OptiCore: Scalable Greedy Coreset Optimization Method for Efficient Deep Learning

André Vinicius Neves Alves*
Federal University of Pará (UFPA)
Belém, Pará, Brazil
andre.neves.alves@itec.ufpa.br

Adriano Madureira dos Santos
Federal University of Pará (UFPA)
Belém, Pará, Brazil
adriano.madureira.santos@itec.ufpa.br

Lyanh Vinicios Lopes Pinto
Federal University of Pará (UFPA)
Belém, Pará, Brazil
lyanh.pinto@itec.ufpa.br

Vitor Hugo Barbosa Melo
Federal University of Pará (UFPA)
Belém, Pará, Brazil
vitor.melo@itec.ufpa.br

Flávio Rafael Trindade Moura
Federal University of Pará (UFPA)
Belém, Pará, Brazil
flavio.moura@itec.ufpa.br

Saulo William da Silva Costa
Federal University of Pará (UFPA)
Belém, Pará, Brazil
saulo.costa@ifpa.edu.br

Marcos César da Rocha Seruffo
Federal University of Pará (UFPA)
Belém, Pará, Brazil
seruffo@ufpa.br

Walter dos Santos Oliveira Júnior
Federal University of Pará (UFPA)
Belém, Pará, Brazil
walter@inteceleri.com.br

Abstract

The growing use of Deep Learning in various domains has amplified the challenges of training models due to high computational costs and the need for large volumes of data. To address these limitations, this study presents the OptiCore method, a new dataset optimization approach based on the Greedy Coreset technique. OptiCore strategically reduces the size of datasets while preserving their representativeness and diversity, integrating computational cost analyses through the Relative Cost Normalized metric. This method balances data efficiency and model performance, offering a scalable solution for practical applications. The methodology is designed for generalization and reproducibility, extending its usefulness to different Deep Learnig contexts. In the case study, Deep Learning models were applied for the classification of three-dimensional shapes, with the ResNet-50 architecture showing the best results. OptiCore reduced the dataset by up to 90%, maintaining competitive accuracy while significantly reducing computational demands.

Keywords

Coreset, Deep Learning, Dataset Reduction, Computational Efficiency

1 Introduction

Deep Learning (DL) has emerged as one of the most promising fields within Artificial Intelligence (AI) in recent decades [1], achieving significant advancements in a variety of complex tasks such as computer vision [2], natural language processing [3], and even speech recognition [4]. These advances are largely attributed to the ability of Deep Neural Networks (DNN) to learn hierarchical representations of data, enabling them to capture complex patterns of high dimensionality. This capability contributes to the automation of tasks, the resolution of complex problems, and enhanced decision-making processes [5].

Most DL models are trained with a significant amount of data in order to learn and achieve satisfactory performance [6]. This approach is strongly accepted by the community because the massive amount and high variability of data is used to ensure that the models show adequate generalization during their life cycle [7]. This makes it possible to simulate the complexity and diversity of real-world data, as well as avoiding overfitting, a phenomenon in which models excessively adjust to specific characteristics of the training data [8].

Despite the benefits of using large volumes of data, the high computational cost is a challenge when developing DL models using a large volume of data [9]. This is because these approaches require the use of high-performance infrastructures, as well as high demand for memory and graphics processing [10]. This process can become a bottleneck in systems with limited resources, which in turn limits the training scalibility and the practical feasibility of using very large databases, this makes DNN training in Big Data an expensive and often impractical task [11].

In this context, the possibility arises of optimizing datasets in order to reduce the volume of data without compromising the quality of the representations essential for training DL models [12]. The dataset reduction techniques offer viables solutions,[13], allowing intelligent reduction of the dataset, preserving the characteristics that proposes the highest impact to DL performances, while minimizing computational costs and, therefore, training time. The idea involves obtaining robust models, but with a significant reduction in computational costs and training time, without the need large amounts of data ([14], [13]).

