

**Dynamic data reduction for
foundational AI models**

**A Thesis Submitted for the
Degree of Doctor of
Philosophy**

By

Dominic Sanderson

**Department of Electronic and
Electrical Engineering,
Brunel University London**

2026

Abstract

Artificial Intelligence (AI) fundamentally relies on data to generate useful outputs. The accuracy of an AI system is not only determined by the effectiveness of the model, but also both the quantity and quality of the data it is trained on. Therefore, a relatively simple way to maximise AI performance is to use more data. However, as we continue to push the boundaries of AI, the demand for larger and more diverse datasets has only intensified, highlighting the importance of high quality data acquisition and management in the development of advanced AI technologies. This causes a growing concern about the environmental impact associated with these methods. The escalating energy requirements for training and operating sophisticated AI models contribute to a larger carbon footprint, prompting discussions on sustainable practices within the field of artificial intelligence. Addressing these concerns necessitates a balance between pursuing technological advancements and implementing strategies to mitigate their environmental repercussions. This thesis explores dynamic data reduction methods as a means of exploring data quality and usefulness. For example, by reducing the amount of training data used, less computation is needed, reducing the environmental cost of the algorithm.

Random dynamic data reduction is explored first by reducing the amount of data used for training at different stages of training, for image classification tasks. Three dynamic data reduction methods are introduced to observe different strategies of data reduction: data step, data increment, and data cut. These strategies present varying amounts of training data application which means that fewer image evaluations are performed to train the model than if the model is trained on the full dataset. Meanwhile, the strategies ensure that the full dataset is used, to ensure maximum data coverage. Results show that the data step method is best performing, and for datasets MNIST and smallNORB, model accuracy may be increased by 0.04% and 0.593% respectively, while 12% and 6% fewer evaluations image are performed.

Random data reduction is so far limited by uncertainty of the quality of data that is removed for a portion of the training. This may be alleviated by introducing a method to determine specific data to remove. By first applying dimensionality reduction algorithms, data may be ordered and then removed based on distance from each classes' centroid. This allows for more intelligent data selection. Data step reduction is performed on larger image datasets, namely Imagenette and the Micro-PCB dataset, yielding inferior accuracy for the Imagenette dataset with an accuracy decrease of 0.114%, and superior accuracy for the Micro-PCB dataset by 0.013%.

The data step method has shown promising results as a dynamic data reduction strategy. An alternative method using a sliding window of data is explored, which ensures unique data each epoch throughout training, instead of a step-up in data use. This method is combined with the previously used non-random data reduction, on image data of differing sizes. Results show that model accuracy is worse for MNIST, CIFAR-10, Imagenette and the Micro-PCB datasets by up to 0.16%, but accuracy increases for the smallNORB dataset by 0.654%. However, statistical significance against benchmark experimentation informs us that after reducing data, results show no statistically significant difference from the benchmark, even though training data has been reduced. Still, despite the underwhelming performance, insight is given into which data is more useful at the beginning or end stages of training by comparing the direction of the sliding data window. Results show that for unstructured datasets MNIST, CIFAR-10 and Imagenette, the forward data flow direction yields superior accuracies, while for the highly structured smallNORB and Micro-PCB datasets the opposite is true. This supports the theory that some data is more useful in the fine-tuning stages of training, depending on how structured the data is.

Having explored dynamic data reduction in image data, data reduction in imbalanced, time-series occupancy datasets is investigated. Occupancy detection from non-invasive sensor readings is useful for building interior management, while maintaining personal privacy. However, real-world data is often imbalanced, often giving bias to training. By using data reduction techniques

inspired by those used for image data, results show that data reduction on the larger class only can maintain and even improve AUC-ROC scores 0.483% with as much as 39% of the dataset removed. Dataset fusion is also explored as a means to improve the generalisability of the classifier. Results of experimentation show a decrease in AUC-ROC of as much as 48% with the full dataset, and 33% with the reduced dataset. This shows that by attempting to increase the available data, overall performance is decreased as the model struggles to specialise. Also, data reduction on a fused dataset can minimise the detrimental effects of too much data. Class density is also calculated to be a metric that measures class balance and data sufficiency, which is vital for determining whether a dataset may be reduced to improve classification performance prior to any model training.

The above investigations show that the exclusion of some data may improve model performance, as well as reduce training cost. The novel data reduction strategies that are introduced also give insight into which data is more important to training at the earlier and later stages. These findings prompt us to consider not just the quantity of data, but the quality and relevance of data during both collection, where focus can be aimed at collecting most useful data, and training, where most relevant data may be applied at the most beneficial stage of training.

Acknowledgements

To my supervisor, Professor Kalganova, through whose guidance and this research was made possible. She will confirm that, not without effort, I finally know what it takes to be a researcher; how to effectively read current and relevant topics, develop relevant hypotheses and programs to suit, and, most dreaded of all, how to write and review my findings.

To staff at Brunel, especially Dr Nikolaos Boulgouris, Dr Lu Gan, Dr Nazmul Huda, Dr Zhengwen Huang, Dr Take Itagaki, Dr Ruiheng Wu and Dr Qiao Cheng, who reviewed my progress and provided encouraging feedback and suggestions. They always reviewed my progress with encouragement and excitement, and a genuine interest in my work.

To my doctoral researcher peers, who kindly allowed me to regularly review my work with them. Their keen observations and judgement kept me on track, as well as serving as another pair of eyes when programs were not working as they should have. And of course, for all the events and activities we took part in over the years, both at Brunel and outside.

