Dataset Condensation with Differentiable Siamese Augmentation

Bo Zhao [1] **Hakan Bilen** [1]

Abstract

In many machine learning problems, large-scale datasets have become the de-facto standard to train state-of-the-art deep networks at the price of heavy computation load. In this paper, we focus on condensing large training sets into significantly smaller synthetic sets which can be used to train deep neural networks from scratch with minimum drop in performance. Inspired from the recent training set synthesis methods, we propose Differentiable Siamese Augmentation that enables effective use of data augmentation to synthesize more informative synthetic images and thus achieves better performance when training networks with augmentations. Experiments on multiple image classification benchmarks demonstrate that the proposed method obtains substantial gains over the state-of-the-art, 7% improvements on CIFAR10 and CIFAR100 datasets. We show with only less than 1% data that our method achieves 99.6%, 94.9%, 88.5%, 71.5% relative performance on MNIST, FashionMNIST, SVHN, CIFAR10 respectively. We also explore the use of our method in continual learning and neural architecture search, and show promising results.

1. Introduction

Deep neural networks have become the go-to technique in several fields including computer vision, natural language processing and speech recognition thanks to the recent developments in deep learning (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016) and presence of large-scale datasets (Deng et al., 2009; Lin et al., 2014; Antol et al., 2015; Abu-El-Haija et al., 2016). However, their success comes at a price, increasing computational expense, as the state-of-the-art models have been primarily fueled by larger models (*e.g.* (Devlin et al., 2018; Radford et al., 2019; Dosovitskiy et al., 2021) and massive datasets (*e.g.* (Kuznetsova et al., 2020; Chen et al., 2020a; Kwiatkowski et al., 2019)). For example, it takes 12.3k TPU days to train EfficientNet-L2 (Xie et al., 2020) on JFT-300M dataset (Sun et al., 2017). To put in a perspective, the energy consumption for training EfficientNet-L2 once is about 3×10^7 J, assuming that the TPU training power is 100W. Ideally, the same energy is sufficient to launch a 30 kg object to the outer space, *i.e.* reaching Kármán line which costs gravitational potential energy 10^6 J/kg. More dramatically, the computational cost significantly increases when better neural architectures are searched and designed due to many trials of training and validation on the dataset for different hyper-parameters (Bergstra & Bengio, 2012; Elsken et al., 2019). Significantly decreasing these costs without degrading the performance of the trained models is one of the long-standing goals in machine learning (Agarwal et al., 2004). To address these challenges, this paper focuses on reducing the training data size by learning significantly smaller synthetic data to train deep neural networks with minimum drop in their performance.

The standard way to reduce the training set size is to use a smaller but equally informative portion of data, namely a *coreset*. In literature, there is a large body of coreset selection methods for various target tasks, *e.g.* accelerating model training in neural architecture search (Shleifer & Prokop, 2019; Such et al., 2020), storing previous knowledge compactly in continual learning (Rebuffi et al., 2017; Toneva et al., 2019) and efficient selection of samples to label in active learning (Sener & Savarese, 2018). However, their selection procedures rely on heuristics and thus do not guarantee any optimal solution for the downstream tasks (*e.g.* image classification). In addition, finding such an informative coreset may not always be possible when the information in the dataset is not concentrated in few samples but uniformly distributed over all of them.

Motivated by these shortcomings, a recent research direction, *training set synthesis* aims at *generating* a small training set which is further used to train deep neural networks for the downstream task (Wang et al., 2018; Sucholutsky & Schonlau, 2019; Bohdal et al., 2020; Such et al., 2020; Nguyen et al., 2021; Zhao et al., 2021). In particular, Dataset Distillation (DD) (Wang et al., 2018) models the network parameters as a function of synthetic training data, and then minimize the training loss on the real training data

[1]School of Informatics, The University of Edinburgh, UK. Correspondence to: Bo Zhao <bo.zhao@ed.ac.uk>, Hakan Bilen <hbilen@ed.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

by optimizing the synthetic data. Sucholutsky & Schonlau (2019) extend DD by learning synthetic images and soft labels simultaneously. Bohdal et al. (2020) simplify DD by only learning the informative soft labels for randomly selected real images. Such et al. (2020) propose to use a generator network instead of directly learning synthetic data. Nguyen et al. (2021) reformulates DD in a kernel-ridge regression which has a closed-form solution. Zhao et al. (2021) propose Dataset Condensation (DC) that "condenses" the large training set into a small synthetic set by matching the gradients of the network parameters w.r.t. large-real and small-synthetic training data. The authors show that DC can be trained more efficiently by bypassing the bi-level optimization in DD while significantly outperforming DD in multiple benchmarks. Despite the recent success of the training set synthesis over the coreset techniques, especially in low-data regime, there is still a large performance gap between models trained on the small synthetic set and those trained on the whole training set. For instance, models that are trained on DD and DC synthetic sets obtain 38.3% and 44.9% accuracy respectively with 10 images per class on the CIFAR10 dataset, while a model trained on the whole dataset (5000 images per class) obtains 84.8% .

An orthogonal direction to increase data efficiency and thus generalization performance is data augmentation, a technique to expand training set with semantic-preserving transformations (Krizhevsky et al., 2012; Zhang et al., 2018; Yun et al., 2019; Chen et al., 2020b; Chen & He, 2020). While they can simply be used to augment the synthetic set that are obtained by a training set synthesis method, we show that naive strategies lead to either drops or negligible gains in performance in Section 4. This is because the synthetic images i) have substantially different characteristics from natural images, ii) are not learned to train deep neural network under various transformations. Thus we argue that an effective combination of these two techniques is non-trivial and demands careful data augmentation design and principled learning procedure.

In this paper we propose a principled method to enable learning a synthetic training set that can be effectively used with data augmentation to train deep neural networks. Our main technical contribution is *Differentiable Siamese Augmentation* (DSA), illustrated in Figure 1, that applies the same randomly sampled data transformation to both sampled real and synthetic data at each training iteration and also allows for backpropagating the gradient of the loss function w.r.t. the synthetic data by differentiable data transformations. Applying various data transformations (*e.g.* 15° clockwise rotation) simultaneously to both real and synthetic images in training has three key advantages. First our method can exploit the information in real training images more effectively by augmenting them in several ways and transfer this augmented knowledge to the synthetic images. Second sharing the same transformation across real and synthetic images allows the synthetic images to learn certain prior knowledge in the real images (*e.g.* the objects are usually horizontally on the ground). Third, most importantly, once the synthetic images are learned, they can be used with various data augmentation strategies to train different deep neural network architectures. We validate these advantages in multiple image classification benchmarks and show that our method significantly outperforms the state-of-the-art with a wide margin, around 7% on CIFAR10/100 datasets[1]. Finally we explore the use of our method in continual learning and neural architecture search, and show promising results.

