MixMix: All You Need for Data-Free Compression Are Feature and Data Mixing

Yuhang Li*†   Feng Zhu†   Ruihao Gong†   Mingzhu Shen†   Xin Dong‡

Fengwei Yu†   Shaoqing Lu†   Shi Gu*

*UESTC   †SenseTime Research   ‡Harvard University

Abstract

*User data confidentiality protection is becoming a rising challenge in the present deep learning research. Without access to data, conventional data-driven model compression faces a higher risk of performance degradation. Recently, some works propose to generate images from a specific pretrained model to serve as training data. However, the inversion process only utilizes biased feature statistics stored in one model and is from low-dimension to high-dimension. As a consequence, it inevitably encounters the difficulties of generalizability and inexact inversion, which leads to unsatisfactory performance. To address these problems, we propose MixMix based on two simple yet effective techniques: (1) **Feature Mixing**: utilizes various models to construct a universal feature space for generalized inversion; (2) **Data Mixing**: mixes the synthesized images and labels to generate exact label information. We prove the effectiveness of MixMix from both theoretical and empirical perspectives. Extensive experiments show that MixMix outperforms existing methods on the mainstream compression tasks, including quantization, knowledge distillation and pruning. Specifically, MixMix achieves up to 4% and 20% accuracy uplift on quantization and pruning, respectively, compared to existing data-free compression work.*

1. Introduction

To enable powerful deep learning models on the embedded and mobile devices without sacrificing task performance, various model compression techniques have been discovered. For example, neural network quantization [12, 23, 27, 55] converts 32-bit floating-point models into low-bit fixed point models and benefits from the acceleration of fixed-point computation and less memory consumption. Network pruning [8, 14, 48] focuses on reducing the redundant neural connections and finds a sparse network. Knowledge Distillation (KD) [18, 41] transfers the knowledge in the large teacher network to small student networks.

However, one cannot compress the neural networks aggressively without the help of data. As an example, most
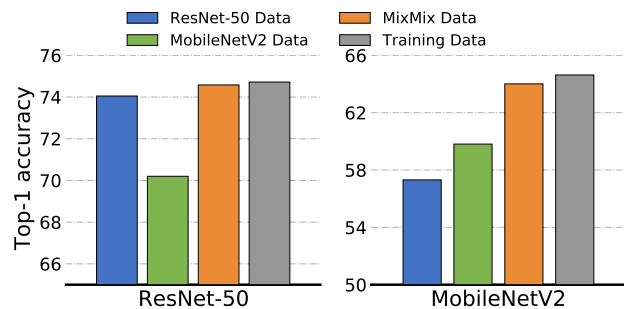

Figure 1. An overview of the results of post-training data-free quantization. Each color bar denotes a data source. Data inverted from ResNet-50 encounters an accuracy deficit on MobileNetV2.

full precision models can be safely quantized to 8-bit by directly rounding the parameters to their nearest integers [26, 39]. However, when the bit-width goes down to 4, we have to perform quantization-aware training using the data collected from users to compensate for the accuracy loss. Unfortunately, due to the increasing privacy protection issue[1], one cannot get user data easily. Moreover, the whole ImageNet dataset contains 1.2M images (more than 100 gigabytes), which consumes much more memory space than the model itself. Therefore, the data-free model quantization is more demanding now.

Recently, many works [4, 15, 50] propose to *invert* images from a specific pretrained model. They try to match the activations' distribution by comparing the recorded running mean and running variance in the Batch Normalization (BN) [24] layer. Data-free model compression with the generative adversarial network is also investigated in [5, 11, 49]. All these works put their focus on developing a better criterion for inverting the data from a specific model. We call this type of data as *model-specific* data.

We identify two problems of model-specific data. First, the synthesized images generated by a specific model are biased and cannot be generalized to another. For example, in DeepInversion [50], synthesizing 215k 224×224 resolu-

---

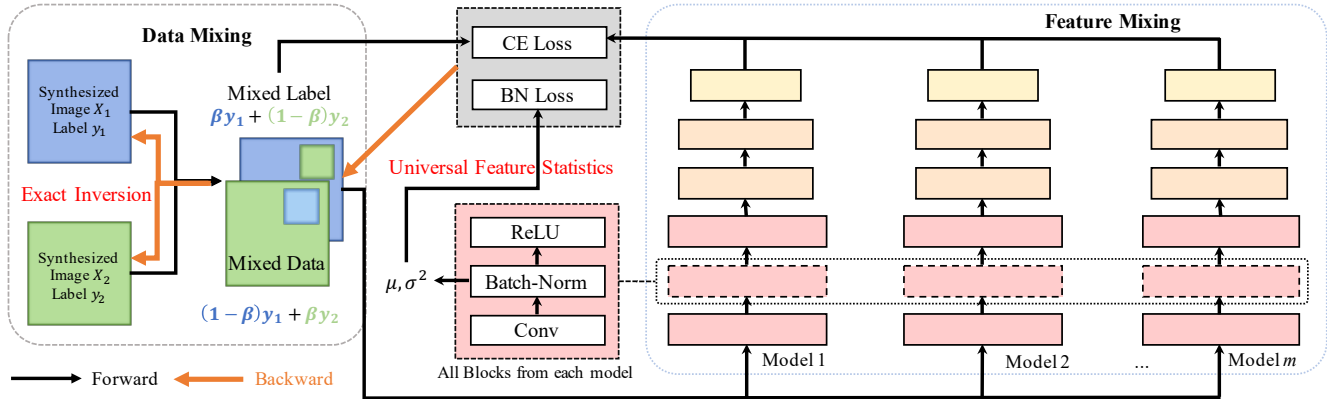[1] https://ec.europa.eu/info/law/law-topic/data-protection_en