And last but by no means least, my loving wife and family. Their support over the years has been a constant motivation.

With fond memories of Professor Abbod and Professor Stonham.

This work was funded by the AFWERX project, project number 11993100, and InnovateUK, project number 10097909.

Contents

1	Introduction	14
1.1	Background	14
1.2	Research Aim, Objectives, and Scope	15
1.3	Contributions to Science	16
1.4	Thesis outline and structure	16
1.4.1	List of Publications	18
2	Literature Review	19
2.1	Fundamental Paradigms of Data Processing	19
2.1.1	Statistical Models & Machine Learning Foundations	19
2.1.2	Deep Learning & Neural Architectures	20
2.1.3	Computational Complexity	21
2.1.4	Summary	21
2.2	Data Manipulation and Transfer Learning	22
2.2.1	Knowledge Transfer & Distillation	22
2.2.2	Data and dataset Fusion	22
2.2.3	Summary	23
2.3	Data Difficulty and Learning Dynamics	24
2.3.1	Easy vs. Hard-to-Learn Samples	24
2.3.2	Dimensionality Reduction and Mapping Sample Importance	24
2.3.3	Curriculum Learning	25
2.3.4	Summary	26
2.4	Deployment Architectures: Cloud vs. Edge	27
2.4.1	Cloud-Native Foundational Models	27
2.4.2	Edge AI & Resource-Constrained Learning	27
2.4.3	Measuring Power Consumption	28
2.4.4	Summary	28
2.5	Data Usage Dynamics	29
2.5.1	Increasing and Decreasing Data Use	29
2.5.2	Dynamic Data Use	30
2.5.3	Summary	31
2.6	CO ₂ Emissions and Energy Efficiency in AI	32
2.6.1	Green AI	32
2.6.2	Carbon Accounting in AI Workflows	32
2.6.3	Energy-to-Accuracy Trade-offs	33
2.6.4	Benchmarking Sustainability Metrics	33
2.6.5	Summary	34
2.7	Training Stop Methods and Early Stopping Strategies	35
2.7.1	Early Stopping for Automated Hyperparameter Tuning	35
2.7.2	Dynamic Stop Criteria	35
2.7.3	Resource-Aware Termination	36

2.7.4	Summary	36
2.8	Summary and Research Gap	37
3	Maintaining Performance with Less Data: Understanding Useful Data	38
3.1	Methodology	39
3.1.1	Datasets	39
3.1.2	Hardware	40
3.1.3	Performance Metrics	40
3.1.4	Network Architecture	40
3.2	Experimental Results	41
3.2.1	Benchmark	41
3.2.2	Data Step	41
3.2.3	Data Increment	49
3.2.4	Data Cut	51
3.3	Discussion	53
3.3.1	Data Step strategy	53
3.3.2	Data Increment strategy	54
3.3.3	Data Cut strategy	54
3.3.4	Comparing with static data reduction	55
3.4	Conclusion	56
4	Dimensionality Reduction to Dynamically Reduce Data	58
4.1	Methodology	58
4.1.1	Datasets	58
4.1.2	Hardware	59
4.1.3	Performance Metrics	59
4.1.4	Network Architecture	59
4.1.5	Data Selection	60
4.2	Experimental Results	61
4.2.1	Benchmark	61
4.2.2	Static Data Reduction	61
4.2.3	Dynamic Data Reduction	63
4.3	Discussion	65
4.4	Conclusion	65
5	Dynamic Data Reduction with Sliding Window	67
5.1	Methodology	67
5.1.1	Data selection	67
5.1.2	Data application	67
5.1.3	Datasets	69
5.1.4	Hardware	69
5.1.5	Performance Metrics	69
5.1.6	Network Architecture	69
5.2	Experimental Results	70
5.2.1	Benchmark	70
5.2.2	MNIST	71
5.2.3	CIFAR-10	72
5.2.4	smallNORB	74
5.2.5	Imagenette	75
5.2.6	Micro-PCB	77
5.3	Discussion	78
5.3.1	Comparing Sliding Window with Benchmark	78

5.3.2	Comparing Sliding Window with Data Step	79
5.4	Conclusion	82
6	Identifying most useful data in imbalanced time-series datasets	83
6.1	Methodology	83
6.1.1	Dataset Preparation and Fusion	83
6.1.2	Centroid Distance Calculation	85
6.1.3	Data Reduction Strategies	85
6.1.4	Class Density Calculation	87
6.1.5	Metrics and Model	88
6.1.6	Hardware and Power Calculation	89
6.2	Experimental Results	90
6.2.1	Experiments on Individual Sites	90
6.2.2	Experiments on Fused Dataset	104
6.3	Discussion	106
6.3.1	Individual Site Datasets	106
6.3.2	Fused Dataset	107
6.4	Conclusion	108
7	Conclusions and Future Work	109
7.1	Conclusion	109
7.2	Future Work	112