2. Related Work

In addition to the coreset selection and training set synthesis methods that are discussed in section 1, our method is also related to data augmentation techniques and Generative Adversarial Networks (GANs).

Data Augmentation. Many deep neural networks adopts data transformations for expanding the effective training set size, reducing overfitting and thus improving their performance. The most popular augmentation strategies include color jittering (Krizhevsky et al., 2012), cropping (Krizhevsky et al., 2012), cutout (DeVries & Taylor, 2017), flipping, scale, rotation. More elaborate augmentation strategies are Mixup (Zhang et al., 2018) and CutMix (Yun et al., 2019). These augmentation strategies are typically applied to various image recognition problems where the label is invariant to transformations of the input and the transformations do not have to be be differentiable w.r.t. the original input image.While we also use several data augmentation techniques, our focus is to synthesize training images that results in gradients that are equivariant to the ones from real images. In addition, we use differentiable augmentations such that gradients can go through augmentation function and back-propagate to synthetic data.

Auto-augmentation. This line of work investigates how to automatically find the best augmentation strategy instead of manually designing by either learning a sequence of transformation functions in an adversarial optimization (Ratner et al., 2017), using a reinforcement learning algorithm Cubuk et al. (2019), or learning the parameters of parametric feature augmentations (Yan et al., 2020). In contrast, our goal is not to find the best augmentation for training data but to synthesize the training data that is equipped with the augmentation ability for the downstream task.

GANs & Differentiable Augmentation. GANs (Goodfellow et al., 2014; Mirza & Osindero, 2014; Radford et al., 2015) typically aim at generating real-looking novel images

[1]The implementation is available at https://github.com/VICO-UoE/DatasetCondensation.

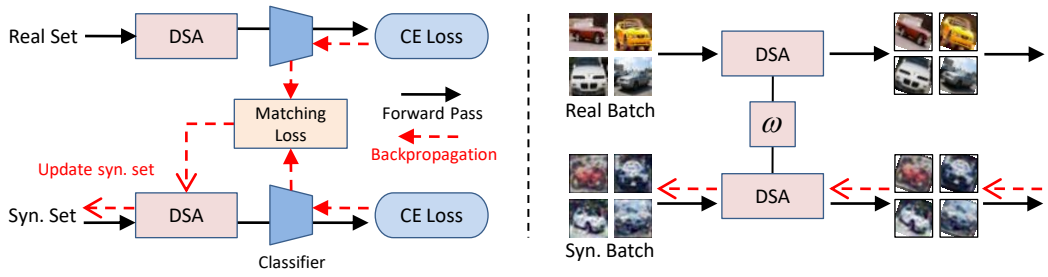

Figure 1. Dataset condensation with differentiable Siamese augmentation. Differentiable Siamese augmentation (DSA) applies the same parametric augmentation (*e.g.* rotation) to all data points in the sampled real and synthetic batches in a training iteration. The gradients of network parameters w.r.t. the sampled real and synthetic batches are matched for updating the synthetic images. A DSA example is given that rotation with the same degree is applied to the sampled real and synthetic batches.

by fooling a discriminator network. Differentiable Augmentation (Zhao et al., 2020a; Tran et al., 2020; Zhao et al., 2020b; Karras et al., 2020) has recently been applied for improving their training and in particular for preventing discriminators from memorizing the limited training set. Though they also apply differentiable augmentation to both real and fake images, augmentations are independently applied to real and fake ones. In contrast we use a Siamese augmentation strategy which is explicitly coordinated to apply the same transformation to both real and synthetic images. In addition, our goal, which is to generate a set of training data that can be used to efficiently train deep neural networks from scratch, differs from theirs and our images do not have to be realistic.

3. Method

Here we first review DC (Zhao et al., 2021), then describe the proposed our DSA method and its training algorithm.

3.1. Dataset Condensation Review

Assume that we are given a large training set $\mathcal{T} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_{|\mathcal{T}|}, y_{|\mathcal{T}|})\}$ with $|\mathcal{T}|$ image and label pairs. DC (Zhao et al., 2021) aims at learning a much smaller set with $|\mathcal{S}|$ synthetic image and label pairs $\mathcal{S} = \{(\boldsymbol{s}_1, y_1), \dots, (\boldsymbol{s}_{|\mathcal{S}|}, y_{|\mathcal{S}|})\}$ from $\mathcal{T}$ such that a deep network trained on $\mathcal{S}$ obtains comparable generalization performance to a deep neural network that is trained on $\mathcal{T}$. Let $\phi_{\boldsymbol{\theta}^\mathcal{T}}$ and $\phi_{\boldsymbol{\theta}^\mathcal{S}}$ denote the deep neural networks with parameters $\boldsymbol{\theta}^\mathcal{T}$ and $\boldsymbol{\theta}^\mathcal{S}$ that are trained on $\mathcal{T}$ and $\mathcal{S}$ respectively. The goal of DC can be formulated as:

$$\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\ell(\phi_{\boldsymbol{\theta}^\mathcal{T}}(\boldsymbol{x}), y)] \simeq \mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\ell(\phi_{\boldsymbol{\theta}^\mathcal{S}}(\boldsymbol{x}), y)] \quad (1)$$

over the real data distribution $P_\mathcal{D}$ with loss function ℓ (*i.e.* cross-entropy loss) . In practice, their generalization performances are measured on an unseen test set.

