)
	4w4a, 50[1]	76.96 (0.15)	74.4 (0.11)	62.78 (0.32)	74.56 (0.15)	58.79 (0.09)
	4w4a, 50	64.52 (0.15)	61.48 (0.08)	51.22 (0.23)	61.82 (0.02)	47.81 (0.19)
KD	4w4a, 50	81.7	79.21	53.64	78.88	64.8
	4w4a, 200	82.11	79.16	54.02	78.75	65.15

[1]First & final layers are in 8 bits [2]First & final layers are in 4 bits

Table 3: ImageNet validation accuracy for low precision models using data from synthetic and real datasets. We use pre-trained weights from *torchvision model-zoo* [25] for several meta-architectures under varying compression settings. BNS results are on par with the real data, while KD performs well on the limited size dataset compared to standard cross-entropy loss, see appendix-B for more details.

	Settings	*Reference*	BNS	$\mathcal{I}$	$BNS + \mathcal{I}$	$\mathcal{G}$
ResNet-18 [8] - ImageNet, fp32 accuracy - 69.75						
Calibration	8w8a, 10[1]	69.63 (0.03)	69.55 (0.01)	69.6 (0.02)	69.57 (0.04)	68.94 (0.02)
	4w4a, 10[2]	54.72 (0.06)	55.29 (0.1)	38.25 (0.1)	55.49 (0.06)	53.02 (0.1)
KD	4w4a, 10[2]	68.63	67.98	62.8	68.06	63.98
	4w4a, 100[2]	68.68	68.14	63.1	67.95	63.58
MobileNet-V2 [28] - ImageNet, fp32 accuracy - 71.88						
Calibration	8w8a, 10[1]	71.26 (0.05)	71.34 (0.03)	71.2 (0.01)	71.32 (0.04)	71.17 (0.02)
	4w4a, 10[3]	15.1 (0.1)	16.17 (0.1)	10.55 (0.04)	16.1 (0.06)	13.36 (0.04)
KD	4w4a, 10[3]	68.5	66.4	53.13	66.07	36.95
DenseNet-121 [12] - ImageNet, fp32 accuracy - 74.65						
Calibration	8w8a, 10	74.41 (0.01)	74.41 (0.03)	74.22 (0.02)	74.23 (0.02)	74.16 (0.02)
	4w4a, 10	45.27 (0.04)	43.54 (0.15)	40.46 (0.1)	43.88 (0.09)	46.89 (0.03)
KD	4w4a, 10	71.08	71.26	63.98	70.72	63.59

[1]Compared to [23] (without weights adjustments): 69.6, 69.7 [2]First & final layers are in 8 bits [3]1x1 convolution layers are in 8 bits

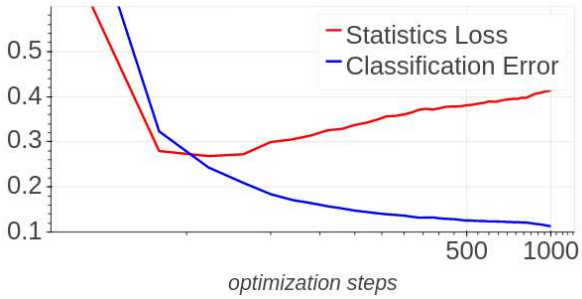
when training on raw teacher outputs.

Finally, we conclude that using synthetic data leads to either equivalent or slightly lower final accuracy compared to real data. We associate the accuracy loss with the synthetic data optimization process (i.e. higher $\mathcal{J}_{KL}$ than real data), as well as with the limited ability of the provided weights to capture the true variance of the data, which in turn leads to a limited sample space.