List of Figures

1.1	Thesis structure, providing a visualisation of the scope and progression between thesis the chapters.	17
2.1	An example of the Curriculum Learning process. Data selection chooses which data subset to use, based on either the current training loop or current model performance.	26
3.1	Common neural network model structure with Online Augmentation	38
3.2	Proposed model structure with Dynamic Data Selection	39
3.3	CNN model architecture used in this chapter: a 9 layer CNN with max pooling between every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.	40
3.4	Data Step Experiments starting with 25% data	43
3.5	Data Step Experiments starting with 50% data	45
3.6	Data Step Experiments starting with 75% data	47
3.7	Data Increment experiments, with 33% data increments and 10% data increments	49
3.8	A dataset split into 6 sections (cuts)	51
3.9	Runtime reduction and accuracy loss with Data Cut strategy, on the CIFAR-10 dataset. With more cuts, there is greater runtime reduction, but also greater accuracy loss.	53
4.1	CNN model architecture used in this chapter: 11 layer CNN with max pooling after every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.	59
4.2	Datapoint distribution for a single class of the Micro-PCB dataset, after UMAP dimensionality reduction to 2D. © 2022 IEEE	60
4.3	Dynamic data allocation with a step from 90% data to full dataset use, for Micro-PCB dataset. © 2022 IEEE	64
5.1	Data Usage graphs with Sliding Window method	68
5.2	CNN model architecture used for MNIST, CIFAR-10 and smallNORB datasets: a 9 layer CNN with max pooling between every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.	70
5.3	CNN model architecture used for the Imagenette and Micro-PCB datasets: 11 layer CNN with max pooling after every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.	70
5.4	Experimental results with MNIST dataset Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.	72
5.5	Experimental results with CIFAR-10 dataset. Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.	73

5.6	Experimental results with smallNORB dataset. Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.	75
5.7	Experimental results with Imagenette dataset. Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.	76
5.8	Experimental results with Micro-PCB dataset. Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.	78
5.9	The p -values for all experiments across all datasets.	79
6.1	Dataset fusion procedure. Data from each site are split into train and test sets at a ratio of 80:20. Training sets are concatenated into one datasets, and each test set is tested separately.	85
6.2	Centroid distance reduction strategies	87
6.3	Experimental results for Site Alpha test set.	91
6.4	Experimental results for Site Beta test set.	93
6.5	Experimental results for Site Charlie test set.	95
6.6	Experimental results for Site Delta test set.	97
6.7	Experimental results for Site Epsilon test set.	100
6.8	Experimental results for Site Fazbear test set.	102

List of Tables

2.1	State of the Art Image Classification models with the data augmentation methods used (as of March 2022). Augmentation increases the available data, giving improved performance.	29
3.1	Dataset properties for the datasets used in this chapter.	39
3.2	Hardware used for the experimentation of each dataset.	40
3.3	Benchmark results for MNIST, CIFAR-10 and smallNORB datasets	41
3.4	Group 1 initial data and epoch distribution	42
3.5	Data Step group 1 experimental results. Values in bold indicate best results. . . .	44
3.6	Group 2 Initial data and epoch distribution	44
3.7	Data Step group 2 experimentation results. Values in bold indicate best results. .	46
3.8	Group 3 Initial data and epoch distribution	47
3.9	Data Step group 3 experimentation results. Values in bold indicate best results. .	48
3.10	Data Increment experimentation results. Values in bold indicate best results. . .	50
3.11	Data Cut experimentation results. Values in bold indicate best results.	52
3.12	Comparison of number of image evaluations between experiment E2 of the Data Step strategy and Data Increments of 25%.	54
3.13	Comparison of number of image evaluations between the Data Increments strategy with increments of 0.33% and the Data Cut strategy with 2 splits.	55
3.14	Comparison of model accuracies between static and dynamic data reduction methods, for the MNIST and CIFAR-10 datasets.	56
4.1	Benchmark accuracies and runtimes. © 2022 IEEE	61
4.2	Imagenette dataset results with static data reduction. © 2022 IEEE	62
4.3	Micro-PCB dataset results with static data reduction. © 2022 IEEE	62
4.4	Imagenette dataset results with dynamic data reduction. © 2022 IEEE	64
4.5	Micro-PCB dataset results with dynamic data reduction. © 2022 IEEE	64
5.1	Properties of each dataset used in this chapter.	69
5.2	Benchmark results for each dataset used in this chapter.	71
5.3	Experimental results for the MNIST dataset. The value in bold is the best average accuracy.	71
5.4	Experimental results for CIFAR-10 dataset. The value in bold is the best average accuracy.	73
5.5	Experimental results for smallNORB dataset. The value in bold is the best average accuracy.	74
5.6	Experimental results for Imagenette dataset. The value in bold is the best average accuracy.	76
5.7	Experimental results for Micro-PCB dataset. The value in bold is the best average accuracy.	77
5.8	Model hyperparameters for each dataset for experiments with the Data Step method of chapters 3 and 4.	79

5.9	Model hyperparameters for all dataset for experiments with the Sliding Window method of this chapter.	80
5.10	Comparison of benchmark accuracies and runtimes between chapters 3 and 4 with chapter 5	80
5.11	Top results for each dataset with Data Step methods of chapters 3 and 4	81
5.12	Top results for each dataset with Sliding Window methods of chapter 5	81
6.1	HPDMobile dataset information	84
6.2	Random Forest and XGBoost model hyperparameters.	88
6.3	Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network model hyperparameters.	89
6.4	Preliminary test results on RF, XGBoost, CNN, and LSTM models with HPDMobile dataset Site Alpha.	89
6.5	HPDMobile class balance and class density properties, and maximum reduction amounts after class balancing.	90
6.6	Experimental benchmarks of accuracy and AUC-ROC with RF model.	90
6.7	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Alpha test set	92
6.8	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Alpha test set	92
6.9	Class density of Site Alpha dataset after data reduction on the majority class only	92
6.10	Class density of Site Alpha dataset after data reduction across both classes . . .	92
6.11	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Beta test set	94
6.12	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Beta test set	94
6.13	Class density of Site Beta dataset after data reduction on the majority class only	94
6.14	Class density of Site Beta dataset after each data reduction method, with reduction on both classes	94
6.15	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Charlie test set	96
6.16	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Charlie test set	96
6.17	Class density of Site Charlie dataset after data reduction on the majority class only	96
6.18	Class density of Site Charlie dataset after data reduction across both classes . . .	96
6.19	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Delta test set	98
6.20	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Delta test set	98
6.21	Class density of Site Delta dataset after data reduction on the majority class only	98
6.22	Class density of Site Delta dataset after data reduction across both classes	99
6.23	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Epsilon test set	99
6.24	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Epsilon test set	100
6.25	Class density of Site Epsilon dataset after data reduction on the majority class only	101
6.26	Class density of Site Epsilon dataset after data reduction across both classes . . .	101
6.27	p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Fazbear test set	102
6.28	p -values of the AUC metrics of each experiment of reduction on both classes, on Site Fazbear test se	102
6.29	Class density of Site Fazbear dataset after data reduction on the majority class only	103

