electronics

MDPI

Article

Advantages and Pitfalls of Dataset Condensation: An Approach to Keyword Spotting with Time-Frequency Representations

Pedro Henrique Pereira, Wesley Beccaro * and Miguel Arjona Ramírez *

Departamento de Engenharia de Sistemas Eletrônicos, Escola Politécnica da Universidade de São Paulo, São Paulo 05508-010, Brazil; pedro3.pereira@usp.br

* Correspondence: wesley@lme.usp.br (W.B.); miguel@lps.usp.br (M.A.R.)

Abstract: With the exponential growth of data, the need for efficient techniques to extract relevant information from datasets becomes increasingly imperative. Reducing the training data can be useful for applications wherein storage space or computational resources are limited. In this work, we explore the concept of data condensation (DC) in the context of keyword spotting systems (KWS). Using deep learning architectures and time-frequency speech representations, we have obtained condensed speech signal representations using gradient matching with Efficient Synthetic-Data Parameterization. From a series of classification experiments, we analyze the models and condensed data performances in terms of accuracy and number of data per class. We also present results using cross-model techniques, wherein models are trained with condensed data obtained from a different architecture. Our findings demonstrate the potential of data condensation in the context of the speech domain for reducing the size of datasets while retaining their most important information and maintaining high accuracy for the model trained with the condensed dataset. We have obtained an accuracy of 80.75% with 30 condensed speech representations per class with ConvNet, representing an addition of 24.9% in absolute terms when compared to 30 random samples from the original training dataset. However, we demonstrate the limitations of this approach in the cross-model tests. We also highlight the challenges and opportunities for further improving the accuracy of condensed data obtained and trained with different neural network architectures.

Keywords: keyword spotting; deep learning; data condensation; spectral analysis; speech analysis; automatic speech recognition

check for updates

Citation: Pereira, P.H.; Beccaro, W.; Arjona Ramírez, M. Advantages and Pitfalls of Dataset Condensation: An Approach to Keyword Spotting with Time-Frequency Representations. *Electronics* **2024**, *13*, 2097. https://doi.org/10.3390/electronics13112097

Academic Editors: Begoña Garcia-Zapirain, Amaia Méndez-Zorrilla and Ibon Oleagordia

Received: 1 March 2024
Revised: 15 May 2024
Accepted: 27 May 2024
Published: 28 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, an unprecedented amount of data has been produced by the proliferation of digital gadgets, online platforms, and connected systems. This remarkable surge in data generation has been an essential factor in the advancement of deep learning, which has led to breakthroughs in several domains, such as natural language processing, computer vision, and speech recognition.

In the context of speech recognition, voice commands are becoming a natural way to interact with consumer electronic devices [1,2]. Systems with speech command recognition such as Amazon's Alexa, Apple's Siri, and Google's Assistant are examples of this popularity. These smart devices often use some embedded systems (e.g., microcontrollers [3], microprocessors, field-programmable gate arrays, or dedicated devices [4]) with limited resources, making the implementation of speech recognition algorithms dependent on hardware limitations [5,6].

Given the hardware limitations, there is a need to obtain low-complexity solutions [7–10]. A major obstacle in training a deep learning model to an edge device is its high memory consumption, which must be decreased to fit within the device. This reduction in the amount of training data can often affect the training of the model and, consequently, its performance metrics.

The handling of such a massive volume of data requires substantial work in terms of gathering, storing, and preprocessing, among other things. Additionally, to achieve satisfactory performance, training over large datasets frequently demands high computational resources. As a result, a number of applications that rely on training in large datasets require more time and computational consumption to deal with hyperparameter optimization and neural architecture search, for example.

One solution to the challenge of training with large datasets is to seek a smaller dataset that can adequately represent the distribution of the original data. A reasonably simple way to obtain such smaller datasets is to choose the most representative or useful samples from the original dataset. In general, models trained on these subsets can then perform as well as the originals. This strategy is referred to as coreset data reduction [11–14].

Another field of study that has received attention is called dataset distillation (DD) [15], also referred to as dataset condensation (DC) [16]. The general idea is to synthesize (i.e., distill) a small number of instances that, when employed as training data, will approximate the model performance to that obtained when training on the original dataset. In other words, the method condenses the rich features of the original dataset into a compact dataset. Synthesized datasets expedite the process of neural architecture search and find diverse applications, including their use in continual learning.

DC also aligns with the few-shot concept [17], offering the benefit of providing only a few examples, yet rich in information compared to the original dataset. Few-shot learning focuses on training models to generalize from a small number of examples, while dataset distillation aims to condense large datasets into smaller, more manageable subsets without losing important information. Hence, both approaches aim to address challenges related to limited data availability.

A key indicator of the performance of a DC method is the generalization of the synthetic dataset in different network architectures. However, the evaluation results of the synthetic data generated by the existing DC methods in heterogeneous models have not reached homogeneous performance [18], and there is a significant performance drop for some DC methods when the condensed data are used to train unseen architectures [19], which motivates further studies and experiments on the subject, especially as regards to audio data.

The novelty of this work can be summarized in three main contributions:

- First, for exploring the DC of time-frequency representations, a strategy for obtaining data, training and testing models by combining different speech representations is proposed, mapping the behavior of the number of data per class in the overall accuracy of each model.
- A methodology is introduced for evaluating the effectiveness of the DC technique in a cross-model setup. In this context, the training architecture used for condensing the data differs from the architecture employed for synthetic data generation. We evaluate the transfer of condensed datasets to unseen architectures, indicating, in some cases, whether the effective distillation of dataset knowledge occurs in an architecture-agnostic manner. This phase encompasses a discussion on the generalization capability as well as bottlenecks identified from the experiments conducted.