Figure 2. The overall pipeline of the proposed MixMix algorithm. Data Mixing can reduce the incorrect solution space by mixing the pixel and label of two trainable images. Feature Mixing incorporates the universal feature space from various models and generates a one-for-many synthesized dataset, after which the synthesized data can be applied to any models and application.

tion images from ResNet-50v1.5 [40] requires 28000 GPU hours. These images cannot be used for another model easily. As we envisioned in Fig. 1, data inverted from MobileNetV2 has 4% higher accuracy than ResNet-50 data on MobileNetV2 quantization and vice versa. Therefore, model-specific data inversion requires additional thousands of GPU hours to adapt to compression on another model. Second, due to the non-invertibility of the pretrained model, model-specific data results in inexact synthesis. A simple example is not hard to find: given a ReLU layer that has 0 in its output tensor, we cannot predict the corresponding input tensor since the ReLU layer outputs 0 for all negative input. As a result, finding the exact inverse mapping of a neural network remains a challenging task.

In this work, we propose *MixMix* data synthesis algorithm that can generalize across different models and applications, which contributes to an overall improvement for data-free compression. MixMix contains two sub-algorithms, the first one is *Feature Mixing* which utilizes the universal feature produced by a collection of pre-trained models. We show that Feature Mixing is equal to optimize the maximum mean discrepancy between the real and synthesized images. Therefore, the optimized data shares high fidelity and generalizability. The second algorithm is called *Data Mixing* that can narrow the inversion solution space to synthesize images with exact label information. To this end, we summarize our core contributions as:

1. **Generalizability:** We propose Feature Mixing that can absorb the knowledge from widespread pre-trained architectures. As a consequence, the synthesized data can generalize well to any models and applications.
2. **Exact Inversion:** We propose Data Mixing which is able to prevent incorrect inversion solutions. Data Mixing can preserve correct label information.
3. **Effectiveness:** We verify our algorithm from both theoretical and empirical perspectives. Extensive data-free

compression applications, such as quantization, pruning and knowledge distillation are studied to demonstrate the effectiveness of MixMix data, achieving up to 20% absolute accuracy boost.

2. Related Works

Data-Driven Model Compression Data is an essential requirement in model compression. For example, automated exploring compact neural architectures [56, 32] requires data to continuously train and evaluate sub-networks. Besides the neural architecture search, quantization is also a prevalent method to compress the full precision networks. For the 8-bit case where the weight quantization nearly does not affect accuracy, only a small subset of calibration images is needed to determine the activation range in each layer, which is called Post-training Quantization [1, 26]. AdaRound [38] learns the rounding mechanism of weights and improves post-training quantization by reconstructing each layer outputs. Quantization-aware finetuning [10] can achieve near-to-original accuracy even when weights and activations are quantized to INT4. But this method requires a full training dataset as we mentioned. Apart from quantization, network pruning and knowledge distillation are also widely explored [18, 17].

Data-Free Model Compression The core ingredient in data-free model compression is image synthesis so that no real images are needed. Currently, the generation process can be classified into two categories, (1) directly learn images by gradient descent or (2) train a generative adversarial network (GAN) to produce images. DAFL [5] and GDFQ [6] apply GAN to generate images and learn the student network. This type of work achieves good results on tiny dataset. However, training large-scale GAN requires significant efforts. A parallel ax in image synthesis is *model*

inversion [35, 36]. Mordvintsev *et al*. proposed Deep-Dream [37] to 'dream' objects features onto images from a single pre-trained model. Recently, DeepInversion [50] uses the BN statistics variable as an optimization metric to distill the data and obtain high-fidelity images. BN scheme has also achieved improvements in other tasks: ZeroQ [4] and *the Knowledge Within* [15] use distilled dataset to perform data-free quantization, but their methods are model-specific, i.e., one generated dataset can only be used for one model's quantization. Our MixMix algorithm mainly focus on the direct optimization of data, however, it is also applicable to generative data-free application.

3. Preliminary

In this section, we will briefly discuss the background of how to synthesize images from a single pretrained model, then we will discuss two challenges of this method.