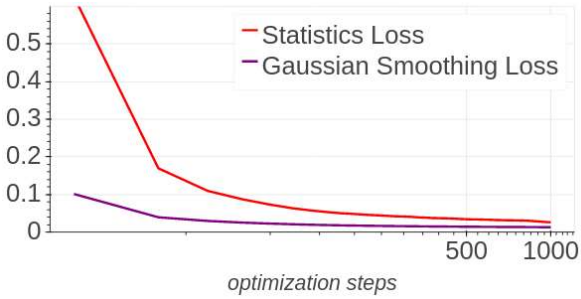
4.3. Analysis of Data Generation Schemes

For this section we consider ResNet44 trained on CIFAR10 as the reference model, in an attempt to understand the relations between generation schemes and provide further explanation to the success of BNS scheme over its counterparts. We will consider the differences between methods from the perspective of the optimization objectives for generating synthetic samples. Additionally, we evaluate the impact of the internal statistics' divergence on KD's potential for recovering lost accuracy.

Monitoring Internal Statistics. As an initial step in our evaluation, we perform a pair of experiments on Inception and BNS schemes. In each experiment, we follow the process of one scheme and observe the other's objective behavior. More specifically, we generate a batch of samples using the Inception scheme while monitoring the behavior of the internal statistics through $\mathcal{J}_{KL}$. For the alternative view, we generate a second batch of samples using the BNS scheme, and observe the input smoothness loss. In both experiments, we perform 1000 optimization steps on a batch of 128 samples. For each iteration we use in-batch augmentations with N=4, additionally, a Gaussian kernel of size 3x3 and sigma=1 is used for the smoothing loss operation. Results are described in Figure 2.



(a) Measured BNS loss quickly diverges when optimizing Inception loss, indicating that the generated samples may be out of the original data distribution.



(b) Gaussian smoothing loss is decreasing during BNS loss optimization, revealing a correlation between statistics loss and the spatial structure within feature maps.

Figure 2: Generating synthetic samples from ResNet-44 trained on CIFAR10.

Under the Inception generation scheme, we find evidence for the divergence of the internal statistics as seen in figure 2a. Additionally, figure 2b indicates that minimizing $\mathcal{J}_{KL}$ leads to lower smoothness loss. Interestingly, figure 2a the initial improvement in $\mathcal{J}_{KL}$ during optimization of $\mathcal{I}$ hints to the existence of a connection in a reversed direction as well. However, smoothing loss alone proves to be insufficient in regularizing the internal statistics' divergence as the Inception loss greedily attempts to maximize the target class predictions and the measured $\mathcal{J}_{KL}$ is increasing after several iterations.

BN-Stats and Model Accuracy. For the next part of our evaluation, we generated samples from the reference model using the BNS scheme. During the sample generation phase, several snapshots of the training data were saved at different step intervals. The snapshots reflect different stages of the $\mathcal{J}_{KL}$ loss curve. We then perform a series of KD experiments on the quantized reference model to observe the impact of dataset size and $\mathcal{J}_{KL}$ loss value on the final model accuracy. Each experiment is repeated with five different seeds, the student model precision configuration is fixed to 4-bit activations and 2-bit weights except for the first and last layers which are using 4 bit.

Final results are presented in Figure 4 as follows: Figure 3b demonstrates that samples with lower statistic loss lead to better validation accuracy, while Figure 3a shows results are very close to real data with slight degradation under identical settings.

4.4. Data Correspondence: BNS Across Datasets

In this section we observe how the internal statistics of a model respond to different inputs, illustrating that similar input data should not lead to significantly different internal activation statistics. In table-4 we demonstrate this by measuring $\mathcal{J}_{KL}$ of various datasets on a set of pre-trained ResNet44 [8] models. We also measure the $\mathcal{J}_{KL}$ divergence in response to the original training data modified with Fast Gradient Sign Method (FGSM [7]). A small perturbation ratio of $\epsilon = 0.1$ is used to reduce model accuracy without considerably altering the perceived input. As expected from a similarity score, table-4 demonstrates that $\mathcal{J}_{KL}$ maintains its proportion compared to the original training dataset, although magnitude may change across models and datasets. We further observe that in some cases, perturbed samples results in a significant increase in the internal statistics' divergence. This invites further exploration in the direction of possible applications of monitoring internal statistics to explain and detect adversarial attacks through tailored outlier sensitive measures.