This article is structured as follows. Section 2 presents theoretical aspects of DC and describes strategies for DC using gradient matching with Efficient Synthetic-Data Parameterization [20]. Sections 3 and 4 detail and discuss the experimental setup and the results, respectively. Section 5 highlights the findings of this paper and summarizes its conclusion.

2. Strategies for Dataset Condensation

As aforementioned, the main goal of DC is to learn a small set of synthetic data from an original large-scale dataset so that models trained in the synthetic dataset can achieve comparable performances to those trained in the original one, as presented in Figure 1.

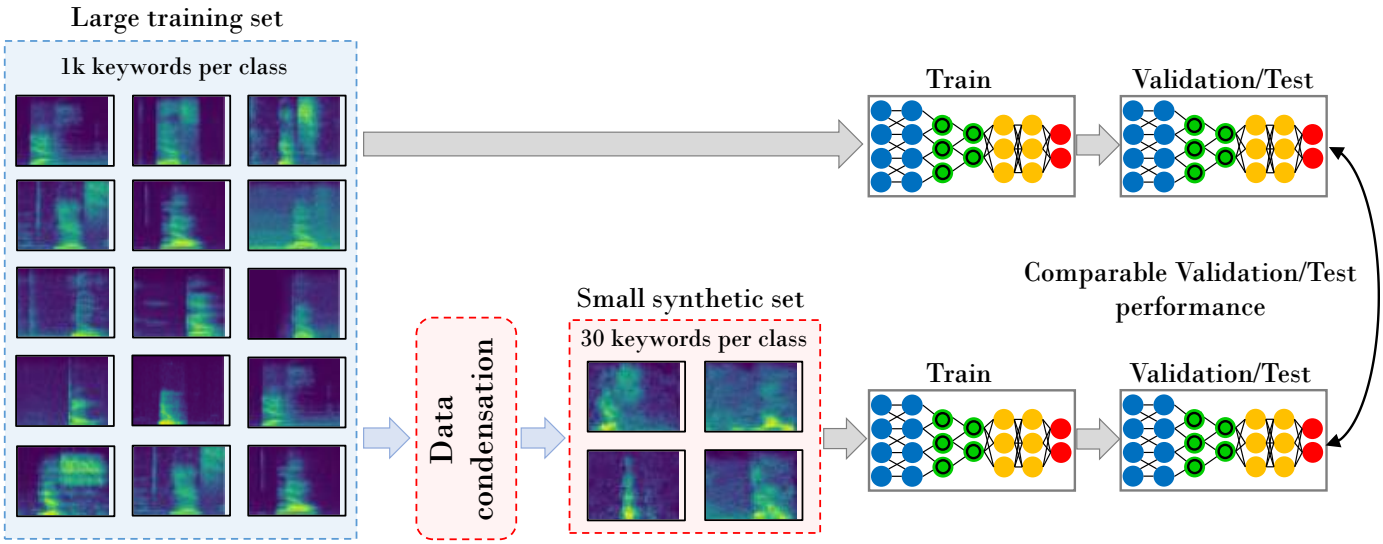

Figure 1. Example of DC applied to KWS. The goal of DC is to synthesize a limited support set of synthetic data for a model trained with these data to achieve a validation/test performance comparable to a model trained with the complete real dataset. In this example, the small synthetic set comprises only 30 keyword representations condensed from the large training dataset.

The structure of the dataset condensation problem introduced by Zhao et al. [16] is described as follows. Given a real large scale dataset consisting of $|\mathcal{T}|$ pairs of a training data and their respective class labels, $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}|_{i=1}^{|\mathcal{T}|}$, where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$, $y \in \mathcal{Y}$ so that $y \in \{0, \ldots, C-1\}$, $\mathcal{X}$ is a d-dimensional input space and C is the number of classes. The purpose is to learn a differentiable function ϕ (e.g., neural network) with parameters $\boldsymbol{\theta}$ that correctly predict labels of previously unseen data, $y = \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$. The parameters of this function can be learned by minimizing an empirical loss term throughout the training set as

$$\boldsymbol{\theta}^{\mathcal{T}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}), \tag{1}$$

where $\mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},y)\in\mathcal{T}} \ell(\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$, $\ell(\cdot,\cdot)$ is a defined loss function (e.g., cross-entropy for classification tasks), and $\boldsymbol{\theta}^{\mathcal{T}}$ are the parameters that minimize the expected loss throughout the observed dataset.

The primary objective is to generate a compact set of synthetic samples along with their corresponding labels, $\mathcal{S} = \{(\boldsymbol{s}_i, y_i)\}|_{i=1}^{|\mathcal{S}|}$ where $\boldsymbol{s} \in \mathbb{R}^d$ and $|\mathcal{S}| \ll |\mathcal{T}|$. Similar to Equation (1), once the condensed dataset is extracted, one can train ϕ on them as follows

$$\boldsymbol{\theta}^{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}), \tag{2}$$

where $\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{s},y)\in\mathcal{S}} \ell(\phi_{\boldsymbol{\theta}}(\boldsymbol{s}), y)$ and $\boldsymbol{\theta}^{\mathcal{S}}$ are the parameters that minimize the expected loss across the synthetic dataset. In this condition, the generalization performance (e.g., classification metrics) of $\phi_{\boldsymbol{\theta}}^{\mathcal{S}}$ is desired to be close to $\phi_{\boldsymbol{\theta}}^{\mathcal{T}}$.

Different studies in DC suggest alternative optimization aims to create synthetic datasets. According to [18], a possible grouping of optimization solutions involving DC can be divided into: performance matching, distribution matching, and parameter matching.