6.30	Class density of Site Fazbear dataset after data reduction across both classes . . .	103
6.31	Properties of the fused dataset created in this chapter.	104
6.32	Fused dataset benchmark experimental results.	104
6.33	Properties of the reduced fused dataset created in this chapter.	105
6.34	Reduced fused dataset accuracies	105
6.35	Reduced fused dataset AUC-ROC scores	105
6.36	Class density of reduced fused dataset	105
6.37	Runtimes for experiments on each individual site at each reduction amount for experiments, for both class balancing and without	107

Acronyms

AI Artificial Intelligence. 7–9, 12–15, 17–21

ANOVA Analysis Of Variance. 52

AUC-ROC Area Under the Receiver Operating Characteristic Curve. 81

AUNL Area Under Normalised Loss. 28

CI Carbon Intensity. 25

CI/CD Continuous Integration / Continuous Deployment. 20

CNN Convolutional Neural Network. 14, 26, 33, 52, 62, 81, 82

CO₂e Carbon Dioxide Equivalent. 26

ESG Environmental, Social and Governance. 25

GAN Generative Adversarial Networks. 13

gCO₂eq Carbon Cost Per Inference. 26

HVAC Heating, Ventilation and Air Conditioning. 13

HVC Homogeneous Vector Capsules. 33, 52

IoT Internet of Things. 20

KNN K-Nearest Neighbour. 23, 77

KPCA Kernel PCA. 18

LCA Lifecycle Assessment. 26, 27

LLM Large Language Models. 13, 26, 29

LSTM Long Short-Term Memory. 13, 81, 82

NLP Natural Language Processing. 13, 26

PCA Principle Component Analysis. 18, 105

PCB Printed Circuit Board. 18, 51

PUE Power Usage Effectiveness. 21, 26

RF Random Forest. 13, 14, 16, 77, 81

RNN Recurrent Neural Networks. 13

SNN Spiking Neural Networks. 21

SNR Signal-to-Noise Ratio. 18, 105

SVM Support Vector Machines. 13

t-SNE t-distributed Stochastic Neighbour Embedding. 18, 105

TTC Test-Time Compute. 29

UMAP Uniform Manifold Approximation Projection. 18, 53

VOC Volatile Organic Compounds. 76

Chapter 1

Introduction

Artificial Intelligence (AI) has become one of the defining technologies of modern society, with its adoption continuing to accelerate across a wide range of industries and applications. Advances in machine learning have enabled AI systems to perform increasingly complex tasks with remarkable accuracy. As demand for AI capabilities grows, so too does the computational infrastructure required to develop and deploy these models.

State-of-the-art AI models often require substantial computational resources for both training and inference. The development of these models typically relies on large-scale data centres equipped with specialised hardware. Consequently, training and operating these models incurs significant financial and environmental costs, driven by high energy consumption and the need for extensive computational infrastructure.

While cloud-based AI provides access to powerful computational resources, there is also a need for alternative, locally deployed AI on personal computers, embedded systems, and edge devices. Running AI models locally can reduce latency, improve user privacy, decrease dependence on network connectivity, and lower ongoing operational costs. However, these benefits come at the expense of significantly reduced computational power compared to cloud environments. Given these hardware constraints, optimisation techniques play a critical role in enabling AI models to operate efficiently on resource-limited devices. One avenue for optimisation involves reducing the amount of training data used during model development. Smaller training datasets can substantially decrease training time, reducing storage requirements and lowering computational costs, but indiscriminately removing training data may reduce the diversity and representativeness of the dataset, potentially leading to poorer model generalisation and degraded performance on unseen data. There is, therefore, a fundamental trade-off between computational efficiency and model accuracy. The question this thesis aims to answer is can this optimal trade-off be found with intelligent data selection and application?

This thesis is a series of experimental works on image classification and occupancy detection tasks with three main goals. The first goal is to investigate the dynamic application of data, where training data is provided not all at once, but in gradually increasing amounts, with the aim of maintaining or improving model accuracy. The second goal is to identify which data reduction strategy introduced in these works is best for the tasks of image classification and occupancy detection. The third goal is to find a metric for identifying the suitability of data reduction for a dataset, prior to any training, to minimise training cost.

1.1 Background

Machine Learning has undeniably become a prominent topic across academia, entertainment, and industrial growth [1]. New Artificial Intelligence (AI) models are being developed and optimised at an incredible rate, to improve the performance and capacity of AI tasks. While the design of an AI model is crucial, the foundation of any successful AI system lies in the quality of the data

that powers it. The adage "garbage in, garbage out" aptly describes the critical role of data in machine learning; if the input data is of poor quality or irrelevant to the task, the resulting outputs will be equally flawed [2]. Therefore, ensuring that data is both relevant and of high quality is paramount to achieving accurate and reliable AI outcomes, highlighting the need for rigorous data collection, curation, and preprocessing.