Among the most common Dataset Reduction approaches there are Dataset Distillation, Coreset Selection and its derivative, Greedy Coreset. Coreset Selection, developed by Sener (2018), focuses on choosing a representative subset of data, selecting points that best capture the characteristics of the original set. In contrast, Dataset Distillation, proposed by Wang (2020), aims to synthesize data, creating new points that encapsulate the essential information of the complete set. The Greedy Coreset presented by Chai (2023), is a

method integrated with the Selection Coreset, but with its own particularities. It uses an iterative process to choose data points based on distance, focusing on maximizing the diversity of the selected data efficiently and ensuring good representativeness, such as variability, value distributions and variations in data characteristics, with a reduced number of samples.

Thinking of ways to optimize datasets in order to reduce computational costs and maintain the quality of data representation, this study proposes a new general and replicable approach to dataset optimization. The OptiCore method seeks to solve problems involving large amounts of data by integrating optimized data collection, applying the Greedy Coreset technique to raw datasets, as well as training and evaluating trained DL models on databases generated using a developed metric. This approach aims to reduce the number of samples and guarantee the variability and representability [18], minimizing performance loss during training and optimizing the development environment.

The method's applicability was done over three dimensional geometric shapes datasets context. This focus was motivated by the needs of the Inteceleri company with the educational game Geometa, publicly available on Google Play[1] and the App Store[2]. The game aims to promote the teaching and learning of Geometry using Virtual and Augmented Reality in the Metaverse environment. Currently, the company proposes immersive teaching in general environments, but intends to adapt it for the Amazonian context. The main challenge in this scenario is related to the lack of contextualized three-dimensional data for this specific environment, which makes it difficult to develop relevant and effective teaching materials for the Game.

In this context, the proposed solution involves using an algorithm to automatically collect relevant examples of geometric shapes in the Amazon environment, which will compose the initial database required for training the Deep Learning models. At a later stage, the Greedy Coreset methodology presented in this study will be applied to select the most representative and highest quality images. The aim, therefore, is to optimize the dataset for training DL models, ensuring a representative and defined database to lauching efficient AI models to Geometa application.

Although the application of the coreset method was initially motivated by the needs of the Inteceleri company and its Geometa game, the approach proposed in this study is widely replicable and generalizable to any type of image database. To this end, the OptiCore method is proposed, which integrates the entire optimization process, including data collection, application of the Greedy Coreset algorithm, training of DP models and analysis of computational efficiency using the NRC metric. This system aims to solve three main challenges: (i) efficiently reducing the size of voluminous datasets, preserving the representativeness and diversity of the data; (ii) optimizing the use of computational resources during model training, reducing memory and processing costs; and (iii) objectively evaluating the efficiency of the reduced models, guaranteeing the viability of their application in scenarios with restrictions.

[1]https://play.google.com/store/apps/details?id=com.Inteceleri.Geometa&hl=en_BR
[2]https://apps.apple.com/br/app/geometa/id1644556020

2 Related Works

The relevant works deal with the concept of Coresets and their application in optimizing the DL models training. Coresets are representative subsets of an original big dataset, constructed to preserve its essential characteristics, allowing optimization and analysis operations to be done using a reduced amount of data [19]. This technique is widely used to reduce computational costs while maintaining the quality of the dataset [20].

Mirzasoleiman et al. (2020) explores the application of Coresets techniques to increase the effectiveness of Incremental Gradient (IG) techniques, such as Stochastic Gradient (SGD), in training DL models. The study shows how the appropriate choice of data subsets can speed up training and considerably reduce computational costs. This strategy is similar to the one presented in this article, which proposes an efficient alternative to optimize training without compromising model training. The study focuses on optimizing model training using Coresets, with the potential to expand to more sophisticated methods aiming for computational efficiency.

In Sener et al. (2018) , the coreset technique is used in the context of active learning, specifically for Convolutional Neural Networks (CNN). They propose the active selection of subsets of training data based on the geometry of the data points in order to maximize learning effectiveness. This active learning approach can complement the construction of coresets, offering new perspectives on optimizing the choice of data for training, which is directly relevant to the development of more efficient DL methods, such as those proposed in this work.

Yang et al. (2022) proposes a hierarchical approach based on coresets to optimize the training of large-scale neural networks in order to reduce the use of computational resources such as memory and processing. The technique creates compact representations of the data while retaining essential information, which makes it possible to improve efficiency without compromising precision. In addition, the application of neuromorphic systems makes the solution scalable and suitable for environments with limited resources, offering a viable alternative to complex neural networks on devices with hardware restrictions.