A possible way to achieve the comparable performance in eq. (1) is obtaining a similar solution to $\boldsymbol{\theta}^\mathcal{T}$, after the parameters of the network are trained on $\mathcal{S}$, *i.e.* $\boldsymbol{\theta}^\mathcal{S} \approx \boldsymbol{\theta}^\mathcal{T}$. However, solving this w.r.t. $\mathcal{S}$ involves nested loop optimization over network parameters $\boldsymbol{\theta}$ and synthetic data $\mathcal{S}$ which is typically not scalable to large models and multi-step optimization. Instead the authors in (Zhao et al., 2021) hypothesize that a similar solution can be achieved, when the parameter updates for $\boldsymbol{\theta}_t^\mathcal{T}$ and $\boldsymbol{\theta}_t^\mathcal{S}$ are approximately equal at each training iteration t, given the same initialization $\boldsymbol{\theta}_0^\mathcal{T} = \boldsymbol{\theta}_0^\mathcal{S}$. In addition, assuming that $\boldsymbol{\theta}_t^\mathcal{S} = \boldsymbol{\theta}_t^\mathcal{T}$ can be satisfied at each iteration, the authors simplify the learning by using a single neural network parameterized by $\boldsymbol{\theta}$ and propose the following minimization problem:

$$\min_\mathcal{S} D(\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{S}, \boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{T}, \boldsymbol{\theta}_t)), \quad (2)$$

where

$$\mathcal{L}(\mathcal{S}, \boldsymbol{\theta}_t) = \frac{1}{|\mathcal{S}|}\sum_{(\boldsymbol{s},y)\in\mathcal{S}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{s}), y)),$$
$$\mathcal{L}(\mathcal{T}, \boldsymbol{\theta}_t) = \frac{1}{|\mathcal{T}|}\sum_{(\boldsymbol{x},y)\in\mathcal{T}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{x}), y))$$

and D is a sum of cosine distances between the two gradients of weights associated with each output node at each layer. We refer the readers to (Zhao et al., 2021) for more detailed explanation.

3.2. Differentiable Siamese Augmentation (DSA)

Here we explain how data augmentation strategies can be effectively used with DC formulation. One naive way is to apply data augmentation to the synthetic images post-hoc, after they are learned. However, this strategy results in negligible performance gains (demonstrated in Section 4), as the synthetic images are not optimized to be augmented. Hence, a more principled way is to apply data augmentation while learning the synthetic images, which can be formulated by rewriting eq. (2):

$$\min_\mathcal{S} D(\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{A}(\mathcal{S}, \omega^\mathcal{S}), \boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{A}(\mathcal{T}, \omega^\mathcal{T}), \boldsymbol{\theta}_t)), \quad (3)$$

where $\mathcal{A}$ is a family of image transformations that preserves the semantics of the input (*i.e.* class label) such as cropping,

color jittering, flipping that are parameterized with $\omega^{\mathcal{S}}$ and $\omega^{\mathcal{T}}$ for the synthetic and real training sets respectively.

Siamese Augmentation. In the standard data augmentation ω is randomly sampled from a predetermined distribution Ω for each image independently. However, randomly sampling both $\omega^{\mathcal{S}}$ and $\omega^{\mathcal{T}}$ is not meaningful in our case, as this results in ambiguous gradient matching problem in eq. (2). For instance, in case of cropping, this would require a particular region of a synthetic image to produce gradients matching to the ones that are generated from different crops of real image at different training iterations. Hence this method results in an averaging affect on the synthetic images and loss of information. To address this issue, we instead use the same transformations across the synthetic and real training sets, *i.e.* $\omega^{\mathcal{S}} = \omega^{\mathcal{T}}$. Thus we use one symbol ω in the remainder of the paper. As two sets have different number of images $|\mathcal{S}| \ll |\mathcal{T}|$ and there is no one-to-one correspondence between them, we randomly sample a single transformation ω and apply it to all images in a minibatch pair at each training iteration. This also avoids the averaging effect in a minibatch. This strategy enables correspondence between the two sets (*e.g.* between 15° clockwise rotation of synthetic and real set) and a more effective way of exploiting the information in the real training images and distilling it to the synthetic images in a more organized way without averaging effect. We illustrate the main idea in Figure 1.

Differentiable Augmentation. Solving eq. (3) for $\mathcal{S}$ involves computing the gradient for the matching loss D w.r.t. the synthetic images $\partial D(\cdot)/\partial \mathcal{S}$ by backpropagation:

$$\frac{\partial D(\cdot)}{\partial \mathcal{S}} = \frac{\partial D(\cdot)}{\partial \nabla_{\boldsymbol{\theta}} \mathcal{L}(\cdot)} \frac{\partial \nabla_{\boldsymbol{\theta}} \mathcal{L}(\cdot)}{\partial \mathcal{A}(\cdot)} \frac{\partial \mathcal{A}(\cdot)}{\partial \mathcal{S}}.$$

Thus the transformation $\mathcal{A}$ has to be differentiable w.r.t. the synthetic images $\mathcal{S}$. Traditionally transformations used for data augmentation are not implemented in a differentiable way, as optimizing input images is not their focus. Note that all the standard data augmentation methods for images are differentiable and can be implemented as differentiable layers. Thus, we implement them as differentiable functions for deep neural network training and allow the error signal to be backpropagated to the synthetic images.

3.3. Training Algorithm

We adopt training algorithm in (Zhao et al., 2021) for the proposed DSA, which is depicted in Alg. 1. To ensure that the generated synthetic images can train deep neural networks from scratch with any randomly initialized parameters, we use an outer loop with K iterations where at each outer iteration we randomly initialize network parameters (*i.e.* $\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}$) from a distribution $P_{\boldsymbol{\theta}_0}$ and train them from scratch. In the inner loop t, we randomly sample an image transformation ω and a minibatch pair $B_c^{\mathcal{T}}$, $B_c^{\mathcal{S}}$ from the real and synthetic sets that contain samples from only class c, compute their average cross-entropy loss and gradients w.r.t. the model parameters separately. Then we compute the gradient matching loss as in eq. (3) and update the synthetic data $\mathcal{S}_c$ by using stochastic gradient descent optimization with $\varsigma_{\mathcal{S}}$ gradient descent steps and $\eta_{\mathcal{S}}$ learning rate. We repeat above steps for every class c in the inner loop t. Alternatively, we update the model parameters $\boldsymbol{\theta}_t$ to minimize the cross-entropy loss on the augmented synthetic data with $\varsigma_{\boldsymbol{\theta}}$ gradient descent steps and $\eta_{\boldsymbol{\theta}}$ learning rate.

Discussion. We observe that using minibatches from multiple classes leads to a slower convergence rate in training. The reason is that when the gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ are averaged over samples from multiple classes, image/class correspondence for synthetic data is harder to retrieve from the gradients.

