sensors

MDPI

Article

Dataset Condensation via Expert Subspace Projection

Zhiheng Ma [1], Dezheng Gao [2], Shaolei Yang [3], Xing Wei [3,*] and Yihong Gong [2,3]

[1] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China; zh.ma@siat.ac.cn
[2] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China; gaodezheng@stu.xjtu.edu.cn (D.G.); ygong@mail.xjtu.edu.cn (Y.G.)
[3] School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China; yangshaolei@stu.xjtu.edu.cn
* Correspondence: weixing@mail.xjtu.edu.cn

Abstract: The rapid growth in dataset sizes in modern deep learning has significantly increased data storage costs. Furthermore, the training and time costs for deep neural networks are generally proportional to the dataset size. Therefore, reducing the dataset size while maintaining model performance is an urgent research problem that needs to be addressed. Dataset condensation is a technique that aims to distill the original dataset into a much smaller synthetic dataset while maintaining downstream training performance on any agnostic neural network. Previous work has demonstrated that matching the training trajectory between the synthetic dataset and the original dataset is more effective than matching the instantaneous gradient, as it incorporates long-range information. Despite the effectiveness of trajectory matching, it suffers from complex gradient unrolling across iterations, which leads to significant memory and computation overhead. To address this issue, this paper proposes a novel approach called Expert Subspace Projection (ESP), which leverages long-range information while avoiding gradient unrolling. Instead of strictly enforcing the synthetic dataset's training trajectory to mimic that of the real dataset, ESP only constrains it to lie within the subspace spanned by the training trajectory of the real dataset. The memory-saving advantage offered by our method facilitates unbiased training on the complete set of synthetic images and seamless integration with other dataset condensation techniques. Through extensive experiments, we have demonstrated the effectiveness of our approach. Our method outperforms the trajectory matching method on CIFAR10 by 16.7% in the setting of 1 Image/Class, surpassing the previous state-of-the-art method by 3.2%.

Keywords: dataset condensation; synthetic data; subspace optimization; deep learning

Citation: Ma, Z.; Gao, D.; Yang, S.; Wei, X.; Gong, Y. Dataset Condensation via Expert Subspace Projection. *Sensors* **2023**, *23*, 8148. https://doi.org/10.3390/s23198148

Academic Editor: Huafeng Li

Received: 12 August 2023
Revised: 19 September 2023
Accepted: 26 September 2023
Published: 28 September 2023

1. Introduction

With the rapid development of the Internet, a growing number of large-scale datasets are being collected for obtaining state-of-art machine learning models in multiple fields, including computer vision, natural language processing, and speech recognition [1]. Such rapid growth of dataset scale results in increasingly expensive model training, and at some scales, even storing and preprocessing the data are burdensome. For instance, the training of a recent language model, GPT-3 [2], consumes an astonishing 190 MWh of electricity, generating approximately 85,000 kg of CO_2, according to US carbon emission standards, which is equivalent to the emissions of driving a car 700,000 km. An intuitive solution is to select a representative subset from the original dataset, commonly referred to as coreset selection. However, previous work shows that, when facing strict compression ratios, coreset selection methods suffer from severe information loss [1] and cannot compete with dataset condensation techniques [3,4], which distill the original dataset into a much smaller synthetic dataset. Typically, this type of method adopts bi-level optimization, which involves an inner optimization for model updates and an outer optimization for

synthetic image updates. To distill the essential information from the original dataset into the synthetic dataset, a suitable matching objective must be defined.

The matching objective includes distribution matching [5], gradient matching [6,7], and meta-model matching [8,9]. Notably, a trajectory matching objective [10] has recently been introduced, demonstrating significant performance improvements over other matching objectives. This objective aims to align the trajectory of the network trained with the synthetic dataset images with the parameter trajectory of the network trained with the original dataset. Unlike gradient matching methods, which only consider the instantaneous training dynamics (i.e., gradients) between the synthetic and real datasets, trajectory matching recognizes that the long-range training dynamics (i.e., training trajectory) provide more informative constraints for dataset condensation.

However, the limitation of trajectory matching is the substantial cost of computation and memory involved in executing multiple unrolled gradient computations for the recursive computational graph when mimicking the training trajectory. This hampers the feasibility of training on the complete set of synthetic images without resorting to slicing and may introduce bias when conducting mini-batch training on the sliced data, ultimately affecting the final performance.

This paper presents a novel matching objective called Expert Subspace Projection (ESP), which effectively guides the dataset condensation process with long-range training dynamics while significantly reducing computational and memory costs compared to trajectory matching. Instead of strictly enforcing the synthetic dataset's training trajectory to mimic the real dataset, ESP only constrains it to lie within the subspace spanned by the training trajectory of the real dataset. Our core technical concept is illustrated in Figure 1. We extract an arbitrary segment from the parameter trajectory obtained through training on the original dataset D_{real}, whose starting point is $\vec{\theta}^*_t$ and ending point is $\vec{\theta}^*_{t+T}$. For training with the synthetic data, the neural network is initialized with $\vec{\theta}^*_t$, and the gradient of the parameters at that point is $\vec{G}$. In gradient matching [6], the objective is to align $\vec{G}$ with $\vec{\theta}^*_{t+1} - \vec{\theta}^*_t$. Conversely, in trajectory matching [10], the objective is to align $\vec{\theta}_{t+T}$ with $\vec{\theta}^*_{t+T}$ after T iterations. Both methods have their advantages and disadvantages. Gradient matching, although computationally simpler, cannot effectively utilize long-range information. On the other hand, trajectory matching has the ability to incorporate long-range information but requires the gradient to be unrolled through T iterations during computation.

Our method, however, circumvents the drawbacks of both approaches while inheriting their respective advantages. First, we construct a subspace $\mathbb{S}_{\tau^*}$ span by means of the training trajectory $\{\vec{\theta}^*_t\}^{t+T}_t$. This subspace effectively captures a substantial amount of information relevant to the training trajectory using a real dataset. Consequently, by confining the optimization gradient $\vec{G}$ of each step within this subspace when training with the synthetic dataset, we are able to distill a significant portion of the information inherent in the real dataset into the synthetic one. Specifically, we define a new objective function $\mathcal{L}_{Proj}$ to penalize the norm of the residual vector between $\vec{G}$ and its projection $\vec{G}_{\mathbb{S}_{\tau^*}}$ within the subspace $\mathbb{S}_{\tau^*}$. This approach not only circumvents the requirement for gradient unrolling across T iterations but also effectively utilizes the long-range information embedded within the optimization process. The memory consumption of our method is not affected by the number of steps in the neural network optimization (inner optimization) since the synthetic images are updated at every step. This significantly reduces the spatial complexity of training compared to previous trajectory methods. Consequently, our method facilitates training on the entire set of synthetic images without the need for slicing and can be seamlessly integrated with other techniques such as distribution matching [5] and KFS [1]. Our method has been extensively tested on four widely used data condensation benchmarks, and the results demonstrate its remarkable effectiveness. In particular, our ESP method outperforms trajectory matching by a significant margin, leading to the establishment of a new state-of-the-art performance.