3.1. Model-Specific Data Inversion

Suppose we have a trainable image X with the size of $[w, h, c]$ (in ImageNet dataset [7], the size is $224 \times 224 \times 3$) and a pretrained network $A : X \rightarrow Y$, the Inceptionism [37] can invert the knowledge by assigning the image with a random label Y, since the networks have already captured the class information. Using the cross-entropy loss, the image can be optimized by

$$\min_X L_{\mathrm{CE}}(A(X), Y). \quad (1)$$

Recently, [4, 15, 50] observe that the pretrained networks have stored the activation statistics in the BN layers (*i.e.* running mean and running variance). Consequently, it is reasonable for synthesized images to mimic the activation distribution of the natural images in the network. Therefore, assuming the activation (regardless of the batch) in each layer is Gaussian distributed, the BN statistics loss can be defined as

$$L_{\mathrm{BN}} = \sum_{i=1}^{\ell} (||\mu_i(X) - \hat{\mu}_i||_2 + ||\sigma_i^2(X) - \hat{\sigma}_i^2||_2), \quad (2)$$

where $\mu_i(X)(\sigma_i^2(X))$ is the mean (variance) of the synthesized images activation in (i)-th layer while $\hat{\mu}_i$ ($\hat{\sigma}_i^2$) is the stored running mean (variance) in the BN layer. Note that we can replace the MSE loss to Kullback-Leibler divergence loss as did in [15].

In addition, an image prior loss can be imposed on X to ensure the images are generally smooth. In [15], the prior loss is defined as the MSE between X and its Gaussian blurred version $\varepsilon(X)$. In this work, we use the prior loss defined in [50]: $L_{\mathrm{prior}}(X) = \mathcal{R}_{\mathrm{TV}}(X) + \lambda_{\ell_2}\mathcal{R}_{\ell_2}(X)$, which is the sum of variance and norm regularization. Combining these three losses, the final minimization objective of

Table 1. BN Loss comparison on ResNet-50 and MobileNetV2, given different types of data.

Model	ImageNet	Res50 Data	MobV2 Data
ResNet-50	0.018	0.049	0.144
MobileNetV2	0.722	4.927	1.498

knowledge inversion for a specific model can be formulated as:

$$\min_X \lambda_1 L_{\mathrm{CE}}(X) + \lambda_2 L_{\mathrm{BN}}(X) + \lambda_3 L_{\mathrm{prior}}(X) \quad (3)$$

3.2. Biased Feature Statistics

For image synthesis tasks, the real ImageNet dataset could be viewed as the global minimum, which can be utilized to perform model compression on any neural architectures. However, we find that the data synthesized from one model cannot be directly applied to another different architecture. Example results demonstrated in Fig. 1 show that data synthesized on ResNet-50 gets bad quantization results on MobileNetV2 (4% lower) and vice versa.

We conjecture the reason for this phenomenon is the different feature statistics learned in different CNNs. In each neural network, the distribution characteristics of the training data are encoded and processed in its unique feature space (*i.e.* unique BN statistics). Therefore, extract the feature information from this neural network leads to biased statistics. To test this, we train images from ResNet-50 and MobileNetV2 by Eq. (3) and validate the synthesized images on these architectures. Results in Table 1 show that L_{BN} of the MobileNetV2 data is high when evaluated on ResNet50. Interestingly, in an opposite way, ResNet-50 data also has poor performance on MobileNetV2. However, as shown in Table 1, L_{BN} always remains in low-level regardless of model in the case of real ImageNet data.

3.3. Inexact Inversion

Apart from BN statistics loss $L_{\mathrm{BN}}(X)$, the cross-entropy loss $L_{\mathrm{CE}}(X)$ desires to learn an inverted mapping from label to input images, denoted by $A^{-1} : Y \rightarrow X$. According to [2], a residual block is invertible if it has less than 1 Lipschitz constant and the same input-output dimension (*i.e.* $\mathbb{R}^d \rightarrow \mathbb{R}^d$). We find the latter one is not true for all classification models, since the label dimension is $Y \in [0, 1]^{1000}$ and input dimension is $X \in \mathbb{R}^{224 \times 224 \times 3}$ for the ImageNet classification task. This dimension difference can produce a huge solution space for image inversion. Let us consider an example of the average pooling layers:

Example 3.1 *Consider a* 2×2 *AvgPool layer* $o = W^T V$, *where* $V \in \mathbb{R}^4$ *and* $W = [0.25, 0.25, 0.25, 0.25]$ *are the input and weight vectors. This AvgPool layers is non-invertible because given output* o, *any inputs that has same mean as* o *will suffice the condition.*
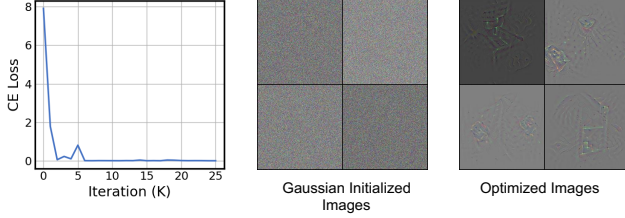
Figure 3. Optimizing cross-entropy loss of the images can be very fast and down to 0, but the images nearly have no class features.

Therefore, finding the exact input is infeasible in such a large space. In fact, almost every CNN has a 7×7 AvgPool layer before the final fully-connected layer. To visualize this, we optimize 4 images using $L_{\mathrm{CE}}(X)$ and $L_{\mathrm{prior}}(X)$ and plot the training curve in Fig. 3. It is clear that CE loss is easy to optimize, yet we cannot invert real images that contain rich class information.

4. Methodology

In this section, we introduce the proposed *MixMix* algorithm, which can improve both the generalizability and the invertibility of the dataset.