It has been shown that increasing the data available to a model improves average accuracy [3]. Because of this, when creating new models for different tasks, researchers and data scientists alike will supply the model with large amounts of data to train the model, without considering if it is truly necessary. This direction is moving rapidly to a state that is environmentally unsustainable and technically unachievable for those outside of big tech companies [4].

Data collection can be a long and arduous process, due to the sheer volume that is required which can reach several hundred terabytes [5]. Data must be cleaned and prepared for use in AI, which is particularly challenging when the data contains sensitive information, such as with medical or personal data. Identifying the specific data required for AI models would significantly enhance efficiency, ensuring that efforts are not wasted on collecting data that does not contribute to the model's performance. Data analysis plays a crucial role in this context, as it can help identify correlations and patterns within the existing data, thereby guiding us toward the data that will most effectively improve model performance. By leveraging advanced data analysis techniques, the data collection process may be streamlined to focus our resources on acquiring the most relevant and impactful data. This approach not only saves time and effort but also enhances the overall quality and accuracy of AI models.

If it is indeed possible to identify data that is particularly beneficial to training, it logically follows that some data may be less useful or even detrimental to the learning process. Such data may hinder model performance and should be removed from training. However, the relationship between data utility and training efficacy is not necessarily binary. Some data may be more valuable during the early stages of training, when model weights are undergoing rapid transformation, while other data may prove more effective during later stages, when the model is refining and fine-tuning its internal representations.

1.2 Research Aim, Objectives, and Scope

The aim of this thesis is to investigate dynamic data reduction by introducing novel, dynamic methods of data application during training, with the goal of enhancing performance and efficiency in foundational AI models. In order to meet this aim, the objectives of this research are:

- to research current methods of optimal data use and data reduction
- to develop and interrogate methods of dynamic data reduction, through which the effects of varying data use may be observed
- to determine if some data is more beneficial to network training, and which dynamic data reduction strategy is best performing
- to identify metrics to measure suitability for data reduction prior to network training.

While the methods described in this thesis may be applied to any domain, the scope of this research will focus on AI for image classification in chapters 3, 4 and 5. Image classification is a popular AI task with uses in medical, manufacturing and object recognition. It is also the gateway to many other AI disciplines such as object detection and tracking. Time-series data is also explored with the task of occupancy detection in chapter 6. Occupancy detection is used to widen the scope of the hypothesis of this research, allowing the observation of the effects of data reduction techniques in an alternative data domain.

1.3 Contributions to Science

The contributions of this thesis can be summarised as below:

1. Dynamic data reduction is introduced and explored as a means to identify data usefulness at different stages of training. Data is reduced in various quantities and through multiple unique reduction strategies, to observe the effect of varying intensities of reduction. These methods use random data exclusion and do not identify which data to remove from training outright, but how much data may be used to train with for the early training stages for a more optimised and efficient training process with shorter runtimes and a lower energy cost. This work is published in a peer reviewed Springer conference paper, by the thesis author, in [6], and discussed in chapter 3.
2. Dynamic data reduction methods are further developed with the introduction of a distance metric to order training data, instead of using random data reduction. Using dimensionality reduction, the dimensionality of 2D image datasets are reduced into 3 dimensions, which allows us to calculate a centroid distance for each datapoint in Euclidian space. This is used to categorise the data before training, and through dynamic data reduction the data is most useful at different stages of training are identified, while reducing runtime once again. This work is published in a peer reviewed IEEE conference paper, by the thesis author, in [7], and discussed in chapter 4.
3. Dynamic data inclusion methods are fine tuned with the introduction of the sliding data window. This method allows the training data to shift from central-most to lateral-most, based on the previously explored centroid distance metric. This method gives each training loop a slightly different training dataset, allowing the AI model to train on ‘easy’ data initially, and move to ‘harder’ data for fine-tuning. The experimental results show that in a majority of cases, data closest to the centroid is easier to learn, while data further is harder, which correlates with the current state-of-art in image classification. It is also possible to improve experimental results over the benchmark through dynamic data reduction on some datasets. This work is published in a peer reviewed Springer conference paper, by the thesis author, in [8], and discussed in chapter 5.
4. Data reduction is explored in the time-series domain with occupancy detection tasks to identify the compatibility of the reduction methods on a different data domain. Experiments include class density as a metric to observe the applicability of data reduction before any AI models need be performed. This work is published in a peer reviewed MDPI journal paper, by the thesis author, in [9], and discussed in chapter 6.

1.4 Thesis outline and structure

This chapter describes the structure of this thesis. Then, the current landscape of data use for AI, and state-of-art solutions are identified in the literature review. The following chapters discuss the experimental work. Figure 1.1 shows the flow of this thesis.