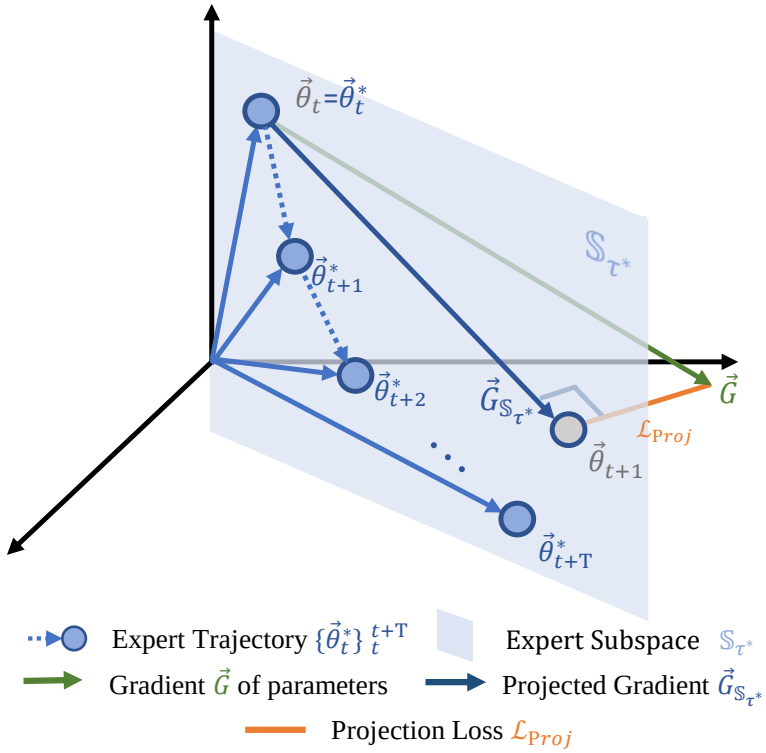

Figure 1. Diagram illustrating the proposed Expert Subspace Projection (ESP) method. The expert trajectory consists of weight snapshots obtained during training with the original dataset D_{real}. Each node $\vec{\theta}_t^*$ represents a saved weight snapshot at the end of step t. The subspace $\mathbb{S}_{\tau^*}$ is spanned by the set of weight snapshots $\{\vec{\theta}_t^*\}_t^{t+T}$. $\vec{G}$ represents the parameter gradient vector generated during training with the synthetic dataset D_{syn}. The subspace projection loss $\mathcal{L}_{Proj}$ penalizes the norm of the residual vector between $\vec{G}$ and its projection $\vec{G}_{\mathbb{S}_{\tau^*}}$ within the subspace $\mathbb{S}_{\tau^*}$.

2. Related Work

2.1. Dataset Condensation

In order to reduce the resources required for deep neural networks, researchers usually use knowledge distillation [11–14] to distill complex and large models into smaller ones, while still ensuring comparable results to those before compression. As technology has evolved, the concept of knowledge distillation began to be transferred to the datasets. Wang et al. [4] first proposed dataset condensation, which uses meta-learning [8,9] methods to compress the knowledge of the entire training dataset into a small amount of synthetic data and achieves high accuracy through several steps of gradient descent on the synthetic data. Subsequently, many works have utilized gradient matching [6,7,10] and distribution matching [1,5] for optimization. Dataset Condensation (DC) [6] assumes that the optimization process of the synthetic dataset is very close to the real dataset, so it optimizes the synthetic dataset by matching the optimization trajectory of the model trained on the synthetic dataset with the optimization trajectory of the real dataset. Differentiable Siamese Augmentation (DSA) [7] is further work based on DC [6] and uses a set of data enhancement strategies while learning the synthetic image, thereby enhancing the information in the real training image and transferring this enhanced knowledge to the synthetic image. Distribution matching (DM) [5] matches the features of the real samples and the synthetic samples that are output at the last layer of the neural network. These neural networks are randomly initialized to ensure computational efficiency and, at the same time, very high accuracy. Knowledge Factorization and Sharing (KFS) [1] also uses distribution matching [5], introducing a new latent code decoder architecture, which greatly increases the number of modalities of synthetic images with the same number of parameters, thus achieving a new state of the art.

Trajectory matching [10] encourages the synthetic dataset to mimic the long-range training dynamic of the real dataset by mimicking the expert trajectory generated by the real dataset. Although long-range information helps it achieves satisfactory results, the method requires accumulating multiple computational graphs, which greatly increases memory consumption and even introduces a subsampling bias as a result of having to reduce the batch size for saving memory. Our ESP method alleviates the memory consumption issue by projecting the model gradient into expert subspace instead of matching model parameters, allowing the model to be trained on the complete synthetic dataset while also utilizing the long-range information of expert trajectories.

2.2. Subspace Training

Generally, deep neural networks come with a large number of parameters, which tend to have strong correlations, thus resulting in great redundancy. Guy et al. [15] first proposed the hypothesis that, in various large-scale deep learning scenarios, gradients dynamically converge to a very small subspace after short-term training, so gradient descent in the subspace will yield a similar loss reduction. Li et al. [16] try to optimize network parameters in a small, random subspace instead of the original parameter space, then slowly increase the dimension of this subspace. Eventually, the authors find that the intrinsic dimension required for model training is smaller than one might think. While this training holds promise for more efficient and scalable optimization schemes, its practical application is limited by poor optimization performance. Gressmann et al. [17] made some optimizations to the stochastic subspace approach, achieving further improvements by applying independent projections to different parts of the network, making the approximation more efficient as the network dimensionality grows. After that, Li et al. [18] extracted the landscape by analyzing the optimization trajectories, while also verifying that many standard neural network structures can be trained well with only 40 independent variables and that the performance is almost the same as conventional training with all parameters. Inspired by [18], we span the parameters from the expert trajectory into an expert subspace and encourage the synthetic data to learn information about real data within the expert subspace, thereby optimizing synthetic data in an efficient way.

2.3. Coreset Selection

Coreset selection [19–23] is an approximate replacement of the original large dataset with a small dataset such that the small dataset still provides rich information, making the accuracy on the test dataset very close to the original dataset. However, such methods often come with a trade-off between performance and dataset size, as they produce a rough approximation of the full dataset. Dataset condensation is very similar to coreset selection, but dataset condensation [1,5,6,10] is more robust. It mainly uses the original dataset to synthesize some learnable pictures and then captures the rich information encoded in the original dataset to realize the compression of the original dataset. These learned images do not appear in the original dataset.

3. Method

The primary objective of dataset condensation is to efficiently compress a real dataset D_{real} into a significantly smaller synthetic dataset D_{syn} while minimizing the loss in performance during downstream training. In line with previous studies [6,7,10], our methods also embrace a bi-level optimization framework. This framework consists of an inner optimization for the neural network's parameters and an outer optimization for synthetic images.

To begin with, we present the comprehensive framework of our Expert Subspace Projection (ESP) method, as illustrated in Figure 2. In the inner optimization, we utilize the binary cross-entropy (BCE) loss as the objective function. This loss penalizes the misclassification of synthetic images based on their pre-defined ground-truth labels. Notably, rather than using the original gradient descent, we update the neural network's parameters using projected gradient descent within the expert subspace. This constraint ensures that

the training trajectory (student trajectory) remains confined within the expert subspace. In the outer optimization, we employ a combination of the projection loss and the distribution matching loss [5] as the objective function. The first loss term penalizes any deviation of gradients from the desired subspace, while the second term aligns the feature distribution between real and synthetic images. During this process, the synthetic images are updated using normal gradient descent.