5. Discussion and Future Work

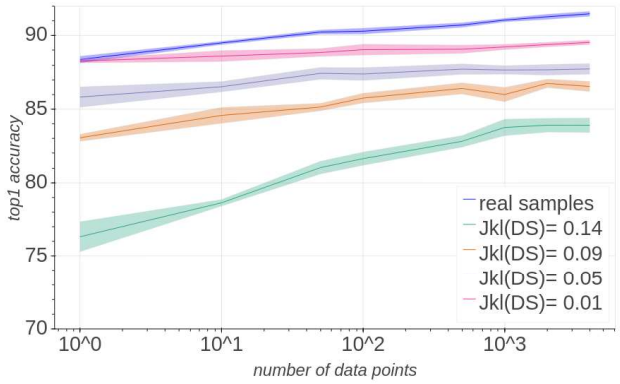
In this work, we addressed the case of data-free KD for model quantization under realistic data and hardware acceleration constrains. To overcome this challenge, we suggested a set of new sample-generation methods, and demonstrated the applicability of such samples for a true data-free model compression in both small and large scale image classification tasks. In the process we explored techniques to leverage such samples in a student-teacher training procedure.

In addition, our best performing method, BNS leveraged the common batch-norm layer to generate examples that mimic the per-channel mean and variance of each BN layer input, by explicitly optimizing $\mathcal{J}_{KL}$ divergence loss. We show evidence that optimizing this loss, results in smooth input features, indicating a strong connection between BN-Stats loss and the local structure of the input (figure-2b). This invites further study to determine the viability of the method for the reproduction of inputs other than images, where prior knowledge may be harder to apply directly.

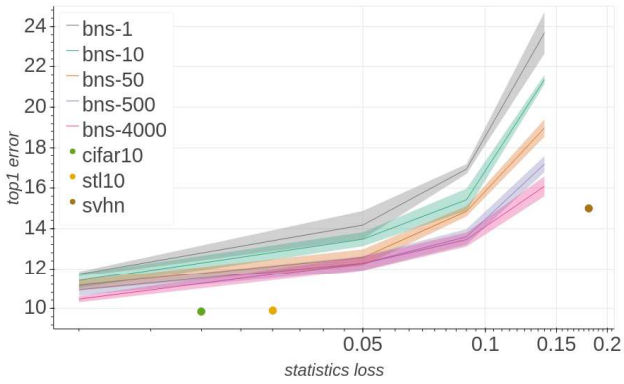
Alternatively, our random data approach $\mathcal{G}$ can serve as a cheap replacement to the real data. In particular, for calibration under mild quantization (8-bit) demands. We also noticed that compared to BNS, the naive inception scheme $\mathcal{I}$ generates samples with a high model prediction confidence, yet causes the internal statistics' divergence to grow significantly (figure-2a). Furthermore, the success of BNS, leads us to consider it as a possible measure for evaluating correspondence between the reference dataset, used during model's training, and alternative samples.

To the best of our knowledge, this is the first time a data-free distillation-based approach was applied to a model and achieved final accuracy results comparable with real samples — under identical optimization conditions and at large scale. Ultimately, we believe our approach can open a path to improve data privacy by reducing the extent of real data exposure during the production phase of deployment.

The release of this paper was followed by [29, 5], which utilize



(a) Dataset size impact on final model accuracy.



(b) Statistics loss impact on final error.

Figure 3: KD on a quantized ResNet-44 model (2w4a, CIFAR10) using real and BNS data with varying number of data points (per-class). Demonstrates the connection between dataset size and $\mathcal{J}_{KL}$ with model accuracy. Providing more synthetic samples improves final accuracy.