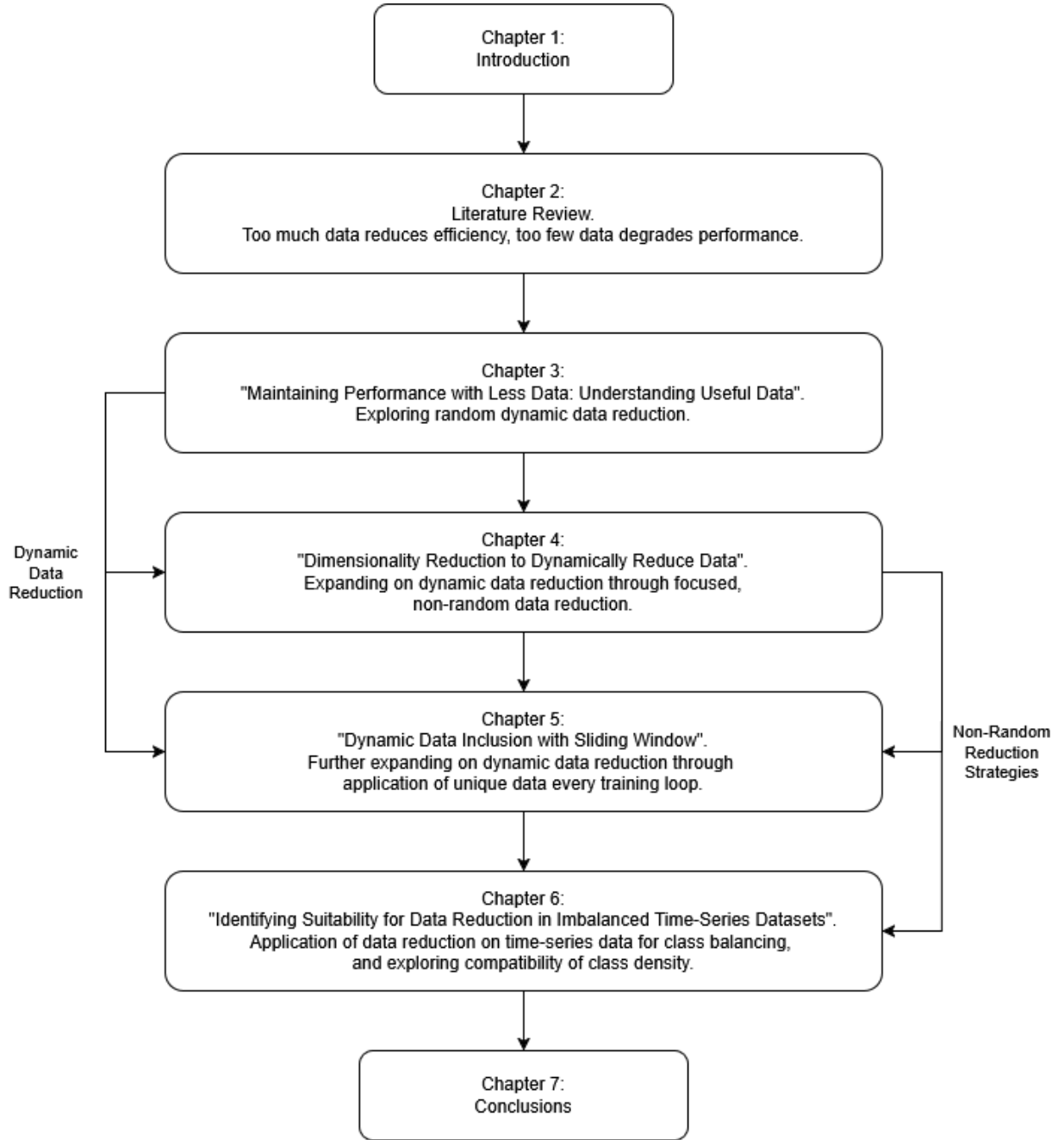


Figure 1.1: Thesis structure, providing a visualisation of the scope and progression between thesis the chapters.

Following this introductory chapter is the literature review, where the topics covered by this thesis will be given context in this chapter by discussing currently used solutions. After this, chapters 3, 4, 5 and 6 show the introduced methodologies and experiments to answer the thesis questions. Conclusions are drawn and given in chapter 7.

1.4.1 List of Publications

- D. Sanderson, B. Malin, T. Kalganova and R. Ott, Dimensionality Reduction to Dynamically Reduce Data, 2022 IEEE Physical Assurance and Inspection of Electronics (PAINE), Huntsville, AL, USA, 2022, pp. 1-5, **Published by IEEE**. <https://doi.org/10.1109/PAINE56030.2022.10014786> © 2022 IEEE. Reprinted with permission.
- Sanderson, D. and Kalganova, T. (2023). Maintaining Performance with Less Data: Understanding Useful Data. In: Yang, X.S., Sherratt, R.S., Dey, N., Joshi, A. (eds) Proceedings of Eighth International Congress on Information and Communication Technology. ICICT 2023. Lecture Notes in Networks and Systems, vol 694. **Published by Springer, Singapore**. https://doi.org/10.1007/978-981-99-3091-3_90
- Sanderson, D., Kalganova, T. (2024). Dynamic Data Inclusion with Sliding Window. In: Nagar, A.K., Jat, D.S., Mishra, D.K., Joshi, A. (eds) Intelligent Sustainable Systems. WorldS4 2023. Lecture Notes in Networks and Systems, vol 817. **Published by Springer, Singapore**. https://doi.org/10.1007/978-981-99-7886-1_44
- Sanderson, D. and Kalganova, T. (2025). Identifying Suitability for Data Reduction in Imbalanced Time-Series Datasets. AI, 6(5), 98. **Published by MDPI**. <https://doi.org/10.3390/ai6050098>

Chapter 2

Literature Review

2.1 Fundamental Paradigms of Data Processing

Data processing may be defined as the conversion of raw data into usable information through data collection preparation, analysis and storage. Data processing allows for the derivation of actionable insights and more informed decision making, if the data is processed effectively. With the current abundance social media, interconnected infrastructures and physical sensors, data collection has become much easier [10]; however, raw data that has not been cleaned or curated is much less useful for drawing accurate conclusions [11]. Data curation is important for preparing the data for use in data modelling. In 2024, a survey on data analytics for modern enterprises showed that robust data cleansing gave a 70% improvement in accuracy, highlighting the importance of robust data processing in not only research environments, but real-world implementations [12]. Unwanted noise can influence the model to give false conclusions, and must be reduced as much as possible, depending on the task and data.