The performance matching strategy was first proposed by Wang et al. [15]. This methodology seeks to enhance a synthetic dataset to enable neural networks trained in it to exhibit minimal loss on the original dataset. Consequently, the performance of models trained in synthetic and real datasets is aligned. The performance matching objective reflects a bi-level optimization algorithm. In the inner loops, the weights of a differentiable model with parameters $\boldsymbol{\theta}$ are updated with $\mathcal{S}$ using gradient descent, and the recursive computation graph is cached. In the outer loops, models trained after inner loops are

validated in $\mathcal{T}$, and the validation loss is backpropagated by the unrolled computation graph to $\mathcal{S}$.

The alternative method, known as the distribution matching approach, aims to generate synthetic data that approximate the distribution of the real data. Instead of matching training effects (e.g., the performance of models trained on $\mathcal{S}$), distribution matching directly optimizes the distance between the two distributions using some metrics, such as the Maximum Mean Discrepancy (MMD) as presented in Zhao et al. [16].

This work focuses on parameter matching, also referred to as gradient matching. The concept of parameter matching within the domain of DC was initially introduced by Zhao et al. [16]. Subsequent studies further built upon this approach, extending its applications [20–22]. The main idea behind parameter matching is to train the same neural network using synthetic and original datasets for some stages, forcing the consistency of their losses by updates of the synthetic sample values.

Initially, $\mathcal{S}$ is updated by minimizing the gradient distance. When updating the synthetic dataset $\mathcal{S}$, Zhao et al. [16] sample each synthetic and real batch pair $\mathcal{S}_c$ and $\mathcal{T}_c$ from $\mathcal{S}$ and $\mathcal{T}$, respectively, containing samples from the c-th class, and each class of synthetic data is updated separately in each iteration. The distance $\mathcal{D}$ between the gradients $\nabla\ell(\mathcal{S}_c;\boldsymbol{\theta})$ and $\nabla\ell(\mathcal{T}_c;\boldsymbol{\theta})$ is calculated as

$$\mathcal{D}(\mathcal{S},\mathcal{T};\boldsymbol{\theta}) = \sum_{c=0}^{C-1} d[\nabla\ell(\mathcal{S}_c;\boldsymbol{\theta}),\nabla\ell(\mathcal{T}_c;\boldsymbol{\theta})], \tag{3}$$

where $d(\cdot,\cdot)$ represents the similarity between two gradients and can be calculated as indicated in the original approach by using the negative cosine similarity [16]. After computing $\mathcal{D}$, the update of the synthetic set is performed as

$$\mathcal{S}_{t+1} = \mathcal{S}_t - \eta_{\mathcal{S}}\nabla_{\mathcal{S}_t}\mathcal{D}(\mathcal{S},\mathcal{T};\boldsymbol{\theta}), \tag{4}$$

where $\eta_{\mathcal{S}}$ denotes the learning rate used for updating the synthetic set.

After each step of updating synthetic data, the network used for computing gradients is trained in $\mathcal{T}$ (some authors also take into account $\mathcal{S}$) for T steps. The parameters of the model are updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{\theta}\nabla_{\theta_t}\mathcal{L}(\mathcal{T}_T;\boldsymbol{\theta}), \tag{5}$$

where $\eta_{\mathcal{T}}$ denotes the learning rate employed in updating the parameters of the model, and $\mathcal{T}_T$ is the T-th mini-batch from the $\mathcal{T}$ set.

The standard approach to dataset distillation in the image classification task is to distill the information in the dataset into a few synthetic images with the same shape and number of channels as the original one. However, because of restricted storage space, the amount of information carried by a small number of data is limited [18]. Furthermore, because synthetic data have the same format as the original data, it is unclear whether it contains worthless or superfluous information. A series of publications focusing on these problems and orthogonal to DC optimization aims, provide several methods of synthetic data parameterization [16,23].

Based on empirical observations, Kim et al. [20] identified that the performance of the synthetic dataset is predominantly influenced by its size rather than its resolution. In response to this insight, they introduce a parameterization strategy for synthetic data known as Information-intensive Dataset Condensation (IDC). This strategy employs multiple formations to enhance the amount of synthetic data generated within the same storage constraints, achieved by lowering the data resolution. Essentially, the structure of synthetic data takes the shape of a downsampled version of an original image (e.g., downsampling by 2), leading to a fourfold increase in the potential number of synthetic samples.

3. Materials and Methods

For evaluating DC in KWS applications, we used 8 of the 35 classes provided by Google's Speech Commands dataset [24], also known as Mini Speech Commands, which contains 8000 one-second audio clips. The dataset was split into training and validation/test subsets. To ensure a rigorous evaluation, we conducted a random split of the data, allocating 80% of the samples for training our model and reserving the remaining 20% for validation/testing. This partitioning strategy allowed us to assess the model generalization capabilities effectively while maintaining a robust evaluation framework.

Speech files were sampled at a rate of 16 kHz. For time-frequency representation, we employed the spectrogram or the Mel spectrogram (number of Mel bands equal to 32) as inputs for the models. The speech signals were segmented into frames of approximately 16 ms and overlapped by 50% (i.e., 8 ms). The discrete Fourier transform with a sliding Hann window was applied to the overlapping segments of the signal. After this, the time-frequency representations were obtained.

The audio files vary in duration and produce time-frequency representations with different dimensions. Thus, it was necessary to standardize the network input data for each set of parameters. A padding with zeros in the original signal was considered, having as a default dimension the longest duration of the training set, which is about 1 s. The speech representations undergo standardization to adjust the input scale before being fed into the models.