Years later, Yang et al. (2024) presented an approach for selecting coresets focused on reconstructing the decision frontier of DL models. The central idea is to identify a representative subset of data that preserves the structure of the decision frontier, which makes it possible to reduce computational complexity during training without compromising the accuracy of the model. The research shows that by selecting only the critical points for defining the frontier, it is possible to improve the efficiency of the training process on large data sets, while maintaining the quality of the predictions.

In Braverman et al. (2021) an innovative approach to building coresets is introduced. The authors propose a sampling technique based on sensitivity, which aims to select the most relevant data for the model, taking into account its contribution to the model's error. This process makes it possible to maximize the quality of the approximation while reducing the size of the coreset, preserving the accuracy of the model with a smaller number of samples. The research by Braverman et al. demonstrates how this approach can be applied to improve computational efficiency in large volumes of data, while maintaining performance in DL models.

Xia el al. (2024) addresses an improved methodology for dataset optimization, with a focus on reducing the size and ensuring the performance constraints of the model. The proposal aims to minimize the amount of data needed to train DL models, while ensuring that performance is not compromised. The selection technique considers both efficiency and the preservation of precision, allowing for the construction of more compact and still representative coresets. The research demonstrates how it is possible to achieve a balance between reducing coreset size and maintaining model quality, offering an effective solution for scenarios with computational limitations.

Existing approaches to related to the coreset application have shown high potential for optimizing the DL models training, especially by reducing computational costs and maintaining the representativeness of the data. However, most techniques focus on specific scenarios or require adaptations for different applications. The key differentiator of this work lies in the application of the OptiCore algorithm in a generalizable way, integrating it into a pipeline that allows for its replication in various contexts. This approach seeks not only to optimize the size of the data, but also to guarantee a scalable process that is easy to implement, meeting both academic and practical demands.

3 Metodology

The following subsections will present the necessary information steps to understand the motivation, development and formulation of the OptiCore method. Figure 1 shows the pipeline followed throughout the proposed methodology, discussed in the following sections.

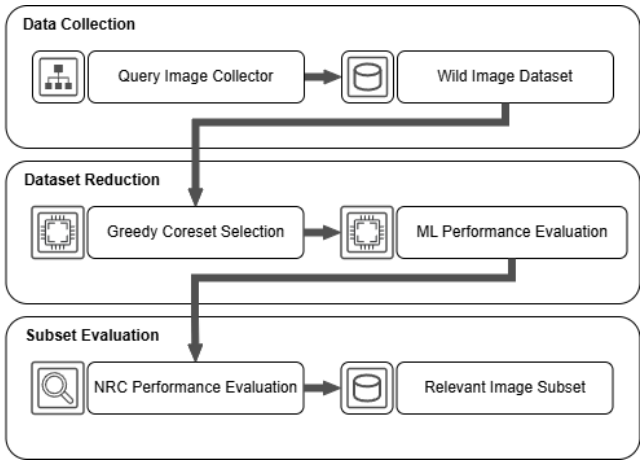

Figure 1: Design of the study's methodological protocol

3.1 Dataset Collection

To simulate the large quantity of images, an image collection algorithm was created, as shown in Figure 1. To do this, we used an algorithm implemented in Python [3] that makes use of the Selenium[4] libraries. First, the algorithm searches for images in a browser based on a search query, randomly generated from a JSON file with a list of adjectives, classes and nouns. After finding the images, they are stored in directories organized by class. The images are processed (resized to 128x128 pixels) and then compressed into a ZIP file. The collection is optimized through parallel execution, using multiple processes, speeding up the download of the images and improving the efficiency of the process. Finally, the final dataset is saved and prepared for use.

For the search query in the JSON file, the classes used were: Parallelepiped, Sphere, Cone and Cylinder. The classes were chosen based on the future supply of images from users, where they will feed geometric shapes into the database. The selection of these shapes aims to create a varied and representative data set, enabling the DL model to recognize and classify different types of three-dimensional shapes. The expectation is that by collecting images of these geometric shapes, simulating the sending of images by users, it will be possible to generate a robust dataset that will contribute to the development of an efficient shape recognition system.