Unique data processing can give improved performance, such as the field-driven processing paradigm proposed by Wang et al. [13], showing that no one general processing method works for all tasks. Therefore, it is important to consider that data processing is an ever growing field with new requirements and solutions being made just as much as new AI models.

Model-centric AI focuses on the development of more complex AI models, deeper networks and new technologies [14]. While model-centric AI has brought about incredible progress in recent years, the increase in computational complexity in order to create generalisable systems means that these models are less effective at highly specific tasks [14]. Also, but large, generalised models are unsustainably expensive to train due to the increased amount of processing inherently required for their size [4]. A different approach from model-centric AI is data-centric AI, where data is probed and optimised to improve AI performance. This thesis discusses how data is commonly used in AI, and how AI training may be manipulated and improved not only by model-centric methods but through data-centric techniques. Through such methods, a user may select any appropriate model for a task and, by optimising data usage, model performance can be maximised at minimal environmental cost, in accordance with the paradigm of ‘Green AI’.

2.1.1 Statistical Models & Machine Learning Foundations

A statistical model may be defined as a set of probability distributions on a sample space to represent relationships within data[15]. A formally defined statistical model may only characterise a single set of data; the same model will not describe another dataset. However, algorithms such as the Decision Tree and Linear Regression exist which statistically model data by adapting to the given data [16]. Such algorithms bridge the gap between statistical modelling and machine learning.

There are three categories of machine learning, depending on the available data: supervised learning, unsupervised learning and reinforcement learning [17]. Supervised learning uses data

that is annotated with labels, and outputs to the algorithm may only be one of these labels. Unsupervised learning uses unlabelled data, and develops knowledge of the data by discovering inherent relationships and patterns within the data. Reinforcement learning gathers data from its surroundings and learns from feedback in the form of rewards or penalties [18]. Which of these three machine learning methods to use depends on the application and the available data, and each have their advantages and disadvantages. For example, with abundantly labelled data, supervised learning is most popular due to its accuracy [19]. Example algorithms include linear regression [20], Support Vector Machines (SVM) [21], Random Forest (RF) [22] and Naive Bayes [23]. Unsupervised learning is useful in data mining and anomaly detection, where raw data is often unclassified. The algorithm is able to organise data without human interaction [24]. Example algorithms include K-Means [25], Autoencoders [26] and Dimensionality Reduction such as Principle Component Analysis [27]. Reinforcement learning is used extensively in Large Language Models (LLM) and robotics, where the algorithm interacts with its surrounding environment and learns over time [28]. When the algorithm, referred to as the ‘agent’, makes a decision based on its environment, referred to as the ‘state’, a good decision earns a reward and a bad decision suffers a penalty. The algorithm aims to achieve the highest reward it can, which requires careful programming of the rewards and penalties to enable the algorithm to learn to give the desired outputs [28].

Some popular algorithms exist which intersect the boundary between statistical model and machine learning algorithm. For example, the decision tree algorithm, which has been the topic of research for over 60 years [29]. Another example is the RF model, which combines multiple decision trees to improve accuracy. RF models use statistical sampling techniques such as data bootstrapping and bagging to draw samples randomly and aggregate results, as well as machine learning techniques such as majority voting [30]. Random forest models are commonly used for tasks such that use time-series data, such as for occupancy detection.

Occupancy detection using AI holds significant potential for optimising Heating, Ventilation and Air Conditioning (HVAC) systems to achieve greater energy efficiency and cost savings [31, 32]. While methods such as camera-based systems can be employed to detect occupancy, these techniques are often seen as intrusive by users, as they may infringe on privacy and personal comfort [33, 34]. To address these concerns, non-intrusive sensing methods are preferred. These alternatives rely on environmental indicators like temperature, humidity, and CO₂ levels within a space to infer human presence. Such non-intrusive techniques offer a viable solution for occupancy detection, achieving notable success while preserving user privacy [33, 35].

2.1.2 Deep Learning & Neural Architectures

Machine learning, while powerful and widely applicable, is limited by its ability to process raw data and required careful programming to achieve acceptable results. Deep learning uses multiple layers of representation learning, which each allow for representations to be discovered in raw data [36]. The raw input is transformed into more abstract levels, allowing for complex functions to be learned.

Recurrent Neural Networks (RNN) are a type of deep learning neural network that show impressive results in Natural Language Processing (NLP) [37]. RNNs are designed to handle sequential data such as sensor outputs and natural language [38]. While the concept of RNNs dates back to the 1970s, they are still widely used, famously as Long Short-Term Memory (LSTM) networks, which have a greater memory capacity and allow for longer input sequences [39].

Generative Adversarial Networks (GAN) have also grown in popularity for their use in generating new content including image and video generation [37]. GANs use two neural networks that compete against each other to generate high quality output; the generator generates the content, and the discriminator aims to determine if an output was generated by the generator or one of the real input data samples. By constantly competing against itself, a GAN is capable of generating very convincing outputs.