To obtain the condensed time-frequency representations, a ConvNet with four layers was employed. To assess the quality of the condensed dataset, six architectures were tested: a four-layer ConvNet, the same employed for generating the synthetic dataset; AlexNet [25], which includes convolutional layers, max-pooling, and dropout for effective feature extraction; SqueezeNet [26], a reduced model suitable for resource-constrained environments; VGG-11 [27], a simplified variant of the VGGNet architecture with 11 weight layers, serving as a robust baseline for image classification; EfficientNet-B0 [28], an optimized model from the EfficientNet family; MobileNet [29], designed for mobile and edge devices, utilizing depthwise separable convolutions to reduce computation while preserving accuracy and DenseNet-121 [30], featuring dense connectivity, enhancing information flow between layers for improved gradient flow and parameter efficiency. These deep learning models have significantly contributed to the field, each addressing specific challenges and diverse application scenarios. All models were initialized with pre-trained weights.

To evaluate the impact of varying data availability on model performance, we tested each of these architectures with four distinct condensed time-frequency representations per class (RPC): 5, 10, 20, and 30 instances.

As the speech command classes are balanced, the overall accuracy was used as a comparison metric between models. This evaluation metric measures the proportion of correctly classified samples in the validation/test set.

The preprocessing and feature extraction were conducted in Python with the PyTorch framework. We built our experiments upon the code provided by [20]. Experiments were carried out on a Google Colab Pro virtual machine, equipped with 16 GB of RAM, an Intel Xeon CPU running at approximately 2.20 GHz with two cores, and a Tesla T4 GPU accelerator boasting 16 GB of memory.

4. Results and Discussion

The first evaluation was conducted using the dataset in its original form, without any process of condensation applied. Accuracy was obtained for all models, taking into account both time-frequency representations. Next, we processed the speech data and obtained time-frequency representations, each of size 32×32. The synthetic data are initialized with random data from the original training dataset, resulting in a faster optimization process compared to random noise initialization [16].

To comprehensively assess the performance of the condensed dataset and understand the implications of cross-model generalization, we conducted two sets of experiments.

First, we evaluated the condensed data using the same architecture from which it was obtained (i.e., the four-layer ConvNet model), aiming to assess how well the architecture could reproduce its own condensed knowledge. This intra-model evaluation provides insights into the effectiveness of the distillation process within the respective model.

In the second set of experiments, we tested the cross-model generalization. Here, we evaluated the condensed data using architectures distinct from the one utilized for distillation. These cross-model tests provide a measure of the extent of knowledge transfer between the models. This investigation also allows for an understanding of the versatility and adaptability of the condensed dataset, offering insights into its potential applicability in different speech recognition contexts.

4.1. Non-Condensed Data Evaluation

The six models were trained using all available instances in the training subset and then assessed within the validation subset. Table 1 presents the accuracy metrics obtained with the models evaluated. The results indicate that all models demonstrate accuracy surpassing 86.34%, with training conducted without any form of distillation.

Table 1. Overall accuracy of the models evaluated in the validation set, considering in the first case the spectrogram and, in the second case, the Mel spectrogram as input of the model.

Time-Frequency Representation	ConvNet	AlexNet	SqueezeNet	VGG-11	EfficientNet-B0	MobileNet	DenseNet-121
Spectrogram	87.84%	94.45%	95.50%	95.51%	93.86%	95.31%	94.30%
Mel Spectrogram	86.34%	93.48%	92.40%	94.76%	92.54%	93.57%	93.42%

To obtain the data shown in Table 1, pre-trained models were employed, followed by fine-tuning. The weights were adjusted using 875 representations per class during training and validated with 125 representations per class. This process was carried out over 30 epochs with a batch size of 32. The optimization algorithm used was SGD with a learning rate of 10^{-2}. These models were trained with the entire dataset to provide a benchmark for comparison when trained with condensed data.

Using spectrograms as input yielded slightly better results in the models evaluated. Besides, except for the ConvNet model, which presented accuracy equal to 87.84% and 86.34%, respectively, for the spectrogram and Mel spectrogram, all the other models exhibited similar accuracy.

4.2. Intra-Model Evaluation

The subsequent step involved evaluating intra-model distillation, which was carried out within the same model trained afterward. The initial step involved distilling the original data utilizing the ConvNet model. Figure 2 presents 16 examples of spectrograms of the keyword "yes" uttered by different speakers and the respective condensed form. The initial process reduces the dimensions of the original time-frequency representation from 125 $\times$ 128 (as shown in Figure 2a) to 32 $\times$ 32 and normalizes the data (Figure 2b). In their original form, the spectrograms present a high variation in maximum amplitude. Additionally, it is possible to observe a variation in the duration of the utterance. On the other hand, the condensed form presents a normalized version of the spectrogram. Overall, the condensed representations maintain the general aspect of the spectrogram, providing a summarized form. Specifically for the keyword "yes", the condensed version is noted to preserve the aspect of the first formant, and in certain instances, the second formant as well.