4.1. Feature Mixing

Before delving into the proposed algorithm, we would like to discuss *the general problem of comparing samples from two probability distributions*. This problem is of great interest in many areas such as bioinformatics where two image types are evaluated to determine if they come from the same tissues. In the case of image synthesis, we also need to evaluate if the generated images preserve high fidelity. Formally, given two distributions p and q with their random variables defined on a topological space $\mathcal{X}$ and observations $(X = \{x_1, \ldots, x_m\}, Z = \{z_1, \ldots, z_n\})$ that are independently and identically distributed (i.i.d.) from p and q, can we find some methods to determine if $p = q$? In [9, Lemma 9.3.2], this problem is well-defined as

Lemma 4.1 ([9]) *Let $(\mathcal{X}, d)$ be a metric space, and let p, q be two Borel probability measures defined on $\mathcal{X}$. Then $p = q$ if and only if $E_{x \sim p}(f(x)) = E_{z \sim q}(f(z))$ for all $f \in C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of bounded continuous functions on $\mathcal{X}$.*

However, evaluating all $f \in C(\mathcal{X})$ in the finite settings is not practical. Fortunately, Gretton *et al*. [13] proposed the maximum mean discrepancy that can be represented by:

$$\mathrm{MMD}[\mathcal{F}, X, Z] = \sup_{f \in \mathcal{F}} \left(E_{x \sim p}[f(x)] - E_{z \sim q}[f(z)] \right), \quad (4)$$

where $\mathcal{F}$ is a class of function which is much smaller than $C(\mathcal{X})$. We will directly give two results that are related

to MMD theory and the BN statistics loss. The detailed derivation is located in the appendix.

1. Let $\mathcal{H}$ be a *reproducing kernel Hilbert space (RKHS)*, and when we set $\mathcal{F}$ to $\mathcal{H}$, the squared MMD is given by.

$$\mathrm{MMD}^2[\mathcal{H}, X, Z] = ||\mu_p - \mu_q||_{\mathcal{H}}^2, \quad (5)$$

where $\mu_p \in \mathcal{H}$ is called the mean embedding of p. we can use the feature extractor of the CNNs and its feature-map to define a reproducing kernel. Therefore the running mean and variance[2] in BN layers can be treat to the mean embedding of $\mathcal{H}$. *Thus optimizing the L_{BN} is equivalent to minimize the MMD^2 between the real images and the synthesized images.*

2. [13, Theorem 5] states that the kernel for feature extraction k must be universal[3] so that we have $\mathrm{MMD}^2[\mathcal{H}, X, Z] = 0$ if and only if $p = q$.

These two results indicate that if the neural network is universal and the L_{BN} of synthesized images is 0, we could say the generated images are very close to real images. But it turns out few neural networks are universal and are only possible for extremely deep or wide ones [31, 33]. In Sec. 3.2, we show that different CNN has different feature statistics. And the data generated from one model is hard to be transferred to another one. This empirical evidence further illustrates the lack of universality in one model. We hereby propose Feature Mixing, which gathers the knowledge in the model zoo and aims to average out the feature bias produced by each individual. We expect the aggregation of pre-trained models can improve the universality of their corresponding RKHS. To demonstrate that mixing features can improve the universality, we have the following theorem:

Theorem 4.2 *Assume there are m neural networks with ReLU activation function ($A_i : X^d \to \mathbb{R}$). Then, the averaged model $\frac{1}{m} \sum_{i=1}^{m} A_i$ is universal if $m \geq \mathrm{ceil}(\frac{d+1}{w})$, where w is the averaged width.*

Proof is available in the appendix. Although sufficing the ultimate universality may require a large m, we anticipate increasing the number of features mixed can improve the quality as well as the generalizability of the generated data. To this end, we will show how to apply Feature Mixing during synthesis. Consider a pretrained model zoo $\mathcal{M} = \{A_1, A_2, \ldots, A_m\}$, the Feature Mixing aims to optimize:

$$\min_X \frac{1}{m} \sum_{i=1}^{m} \left(\lambda_1 L_{\mathrm{CE}}(X, A_i) + \lambda_2 L_{\mathrm{BN}}(X, A_i) \right). \quad (6)$$

Note that we also add a prior loss on X. However, assuming the identical size of each model A_i, the training memory

---

[2]The variance can be defined in the kernel's second-order space.

[3]The universal $\mathcal{H}$ means that, for any given $\epsilon > 0$ and $f \in C(\mathcal{X})$ there exists a $g \in \mathcal{H}$ such that the max norm $||f - g||_\infty < \epsilon$.

and computation is linearly scaled by the number of features we mix. Therefore, we add a hyper-parameter $m' \leq m$, which will decide how many models will be sampled from the model zoo for each batch of the data. The effect of m' will be studied in the experiments. Another problem is how to select model zoo, we expect different architecture families will contain different feature statistics, therefore we choose model families as many as possible in our model zoo.