4. Experiments

4.1. Datasets & Implementation Details

Datasets. We evaluate our method on 5 image classification datasets, MNIST (LeCun et al., 1990), FashionMNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). Both the MNIST and FashionMNIST datasets have 60,000 training and 10,000 testing images of 10 classes. SVHN is a real-world digit dataset which has 73,257 training and 26,032 testing images of 10 numbers. CIFAR10 and CIFAR100 both have 50,000 training and 10,000 testing images from 10 and 100 object categories respectively.

Network Architectures. We test our method on a wide range of network architectures, including multilayer perceptron (MLP), ConvNet (Gidaris & Komodakis, 2018), LeNet (LeCun et al., 1998), AlexNet (Krizhevsky et al., 2012), VGG-11 (Simonyan & Zisserman, 2014) and ResNet-18 (He et al., 2016). We use the ConvNet as the default architecture in experiments unless otherwise indicated. The default ConvNet has 3 duplicate convolutional blocks followed by a linear classifier, and each block consists of 128 filters, average pooling, ReLu activation (Nair & Hinton, 2010) and instance normalization (Ulyanov et al., 2016). We refer to (Zhao et al., 2021) for more details about the above-mentioned architectures. The network parameters for all architectures are randomly initialized with Kaiming initialization (He et al., 2015). The labels of synthetic data are pre-defined evenly for all classes, and the synthetic images are initialized with randomly sampled real images of corresponding class. While our method works well when initialising synthetic images from random noise, initialising them from randomly picked real images leads to better performance. We evaluate the initialization strategies in Section 5.2.

Algorithm 1: Dataset condensation with differentiable Siamese augmentation.

Input: Training set $\mathcal{T}$

Required: Randomly initialized set of synthetic samples $\mathcal{S}$ for C classes, probability distribution over randomly initialized weights $P_{\boldsymbol{\theta}_0}$, deep neural network $\phi_{\boldsymbol{\theta}}$, number of training iterations K, number of inner-loop steps T, number of steps for updating weights $\varsigma_{\boldsymbol{\theta}}$ and synthetic samples $\varsigma_{\mathcal{S}}$ in each inner-loop step respectively, learning rates for updating weights $\eta_{\boldsymbol{\theta}}$ and synthetic samples $\eta_{\mathcal{S}}$, differentiable augmentation $\mathcal{A}_\omega$ parameterized with ω, augmentation parameter distribution Ω, random augmentation $\mathcal{A}$.

for $k = 0, \cdots, K-1$ **do**
  Initialize $\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}$
  for $t = 0, \cdots, T-1$ **do**
    for $c = 0, \cdots, C-1$ **do**
      Sample $\omega \sim \Omega$ and a minibatch pair $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ ▷ $B_c^{\mathcal{T}}$, $B_c^{\mathcal{S}}$ are of class c.
      Compute $\mathcal{L}_c^{\mathcal{T}} = \frac{1}{|B_c^{\mathcal{T}}|} \sum_{(\boldsymbol{x},y)\in B_c^{\mathcal{T}}} \ell(\phi_{\boldsymbol{\theta}_t}(\mathcal{A}_\omega(\boldsymbol{x})), y)$ and $\mathcal{L}_c^{\mathcal{S}} = \frac{1}{|B_c^{\mathcal{S}}|} \sum_{(\boldsymbol{s},y)\in B_c^{\mathcal{S}}} \ell(\phi_{\boldsymbol{\theta}_t}(\mathcal{A}_\omega(\boldsymbol{s}), y)$
      Update $\mathcal{S}_c \leftarrow \mathtt{sgd}_{\mathcal{S}}(D(\nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{S}}(\boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{T}}(\boldsymbol{\theta}_t)), \varsigma_{\mathcal{S}}, \eta_{\mathcal{S}})$
    Update $\boldsymbol{\theta}_{t+1} \leftarrow \mathtt{sgd}_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta}_t, \mathcal{A}(\mathcal{S})), \varsigma_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\theta}})$ ▷ Use $\mathcal{A}$ for the whole $\mathcal{S}$

Output: $\mathcal{S}$

Hyper-parameters and Augmentation. For simplicity and generality, we use one set of hyper-parameters and augmentation strategy for all datasets. We set $K = 1000$, $\varsigma_{\mathcal{S}} = 1$, $\eta_{\boldsymbol{\theta}} = 0.01$, $\eta_{\mathcal{S}} = 0.1$, $T = 1/10/50$ and $\varsigma_{\boldsymbol{\theta}} = 1/50/10$ for 1/10/50 image(s)/class learning respectively as in (Zhao et al., 2021). The minibatch sizes for both real and synthetic data are 256. When the synthetic set has fewer images than 256, we use all the synthetic images of a class c in each minibatch. For data augmentation, we randomly pick one of several augmentations to implement each time. More details can be found in section 4.4.

Experimental Setting. We evaluate our method at three settings, 1/10/50 image(s)/class learning. Each experiment involves two phases. First, we learn to synthesize a small synthetic set (*e.g.* 10 images/class) from a given large real training set. Then we use the learned synthetic set to train randomly initialized neural networks and evaluate their performance on the real testing set. For each experiment, we learn 5 sets of synthetic images and use each set to train 20 randomly initialized networks, report mean accuracy and its standard deviation over the 100 evaluations.

4.2. Comparison to State-of-the-Art

Competitors. We compare our method to several state-of-the-art coreset selection and training set synthesis methods. The coreset selection competitors are *random*, *herding* (Chen et al., 2010; Rebuffi et al., 2017; Castro et al., 2018; Belouadah & Popescu, 2020) and *forgetting* (Toneva et al., 2019). *Random* is a simple baseline that randomly select samples as the coreset. *Herding* is a distance based algorithm that selects samples whose center is close to the distribution center, *i.e.* each class center. *Forgetting* is a statistics based metric that selects samples with the maximum misclassification frequencies during training. Training set synthesis competitors are Dataset Distillation (DD) (Wang et al., 2018), Label Distillation (LD) (Bohdal et al., 2020), Dataset Condensation (DC) (Zhao et al., 2021) which we built our method on. We also provide baseline performances for an approximate upperbound that are obtained by training the models on the whole real training set. Note that we report the results of coreset selection methods and upperbound performances presented in DC (Zhao et al., 2021), as we use the same setting and hyper-parameters, and present the original results for the rest methods.