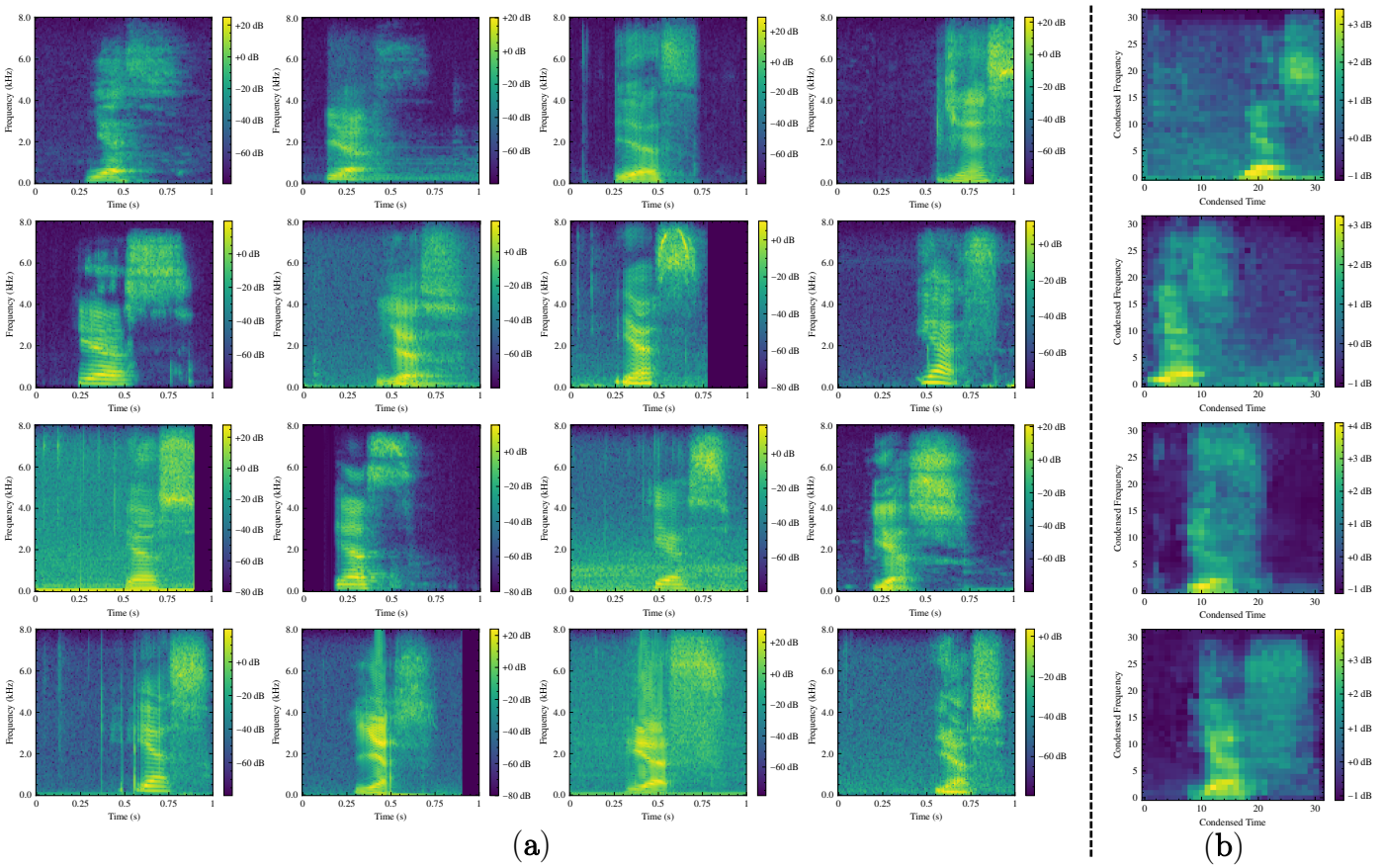

Figure 2. Representations of the keyword "yes": (**a**) examples of spectrograms from the original dataset with a resolution of 125 × 128, and (**b**) samples from the condensed dataset with a resolution of 32 × 32.

Combinations of values for typical neural network training parameters were tested. This study encompassed a range of configurations, including different types of neural networks such as three-layer and four-layer convolutional neural networks, optimization algorithms like ADAM and SGD, learning rates spanning from 10^{-4} to 10^{-2}, momentum values of 0.3, 0.6, and 0.9, different numbers of epochs (100, 200, and 300), varied batch sizes (64 and 96), and subsampling of temporal-frequency representation with interpolation at resolutions of 32 × 32 and 64 × 64. The results, presented in Tables 2–8, showcase the performance of our approach under an optimal setup. Specifically, the best configuration was identified utilizing a four-layer convolutional neural network, the SGD optimization algorithm, a learning rate of 10^{-2}, a momentum of 0.9, 100 epochs, a batch size of 64, and subsampling of temporal-frequency representation with interpolation at a 32 × 32 resolution.

To establish a baseline for comparison during the validation, we selected random samples from the training set for subsequent training and testing/validation of the model. This process is performed 10 times, with accuracy calculated each time. From these runs, we computed the average accuracy obtained from the original dataset for 10 different random sets (indicated in the tables as "Random Mean"), the standard deviation, the lowest accuracy (indicated as "Random Min."), and the highest accuracy obtained (indicated as "Random Max."). This baseline provides a reference point for understanding the performance of our condensed data relative to random chance classification.