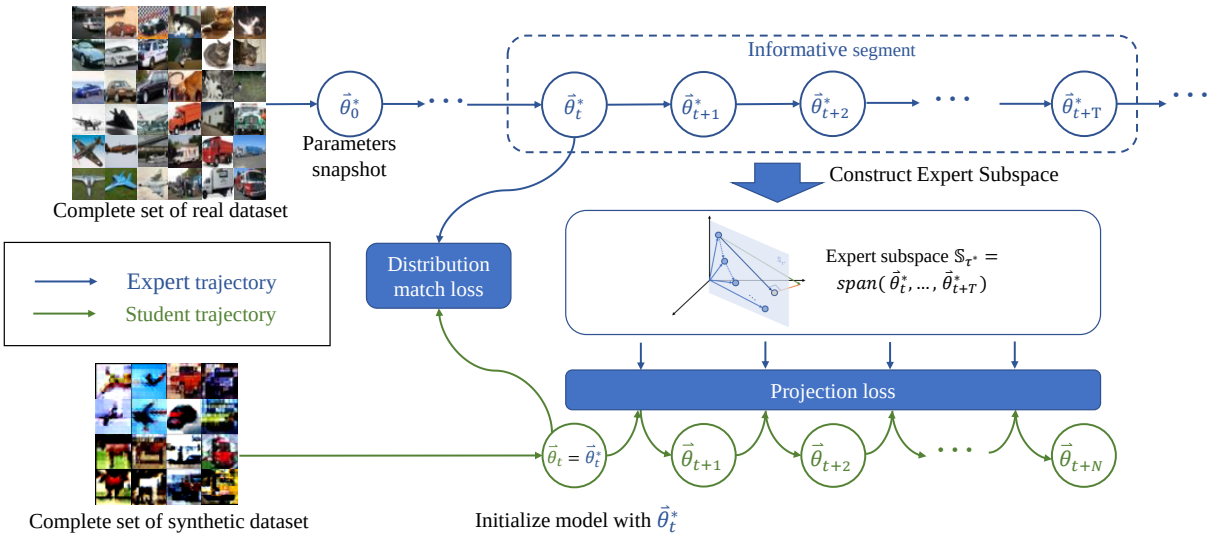

Figure 2. Overview of our ESP method for dataset condensation. We train the model with the complete real dataset to obtain the expert trajectory. An informative segment of length T is extracted and flattened into one-dimensional vectors, forming a subspace. The condensation process utilizes a bi-level optimization framework. The inner stage refines the model's parameters within the expert subspace using projected gradient descent, while the outer stage updates the synthetic dataset. Distribution matching loss penalizes feature distribution discrepancy, and projection loss penalizes gradients outside the expert subspace.

In subsequent sections, we delve into the technical details of our method, providing a comprehensive explanation of its underlying mechanisms. Furthermore, we conduct a thorough analysis to demonstrate the memory consumption advantage offered by our approach.

3.1. Preliminaries

The long-range training trajectory of the original dataset, referred to as the expert trajectory, has been empirically demonstrated to be effective in guiding the condensation of the synthetic dataset in trajectory matching [10]. As illustrated in Figure 2, an expert trajectory $\tau^* = \{\vec{\theta}_t^*\}_t^{t+T}$ consists of a series of parameter snapshots during the training process with D_{real}. Each snapshot, denoted as $\vec{\theta}_t^*$, represents the parameters saved after the t-th epoch and is flattened into a one-dimensional vector. In the context of trajectory matching, the inner optimization process involves T steps of updates on the neural network's parameters, i.e., from $\vec{\theta}_t$ to $\vec{\theta}_{t+T}$, supervised by the classification loss of the synthetic images D_{syn}. To ensure that the student trajectory begins from the same starting point as the expert trajectory, $\vec{\theta}_t$ is initialized with $\vec{\theta}_t^*$. The outer optimization process involves a single update step for the synthetic images, supervised by the trajectory matching loss:

$$\mathcal{L}_{TM} = \frac{\|\vec{\theta}_{t+N} - \vec{\theta}_{t+T}^*\|_2^2}{\|\vec{\theta}_t^* - \vec{\theta}_{t+T}^*\|_2^2}, \tag{1}$$

where N is the length of the student trajectory, and T is the length of the expert trajectory. It is worth noting that the student trajectory's length is not necessarily the same as that of

the expert trajectory due to the much smaller size of the synthetic dataset compared to the original dataset. In practice, N is intentionally set to be smaller than T.

In most cases, the student trajectory is indeed much shorter. However, for the sake of simplicity and without loss of generality, we assume that both trajectories have the same length in the equation provided above. $\mathcal{L}_{TM}$ promotes D_{syn} to mimic the long-range training dynamic of D_{real}. However, due to the inclusion of T steps of updates in the inner optimization process, the calculation of gradients for the outer optimization requires unrolling the gradients through T iterations. This unrolling process and the need to save the computation graph from multiple updates result in significantly higher memory consumption compared to other methods [5–7]. This high memory usage necessitates the use of a slice of the synthetic dataset, which may introduce a biased optimization. In Section 3.5, we discuss the impact of this issue in detail and analyze the memory usage.

3.2. Expert Subspace Projection

Our proposed ESP method leverages the benefits of long-range training dynamics obtained from expert trajectories. Importantly, it addresses the issue of linear memory growth that arises when unrolling the gradient through iterations, as the inner optimization solely entails a single-step update. The crux of the challenge lies in emulating the long-range training dynamics despite the constraint of having only one update.

Our solution is simple yet effective. In our approach, we confine each optimization step to remain within the subspace spanned by the expert trajectory during the inner optimization. Simultaneously, during the outer optimization, we penalize the residual gradient that deviates from this subspace. The expert subspace $\mathbb{S}_{\tau^*}$ is spanned by the vector set $\{\vec{\theta}_t^*\}_t^{t+T}$:

$$\mathbb{S}_{\tau^*} = \boldsymbol{span}\left(\vec{\theta}_t^*, \ldots, \vec{\theta}_{t+T}^*\right). \tag{2}$$

By using Schmidt's orthogonal standardization, we can obtain a set of standardized bases for the subspace $\mathbb{S}_{\tau^*}$:

$$\begin{aligned} \boldsymbol{E} &= [\vec{\epsilon_0}, \vec{\epsilon_1}, \ldots, \vec{\epsilon_T}] \\ &= OS([\vec{\theta}_t^*, \ldots, \vec{\theta}_{t+T}^*]), \end{aligned} \tag{3}$$

where the notation $OS(\cdot)$ represents the orthogonal standardization operator. It is worth emphasizing that the length of the expert trajectory is considerably smaller than the dimension of the parameter vector. Our experimental findings provide evidence that this expert subspace encapsulates the majority of long-range training dynamics exhibited by the expert trajectories. Therefore, it can serve as a reliable proxy for capturing these dynamics. Consistent with trajectory matching [10], the expert trajectory can be generated and stored offline prior to commencing the data condensation process. This approach helps to conserve memory and reduce computation costs.

3.3. Inner Optimization

At the onset of the dataset condensation process, we initialize the synthetic images with random Gaussian distribution, along with pre-assigned category labels. The neural network is initialized with the starting point of the expert trajectory, i.e., $\vec{\theta}_t \coloneqq \vec{\theta}_t^*$.