Results for 10 Category Datasets. Table 1 presents the results of different methods on MNIST, FashionMNIST, SVHN and CIFAR10, which all have 10 classes. We first see that *Herding* performs best among the coreset methods for a limited number of images *e.g.* only 1 or 10 image(s)/class and *random* selection performs best for 50 images/class. Overall the training set synthesis methods outperform the coreset methods which shows a clear advantage of synthesizing images for the downstream tasks, especially for 1 or 10 image(s)/class. Our method achieves the best performance in most settings and in particular obtains significant gains when learning 10 and 50 images/class, improves over the state-of-the-art DC by 7.2% and 6.7% in CIFAR10 dataset for 10 and 50 images per class. Remarkably in MNIST with less than 1% data (50 images/class), it achieves 99.2% which on par with the upperbound 99.6%. We also observe that our method obtains comparable or worse performance than DC in case of 1 image/class. We argue that our method acts as a regularizer on DC, as the synthetic images are forced to match the gradients from real training images under different transformations. Thus we expect that our method works better when the solution space (synthetic set) is larger. Finally, the performance gap between the training

	Img/Cls	Ratio %	Coreset Selection			Training Set Synthesis				Whole Dataset
			Random	Herding	Forgetting	DD†	LD†	DC	*DSA*	
MNIST	1	0.017	64.9±3.5	89.2±1.6	35.5±5.6		60.9±3.2	**91.7±0.5**	88.7±0.6	99.6±0.0
	10	0.17	95.1±0.9	93.7±0.3	68.1±3.3	79.5±8.1	87.3±0.7	97.4±0.2	**97.8±0.1**	
	50	0.83	97.9±0.2	94.8±0.2	88.2±1.2	-	93.3±0.3	98.8±0.2	**99.2±0.1**	
FashionMNIST	1	0.017	51.4±3.8	67.0±1.9	42.0±5.5	-	-	**70.5±0.6**	**70.6±0.6**	93.5±0.1
	10	0.17	73.8±0.7	71.1±0.7	53.9±2.0	-	-	82.3±0.4	**84.6±0.3**	
	50	0.83	82.5±0.7	71.9±0.8	55.0±1.1	-	-	83.6±0.4	**88.7±0.2**	
SVHN	1	0.014	14.6±1.6	20.9±1.3	12.1±1.7	-	-	**31.2±1.4**	27.5±1.4	95.4±0.1
	10	0.14	35.1±4.1	50.5±3.3	16.8±1.2	-	-	76.1±0.6	**79.2±0.5**	
	50	0.7	70.9±0.9	72.6±0.8	27.2±1.5	-	-	82.3±0.3	**84.4±0.4**	
CIFAR10	1	0.02	14.4±2.0	21.5±1.2	13.5±1.2	-	25.7±0.7	**28.3±0.5**	**28.8±0.7**	84.8±0.1
	10	0.2	26.0±1.2	31.6±0.7	23.3±1.0	36.8±1.2	38.3±0.4	44.9±0.5	**52.1±0.5**	
	50	1	43.4±1.0	40.4±0.6	23.3±1.1	-	42.5±0.4	53.9±0.5	**60.6±0.5**	

Table 1. The performance comparison to coreset selection and training set synthesis methods. This table shows the testing accuracies (%) of models trained from scratch on the small coreset or synthetic set. Img/Cls: image(s) per class, Ratio (%): the ratio of condensed images to whole training set. DD† and LD† use LeNet for MNIST and AlexNet for CIFAR10, while the rest use ConvNet for training and testing.

set synthesis methods and upperbound gets larger when the task is more challenging. For instance, in the most challenging dataset CIFAR10, the gap between ours and the upperbound is 24.2%, while it is 0.4% in MNIST in the 50 images/class setting.

Note that we are aware of two recent work, Generative Teaching Networks (GTN) (Such et al., 2020) and Kernel Inducing Point (KIP) (Nguyen et al., 2021). GTN provides only their performance curve on MNIST for 4,096 synthetic images ($\approx$ 400 images/class) but no numerical results, which is slightly worse than our performance with 50 images/class. KIP achieves 95.7±0.1% and 46.9±0.2% testing accuracies on MNIST and CIFAR10 when learning 50 images/class with kernels and testing with one-layer fully connected network, while our results with ConvNet are 99.2±0.1% and 60.6±0.5% respectively. Though our results are significantly better than theirs, two methods are not directly comparable, as KIP and our DSA use different training and testing architectures.

We visualize the generated 10 images/class synthetic sets of MNIST and CIFAR10 in Figure 2. Overall the synthetic images capture diverse appearances in the categories, various writing styles in MNIST and a variety of viewpoints and background in CIFAR10. Although it is not our goal, our images are easily recognizable and more similar to real ones than the ones that are reported in (Wang et al., 2018; Such et al., 2020; Nguyen et al., 2021).

CIFAR100 Results. We also evaluate our method in the more challenging CIFAR100 dataset in which few works report their results. Note that compared to CIFAR10, CIFAR100 is more challenging, as recognizing 10 times more categories requires to learn more powerful features and also there are $\frac{1}{10}$ fewer images per class in CIFAR100. We present our results in Table 2 and compare to the competitive coreset methods (*random*, *herding*) and train set synthesis methods (*LD*, *DC*). Our method obtains 13.9% and 32.3% testing accuracies for 1 and 10 images/class, which improves over DC by 1.1% and 7.1% respectively. Compared to the 10 category datasets, the relative performance gap between the upperbound (56.2±0.3%) and the best performing method (DSA) is significantly bigger in this dataset.

Figure 2. Visualization of the generated 10 images/class synthetic sets of MINIST and CIFAR10. The synthetic images are easy to recognize for human beings.

Img/Cls	Random	Herding	LD†	DC	*DSA*	Whole Dataset
1	4.2±0.3	8.4±0.3	11.5±0.4	12.8±0.3	**13.9±0.3**	56.2±0.3
10	14.6±0.5	17.3±0.3	-	25.2±0.3	**32.3±0.3**	

Table 2. The performance (%) comparison on CIFAR100 dataset. LD† use AlexNet for CIFAR100, while the rest use ConvNet.