4.2. Data Mixing

Another problem in image synthesis is that certain layers or blocks in neural network cause inexact inversion, as we described in Sec. 3.3. In this section, we show that we can mitigate this problem through Data Mixing. Denote two image-label pairs as (X_1, Y_1) and (X_2, Y_2), we first randomly generate a binary mask $\alpha \in \{0, 1\}^{w \times h}$. The elements in this mask will be set to 1 if it is located in the bounding box:

$$\alpha_{ij} = \begin{cases} 1 & \text{if } x_l \leq i \leq x_r \text{ and } y_d \leq j \leq y_u \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

where x_l, x_r, y_d, y_u are the left, right, down, up boundaries of the box. The center coordinate of the bounding box can be computed by $\left(\frac{x_l + x_r}{2}, \frac{y_d + y_u}{2}\right)$. With this binary mask, we can mix the feature of data, given by

$$\hat{X} = (1 - \alpha)X_1 + \alpha g(X_2). \quad (8)$$

Here $g(\cdot)$ is a linear interpolation function that can resize the images to the same size of the bounding box. The mixed data $\hat{X}$ now contains information from two images, thus we mix the label so that the CE loss becomes

$$L_{\text{CE}}(\hat{X}) = (1 - \beta)L_{\text{CE}}(\hat{X}, Y_1) + \beta L_{\text{CE}}(\hat{X}, Y_2) \quad (9)$$

where β is computed as the ratio of bounding box area and the image area $w \times h$. Data Mixing is inspired by the success of mixing-based data-augmentations, such as CutMix [51] and Mixup [52]. They use mixed data and label to train a model that has stronger discriminative power. Indeed, such augmentation techniques is also helpful in our scope by generating the robust features that remain discriminative when being mixed. Moreover, in this work, data mixing is also used to reduce the inexact inversion solutions of the neural networks. Back to Example 3.1, for each iteration t, we must suffice $\text{mean}(\hat{V}^t) = \hat{o}^t$, where the input and output are mixed differently and therefore more restrictions are added when inverting the input. We also give an example to illustrate this:

Example 4.3 *Consider the same 2×2 AvgPool layer in former discussion. If $o = 1$, then we have*
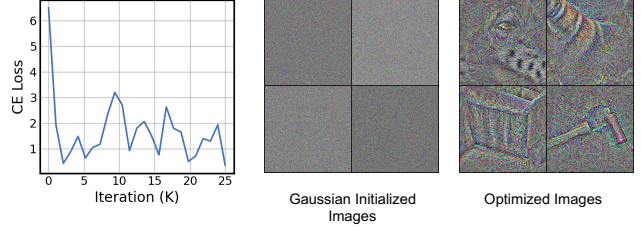
$$V_1 + V_2 + V_3 + V_4 = 4. \quad (10)$$

Figure 4. Training images with Data Mixing.

Algorithm 1: MixMix data synthesis

Input: Pretrained model zoo, subset size m'
Initialize random label Y_i;
Randomly select m' models;
for *all $t = 1, 2, \ldots, T$-iteration* **do**
    Generate mask and mix data features;
    for *all $j = 1, 2, \ldots, m'$-th pretrained model A_j*
    do
        Compute BN statistic loss $L_{\text{BN}}(\hat{X}, \hat{\mu}_j, \hat{\sigma}_j)$;
        Compute mixed CE loss $L_{\text{CE}}(A_j(\hat{X}))$;
    Compute image prior loss $L_{\text{prior}}(X_i)$;
    Descend final loss objective and update X
return MixMix data X

Now assume a mixed input $\hat{V} = [\hat{V}_1, \hat{V}_2, V_3, V_4]$ and output $\hat{o} = 0$ where the first two elements in $\hat{V}$ come from another input image. Then we can obtain following relationships:

$$\begin{cases} V_1 + V_2 + V_3 + V_4 = 4 \\ \hat{V}_1 + \hat{V}_2 + V_3 + V_4 = 0 \end{cases} = \begin{cases} V_3 + V_4 = 4 \\ V_1 + V_2 = 0 \\ \hat{V}_1 + \hat{V}_2 = -4 \end{cases} \quad (11)$$

We can see that Data Mixing can help image inversion because the solution space in Eq. (11) is much smaller than Eq. (10). We also visualize the data-feature mixing using only CE and prior loss for image generation. The training curve as well as the optimized images are shown in Fig. 4. Notably, there are some basic shapes or textures in optimized images if we mix the data features.

Together with Feature Mixing, we formalize the MixMix data synthesis procedure in algorithm 1.

5. Experiments

We conduct our experiments on both CIFAR10 and ImageNet dataset. We set the subset size of Feature Mixing as $m' = 3$ except we mention it. We select 21 models in our pretrained model zoo, including ResNet, RegNet, MobileNetV2, MobileNetV3, MNasNet, VGG, SE-Net, DenseNet, ShuffleNetV2 [16, 42, 44, 19, 47, 46, 34, 21, 22], *etc*. See detailed descriptions in appendix. The width and the height of the bounding box for data mixing are sampled

from a uniform distribution. We use Adam [25] optimizer to optimize the images. Most hyper-parameters and implementation are aligned up with [50], such as multi-resolution training pipeline and image clip after the update. We optimize the images for 5k iteration and use a learning rate of 0.25 followed by a cosine decay schedule. To determine λ, we set it learnable and optimize it by gradient descent, details can be found in the appendix. Training 1024 MixMix images requires approximately 2 hours on 8 1080TIs.