Table 2. Data condensation with the ConvNet model and trained with ConvNet considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed [2]**	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed**
5	27.30 ± 0.85%	25.71%	30.13%	**57.52%**	30.11 ± 0.67%	28.72%	31.22%	**51.73%**
10	38.81 ± 0.73%	37.72%	39.84%	**69.14%**	46.89 ± 1.24%	45.37%	48.92%	**64.47%**
20	50.93 ± 1.05%	49.94%	53.41%	**76.66%**	56.47 ± 0.91%	55.61%	59.52%	**71.89%**
30	55.87 ± 0.87%	54.77%	57.94%	**80.78%**	59.38 ± 0.56%	58.27%	60.11%	**77.47%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 3. Data condensation performed with the ConvNet model and employed for training the AlexNet model considering the spectrogram in the first case and, in the second case, the Mel spectrogram as input of the model.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed [2]**	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed**
5	30.45 ± 2.50%	27.84%	36.00%	**46.25%**	30.72 ± 1.85%	27.74%	33.52%	**37.93%**
10	41.18 ± 3.2%	38.24%	51.25%	**58.56%**	40.71 ± 2.70%	37.12%	45.68%	**65.14%**
20	52.64 ± 2.53%	50.18%	57.59%	**64.87%**	56.16 ± 1.61%	53.94%	59.15%	**71.76%**
30	60.41 ± 2.46%	54.84%	63.35%	**77.97%**	66.21 ± 1.53%	63.42%	68.45%	**74.78%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 4. Data condensation performed with the ConvNet model and employed for training the SqueezeNet model considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed [2]**	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed**
5	25.12 ± 2.03%	23.90%	30.10%	**44.15%**	28.75 ± 2.19%	25.71%	33.42%	**45.16%**
10	34.52 ± 2.15%	31.21%	37.87%	**55.53%**	37.54 ± 2.18%	34.00%	41.10%	**58.94%**
20	47.13 ± 2.47%	42.68%	54.67%	**62.94%**	54.67 ± 1.68%	51.47%	57.44%	**66.26%**
30	53.74 ± 2.12%	50.14%	58.12%	**68.79%**	59.87 ± 2.66%	56.91%	64.76%	**70.49%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 5. Data condensation performed with the ConvNet model and employed in the training of the VGG-11 model considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed [2]**	**Random Mean [1]**	**Random Min.**	**Random Max.**	**Condensed**
5	27.75 ± 2.28%	22.91%	30.82%	**49.85%**	28.45 ± 1.48%	26.35%	29.45%	**32.72%**
10	38.65 ± 1.87%	35.43%	41.92%	**54.86%**	39.76 ± 1.35%	36.22%	41.44%	**64.20%**
20	47.88 ± 2.21%	45.53%	52.34%	**67.97%**	53.86 ± 1.59%	51.70%	57.27%	**66.70%**
30	57.43 ± 1.31%	54.47%	57.00%	**69.97%**	61.48 ± 2.05%	57.11%	63.74%	**68.50%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 6. Data condensation performed with the ConvNet model and employed for training the EfficientNet-B0 model considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed** [2]	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed**
5	21.56 ± 0.99%	19.10%	22.74%	**24.24%**	28.58 ± 0.54%	22.45%	31.54%	**19.10%**
10	38.68 ± 2.06%	34.85%	40.90%	**24.84%**	38.45 ± 1.54%	28.12%	42.54%	**18.20%**
20	42.28 ± 2.15%	37.33%	45.00%	**22.43%**	43.15 ± 1.78%	35.14%	44.28%	**20.20%**
30	48.46 ± 1.95%	44.40%	50.94%	**19.77%**	56.75 ± 2.75%	41.86%	58.74%	**17.50%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 7. Data condensation performed with the ConvNet model and employed for training the MobileNet model considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed** [2]	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed**
5	21.84 ± 1.62%	19.00%	24.32%	**36.21%**	27.42 ± 1.99%	24.12%	30.52%	**27.20%**
10	32.50 ± 2.61%	28.37%	36.35%	**26.06%**	34.44 ± 3.29%	28.32%	38.43%	**24.22%**
20	42.42 ± 4.54%	36.40%	50.14%	**26.01%**	50.14 ± 2.83%	46.17%	54.40%	**32.57%**
30	53.96 ± 4.84%	44.11%	60.38%	**26.70%**	56.80 ± 3.34%	49.80%	61.51%	**27.63%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 8. Data condensation performed with the ConvNet model and employed in the training of the DenseNet model considering the spectrogram in the first case and the Mel spectrogram as input of the model in the second case.

	Spectrogram				Mel Spectrogram			
RPC	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed** [2]	**Random Mean** [1]	**Random Min.**	**Random Max.**	**Condensed**
5	17.79 ± 3.06%	13.21%	22.52%	**36.05%**	21.12 ± 1.95%	18.25%	24.42%	**23.94%**
10	22.13 ± 2.28%	19.03%	25.86%	**15.98%**	23.46 ± 3.41%	19.10%	28.53%	**35.05%**
20	30.08 ± 3.49%	25.75%	38.28%	**22.25%**	37.66 ± 7.47%	23.40%	48.88%	**23.93%**
30	32.09 ± 3.51%	27.98%	40.70%	**30.49%**	42.95 ± 5.85%	34.04%	54.22%	**24.29%**

[1] Classification results (mean ± standard deviation). [2] The values in bold indicate the accuracy obtained with the condensed data.

Table 2 presents the accuracy for the validation/test set obtained from the four-layer ConvNet trained with condensed data generated using the same ConvNet architecture, employing both spectrograms and Mel spectrograms. To assess whether the condensed information is capable of improving the models' performance, a comparison was conducted for different RPCs.

In the first case, five instances from the original training set and five condensed instances were used to train the model. Additionally, 10, 20, and 30 instances were also tested. The "Condensed" column presents the accuracy values obtained with the distilled data. Note, that for all cases, training with condensed data significantly outperformed the maximum results achieved when the same architecture was trained with an equal number of samples from the original dataset. This behavior holds true for both the spectrogram and the Mel spectrogram.

Due to the limited number of instances provided to the models during training, a decrease in accuracy is expected when compared to the results from Section 4.1, as the model is presented with a restricted number of instances.

Notice that the results from the spectrogram are slightly superior to those from the Mel spectrogram. In addition, as the number of RPCs increases, the accuracy of the models tends to improve.

With only five instances of condensed data, it was possible to achieve accuracy exceeding 51.73%. In the same context, when presented with 30 time-frequency representations, accuracy exceeding 77.47% was attained. The results indicate that condensed data can extract information from the original dataset for training the model.

Comparing the results from RPC equal to 30, the performance achieved with condensed data is observed to be 39.42% (Spectrogram) and 28.88% (Mel spectrogram) higher than the best result from the "Random Max" column. Such results are also evident for other values of RPC.

4.3. Cross-Model Evaluation

The second condensation assessment was performed using the cross-model strategy, whereby one model is used to distill the data, and the distilled data are then employed to train other models. The condensed data were obtained using a ConvNet and trained with AlexNet, SqueezeNet, VGG-11, EfficientNet-B0, MobileNet, and DenseNet. Tables 3–8 depict the results of applying dataset condensation to cross-model scenarios.