3.2 Greedy Coreset

The Greedy Coreset algorithm was used to select a representative subset of embeddings extracted from images in a dataset. The approach aims to significantly reduce the number of samples while maintaining the diversity and representativeness of the original data. To do this, a pre-trained ResNet-50 was used to obtain the embeddings of the images, which are vectors that represent latent characteristics of each sample. The selection process follows a greedy strategy in which the most distant points from the set of selected samples are added to the coreset iteratively.

Algorithm 1 OptiCore Dataset Optimization

Require: Original Image Dataset D, number of points in the coreset k
Ensure: Selected points for the coreset C, New dataset with the coreset D_{coreset}
Load the pre-trained ResNet-50
Remove the last classification layer of ResNet-50 to obtain embeddings
For each image i in D, perform:
Preprocess the image i (resize, normalize, etc.)
Extract the embedding of the image i using ResNet-50
Convert the extracted embeddings into a set D' of vectors
Randomly select a point from D' and add it to C
for $i = 2$ to k **do**
    Compute the minimum distance of each point in D' to the points in C
    Select the point in D' with the largest minimum distance to C
    Add the selected point to C
end for
Create the new dataset with the coreset:
For each selected point in C, add the corresponding image to the new dataset directory D_{coreset}
Create the folder structure in the directory D_{coreset} to store the coreset images according to their classes
Copy the selected images to the new coreset directory
return C, D_{coreset}

Algorithm 1 describes an approach for creating a representative coreset from an original big dataset using a greedy strategy. Initially, the pre-trained ResNet-50 is loaded and adjusted, with the final classification layer removed, allowing for the extraction of embeddings, which capture latent characteristics of the images. After extraction, all images are normalized to ensure consistency in the data, and the embeddings generated are organized into a set, forming the basis for coreset selection.

[3] https://www.python.org/
[4] https://selenium-python.readthedocs.io/#

The selection process begins with the random selection of a point from the set of embeddings, which is added to the coreset. Iteratively, the algorithm identifies the most distant point from the current set, maximizing diversity among the selected samples. Finally, the images corresponding to the chosen embeddings are organized in a new directory, maintaining the original class structure. This process results in a reduced, efficient and representative dataset, ideal for optimizing computational costs without compromising training quality.

3.3 Normalized Relative Cost (NRC)

To evaluate the efficiency of the model in terms of cost per sample, considering the training and testing times, the Normalized Relative Cost (NRC) metric was developed. This metric is defined as:

$$\text{NRC} = \frac{T_{\text{train}} + T_{\text{test}}}{N_{\text{imgs}}} \tag{1}$$

In the formula, T_{train} is the total training time, T_{test} is the total testing time and N_{imgs} refers to the total number of images in the dataset. The NRC makes it possible to evaluate the model's efficiency in different contexts by weighting the computational times by the size of the dataset and the total processing time. This approach makes it possible to make a fair analysis between approaches with or without the use of coresets. The lower the NRC value, the greater the model's efficiency in relation to the computational cost per sample, which is particularly relevant in scenarios where resource optimization is a critical factor.

During the development of the NRC, the need to adjust the metric was identified. This was because the reduction in the number of images in the dataset after applying the coreset resulted in inconsistent NRC values, which were often higher, even with a smaller volume of data. To correct this discrepancy, an adjusted version of the NRC was proposed, defined as:

$$\text{NRC}_{\text{adjusted}} = \frac{(T_{\text{train}} + T_{\text{test}})}{N_{\text{imgs}}} \cdot \frac{1}{T_{\text{total}}} \tag{2}$$

In the formula, T_{total} represents the total processing time, which includes both the training and testing phases. This additional normalization incorporates the relative efficiency in relation to the total time, ensuring a fairer comparison between the model trained with the raw dataset and the one trained with the coreset. The adjusted NRC more accurately reflects the efficiency of the coreset, avoiding undue penalties due to a reduction in the number of samples.

In the literature, there is no metric equivalent to NRC, which makes this approach innovative for evaluating the efficiency of models in relation to the computational cost per sample. Although there are traditional metrics that evaluate the processing time and computational complexity of models, [27], none of them combine training and testing time with the total amount of samples to provide a more balanced view of relative cost. Furthermore, the adjusted version of the NRC introduces an additional normalization that corrects for discrepancies observed when applying correction techniques, such as the use of coresets. This lack of similar metrics reinforces the relevance of NRC as a new indicator for measuring the computational efficiency of machine learning models, especially in scenarios where resource optimization is key.