Following the bi-level optimization framework, we commence by updating the neural network in the inner optimization phase, followed by updating the synthetic images in the outer optimization phase. The gradient for the neural network in the inner optimization is computed as follows:

$$\vec{G} = \nabla_{\vec{\theta}_t} \mathcal{L}_{BCE}(\mathcal{M}(D_{syn}; \vec{\theta}_t)), \tag{4}$$

where $\mathcal{M}$ represents the neural network model, and $\mathcal{L}_{BCE}$ represents binary cross-entropy loss. Rather than directly utilizing this gradient to update the neural network, we project it

onto the expert subspace to align with the long-range training dynamics of the original data:

$$\vec{G}_{\mathbb{S}_{\tau^*}} = \mathrm{Proj}_{\mathbb{S}_{\tau^*}}(\vec{G}) = \boldsymbol{E}\boldsymbol{E}^T\vec{G}, \tag{5}$$

where $\boldsymbol{E}^T$ represent the transposition of $\boldsymbol{E}$, and the neural network is updated by the projected gradient in expert subspace:

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \alpha\vec{G}_{\mathbb{S}_{\tau^*}}, \tag{6}$$

where α represents the learning rate. In the equation, we have introduced the naive gradient descent method. However, it is essential to emphasize that there are other gradient-based optimization algorithms that can also be employed in this context.

3.4. Outer Optimization

In the outer optimization, we perceive the gradient deviation outside the expert subspace as the disparity between the synthetic dataset and the original dataset. To alleviate this dissimilarity, we introduce a penalization function that aims to minimize the norm of the residual gradient:

$$\mathcal{L}_{Proj} = \|\vec{G} - \vec{G}_{\mathbb{S}_{\tau^*}}\|_1, \tag{7}$$

$$\mathcal{L}_{DM} = \frac{1}{C}\sum_{c=1}^{C}\left\|\mathcal{M}(D_{real};\vec{\theta}_t) - \mathcal{M}(D_{syn};\vec{\theta}_t)\right\|_2^2, \tag{8}$$

where $\|\cdot\|$ indicates the ℓ_2 norm. In subsequent experimental analyses, we observed that the integration of this loss function resulted in improved accuracy in downstream training and enhanced the level of detail in synthesized images. These findings provide compelling evidence for the complementary nature of these two loss functions. The final objective function is defined as follows, where β is the hyper-parameter to balance these two losses:

$$\mathcal{L}_{Out} = \mathcal{L}_{DM} + \beta\mathcal{L}_{Proj}. \tag{9}$$

Finally, we can update the synthetic dataset with gradient descent:

$$D_{syn} = D_{syn} - \alpha\nabla_{D_{syn}}\mathcal{L}_{Out}. \tag{10}$$

Our comprehensive data condensation process is summarized in Algorithm 1.

3.5. Memory Consumption

Trajectory matching [10] involves N update steps in the inner optimization and a single update step in the outer optimization. Therefore, the calculation of the gradient for the outer optimization needs to unroll the gradient through all N steps. As a result, all intermediate computational graphs and variables must be stored in GPU memory. Accurately estimating GPU memory consumption, also known as the memory footprint, is a complex task that depends on various factors, such as the specific operators utilized in the neural network, the connectivity of the computational graph, and the choice of deep learning framework. However, in theory, the memory footprint scales approximately linearly with the number of inner optimization steps N. Our method involves a single step of inner optimization followed by a single step of outer optimization. Therefore, our memory consumption is significantly smaller compared to trajectory matching, while still allowing us to benefit from the valuable long-range expert trajectory.

Algorithm 1: Expert Subspace Projection

Input: the set of expert trajectories $\{\tau_k^*\}_{k=1}^K$; the length of the selected expert trajectory T; the length of the student trajectory N; the learning rate α;
Output: synthetic dataset D_{syn};
Initialize D_{syn} with Gaussian distribution;
while *not converged* **do**
  Randomly select a trajectory from the set $\{\tau_k^*\}_{k=1}^K$;
  Randomly extract a segment of length T: $\tau^* = \{\vec{\theta}_t^*\}_t^{t+T}$;
  Construct the expert subspace $\mathbb{S}_{\tau^*}$;
  Initialize $\vec{\theta}_t$ with $\vec{\theta}_t^*$;
  for $n = 1$ **to** N **do**
    Calculate $\vec{G}$ with Equation (4);
    Project $\vec{G}$ onto $\mathbb{S}_{\tau^*}$ to obtain $\vec{G}_{\mathbb{S}_{\tau^*}}$ with Equation (5);
    Calculate $\mathcal{L}_{Out}$ with Equation (9);
    `// Inner Optimization`
    $\vec{\theta}_{t+n} = \vec{\theta}_{t+n-1} - \alpha \vec{G}_{\mathbb{S}_{\tau^*}}$;
    `// Outer Optimization`
    $D_{syn} = D_{syn} - \alpha \nabla_{D_{syn}} \mathcal{L}_{Out}$;
  end
end
return D_{syn}

4. Experiments

We validate the effectiveness of our proposed method on various classification benchmark datasets, assessing its ability to generalize across different architectures. In addition, we conduct ablation experiments and provide visualization results to further substantiate the efficacy of our approach.

4.1. Datasets

CIFAR10/100 [24]. The CIFAR10 dataset is a collection of 60,000 color images, each measuring 32×32 pixels. These images are classified into 10 distinct categories representing common objects and animals such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each category contains 6000 images, and the dataset is split into a training set with 50,000 images and a test set with 10,000 images. CIFAR10 is widely recognized as a benchmark for image classification tasks, playing a crucial role in the development and evaluation of various machine learning and deep learning algorithms.

CIFAR100 consists of 100 classes, with each class containing 600 images. These 100 classes are grouped into 20 superclasses, with each superclass containing 5 classes. The CIFAR100 dataset covers a diverse range of object categories, including animals, vehicles, household items, and natural objects. Similar to CIFAR10, CIFAR100 is divided into a training set (50,000 images) and a test set (10,000 images). It serves as a benchmark for addressing more intricate and fine-grained classification tasks, enabling researchers to delve into complex image recognition problems.

TinyImageNet [25]. TinyImageNet is extensively utilized in computer vision research and benchmarking as a downsized variant of the renowned ImageNet dataset [26]. Its purpose is to offer a more manageable alternative for training and evaluating deep learning models, given the vast scale of the original ImageNet dataset. Comprising 200 distinctive object classes, the TinyImageNet dataset encompasses an approximate quantity of 500 training images, 50 validation images, and 50 test images per class. It boasts a diverse array of object categories, encompassing animals, commonplace objects, as well as a variety of natural and human-created items. Notably, each image within the dataset maintains a resolution of 64×64 pixels.

SVHN [27], The SVHN (Street View House Number) Dataset is a valuable collection of real-world images extracted from door numbers captured by Google Street View. It provides a significant amount of data, with more than 70,000 numbers specifically designated for training and an additional 20,000 numbers reserved for testing. Similar to the well-known MNIST [28] dataset, each image in SVHN is 32 $\times$ 32 pixels in size and focuses on a single character situated at the center. Notably, many images in SVHN also contain distractors placed alongside the main character of interest. This dataset is highly useful for tasks like digit recognition and character segmentation, allowing researchers to tackle challenges posed by real-world scenarios and evaluate the performance of various machine learning algorithms.

4.2. Implementation Details

For fairness and convenience of comparison, we use the same suite of differentiable augmentations as in previous work, as well as the same number of latent code decoder parameters as in [1], which matches the size of 1, 10, and 50 images per class. Prior to the condensation step, we pre-compute 1000 expert trajectories on ConvNet-3 for each dataset. These pre-computed trajectories are then utilized in all our experiments to expedite the condensation process. This approach follows a similar technique used in trajectory matching [10]. During the distillation process, we have the flexibility to randomly select one expert trajectory as the teacher, which saves time in optimizing the expert network.