C\T	MLP	ConvNet	LeNet	AlexNet	VGG	ResNet
MLP	76.5±1.2	73.1±3.6	80.4±2.9	78.2±6.4	58.7±6.5	78.7±3.9
ConvNet	75.6±1.1	88.8±0.8	84.8±1.5	84.7±1.5	83.5±1.7	89.3±0.8
LeNet	76.5±0.9	86.6±1.5	83.9±1.6	83.9±1.2	81.1±2.3	88.2±0.9
AlexNet	76.1±0.8	87.6±0.8	84.2±1.6	84.6±1.7	82.0±2.1	88.8±0.8
VGG	75.8±1.0	88.9±0.7	84.5±1.6	85.0±1.4	83.2±1.9	88.9±1.0
ResNet	75.8±1.0	88.6±0.8	84.8±1.7	84.8±1.2	82.4±1.9	89.5±1.0

Table 3. Cross-architecture performance (%). We learn to condense the training set on one architecture (C), and test it on another architecture (T). We learn 1 image/class condensed set on MNIST.

4.3. Cross-Architecture Generalization

Here we study the cross-architecture performance of our model and report the results in Table 3 in MNIST for 1 image/class. To this end, we use different neural network architectures to learn the synthetic images and further use them to train classifiers. The rows indicate the architecture which is used to learn the synthetic images and columns show the architectures that we train classifiers. The results show that synthetic images learned by the convolutional architectures (ConvNet, LeNet, AlexNet, VGG and ResNet) perform best and generalizes to the other convolutional ones, while the MLP network produces less informative synthetic images overall. Finally the most competitive architecture, ResNet provides the best results when trained as a classifier on the synthetic images.

4.4. Ablation Study

Effectiveness of DSA. Here we study the effect of design choices in the proposed DSA in terms of test performance on CIFAR10 for 10 images/class and report it in Table 4. One can apply image transformations to the real and synthetic set while learning the synthetic images, also to the synthetic images while training a classifier in the second stage. In addition, the same image transformation can be applied to all images in a real and synthetic minibatch pair (denoted as $\mathcal{A}_\omega$) or an independently sampled image transformation can be applied to each image (denoted as $\mathcal{A}$). Note that the former corresponds to our proposed Siamese augmentation and we test different augmentation schemes for cropping, flipping, scaling and rotation.

The results verify that the proposed Siamese augmentation always achieves the best performance when used with individual augmentation. The largest improvement is obtained by applying our Siamese augmentation with cropping. Specifically, using Siamese augmentation with cropping achieves 3.6% improvement compared to no data augmentation (A). Note that (A) corresponds to DC (Zhao et al., 2021) with initialization from real images. While smaller improvement of 1.4% can be obtained by applying cropping only to synthetic data in test phase (B), DSA provides a further 2.2% over this. Applying cropping only to the real or synthetic images (C and D) degrades the performance and obtains worse performance than no data augmentation (A). Similarly, applying independent transformations to the real and synthetic images when learning synthetic images, *i.e*. (F), leads to worse performance than (A). Finally, the Siamese augmentation method performs worse than (A) when no data augmentation is used to train the classifier (E). This shows that it is important to apply data augmentation consistently in both stages. The effects on other augmentation strategies may be slightly different but similar to those on cropping augmentation.

	Condense		Test	Test Performance (%)			
	Real	Synthetic	Synthetic	Crop	Flip	Scale	Rotation
Ours	$\mathcal{A}_\omega$	$\mathcal{A}_\omega$	$\mathcal{A}$	49.1±0.6	47.9±0.7	46.9±0.5	46.8±0.6
(A)	-	-	-	45.5±0.6	45.5±0.6	45.5±0.6	45.5±0.6
(B)	-	-	$\mathcal{A}$	46.9±0.6	46.1±0.6	45.7±0.5	45.0±0.5
(C)	$\mathcal{A}$	-	$\mathcal{A}$	42.8±0.7	46.2±0.6	44.5±0.6	44.5±0.6
(D)	-	$\mathcal{A}$	$\mathcal{A}$	44.6±0.7	46.8±0.6	45.4±0.6	45.9±0.7
(E)	$\mathcal{A}_\omega$	$\mathcal{A}_\omega$	-	43.4±0.5	46.4±0.6	45.7±0.6	46.3±0.5
(F)	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	44.5±0.5	46.9±0.6	45.7±0.5	45.8±0.5

Table 4. Ablation study on augmentation schemes in CIFAR10 for 10 synthetic images/class. $\mathcal{A}_\omega$ denotes Siamese augmentation when applied to both real and synthetic data, while $\mathcal{A}$ denotes augmentation that is not shared across real and synthetic minibatches.

Augmentation Strategy. Our method can be used with the common image transformations. Here we investigate the performance of our method with several popular transformations including color jittering, cropping, cutout, flipping, scaling, rotation on MNIST, FashionMNIST, SVHN and CIFAR10 for 10 images/class setting. We also show a simple combination strategy that is used as the default augmentation in experiments by randomly sampling one of these six transformations at each time. The exception is that flipping is not included in the combination for the two number datasets - MNIST and SVHN, as it can change the semantics of a number. Note that our goal is not to exhaustively find the best augmentation strategy but to show that our augmentation scheme can be effectively used for dataset condensation and we leave a more elaborate augmentation strategy for future work.

Table 5 depicts the results for no transformation, individual transformations and as well as the combined strategy. We find that applying all the augmentations improve the performance on CIFAR10 compared to the baseline (None). Cropping is the most effective single transformation that can increase the testing accuracy from 45.5% to 49.1%. The combination of these augmentations further improves the performance to 52.1%. Interestingly cropping and cutout transformations degrade the performance of SVHN, as SVHN images are noisy and some include multiple digits and these transformations may pick the wrong patch of images. Nevertheless, we still observe that the combined strategy obtains the best performance in all datasets.