3.4 Fine Tuning

The pre-trained ResNet-50 [28] was used in the model training phase, in wich the model had its number of classes adjusted by just replacing and training its final layer. Training was carried out using a normalized dataset based on ImageNet default pretraining weights. The model was trained using the Adam optimizer and CrossEntropy loss function, with an initial learning rate of 1×10^{-3}. Finally, it was developed the stop critetion in wich the training could be stopped if the validation loss increased by 5 consecutive epochs, thus avoiding overfitting and optimizing training time.

In addition, the best validation loss metric was monitored and the corresponding model was saved as a checkpoint to ensure that the best performing state was preserved. The approach adopted considered not only precision, but also metrics such as accuracy, recall and F1-score, in order to comprehensively evaluate the model's performance at each epoch. The total training time was taken into account for future efficiency analysis.

4 Results

The results presented in this study explore the application of the Greedy Coreset algorithm, in conjunction with the system built, to select a representative subset of data, followed by the evaluation of models trained on the basis of the NRC. The Greedy Coreset algorithm was used to reduce the number of samples in the dataset while maintaining its diversity and representativeness, with the aim of minimizing the computational cost without significantly comprimising the model's performance.

Table 1: Post-Coreset Results

Datasets	Total Images	Size (Mb)	Test Time (s)
Original Dataset	7844	148	1.60
Coreset 30 %	1132	27.98	1.12
Coreset 20 %	755	20.43	0.85
Coreset 10 %	377	10.94	0.61

After applying the Greedy Coreset algorithm, there was a significant reduction in the size of the original big dataset, as shown in Table 1, while maintaining the diversity and representativeness of the selected samples. The algorithm selected a subset of data that captured the essential characteristics of the original big dataset, based on maximizing the minimum distance between image embeddings, seeking the greatest variability of data within the dataset.

As a result, the size of the dataset was reduced to a fraction of the original big dataset, without compromising the quality of the data for training, with the application of the coreset in 10% being the smallest in size. This reduction not only reduced the computational cost, making training more efficient, but also demonstrated the ability of the Greedy Coreset to preserve the fundamental structure of the data, with little or no loss of performance in the trained models.

The application of the Greedy Coreset resulted in notable gains in time and resource savings, as shown in Table 1. The reduction in test time from 1.60s to 0.61s for the 10% coreset demonstrates the efficiency of the technique, which is crucial for real-time systems. In addition, the dataset size was reduced from 148 MB to 10.94 MB,

making the model lighter and more suitable for devices with limited resources. By efficiently reducing the size of voluminous datasets while preserving the representativeness and diversity of the data, the method successfully meets criterion (i). In larger datasets, these improvements can scale, providing benefits such as lower operating costs, reduced latency, and greater feasibility for deployments in environments with restricted infrastructure.

Table 2: Metrics of assertiveness

Dataset	Precision	Accuracy	Recall	F1-Score
Original Dataset	0.91	0.90	0.90	0.90
Coreset 10%	0.89	0.88	0.90	0.89
Coreset 20%	0.90	**0.90**	**0.90**	**0.90**
Coreset 30%	**0.93**	0.89	0.90	0.90

As shown in Table 2, the results presented highlight the impact of using the Greedy Coreset to train a ResNet50, demonstrating the feasibility of significantly reducing the size of the dataset without substantially compromising the model's performance. The raw dataset establishes a solid base, with balanced metrics (precision, accuracy, recall and F1-score around 0.90).

However, when applying the coreset, it is possible to observe a slight variation in the metrics, depending on the level of reduction. The 10% coreset shows a slight drop in precision and F1-score (both at 0.89), while the 20% coreset maintains identical metrics to the raw dataset, demonstrating an excellent balance between efficiency and performance. The 30% coreset, on the other hand, stands out for its greater precision (0.93), although it registers a slight reduction in accuracy (0.89), proving to be ideal for scenarios where precision is critical. By training the ResNet-50 model on reduced datasets and on the original big dataset to compare performance metrics, the method successfully meets criterion (ii). These results reinforce the efficiency of the Greedy Coreset in preserving the quality of the model even with reduced datasets, making it a practical and scalable solution for applications in environments with computational constraints.