For the distillation process, we use $\beta = 1 \times 10^{-5}$ to balance the loss of $\mathcal{L}_{DM}$ and $\mathcal{L}_{Proj}$. The learning rate of the neural network (ConvNet-3 by default) is set as a constant value of 0.1. The synthetic dataset, represented by the latent code and decoder, follows a linear decay schedule for their learning rates. The initial learning rate for the latent code is set to 0.1, while the initial learning rate for the decoder is set to 0.01. To optimize the model, we employ the SGD as the inner optimizer and Adam [29] as the outer optimizer. More details can be found in Algorithm 1.

For the evaluation, we train the condensed data on ConvNet-3 for accuracy, while using ResNet [30] and DenseNet [31] for cross-architecture ability. To evaluate the performance of classification models trained with the condensed datasets, we report the mean classification accuracy and its corresponding standard deviation across 5 runs with different random seeds.

4.3. Comparison with State-of-the-Art Methods

We first compare our ESP method with coreset selection [19–23] methods, such as randomizing [32,33] and herding [34,35] methods. Secondly, we also compare our method with some recent condensation works, namely DC (dataset condensation) [6], DSA (differentiable Siamese augmentation for DC) [7], DM (distribution matching) [5], KIP to NN (infinitely wide convolutional networks) [36], CAFE (aligning features) [37], MTT (matching training trajectories) [10], IDC [38], and KFS (latent space knowledge factorization and sharing) [1]. Our ESP method achieves the state of the art on SVHN [27], CIFAR10 [24], CIFAR100 [24], and TinyImageNet [25] datasets. Table 1 illustrates the performance of our method and competitor methods on four datasets. Particularly when IPC = 1, our ESP method achieves a significant improvement under the same parameters on all four datasets. In particular, the improvements on SVHN, CIFAR10, CIFAR100, and TinyImageNet are 2.3%, 5.4%, 2.3%, and 9.7%, respectively, over other methods, indicating that the subspace projection method improves the data efficiency of dataset condensation. In order to illustrate the effect of our condensed dataset, we visualized our synthetic image of SVHN in Figure 3, CIFAR10 in Figure 4, CIFAR100 in Figure 5, and TinyImageNet in Figure 6.

Figure 3. The synthetic images of SVHN [27].

Figure 4. The synthetic images of CIFAR10 [24].

Figure 5. The synthetic images of CIFAR100 [24].

Figure 6. The synthetic images of TinyImageNet [25].

Table 1. Classification accuracies (%) on ConvNet-3. The results of other methods are reported in their respective papers [1,5–7,10,36–38]. For our method, we report mean accuracy and standard deviation over five runs with different random seeds.

Dataset	**SVHN**			**CIFAR10**			**CIFAR100**		**TinyImageNet**	
Metric	**Accuracy (%)**			**Accuracy (%)**			**Accuracy (%)**		**Accuracy (%)**	
Images/Class	**1**	**10**	**50**	**1**	**10**	**50**	**1**	**10**	**1**	**10**
Param./Class	**3072**	**30,720**	**153,600**	**3072**	**30,720**	**153,600**	**3072**	**30,720**	**12,288**	**122,880**
Random	14.6±1.6	35.1±4.1	70.9±0.9	14.4±2.0	26.0±1.2	43.4±1.0	4.2±0.3	14.6±0.5	1.4±0.1	5.0±0.2
Herding	20.9±1.3	50.5±3.3	72.6±0.8	21.5±1.2	31.6±0.7	40.4±0.6	8.4±0.3	17.3±0.3	2.8±0.2	6.3±0.2
DC [6]	31.2±1.4	76.1±0.6	82.3±0.3	28.3±0.5	44.9±0.5	53.9±0.5	12.8±0.3	25.2±0.3	-	-
DSA [7]	27.5±1.4	79.2±0.5	84.4±0.4	28.8±0.7	52.1±0.5	60.6±0.5	13.9±0.3	32.3±0.3	-	-
DM [5]	20.3±2.1	73.5±1.0	84.2±0.0	26.0±0.8	48.9±0.6	63.0±0.4	11.4±0.3	29.7±0.3	3.9±0.2	12.9±0.4
KIP to NN [36]	57.3±0.1	75.0±0.1	80.5±0.1	49.9±0.2	62.7±0.3	68.6±0.2	15.7±0.2	28.3±0.1	-	-
CAFE + DSA [37]	42.9±3.0	77.9±0.6	82.3±0.4	31.6±0.8	50.9±0.5	62.3±0.4	14.0±0.3	31.5±0.2	-	-
Traj. Matching [10]	-	-	-	46.3±0.8	65.3±0.7	71.6±0.2	24.3±0.3	40.1±0.4	8.8±0.3	23.2±0.2
IDC [38]	68.1±0.1	87.3±0.2	90.2±0.1	50.0±0.4	67.5±0.5	74.5±0.1	-	44.8±0.2	-	-
KFS [1]	82.9±0.4	91.4±0.2	92.2±0.1	59.8±0.5	72.0±0.3	75.0±0.2	40.0±0.5	**50.6±0.2**	22.7±0.2	**27.8±0.2**
ESP (ours)	**84.8±0.3**	**91.6±0.1**	**92.8±0.1**	**63.0±0.4**	**73.8±0.2**	**76.1±0.3**	**41.1±0.3**	48.0±0.1	**24.9±0.3**	26.6±0.5
Full dataset		95.4±0.1			84.8±0.1			56.2±0.3		37.6±0.4

4.4. Cross-Architecture Generalization

In the context of dataset condensation, it is imperative that an ideal synthetic dataset exhibit similar training effects as the original dataset on downstream models with arbitrary structures. Consequently, the performance of cross-architecture generalization holds significant importance as a key metric. We used our synthetic data generated on ConvNet-3 (0.32 M parameters) to train different models, including ResNet-10 [30] (4.90 M parameters), DenseNet-121 [31] (6.96 M parameters), and EfficientNet-V2-s [39] (22 M parameters), and the results are presented in Tables 2 and 3.

Table 2. Cross-architecture experiments. Conv3, RN10, and DN121 denote ConvNet-3, ResNet-10, and DenseNet-121, respectively. We train on ConvNet-3 and evaluate on the three architectures. The results of other methods are reported in their respective papers [1,5,7,38]. For our method, we report mean accuracy and standard deviation over five runs with different random seeds.