Table 4: The value of $\mathcal{J}_{KL}$ normalized by the value of $\mathcal{J}_{KL}$ that is measured on the original training dataset. Values are computed on the entire train split, raw measurements are available in appendix-D.

Train/Measure	*CIFAR10*	*CIFAR100*	*MNIST*	*SVHN*	*STL10*	*Random*	*FGSM*[1]
CIFAR10	1.0	1.0	6.6	7.9	1.3	21.7	3.3
CIFAR100	1.1	1.0	3.9	4.1	1.3	14.8	1.9
MNIST	103.5	111.5	1.0	196.0	138.5	327.5	2.0
SVHN	1.4	1.2	1.2	1.0	2.7	4.1	0.9

[1]Ratio grows with the ϵ parameter of FGSM, as the image perturbation becomes more apparent.

the same principle of leveraging the internal statistics' divergence for a range of data-free tasks, including low precision calibration and knowledge distillation. These papers explore interesting ideas such as adaptive sample generation and layer sensitivity heuristics for mixed-precision, while their results further support our findings. In contrast to [29], which presents a wide range of experiments on data-free applications such as classical KD and pruning with a limited analysis, we focused our attention on the applicability of the method for DNN's deployment on low precision accelerators and provide further insight into the success of the method and to the importance of the internal divergence measure. Additionally, we present data-free KD results for low precision models, which are not explored by [5], and show highly compressed models can recover from extreme accuracy degradation which enables much more demanding configurations without requiring access to the original training data.

We consider two drawbacks of the proposed method. One is the computational cost of generating samples through back-propagation, which can impede the practical use with large scale models for continuous train-deploy scenarios. However, we note that as long as the new training data does not significantly change, the behavior of the internal statistics of the generated samples can potentially be shared — to avoid reproducing an entirely new synthetic dataset at each deployment cycle. We leave the exploration of cross-model and cross-dataset applications for future work. Second, we find the BNS method produces datasets which are unbalanced in terms of the mean output distribution of the reference model, due to the lack of explicit conditioning on the model output. However, our experiments with $BNS + \mathcal{I}$ did not show a dramatic improvement despite their added balance control and the additional information injection from the final layer weights. We suspect this is due to the tension between the BNS objective and $\mathcal{I}$ which may require balancing or a longer optimization to reach comparable $\mathcal{J}_{\mathcal{KL}}$. Still, there is a lot to unveil and we are excited by the diverse opportunities to exploit the suggested scheme beyond model compression. For instance, applying BNS measure to detect outliers and adversarial examples, or to avoid *catastrophic forgetting* in a continual learning setting.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. 2019. 1, 4, 5

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[3] Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework. *arXiv preprint arXiv:1905.07072*, 2019. 2

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1

[5] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework, 2020. 7, 8

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 7

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 5, 6, 7

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017. 1

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[11] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: better training with larger batches. *arXiv preprint arXiv:1901.09335*, 2019. 4

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 6

[13] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017. 1, 4

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 2

[15] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 1

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. 1

[17] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 4, 5

[18] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1, 5

[19] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017. 2

[20] Jeffrey L. McKinstry, Steven K. Esser, Rathinakumar Appuswamy, Deepika Bablani, John V. Arthur, Izzet B. Yildiz, and Dharmendra S. Modha. Discovering low-precision networks close to full-precision networks for efficient embedded inference, 2018. 4, 11

[21] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different-recovering neural network quantization error through weight factorization. *arXiv preprint arXiv:1902.01917*, 2019. 5

[22] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015. 2, 3

[23] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. *arXiv preprint arXiv:1906.04721*, 2019. 2, 4, 5, 6

[24] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019. 2

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 5, 6

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1, 5

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 1, 5

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. 6

[29] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion, 2019. 7, 8

[30] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *Procedings of the British Machine Vision Conference 2016*, 2016. 4, 5

[31] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5