5.1. Analysis of the Synthesized Images

We present some qualitative evaluation in Fig. 5. Notably, MixMix data preserves high fidelity and resembles real images. To test the generalizability of the synthesized images, we report the average classification accuracy (as well as the standard deviation) on 21 different models in the model zoo. We also report the Inception Score (IS) [43] to evaluate the image quality.

Table 2. Classification accuracy evaluated on 21 different models and the Inception Score (IS) metric of the synthesized images.

Methods	Size	Average Acc.	IS
DeepDream-R50 [37]	224	24.9±8.23	6.2
DeepInversion-R50 [50]	224	85.96±5.80	60.6
BigGAN [3]	256	N/A	178.0
SAGAN [53]	128	N/A	52.5
MixMix	224	**96.95±1.53**	92.9

It can be noted from Table 2 that model-specific data (from ResNet-50) has lower average accuracy. The proposed MixMix data achieves near 97% average accuracy and the stablest result. This means that our generated images have obvious class characteristics that all models can recognize. As a consequence, we can safely adopt it for all data-free applications and any architecture. We also compare with some GAN-based image synthesis methods where MixMix can achieve a comparable Inception Score.

5.2. Data-Free Quantization

In this section, we utilize the images generated by MixMix to conduct Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) on ImageNet. Here we adopt two state-of-the-art PTQ and QAT methods, namely BRECQ [28] and LSQ [10].

Post-Training Quantization [28] uses the block output reconstruction to optimize the rounding policy of weight quantization and the activation quantization range. We use 1024 images with a batch size of 32 to optimize the quantized model. Each block is optimized by 20k iterations. We compare ZeroQ [4] and *The Knowledge Within* [15]
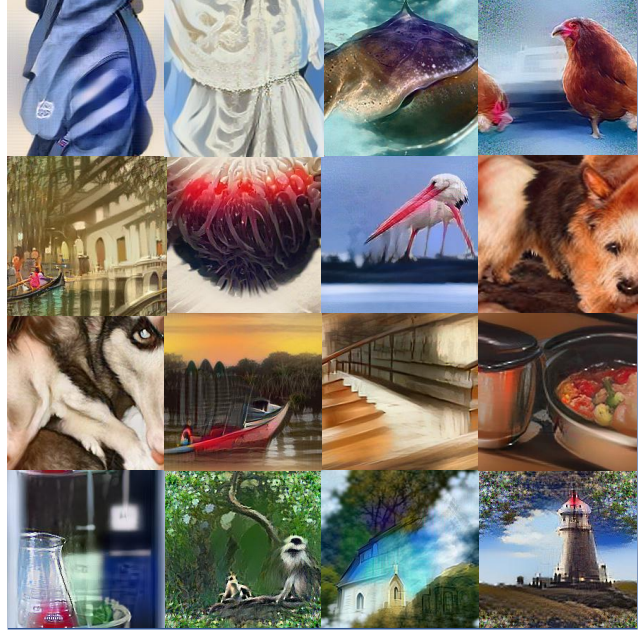

Figure 5. Example images synthesized by 21 different models (Labels: backpack, overskirt, stingray, hen, gondola, anemone, white stork, Norwich terrier, husky, canoe, banister, crock pot, beaker, langur, church, lighthouse).

(abbreviated as KW in the this paper) data on ResNet-50, MobileNet-b (modified version of MobileNetV1 [20] without BN and ReLU after the depthwise convolutional layers), MobileNetV2, and MNasNet. To fairly compare results, we implement ZeroQ and KW to the same pre-trained models and apply the same quantization algorithm. Results are demonstrated in Table 3. For ResNet-50 4-bit quantization, MixMix data has comparable results with the real images (only 0.14% accuracy drop). When the bit-width of weight goes down to 2, the quantization becomes more aggressive and requires high-fidelity data for calibration. In that case, MixMix still achieves the lowest degradation among the existing methods. We next verify data-free quantization on three mobile-platform networks, which face a higher risk of performance degeneration in compression. It can be seen that 4-bit quantization on lightweight networks generally will have much higher accuracy loss even using real images. However, MixMix data still reaches close-to-origin results. As an example, ZeroQ and KW only reach 49.83% and 59.81% accuracy on MobileNetV2, respectively. Whereas, MixMix can boost the performance to 64.01%.

Quantization-Aware Training QAT aims to recover the performance of the quantization neural network in low-bit scenarios. In this work, we leverage the state-of-the-art QAT baseline: Learned Step Size Quantization [10]. In QAT, Straight-Through Estimator (STE) is adopted to compute the gradients of the latent weights. In the case of LSQ,

Table 3. ImageNet top-1 accuracy comparison on PTQ.

Model	Bits(W/A)	Data Source	Top-1 Acc.
ResNet-50 FP: 77.00	4 / 4	Training data	74.72
	4 / 4	ZeroQ [4]	73.73
	4 / 4	KW [15]	74.05
	4 / 4	MixMix	**74.58**
	2 / 4	Training data	68.87
	2 / 4	ZeroQ [4]	64.16
	2 / 4	KW [15]	57.74
	2 / 4	MixMix	**66.49**
MobileNet-b FP: 74.53	4 / 4	Training data	66.11
	4 / 4	ZeroQ [4]	55.93
	4 / 4	KW [15]	61.94
	4 / 4	MixMix	**65.38**
MobileNetV2 FP: 72.49	4 / 4	Training data	64.61
	4 / 4	ZeroQ [4]	49.83
	4 / 4	KW [15]	59.81
	4 / 4	MixMix	**64.01**
MNasNet FP: 73.52	4 / 4	Training data	58.86
	4 / 4	ZeroQ [4]	52.04
	4 / 4	KW [15]	55.48
	4 / 4	MixMix	**57.87**

Table 4. ImageNet top-1 accuracy comparison on QAT.