In our experiments, we observed that data condensed by simpler neural networks such as ConvNet tend to exhibit reasonable performance in various neural network architectures, such as SqueezeNet, AlexNet, and VGG-11. However, the same cannot be said for more complex architectures, such as EfficientNet-B0, MobileNet, and DenseNet. This result highlights a constraint in the respective strategy of DC.

Considering the models SqueezeNet, AlexNet, and VGG-11, we consistently obtained higher accuracy performances with the condensed data compared to those obtained with the original data for each RPC. The AlexNet model achieved the best performance in this group. For an RPC equal to 30, the accuracy obtained with the condensed data were 77.97% with the spectrogram and 74.78% with the Mel spectrogram. A slight decrease in accuracy is noticeable when compared to the previous results from the intra-model evaluation, in which the accuracy reached 80.78%.

Conversely, the EfficientNet-B0, MobileNet, and DenseNet models show a decline in performance when trained with condensed data. In the worst case, observed with the EfficientNet-B0 model, the accuracy performance varied between 17.50% (when using the Mel spectrogram as input) and 24.84% (when using the spectrogram as input). In many cases, the accuracy obtained with condensed data is lower than that obtained with the original data. The same can also be observed in other models, such as MobileNet and DenseNet.

Certain insights can be derived based on these results. Models such as ConvNet, SqueezeNet, AlexNet, and VGG-11 tend to perform better when handling small amounts of data and low-resolution inputs. While these models reach mean accuracy greater than 53.74% for RPC equal to 30 in the case of the spectrogram, EfficientNet-B0 and DenseNet achieve mean accuracy of 48.46% and 32.09%, respectively. EfficientNet-B0 and DenseNet require more data to train and also inputs with a higher resolution compared to ConvNet, AlexNet, and VGG-11.

This pitfall demonstrates that data condensed by the evaluated algorithm suffers from issues related to generalizability. In this condition, it is not possible to assert that condensed data are agnostic to different models, as is typically expected when dealing with data. The distilled data from a simple ConvNet model may not generalize well for complex models, thus emphasizing the need for tailored strategies when working with different architectures.

While simple ConvNet might offer a degree of cross-model generalization, using complex models for data distillation introduces a new set of challenges. The process of generating a condensed dataset using matching loss techniques can be particularly arduous when dealing with intricate models. The high dimensionality and intricacy of these models can make it challenging to ensure the effective convergence of distilled representations.

In our exploration, we conducted experiments to condense data using complex models. However, we encountered issues with convergence. These models, which often consist of numerous layers and parameters, require a large number of instances to be trained. In addition, distillation using complex models demands computational and time resources. The intricate architecture and numerous parameters may require significant resources during the distillation process, and results may not always meet expectations.

The same can be said regarding the assessment of the impact of certain hyperparameters, including learning rates, optimizers, and the number of layers in ConvNet. Despite our diligent investigation, we found that these adjustments did not lead to significant improvements that warranted their inclusion in this article.

Further studies will be conducted on diverse datasets, encompassing both general and comprehensive scenarios like speech commands, as well as more specific contexts such as isolated keywords or wake-words, similar to those presented in this work.

5. Conclusions

Dataset condensation is a valuable technique for reducing the size of training datasets, enabling more efficient model training and deployment. From the tests performed, we observed that data condensed by simpler neural networks, such as ConvNet, consistently delivers promising results when used to train a group of models including the ConvNet itself, the AlexNet, the SqueezeNet, and the VGG-11 models. These models appear to handle condensed data effectively, showcasing the efficacy of the technique as a means of enhancing their performance in a KWS use case. However, the results highlight the constraints of data condensation in cross-model generalization, especially when dealing with intricate neural network architectures, such as EfficientNet-B0, MobileNet, and DenseNet.

In the tests conducted, training models with condensed data yielded positive outcomes for models that do not require a large volume of data or high-resolution input. In these instances, the accuracy achieved with the model trained using condensed data exceeds the results with the same number of representations per class from the original data. The ConvNet model achieved an accuracy of 80.78% with the spectrogram and 77.47% with the Mel spectrogram using only 30 time-frequency representations condensed. The same good results are also obtained with SqueezeNet (68.79% for spectrogram and 70.49% for Mel spectrogram), AlexNet (77.97% for spectrogram and 74.78% for Mel spectrogram), and VGG-11 (69.97% for spectrogram and 68.50% for Mel spectrogram) in the same conditions.

However, complex models presented opposite results with the distilled data. The EfficientNet-B0 model, for example, is incapable of training with condensed data. The maximum accuracy obtained was 24.84% employing, as input of the model, 10 time-frequency representations extracted from spectrograms. The same pattern was observed with the MobileNet and DenseNet models, exhibiting poorer performance compared to when the model was trained with the original data. Therefore, the performance of the condensed data generated is not consistent across different models. Furthermore, determining the performance of condensed data representations generated by neural models in unseen architectures is not always clear. The challenge is to ensure that the condensed data remain representative and usable across various applications.

The results demonstrate that there is a pitfall in data condensation generated by the evaluated algorithm, which faces challenges related to generalizability. Although some authors may choose not to conduct cross-model evaluations, the results illustrate the necessity of such tests. The performed tests also demonstrate that efficient hyperparameter tuning is crucial for optimizing the distillation process.

Additionally, when attempting to distill the original dataset with complex models, issues related to non-convergence were noted. The use of complex models for data distillation can be computationally demanding, requiring careful consideration of the resources available. Therefore, a trade-off exists between the complexity of the model and its ability to distill (i.e., condensate) the data.

By addressing these challenges and understanding their nuances, the field of data condensation can continue to evolve and provide valuable solutions for data-intensive machine learning applications.