Dataset	**Images/Class**	**1**			**10**			**50**		
	Metric	**Accuracy (%)**			**Accuracy (%)**			**Accuracy (%)**		
	Test Architecture	**Conv3**	**RN10**	**DN121**	**Conv3**	**RN10**	**DN121**	**Conv3**	**RN10**	**DN121**
SVHN	DSA [7]	27.5±1.4	13.2±1.1	13.3±1.4	79.2±0.5	19.5±1.5	23.1±1.9	84.4±0.4	41.6±2.1	58.0±3.1
	DM [5]	20.3±2.1	10.5±2.8	13.6±1.0	73.5±1.0	28.2±1.5	24.8±2.5	84.2±0.0	54.7±1.3	58.4±2.7
	IDC [38]	68.1±0.1	39.6±1.5	39.9±2.9	87.3±0.2	83.3±0.2	82.8±0.2	90.2±0.1	89.1±0.2	91.0±0.3
	KFS [1]	82.9±0.4	75.7±0.8	81.0±0.7	91.4±0.2	90.3±0.2	89.7±0.2	92.2±0.1	90.9±0.2	90.2±0.2
	ESP (ours)	**84.8±0.3**	**84.7±0.6**	**82.0±0.1**	**91.6±0.2**	**93.5±0.1**	**90.7±0.3**	**92.8±0.1**	**93.7±0.1**	**91.0±0.5**
	Full dataset	95.4±0.1	93.8±0.5	89.1±0.8	95.4±0.1	93.8±0.5	89.1±0.8	95.4±0.1	93.8±0.5	89.1±0.8
CIFAR10	DSA [7]	28.8±0.7	25.1±0.8	25.9±1.8	52.1±0.5	31.4±0.9	32.9±1.0	60.6±0.5	49.0±0.7	53.4±0.8
	DM [5]	26.0±0.8	13.7±1.6	12.9±1.8	48.9±0.6	31.7±1.1	32.2±0.8	63.0±0.4	49.1±0.7	53.7±0.7
	IDC [38]	50.0±0.4	41.9±0.6	39.8±1.2	67.5±0.5	63.5±0.1	61.6±0.6	74.5±0.1	72.4±0.5	71.8±0.6
	KFS [1]	59.8±0.5	47.0±0.8	49.5±1.3	72.0±0.3	70.3±0.3	69.2±0.4	75.0±0.2	75.1±0.3	**76.3±0.4**
	ESP (ours)	**63.0±0.4**	**50.4±0.5**	**51.6±0.6**	**73.0±0.3**	**71.8±0.4**	**69.4±0.3**	**75.9±0.3**	**75.3±0.2**	73.0±0.5
	Full dataset	84.8±0.1	87.9±0.2	90.5±0.3	84.8±0.1	87.9±0.2	90.5±0.3	84.8±0.1	87.9±0.2	90.5±0.3

Table 3. Cross-architecture experiments on EfficientNet. ENs denotes EfficientNetv2-s [39]. We train on ConvNet-3 and evaluate on EfficientNetv2-s. The reported results are solely based on our own experiments.

Dataset	Images/Class Metric Test Architecture	1 Accuracy (%) ENs	10 Accuracy (%) ENs	50 Accuracy (%) ENs
CIFAR10	DSA [7]	16.5$_{\pm 0.3}$	25.6$_{\pm 0.4}$	33.7$_{\pm 0.2}$
	DC [6]	16.1$_{\pm 1.7}$	22.3$_{\pm 1.4}$	25.7$_{\pm 0.9}$
	Traj. Matching [10]	17.7$_{\pm 0.2}$	24.0$_{\pm 0.4}$	33.9$_{\pm 0.6}$
	ESP (ours)	**35.3**$_{\pm 0.4}$	**55.0**$_{\pm 1.2}$	**63.9**$_{\pm 0.8}$
	Full dataset	98.7$_{\pm 0.2}$	98.7$_{\pm 0.2}$	98.7$_{\pm 0.2}$

We can see that the performance of ESP is more robust to the change in network architectures and achieves state-of-the-art performance on most of the network architectures. For the SVHN [27] dataset, the accuracy on ResNet-10 is generally higher than our baseline model (ConvNet-3), up to 2.1%. These experiments provide evidence that the synthetic dataset generated by ESP exhibits better generalizability compared to other datasets, showcasing the superior ability of our ESP method to capture representative information from the original datasets. However, the experimental results also reveal that when there is a significant architectural difference between the training and testing phases, the cross-architecture performance is weakened. This suggests that a cross-architecture generalization problem persists across all dataset condensation methods.

4.5. Memory Analysis

We perform an experimental comparison of the memory consumption between our ESP method and trajectory matching [10] on the CIFAR10 [24] dataset. In line with our analysis in Section 3.5, trajectory matching exhibits a linear increase in memory consumption with the length of the student trajectory due to its inner optimization steps being equal to the student trajectory length. This theoretical analysis is confirmed by Figure 7a. Notably, the memory consumption of trajectory matching increases approximately linearly with the length of the student trajectory, while our ESP method remains unaffected.

We conducted an additional experiment to demonstrate that our ESP method has a significantly lower growth rate in memory consumption compared to trajectory matching when increasing the size of the synthetic dataset. Figure 7b illustrates this observation, where our ESP method only exhibits a slight increase in memory usage as the IPC increases. In contrast, the memory consumption of the trajectory matching [10] method experiences a substantial surge, rendering it ineffective for training on the complete synthetic dataset with higher IPC. Conversely, our method can directly handle the complete synthetic dataset, enabling unbiased training, as we elaborate on in Section 4.6.

4.6. Synthetic Batch Size Analysis

Based on the analysis presented in [1], it is noteworthy that training the synthetic dataset using batch optimization, where the complete synthetic dataset is divided into several batches, introduces a bias in the gradient. This bias stems from disregarding the interactions between different synthetic images, resulting in a reduction in diversity among images within classes. The ablation experiments shown in Figure 8 quantify the impact of this issue. In the figure, we trained synthetic images on the CIFAR10 dataset with various batch sizes, ranging from small (batch size = 64) to large (batch size = 1024). It is evident that the accuracy increases with batch size and stabilizes with larger batch sizes, reflecting the property that the bias of the gradient of the synthetic dataset decreases as the batch size increases. Thanks to the efficient use of memory in our ESP method, we are able to train on the complete set of the synthetic dataset, which allows us to achieve better performance than trajectory matching.

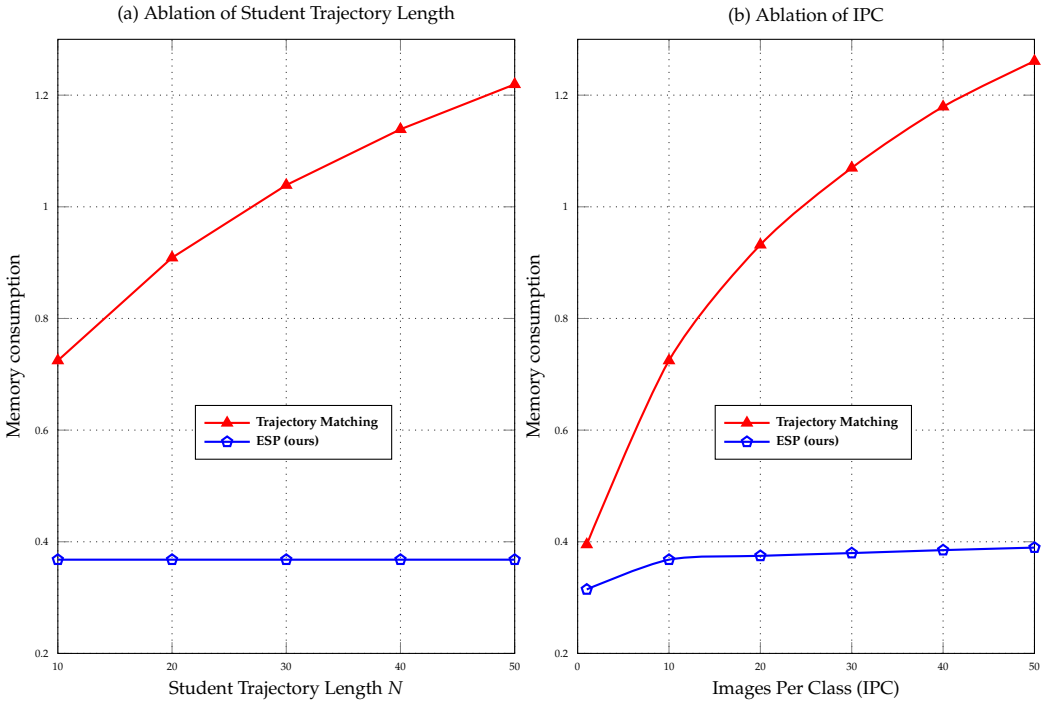

Figure 7. Memory consumption of trajectory matching [10] and our method. The right part shows the memory consumption of different synthetic steps for IPC = 10, and the left part shows the memory consumption of the two methods with different IPC for the CIFAR10 dataset. The results of trajectory matching are obtained by executing their officially released code [10].