4.5. Continual Learning

Here we apply our method to a continual learning task (Rebuffi et al., 2017; Castro et al., 2018; Aljundi et al., 2019) where the tasks are incrementally learned on three digit recognition datasets, SVHN (Netzer et al., 2011), MNIST (LeCun et al., 1998) and USPS (Hull, 1994) as in (Zhao et al., 2021) and the goal is to preserve the performance in the seen tasks while learning new ones. We build our model on the popular continual learning baseline – EEIL (Castro et al., 2018) which leverages memory rehearsal and

	Img/Cls	None	Color	Crop	Cutout	Flip	Scale	Rotate	Combination
MNIST	10	96.4±0.1	96.5±0.1	97.2±0.1	96.5±0.1	-	97.2±0.1	97.2±0.1	97.8±0.1
FashionMNIST	10	82.5±0.3	82.9±0.3	83.3±0.3	84.0±0.3	83.1±0.2	84.0±0.4	83.1±0.3	84.6±0.3
SVHN	10	76.7±0.6	77.4±0.5	75.9±0.8	73.1±0.6	-	78.0±0.5	77.4±0.4	79.2±0.5
CIFAR10	10	45.5±0.6	47.6±0.5	49.1±0.6	48.1±0.5	47.9±0.7	46.9±0.5	46.8±0.6	52.1±0.5

Table 5. Performance with different augmentation strategies. Flipping is not suitable for number datasets - MNIST and SVHN. Combination means randomly sampling one from the six/five transformations to implement each time.

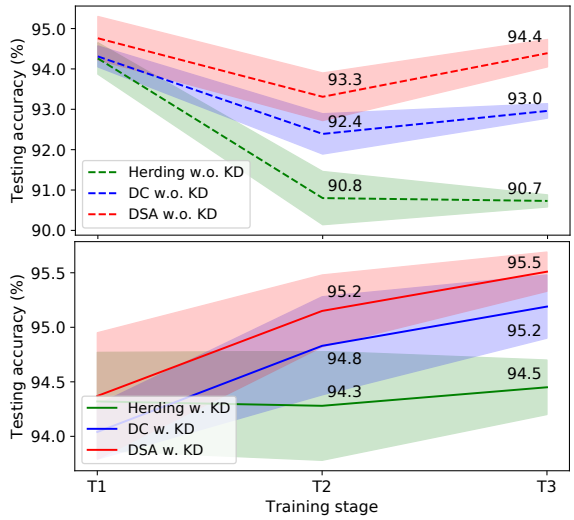

Figure 3. Continual learning performance. We compare to the original EEIL (Castro et al., 2018) denoted as Herding and DC (Zhao et al., 2021) under two settings: with and without knowledge distillation. T1, T2, T3 are three learning stages.

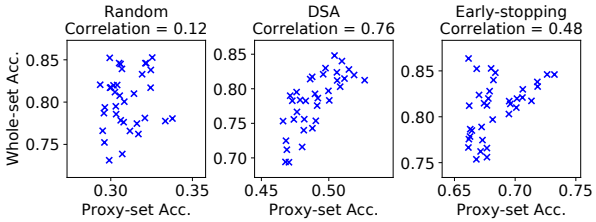

Figure 4. The distribution of correlation between proxy-set performance and whole-dataset performance on top 5% architectures.

knowledge distillation (Hinton et al., 2015) to mitigate catastrophic forgetting of old tasks. We replace the sample selection strategy, *i.e*. herding, with our dataset condensation method for memory construction and keep the rest the same. The memory budget is 10 images/class for all seen classes. We refer to (Zhao et al., 2021) for more details.

Figure 3 depicts the results of EEIL with three memory construction strategies - herding (Castro et al., 2018), DC (Zhao et al., 2021) and our DSA under two settings - with and without knowledge distillation. The results show that DSA always outperforms the other two memory construction methods. Especially, DSA achieves 94.4% testing accuracy after learning all three tasks without knowledge distillation, which surpasses DC and herding by 1.4% and 3.7% respectively. It indicates that the synthetic images learned by our method are more informative for training models than those produced by competitors.

4.6. Neural Architecture Search

Our method has substantial practical benefits when one needs to train many neural networks in the same dataset. One such application is neural architecture search (NAS) (Zoph et al., 2018) which aims to search the best network architecture for a given dataset. Here our goal is to verify that a small set of synthetic images learned by our method can be used as a proxy set to efficiently train many networks and the test performances of the neural architectures are correlated to the ones trained on the original training set.

To this end, we design a set of candidate neural network architectures based on the modular ConvNet by varying its depth, width, pooling, activation and normalization layers which produces 720 candidates in total. We refer to (Zhao et al., 2021) for more details. We train these models on the whole CIFAR10 training set for obtaining the ground-truth performance and also three small proxy-sets that are obtained with *random*, *DSA* and *early-stopping* (Li & Talwalkar, 2020). We randomly select 10 images/class in *random* and learn 10 images/class condensed set with the default ConvNet in *DSA* as the proxy-sets. We train models 300 epochs in *random* and *DSA*. In *early-stopping* we train models for same amount of iterations to DSA (300 iterations with batch size 100) on the whole original training set. Both *random* and *DSA* use 100 images (0.2% of whole dataset) in total, while *early-stopping* uses 3×10^4 images (60% of whole dataset). For the whole set baseline, we train models for 100 epochs, which is sufficiently long to converge. Finally we pick the best performing architectures that are trained on each proxy set, train them on the original training set from scratch and report their test set performance.

We report the results in Table 6 in the performance of selected best architecture, correlation between performances of proxy-set and whole-dataset training, training time cost and storage cost. The correlation, *i.e*. Spearman's rank correlation coefficient, is calculated on the top 5% candidate architectures of each proxy-set, which is also illustrated in Figure 4. The proxy-set produced by our method achieves

	Random	DSA	Early-stopping	Whole Dataset
Performance (%)	78.2	81.3	**84.3**	85.9
Correlation	0.12	**0.76**	0.48	1.00
Time cost (min)	**32.5**	**44.5**	**32.6**	3580.2
Storage (imgs)	$\mathbf{10^2}$	$\mathbf{10^2}$	3×10^4	5×10^4

Table 6. Neural architecture search on proxy-sets and whole dataset. The search space is 720 ConvNets. We do experiments on CIFAR10 with 10 images/class randomly selected coreset and synthetic set learned by *DSA*. *Early-stopping* means training models on whole dataset but with the same iterations as *random* and *DSA*.