Author Contributions: Conceptualization, P.H.P. and W.B.; methodology, P.H.P. and W.B.; software, P.H.P. and W.B.; validation, P.H.P., W.B. and M.A.R.; writing—original draft preparation, P.H.P., W.B. and M.A.R.; writing—review and editing, P.H.P., W.B. and M.A.R.; supervision, P.H.P., W.B. and M.A.R.; funding acquisition, W.B. and M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)-Finance Code 001. This research also received partial support from the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) under Grant 2022/10909-5.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge the financial support of the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

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

References

1. Leem, S.G.; Yoo, I.C.; Yook, D. Multitask Learning of Deep Neural Network-Based Keyword Spotting for IoT Devices. *IEEE Trans. Consum. Electron.* **2019**, *65*, 188–194. [CrossRef]
2. López-Espejo, I.; Tan, Z.H.; Hansen, J.H.L.; Jensen, J. Deep Spoken Keyword Spotting: An Overview. *IEEE Access* **2022**, *10*, 4169–4199. [CrossRef]
3. Cerutti, G.; Cavigelli, L.; Andri, R.; Magno, M.; Farella, E.; Benini, L. Sub-mW Keyword Spotting on an MCU: Analog Binary Feature Extraction and Binary Neural Networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2002–2012. [CrossRef]
4. Shan, W.; Yang, M.; Wang, T.; Lu, Y.; Cai, H.; Zhu, L.; Xu, J.; Wu, C.; Shi, L.; Yang, J. A 510-nW Wake-Up Keyword-Spotting Chip Using Serial-FFT-Based MFCC and Binarized Depthwise Separable CNN in 28-nm CMOS. *IEEE J. Solid-State Circuits* **2021**, *56*, 151–164. [CrossRef]
5. Pereira, P.H.; Beccaro, W.; Ramírez, M.A. Evaluating Robustness to Noise and Compression of Deep Neural Networks for Keyword Spotting. *IEEE Access* **2023**, *11*, 53224–53236. [CrossRef]
6. Gong, Y.; Li, Y.; Ding, X.; Yang, H.; Zhang, Z.; Zhang, X.; Ge, W.; Wang, Z.; Liu, B. QCNN Inspired Reconfigurable Keyword Spotting Processor with Hybrid Data-Weight Reuse Methods. *IEEE Access* **2020**, *8*, 205878–205893. [CrossRef]
7. Heittola, T.; Mesaros, A.; Virtanen, T. Acoustic Scene Classification in DCASE 2020 Challenge: Generalization Across Devices and Low Complexity Solutions. *arXiv* **2020**, arXiv:2005.14623.
8. Martín-Morató, I.; Paissan, F.; Ancilotto, A.; Heittola, T.; Mesaros, A.; Farella, E.; Brutti, A.; Virtanen, T. Low-Complexity Acoustic Scene Classification in DCASE 2022 Challenge. *arXiv* **2022**, arXiv:2206.03835.
9. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. *arXiv* **2018**, arXiv:1807.06521.
10. Chinnamuniyandi, M.; Chandran, S.; Xu, C. Fractional Order Uncertain BAM Neural Networks with Mixed Time Delays: An Existence and Quasi-Uniform Stability Analysis. *J. Intell. Fuzzy Syst.* **2024**, *46*, 4291–4313. [CrossRef]
11. Agarwal, P.K.; Har-Peled, S.; Varadarajan, K.R. Approximating Extent Measures of Points. *J. ACM (JACM)* **2004**, *51*, 606–635. [CrossRef]
12. Feldman, D.; Schmidt, M.; Sohler, C. Turning Big Data Into Tiny Data: Constant-size Coresets for k-means, PCA, and Projective Clustering. *SIAM J. Comput.* **2020**, *49*, 601–657. [CrossRef]
13. Phillips, J.M. Coresets and Sketches. In *Handbook of Discrete and Computational Geometry*; Chapman and Hall/CRC: London, UK, 2017; pp. 1269–1288.
14. Har-Peled, S.; Mazumdar, S. On Coresets for k-means and k-median Clustering. In Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 13–16 June 2004; pp. 291–300.
15. Wang, T.; Zhu, J.Y.; Torralba, A.; Efros, A.A. Dataset Distillation. *arXiv* **2020**, arXiv:1811.10959.
16. Zhao, B.; Mopuri, K.R.; Bilen, H. Dataset Condensation with Gradient Matching. *arXiv* **2021**, arXiv:2006.05929.

17. Chen, J.; Xu, K.; Ning, Y.; Jiang, L.; Xu, Z. CRTED: Few-Shot Object Detection via Correlation-RPN and Transformer Encoder–Decoder. *Electronics* **2024**, *13*, 1856. [CrossRef]
18. Yu, R.; Liu, S.; Wang, X. Dataset Distillation: A Comprehensive Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 150–170. [CrossRef]
19. Lei, S.; Tao, D. A Comprehensive Survey of Dataset Distillation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 17–32. [CrossRef] [PubMed]
20. 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. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 11102–11118.
21. Lee, S.; Chun, S.; Jung, S.; Yun, S.; Yoon, S. Dataset Condensation with Contrastive Signals. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 12352–12364.
22. 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, New Orleans, LA, USA, 18–24 June 2022; pp. 4750–4759.
23. Nguyen, T.; Novak, R.; Xiao, L.; Lee, J. Dataset Distillation with Infinitely Wide Convolutional Networks. *arXiv* **2022**, arXiv:2107.13034.
24. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**, arXiv:1804.03209.
25. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–6 December 2012. Available online: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (accessed on 26 May 2024). [CrossRef]
26. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level Accuracy with 50× Fewer Parameters and <0.5 MB Model Size. *arXiv* **2016**, arXiv:1602.07360.
27. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representation. Computational and Biological Learning Society, San Diego, CA, USA, 7–9 May 2015; pp. 1–14.
28. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
29. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
30. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.