Table 3: NRC Results

Dataset	NRC
Original Dataset	3.0207×10^{-2}
Coreset 10%	1.114×10^{-3}
Coreset 20%	1.325×10^{-3}
Coreset 30%	4.321×10^{-3}

The results presented in Table 3 show that the application of Coreset significantly reduces the computational cost, with lower values indicating better efficiency. The NRC of the raw dataset is 3.0207×10^{-2}, the highest among the cases, reflecting the high computational cost due to the greater number of images. With the 10% coreset, the NRC decreases to 1.114×10^{-3}, showing the highest efficiency with the lowest cost per sample.

The 20% coreset has an NRC of 1.325×10^{-3}, which is better than the raw dataset, balancing data reduction and performance preservation well. The 30% coreset has an NRC of 4.321×10^{-3}, which is even more efficient than the original big dataset, although it costs more than the 10% and 20% coresets. By analyzing the reductions in computational costs using the NRC metric, the method successfully meets criterion (iii). These results reinforce the effectiveness of the Greedy Coreset in optimizing resources and improving computational efficiency.

Trough the RNC metric results, it was possible to evaluate the efficiency of the models in terms of cost per sample, taking into account both training and testing times. The results show that, after applying the coreset, the models maintained a robust performance, with little or no loss of accuracy, demonstrating the effectiveness of the method in creating compact and efficient datasets for training, without compromising the quality of the results, as shown in Table 2.

5 Final considerations

This paper proposed a scalable dataset optimization methodology using the Greedy Coreset algorithm, aimed at addressing the challenges of training DL models on large datasets, such as long training times and excessive memory consumption. The OptiCore method was evaluated using a case study with the ResNet-50 model applied to a dataset of 3D geometric shapes. The approach proved to be effective by: (i) reducing the size of the dataset while preserving its representativeness by selecting coresets; (ii) training the ResNet-50 model on reduced datasets and on the original big dataset to compare performance metrics; and (iii) analyzing the reductions in computational costs using the Normalized Relative Cost (NRC) metric.

The experiments results showed that reducing the dataset by up to 90% could offer significant gains in computational efficiency, with little or no loss in model performance. It was also observed that the use of Coresets enabled substantial savings in memory and processing time, while maintaining precision, recall and F1-score metrics comparable to those obtained with the original big dataset, showing that the OptiCore method is a scalable and adaptable solution for different contexts and data types.

Despite the promising results of the OptiCore method, limitations remain regarding computational cost and the Normalized Relative Cost (CRN) metric. The initial cost of extracting embeddings and performing iterative selection with the Greedy Coreset can be prohibitive for systems with limited resources. Additionally, while CRN effectively evaluates computational efficiency, it may require adjustments to ensure robustness across diverse datasets and varying training-test dynamics.

Future work will include developing more sophisticated metrics to assess the representativeness of coresets on different types of data, such as time series and text, as well as exploring combinations of the Greedy Coreset with techniques such as distillation and active learning. The definition of more robust selection thresholds will also be investigated, seeking an ideal balance between reducing the size of the dataset and preserving the model's performance. Finally, we intend to apply the proposed method to other domains and scenarios, expanding its impact and practical viability in deep learning contexts.