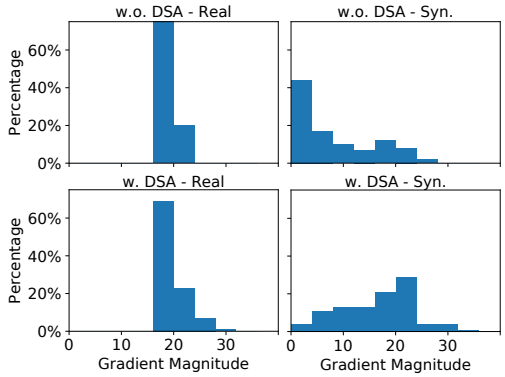

Figure 5. Gradient magnitude distribution w.r.t. real/synthetic data.

the strongest correlation - 0.76, while the time cost of implementing NAS on our proxy-set is only 1.2% of implementing NAS on the whole dataset. This promising performance indicates that our DSA can speed up NAS by training models on small proxy-set. Although the model chosen by *early-stopping* achieves better performance than ours, *early-stopping* requires two orders of magnitude more training images than ours. In addition, the correlation (0.48) between *early-stopping* performance and whole-dataset training performance is significantly lower than ours (0.76).

5. Discussion

5.1. Why Does DSA Work?

In this section, we attempt to shed light onto why DSA leads to better synthetic data. We hypothesize that the Siamese augmentation acts as a strong regularizer on the learned high-dimensional synthetic data $\mathcal{S}$ and alleviates its overfitting to the real training set. We refer to (Hernández-García & König, 2018) for more elaborate analysis of the relation between data augmentation and regularization. This can be shown more explicitly by reformulating eq. (3) over multiple randomly sampled augmentations:

$$\min_{\mathcal{S}} \sum_{\omega \sim \Omega} D(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{A}(\mathcal{S}, \omega), \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{A}(\mathcal{T}, \omega), \boldsymbol{\theta})), \quad (4)$$

which forces the synthetic set to match the gradients from the real set under multiple transformations ω when sampled from the distribution Ω and renders the optimization harder and less prone to overfitting.

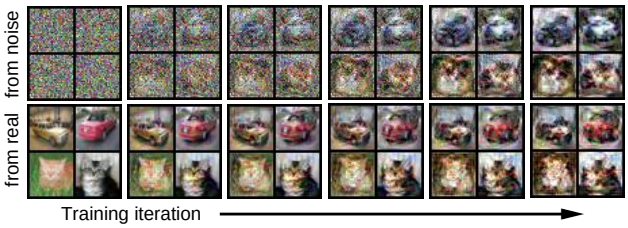

Figure 6. The learning/rendering process of two classes in CIFAR10 initialized from random noise and real images respectively.

We also quantitatively analyze this by reporting the gradient magnitude distribution $||\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{T})||$ and $||\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{S})||$ for real and synthetic sets respectively in Figure 5 when learning 10 images/class synthetic set on CIFAR10 with and without DSA. The gradients are obtained at the training iteration $k = 1000$ (see Alg. 1). We see that the gradient magnitudes from the synthetic data quickly vanishes and thus leads to a very small updates in absence of DSA, while the synthetic images can still be learned with DSA. Note that as backpropagation involves successive products of gradients, the updates for $\mathcal{S}$ naturally vanishes when multiplied with small $||\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{S})||$.

5.2. Initializing Synthetic Images

In our experiments, we initialize each synthetic image with a randomly sampled real training image (after standard image normalization) from the corresponding category. After the initialization, we update them by using the optimization in eq. (3). Once they are trained, they are used to train neural networks without any post-processing. In Figure 6, we illustrate the evolution of synthetic data initialized from random noise and real images from car and cat categories through our training in CIFAR10. While we see significant changes over the initialization in both cases, the ones initialized with real images preserve some of their contents such as object pose and color.

6. Conclusion

In this paper, we propose a principled dataset condensation method – Differentiable Siamese Augmentation – to enable learning synthetic training set that can be effectively used with data augmentation when training deep neural networks. Experiments and ablation study show that the learned synthetic training set can be used to train neural networks with data augmentation and achieve significantly better performance (about 7% improvement on CIFAR10/100) than state-of-the-art methods. We also show promising results when applying the proposed method to continual learning and neural architecture search.

Acknowledgment. This work is funded by China Scholarship Council 201806010331 and the EPSRC programme grant Visual AI EP/T028572/1.

References

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

Belouadah, E. and Popescu, A. Scail: Classifier weights scaling for class incremental learning. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

Bohdal, O., Yang, Y., and Hospedales, T. Flexible dataset distillation: Learn labels instead of images. *Neural Information Processing Systems Workshop*, 2020.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 233–248, 2018.

Chen, H., Xie, W., Vedaldi, A., and Zisserman, A. Vgg-sound: A large-scale audio-visual dataset. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020a.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020b.

Chen, X. and He, K. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Chen, Y., Welling, M., and Smola, A. Super-samples from kernel herding. *The Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence*, 2010.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021.

Elsken, T., Metzen, J. H., Hutter, F., et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

Gidaris, S. and Komodakis, N. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hernández-García, A. and König, P. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

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

Hull, J. J. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. Training generative adversarial networks with limited data. *Neural Information Processing Systems*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., and Ferrari, V. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377. PMLR, 2020.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel-ridge regression. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=l-PrrQrK0QR.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Ratner, A. J., Ehrenberg, H. R., Hussain, Z., Dunnmon, J., and Ré, C. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30:3239, 2017.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. *ICLR*, 2018.

Shleifer, S. and Prokop, E. Using small proxy datasets to accelerate hyperparameter search. *arXiv preprint arXiv:1906.04887*, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Such, F. P., Rawal, A., Lehman, J., Stanley, K. O., and Clune, J. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. *International Conference on Machine Learning (ICML)*, 2020.

Sucholutsky, I. and Schonlau, M. Soft-label dataset distillation and text dataset distillation. *arXiv preprint arXiv:1910.02551*, 2019.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. *ICLR*, 2019.

Tran, N.-T., Tran, V.-H., Nguyen, N.-B., Nguyen, T.-K., and Cheung, N.-M. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 2020.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

Yan, B., Zhou, C., Zhao, B., Guo, K., Yang, J., Li, X., Zhang, M., and Wang, Y. Augmented bi-path network for few-shot learning. *International Conference on Pattern Recognition (ICPR)*, 2020.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations (ICLR)*, 2018.

Zhao, B., Mopuri, K. R., and Bilen, H. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=mSAKhLYLSsl.

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. Differentiable augmentation for data-efficient gan training. *Neural Information Processing Systems*, 2020a.

Zhao, Z., Zhang, Z., Chen, T., Singh, S., and Zhang, H. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020b.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.