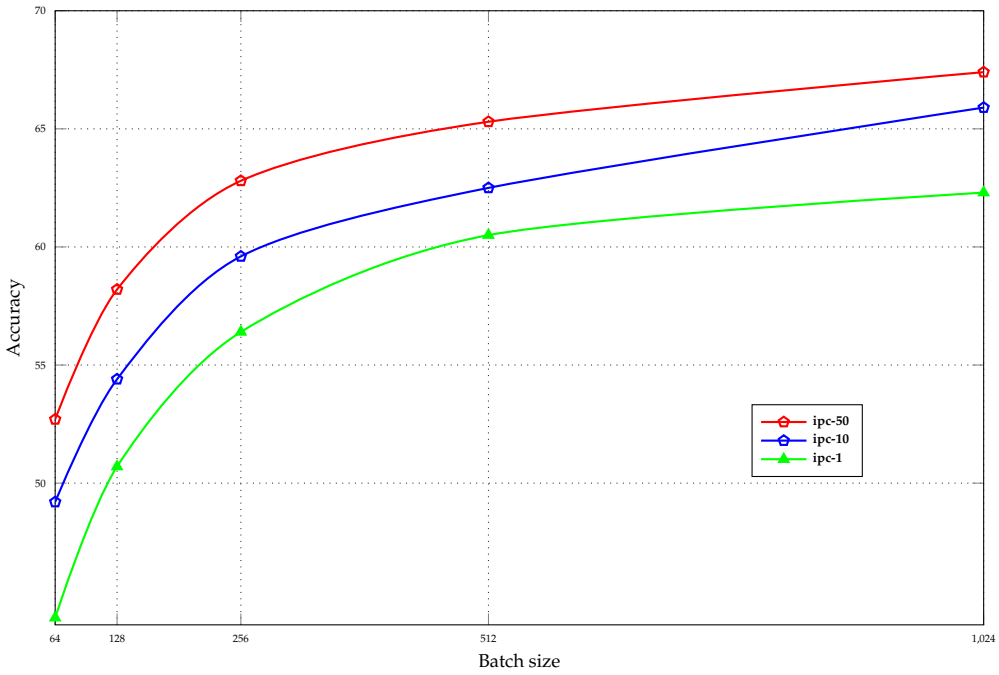

Figure 8. Accuracy on ConvNet-3 of synthetic images trained with different batch sizes.

4.7. Ablation Study

As discussed in Section 3.4, the projection loss and the distribution matching loss exhibit a high level of complementarity. The projection loss focuses on aligning the long-range training dynamics between the synthetic and original datasets, while the distribution matching loss aims to match the static feature distribution between the two datasets. These two loss components work together to ensure a comprehensive alignment of both the

dynamic and static aspects of the datasets. This observation is further supported by the results of our ablation experiments, as presented in Table 4. The individual losses alone show suboptimal performance, whereas their combination yields excellent results. We further visualize the synthetic images with and without the distribution loss. As can be observed in Figure 9, the introduction of the distribution loss results in synthetic images that exhibit more detailed textures and recognizable visual concepts. This phenomenon may be attributed to the fact that the projection loss primarily constrains higher-order information, such as gradients. On the other hand, low-level information like texture is predominantly constrained by the distribution matching loss.

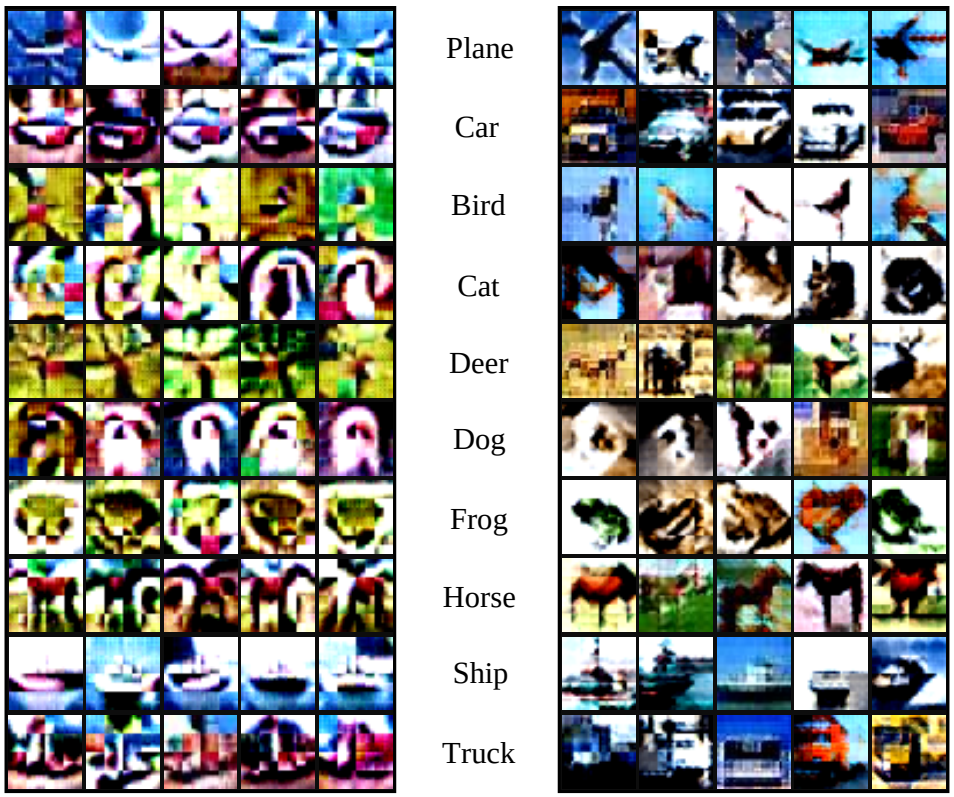

Figure 9. The condensed synthetic images produced by our method exhibit a noticeable difference when comparing the left and right parts. The left portion corresponds to images trained without the distribution loss, resulting in a more abstract style. On the other hand, the right portion showcases images with more pronounced and detailed texture information. Experiments are conducted on CIFAR10 [24].

Table 4. Classification accuracies (%) on CIFAR10 [24]. We maintain the other hyper-parameters and only change the loss.

Loss	$\mathcal{L}_{Proj}$	$\mathcal{L}_{DM}$	Images/Class 1	10	50
Proj	✓		41.3$_{\pm 0.2}$	40.6$_{\pm 0.4}$	39.5$_{\pm 0.3}$
DM		✓	48.0$_{\pm 0.3}$	71.3$_{\pm 0.3}$	74.0$_{\pm 0.1}$
Proj + DM	✓	✓	**62.6**$_{\pm 0.1}$	**73.0**$_{\pm 0.3}$	**75.9**$_{\pm 0.2}$

5. Conclusions

In this paper, we have proposed a novel dataset condensation method called Expert Subspace Projection (ESP) that effectively utilizes long-range training dynamics while reducing computational overhead compared to prior trajectory matching techniques. Our

key insight is to constrain model optimization to remain within the subspace spanned by expert trajectories from the original dataset. This avoids expensive unrolling of gradients across multiple steps, enabling memory-efficient training of the complete set of synthetic data. We have validated ESP extensively on image classification tasks, demonstrating state-of-the-art results on CIFAR, SVHN, and TinyImageNet datasets compared to existing condensation methods. Importantly, we have shown ESP's superior ability to transfer condensed datasets to unseen architectures, indicating it effectively distills dataset knowledge in an architecture-agnostic manner. Overall, ESP provides an effective and scalable solution for dataset condensation, resulting in the synthesis of highly informative compact datasets. This technique enables the application of modern deep learning approaches in resource-constrained settings, where memory or computational resources are limited. Moreover, ESP contributes to minimizing the energy consumption needed for training models.