References

[1] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[2] Jia Liu and Yaochu Jin. A comprehensive survey of robust deep learning in computer vision. *Journal of Automation and Intelligence*, 2023.
[3] Jasmin Bharadiya. A comprehensive survey of deep learning techniques natural language processing. *European Journal of Technology*, 7(1):58–66, 2023.
[4] Tala Talaei Khoei, Hadjar Ould Slimane, and Naima Kaabouch. Deep learning: Systematic review, models, challenges, and research directions. *Neural Computing and Applications*, 35(31):23103–23124, 2023.
[5] NL Rane, M Paramesha, J Rane, and O Kaya. Emerging trends and future research opportunities in artificial intelligence, machine learning, and deep learning. *Artificial Intelligence and Industry in Society*, 5:2–96, 2024.
[6] Hui Luan, Peter Geczy, Hollis Lai, Janice Gobert, Stephen JH Yang, Hiroaki Ogata, Jacky Baltes, Rodrigo Guerra, Ping Li, and Chin-Chung Tsai. Challenges and future directions of big data and artificial intelligence in education. *Frontiers in psychology*, 11:580820, 2020.
[7] Muhammad Naeem, Tauseef Jamal, Jorge Diaz-Martinez, Shariq Aziz Butt, Nicolo Montesano, Muhammad Imran Tariq, Emiro De-la Hoz-Franco, and Ethel De-La-Hoz-Valdiris. Trends and future perspective challenges in big data. In *Advances in Intelligent Data Analysis and Applications: Proceeding of the Sixth Euro-China Conference on Intelligent Data Analysis and Applications, 15–18 October 2019, Arad, Romania*, pages 309–325. Springer, 2022.
[8] Mohammad Mahdi Bejani and Mehdi Ghatee. A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, 54(8): 6391–6438, 2021.
[9] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 10, 2020.
[10] Kewei Bian and Rahul Priyadarshi. Machine learning optimization techniques: a survey, classification, challenges, and future research issues. *Archives of Computational Methods in Engineering*, pages 1–25, 2024.
[11] Chunlei Chen, Peng Zhang, Huixiang Zhang, Jiangyan Dai, Yugen Yi, Huihui Zhang, and Yonghui Zhang. Deep learning on computational-resource-limited platforms: A survey. *Mobile Information Systems*, 2020(1):8454327, 2020.
[12] Jasmin Praful Bharadiya. A tutorial on principal component analysis for dimensionality reduction in machine learning. *International Journal of Innovative Science and Research Technology*, 8(5):2028–2032, 2023.
[13] Zheng Ju, Honghui Du, Elias Tragos, Neil Hurley, and Aonghus Lawlor. Exploring coresets for efficient training and consistent evaluation of recommender systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1152–1157, 2024.
[14] Animesh Gupta, Irtiza Hasan, Dilip K Prasad, and Deepak K Gupta. Data-efficient training of cnns and transformers with coresets: A stability perspective. *arXiv preprint arXiv:2303.02095*, 2023.
[15] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach, 2018. URL https://arxiv.org/abs/1708.00489.
[16] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020. URL https://arxiv.org/abs/1811.10959.
[17] Chengliang Chai, Jiayi Wang, Nan Tang, Ye Yuan, Jiabin Liu, Yuhao Deng, and Guoren Wang. Efficient coreset selection with cluster-based methods. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 167–178, 2023.
[18] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data. *Proceedings of the ACM on Management of Data*, 1(2):1–27, 2023.
[19] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets survey. *Current Trends in Combinatorial and Computational Geometry, E. Welzl, ed., Cambridge University Press, Cambridge*, 2006.
[20] Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer, 2022.
[21] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.
[22] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.
[23] Liwei Yang, Huaipeng Zhang, Tao Luo, Chuping Qu, Myat Thu Linn Aung, Yingnan Cui, Jun Zhou, Ming Ming Wong, Junran Pu, Anh Tuan Do, et al. Coreset: Hierarchical neuromorphic computing supporting large-scale neural networks with improved resource efficiency. *Neurocomputing*, 474:128–140, 2022.
[24] Shuo Yang, Zhe Cao, Sheng Guo, Ruiheng Zhang, Ping Luo, Shengping Zhang, and Liqiang Nie. Mind the boundary: Coreset selection via reconstructing the decision boundary. In *Forty-first International Conference on Machine Learning*, 2024.
[25] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. Efficient coreset constructions via sensitivity sampling. In *Asian Conference on Machine Learning*, pages 948–963. PMLR, 2021.
[26] Xiaobo Xia, Jiale Liu, Shaokun Zhang, Qingyun Wu, Hongxin Wei, and Tongliang Liu. Refined coreset selection: Towards minimal coreset size under model performance constraints. In *Forty-first International Conference on Machine Learning*, 2024.
[27] Sabir Hossain and Deok-jin Lee. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices. *Sensors*, 19(15), 2019. ISSN 1424-8220. doi: 10.3390/s19153371. URL https://www.mdpi.com/1424-8220/19/15/3371.
[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.