Model	Bits(W/A)	Data Source	Top-1 Acc.
ResNet-50 FP:77.00	4 / 4	Training data	76.09
	4 / 4	MixMix	**73.39**
	2 / 4	Training data	70.20
	2 / 4	MixMix	**64.60**
MobileNetV2 FP:72.49	4 / 4	Training data	68.50
	4 / 4	KW [15]	66.07
	4 / 4	MixMix	**67.74**

Table 5. ImageNet top-1 accuracy comparison on pruning. UP and SP refer to unstructured and structured pruning, respectively.

Model	Data Source	UP Acc.	SP Acc.
ResNet-50 FP: 77.00	Training data	76.53	70.95
	DeepInversion [50]	71.58	65.07
	MixMix	**75.41**	**69.80**
MobileNet-b FP: 74.53	Training data	72.96	48.38
	DeepInversion [50]	70.56	40.62
	MixMix	**70.64**	**44.82**
MobileNetV2 FP: 72.49	Training data	68.96	45.24
	DeepInversion [50]	47.08	15.32
	MixMix	**66.74**	**42.47**
MNasNet FP: 73.52	Training data	70.43	47.81
	DeepInversion [50]	57.42	22.62
	MixMix	**67.98**	**43.41**

STE is also used to estimate the gradients of quantization step size. Note that our QAT uses per-layer quantization which is more challenging than per-channel quantization used in PTQ. We synthesize 100k images and use a batch size of 128 to finetune the quantization neural network. During QAT, the full precision model serves as the teacher and we use KL loss with temperature $\tau = 3$ as the criterion. We also incorporate the intermediate feature loss proposed in [15]. We finetune the quantized model for 44000 steps and it only takes 3 hours on 8 1080TIs to complete the fine-tuning. We perform W4A4 quantization-aware training on ResNet-50 and MobileNetV2. Additionally, W2A4 quantization is applied to ResNet-50. Results are presented in Table 4. Note that the real training dataset only contains 100k images. In W4A4, we show the MixMix dataset only has 2.6% accuracy reduction compared to the natural images. In W2A4 quantization, the gap between synthesized images and natural images is bigger. We compare an existing work, KW [15], where our MixMix can recover 1.7% accuracy in 4-bit MobileNetV2.

5.3. Data-Free Pruning

Another important compression technique is network pruning. In this section, we validate both unstructured pruning [14] and structured channel pruning [17] using L1-norm magnitude measure. The pruning ratio (or sparsity) is set to 0.5 and 0.2 for unstructured and structured pruning. We also use 1024 images to reconstruct the output feature-maps after pruning as did in [17]. We mainly compare DeepInversion [50] as the baseline method. Table 5 summarizes the results, where the baseline still produces a large gap between the real images. Quantitatively, on sparse ResNet-50, MixMix is nearly 4% higher than DeepInversion. We find the hard case for pruning is MobileNetV2 and MNasNet, especially when conducting channel pruning. In these cases, the improvement of MixMix is much more evident, up to 27% absolute accuracy uplift on MobileNetV2 and 21% on MNasNet.

5.4. Data-Free Knowledge Distillation

In this section, we perform knowledge distillation to verify MixMix. Due to the reproducibility issue, we retrieve the code of DAFL [5] as our code-base to conduct experiments on CIFAR10. We add our MixMix training objective during the GAN training. See the detailed description of the KD experiment settings in the appendix. Here we compare DeepInversion [50] and DAFL [5] as our baseline methods. The results are shown in Table 7. We can see that our MixMix is 3% higher than DeepInversion when distilling VGG11 to learn VGG11. Compared with the original DAFL, our method is also 2.5% higher.

Table 6. Generalizability study via cross-validation. We apply post-training quantization to target models and quantized them to W4A4 (except for EfficientNet which is W4A8). MixMix data only requires one-time synthesis and is generaliable to all models.

Source Data \ Target Model	ResNet-18*	ResNet-50	MobileNet-b*	MobileNetV2	MNasNet	EfficientB0*
KW-ResNet-18	69.08	73.84	61.06	59.79	53.12	67.67
KW-ResNet-50	67.34	74.05	61.07	57.31	47.57	68.33
KW-MobileNet-b	68.28	72.83	61.95	53.04	50.02	58.84
KW-MobileNetV2	63.58	70.20	60.81	59.81	54.08	68.59
KW-MNasNet	66.03	71.45	54.86	59.03	55.48	68.15
KW-EfficientNetB0	65.87	71.34	46.28	60.47	42.84	69.59
MixMix	**69.46**	**74.58**	**65.38**	**64.01**	**57.87**	**70.59**
Training Set	69.52	74.72	66.11	64.63	58.86	70.64

*These models are exceluded from our pretrained model zoo, thus they can test the generalizability of MixMix data.