6. Limitations and Future Work

Despite the effectiveness of our Expert Subspace Projection (ESP) approach in reducing memory usage and computational requirements compared to the previous trajectory matching [10] approach, it is crucial to acknowledge that ESP still operates within a bi-level optimization framework. Consequently, extending ESP to large datasets that contain high-resolution images presents a significant challenge, similar to previous bi-level optimization methods [1,6,7,10,36,37]. This limitation hampers the application of dataset condensation to tasks such as fine-grained classification, which heavily depends on high-resolution images for capturing intricate details. Therefore, it is imperative to focus further efforts on exploring strategies to minimize memory usage and computational requirements. Promising directions include disentangling the outer optimization from the inner optimization and approximating the inner optimization using a convex proxy model.

Author Contributions: Conceptualization, Z.M.; methodology, Z.M.; software, Z.M., D.G. and S.Y.; validation, Z.M. and X.W.; formal analysis, Z.M., D.G. and X.W.; investigation, Z.M.; resources, Z.M. and Y.G.; data curation, Z.M. and X.W.; writing—original draft preparation, Z.M., D.G. and S.Y.; writing—review and editing, Z.M., X.W. and Y.G.; visualization, D.G. and S.Y.; supervision, X.W. and Y.G.; project administration, Y.G.; funding acquisition, Z.M., X.W. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by Shenzhen Key Technical Projects (202208313000248, 202205173000112), the Key Program of the Natural Science Foundation of Shenzhen (JCYJ20220818101406014), the National Natural Science Foundation of China (62006183, 62206271), the National Key Research and Development Project of China (2020AAA0105600), and the China Postdoctoral Science Foundation (2020M683489).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are all openly available. CIFAR10/100 is found at https://www.cs.toronto.edu/kriz/cifar.html, accessed on 11 August 2023; SVHN is found at http://ufldl.stanford.edu/housenumbers/, accessed on 11 August 2023; and TinyImageNet is found at https://www.kaggle.com/c/tiny-imagenet or https://www.image-net.org/download, accessed on 11 August 2023.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, H.B.; Lee, D.B.; Hwang, S.J. Dataset Condensation with Latent Space Knowledge Factorization and Sharing. *arXiv* **2022**, arXiv:2208.10494.
2. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)* **2020**, *33*, 1877–1901.
3. Cui, J.; Wang, R.; Si, S.; Hsieh, C.J. DC-BENCH: Dataset condensation benchmark. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 810–822.
4. Wang, T.; Zhu, J.Y.; Torralba, A.; Efros, A.A. Dataset distillation. *arXiv* **2018**, arXiv:1811.10959.
5. Zhao, B.; Bilen, H. Dataset Condensation with Distribution Matching. *arXiv* **2021**, arXiv:2110.04181.

6. Zhao, B.; Mopuri, K.R.; Bilen, H. Dataset Condensation with Gradient Matching. *Int. Conf. Learn. Represent. (ICLR)* **2021**, *1*, 3.
7. Zhao, B.; Bilen, H. Dataset condensation with differentiable siamese augmentation. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, Virtual Event, 18–24 July 2021; pp. 12674–12685.
8. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, Sydney, Australia, 6–11 August 2017; pp. 1126–1135.
9. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv* **2018**, arXiv:1803.02999.
10. Cazenavette, G.; Wang, T.; Torralba, A.; Efros, A.A.; Zhu, J.Y. Dataset distillation by matching training trajectories. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 4750–4759.
11. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
12. Turc, I.; Chang, M.W.; Lee, K.; Toutanova, K. Well-read students learn better: On the importance of pre-training compact models. *arXiv* **2019**, arXiv:1908.08962.
13. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
14. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. Tinybert: Distilling bert for natural language understanding. *arXiv* **2019**, arXiv:1909.10351.
15. Gur-Ari, G.; Roberts, D.A.; Dyer, E. Gradient descent happens in a tiny subspace. *arXiv* **2018**, arXiv:1812.04754.
16. Li, C.; Farkhoor, H.; Liu, R.; Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *arXiv* **2018**, arXiv:1804.08838.
17. Gressmann, F.; Eaton-Rosen, Z.; Luschi, C. Improving neural network training in low dimensional random bases. *Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)* **2020**, *33*, 12140–12150.
18. Li, T.; Tan, L.; Tao, Q.; Liu, Y.; Huang, X. Low dimensional landscape hypothesis is true: DNNs can be trained in tiny subspaces. *arXiv* **2021**, arXiv:2103.11154.
19. Bachem, O.; Lucic, M.; Krause, A. Practical coreset constructions for machine learning. *arXiv* **2017**, arXiv:1703.06476.
20. Borsos, Z.; Mutny, M.; Krause, A. Coresets via bilevel optimization for continual learning and streaming. *Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)* **2020**, *33*, 14879–14890.
21. Har-Peled, S.; Kushal, A. Smaller coresets for k-median and k-means clustering. In Proceedings of the Twenty-First Annual Symposium on Computational Geometry, Pisa, Italy, 6–8 June 2005; pp. 126–134.
22. Sener, O.; Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv* **2017**, arXiv:1708.00489.
23. Tsang, I.W.; Kwok, J.T.; Cheung, P.M.; Cristianini, N. Core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res. JMRL* **2005**, *6*, 363–392.
24. Krizhevsky, A.; Vinod, N.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. Technical Report. University of Toronto. 2009. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 11 August 2023).
25. Le, Y.; Yang, X. Tiny imagenet visual recognition challenge. *CS 231N* **2015**, *7*, 3.
26. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. (IJCV)* **2015**, *115*, 211–252. [CrossRef]
27. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 16–17 December 2011.
28. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
29. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, 7–9 May 2015.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
32. Chen, Y.; Welling, M.; Smola, A. Super-samples from kernel herding. *arXiv* **2012**, arXiv:1203.3472.
33. Rebuffi, S.A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. icarl: Incremental classifier and representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2001–2010.
34. Belouadah, E.; Popescu, A. Scail: Classifier weights scaling for class incremental learning. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass Village, CO, USA, 1–5 March 2020; pp. 1266–1275.
35. Castro, F.M.; Marín-Jiménez, M.J.; Guil, N.; Schmid, C.; Alahari, K. End-to-end incremental learning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 233–248.
36. Nguyen, T.; Novak, R.; Xiao, L.; Lee, J. Dataset distillation with infinitely wide convolutional networks. *Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)* **2021**, *34*, 5186–5198.
37. Wang, K.; Zhao, B.; Peng, X.; Zhu, Z.; Yang, S.; Wang, S.; Huang, G.; Bilen, H.; Wang, X.; You, Y. Cafe: Learning to condense dataset by aligning features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 12196–12205.

38. Kim, J.H.; Kim, J.; Oh, S.J.; Yun, S.; Song, H.; Jeong, J.; Ha, J.W.; Song, H.O. Dataset Condensation via Efficient Synthetic-Data Parameterization. *arXiv* **2022**, arXiv:2205.14959.
39. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual Event, 18–24 July 2021; pp. 10096–10106.