Table 7. CIFAR10 Data-Free Knowledge Distillation comparison.

Teacher	VGG11	VGG11	ResNet-34
Student	VGG11	ResNet-18	ResNet-18
Teacher Acc.	94.31	94.31	95.42
DeepInversion	90.78	90.36	93.26
DAFL (original)	-	-	92.22
DAFL (+ MixMix)	**93.97**	**91.57**	**94.79**

Table 8. ImageNet pruning on ResNet-50, given different policies when generating the data.

Method	Acc.	Method	Acc.
FMix ($m' = 1$)	64.46	FMix ($m' = 1$)+DMix	68.16
FMix ($m' = 2$)	68.87	FMix ($m' = 2$)+DMix	69.29
FMix ($m' = 3$)	69.49	FMix ($m' = 3$)+DMix	69.80
FMix ($m' = 4$)	69.54	FMix ($m' = 4$)+DMix	69.47

5.5. Generalizability Study

In this section, we verify the generalizability and the transferability of the MixMix data. We consider it as an essential property for the optimal synthesized data, since the real images perform well whatever the model is. We conduct *cross-validation*, *i.e.*, using data inverted from a single model to validate on multiple model compression. We conduct 4-bit post-training quantization and use 1024 synthesized images. Target test models include ResNet-18, ResNet-50, MobileNet-b, MobileNet-V2, MNasNet and EfficientNetB0. Note that *ResNet-18, MobileNet-b, EfficientB0 are not utilized in our pre-trained model zoo, thus their compression performances are effective evaluations of data generalizability*. As a baseline, we implement KW[15] to generate images for the tested models. Thus it sufficiently extracts the information inside the model that is ready to compress.

The results are summarized in Table 6. A general rule for model-specific data is that its good performance is restricted to the original model compression. Take ResNet-50 as an example, synthesized data from the original model achieves 74.05% performances, which is higher than data synthesized from other models. However, ResNet-50 data is yet 2.6% lower than ResNet-18 data on ResNet-18 quantization. Evidence from this table proves our conjecture that the feature bias cannot be eliminated when synthesizing from one specific model. In contrast, MixMix data preserves high generalizability and is applicable to all models even when they are not used in the pre-trained model zoo.

5.6. Ablation Study

In this section, we study the design choice of Feature Mixing (denoted as FMix) and Data Mixing (denoted as DMix). We verify the number of features we mixed and the usage of DMix during image synthesis. We test it on ResNet-50 structured pruning. Table 8 presents the results of ablation experiments. We can find that large m' is helpful to increase the image quality for compression. However, we did not obtain significant accuracy improvement when m' is higher than 3. As for DMix, we find it can consistently improve the performance of image synthesis. However, when $m' = 4$, the improvement is also trivial. This result can be improved by more hyper-parameter tuning. Nevertheless, the effectiveness of MixMix is still evident, and in this work, we mainly use $m' = 3$ with DMix to synthesize data by taking the synthesis time into consideration.

6. Conclusion

In this work, we identify two drawbacks in model-specific inversion method, namely insufficient generalizability and inexact inversion process. The proposed MixMix algorithm improves the existing method by leveraging the knowledge of a collection of models and data mixing. MixMix algorithm is both effective and efficient as it only requires one-time synthesis to generalize any models. Experimental results demonstrate that MixMix establishes a new state-of-the-art for data-free compression.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 2019.

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

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

[4] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[5] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3514–3522, 2019.

[6] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 710–711, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 2017.

[9] Richard M Dudley. *Real analysis and probability*. CRC Press, 2018.

[10] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

[11] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *CoRR*, abs/1912.11006, 2019.

[12] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *arXiv preprint arXiv:1908.05033*, 2019.

[13] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[15] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020.

[16] 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.

[17] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017.

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

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[21] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[23] 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.

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

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[27] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019.

[28] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

[29] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards reproducible and deployable model quantization benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[30] Yuhang Li, Wei Wang, Haoli Bai, Ruihao Gong, Xin Dong, and Fengwei Yu. Efficient bitwidth search for practical mixed precision neural network. *arXiv preprint arXiv:2003.07577*, 2020.

[31] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[33] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *arXiv preprint arXiv:1709.02540*, 2017.

[34] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[35] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[36] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

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

[38] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *arXiv preprint arXiv:2004.10568*, 2020.

[39] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1325–1334, 2019.

[40] NVIDIA. Resnet50v1.5 training. https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/ConvNets/resnet50v1.5. Accessed: Nov-9-2020.

[41] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[42] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[43] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2234–2242. Curran Associates, Inc., 2016.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[45] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for accurate binary neural architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[47] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[48] Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*, 2018.

[49] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 1–17, Cham, 2020. Springer International Publishing.

[50] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep-inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[51] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

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

[53] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

[54] Xiangguo Zhang, Haotong Qin, Yifu Ding, Ruihao Gong, Qinghua Yan, Renshuai Tao, Yuhang Li, Fengwei Yu, and Xianglong Liu. Diversifying sample generation for accurate data-free quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15658–15667, 2021.

[55] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[56] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.