Convolutional Neural Network (CNN) have revolutionised the automation of numerous tasks by enabling machines to achieve, and in some cases surpass, human-level performance, particularly in domains such as image classification, object detection, and pattern recognition. Originating from early work in the 1980s, most notably with LeNet [40], CNNs gained widespread recognition following the landmark success of AlexNet in 2012. Since then, as computer hardware has become more openly available, CNNs have become foundational to modern computer vision, driving progress in applications such as medical imaging [41], autonomous vehicles [42], and facial recognition [43]. Functionally, CNNs are trained over multiple iterations to allow their model weights to gradually converge to the required values. Usually, each training loop, (or ‘epoch’), provides the model with the same data, so that the data is steadily learned. The quality of the data used is just as important as the design of the model and, careful attention must be given to the data collected and admitted to the training process.

2.1.3 Computational Complexity

There are many solutions to the same problem, each with their own pros and cons. To give an impression of which solution might be most suitable, the computational complexity of each algorithm can be calculated. Computational complexity is the measurement of how computation time and space requirements are affected by the size of the input data, to gauge the practical difficulty of solving problems [44]. For example, two algorithms, such as sorting algorithms, that achieve the same result through different processes, may have a different number of steps to achieve their results. One algorithm may be more efficient for smaller input data while the other is preferable for larger data. A measurement of computational complexity allows for a standard comparison between methods to enable the best selection.

In the context of AI, some problems require extremely fast processing to achieve real-time responses, such as with computer vision and chatbots. For such systems, algorithms that give the quickest response are necessary, which usually increases the cost in computation and energy usage. As the inference domains expand and input data increases, a once efficient model may become ineffective, calling for careful consideration that the method being used is still the most efficient for the entire model lifecycle [45].

2.1.4 Summary

AI has allowed for great progress in recent years, allowing raw data to be turned into meaningful action. Fundamentally, the data used to train AI is essential to good results. However, as AI progresses, it is becoming more expensive, demanding more data and computational resources. This thesis explores alternative methods of data application as a way of alleviating some of the need for training with such large quantities of data. To explore such alternatives, experimentation with Convolutional Neural Network (CNN) for image classification and Random Forest (RF) for occupancy detection are demonstrated in the proceeding chapters of this thesis.

2.2 Data Manipulation and Transfer Learning

Any method to speed up or otherwise improve training gives benefits to model performance and efficiency. Transfer learning is the task of transferring knowledge across domains, if both the source and destination domains contain some similarities [46, 47]. For example, a successful cat and dog classifier will likely perform better in attempting to classify tigers and wolves than a classifier for trucks and cars. It may also be used to transfer knowledge within the same domain, to reinforce the model when new data becomes available [48]. Foremost, transfer learning is a technique used to reduce the amount of time needed to train a model [48]. For example, a popular use for Transfer Learning is to assist the classification of new data; instead of training a model from the very beginning when new data is received, a model can use previously attained weights to speed training up, cutting out the initial training steps [49]. Transfer learning is particularly useful in situations where sensitive data is required for training, but not available for distribution. In such cases, a pre-trained model can be distributed and expanded on with new data while the original data remains protected. However, an issue that arises when training on new data without including the old data is catastrophic forgetting [50], where the model changes to accommodate the new data, but in doing so becomes unable to classify the old samples. To remedy catastrophic forgetting, some or all of the old samples must be included in the training of the new samples. Alternatively, training a new model on a data subset and then moving onto the full dataset is a type of transfer learning.

2.2.1 Knowledge Transfer & Distillation

Knowledge transfer is a non-AI specific term for passing on knowledge from one entity to another. Knowledge transfer is important for training new entities, which not only applies to humans, but is also important for updating processes to use the newest tools. For example, a new version of an AI model is available but must have the same knowledge as the previous system, knowledge transfer is key. Transfer learning is a common example of knowledge transfer used in AI. Knowledge transfer is the principle of giving knowledge from one model to another, in a similar way to transfer learning.

Knowledge Distillation [51] is a form of knowledge transfer, but specifically from a larger, more cumbersome AI model to a smaller, faster, less resource demanding model. These models are described as the teacher model (the larger model) and the student model (the smaller model). Instead of simply training the student model on the data as normal, the student model is provided the probability distribution of the teacher model, allowing the student model to have more awareness of the teacher’s confidence scores.

2.2.2 Data and dataset Fusion

AI models are most often trained on data specific to the domain they are to operate in. However, data from the real world may be combined to make it more heterogeneous and informative, increasing the reliability of the classification and quality of the extracted information [52]. There are two processes of combining training data: data fusion and dataset fusion. Data fusion refers to the combination of multiple features into a single dataset for used in regression or classification [53]. In the context of occupancy detection, data fusion is often referred to as sensor fusion, as multiple sensor readings are concatenated into one complete dataset. Dataset fusion allows AI models to perform more reliably in new domains by introducing data from multiple sources. By combining datasets, information from other domains becomes available, giving improved performance when tested in new domains. Dataset fusion is a popular method in image classification tasks, as new images are easily resized to match the original data [54]. However, time-series datasets are seldom in the same format as each other, due to the domain-specific information they capture. Formatting disparity is possible even for datasets that aim

to capture the same information; for example, in the task of occupancy detection, occupancy datasets can capture different types of data such as temperature and humidity, image and sound, altitude and location. When attempting to fuse multiple occupancy datasets, there can be issues such as mismatching data formats or missing data, which makes training difficult. Data pre-processing must be performed in order to homogenise datasets prior to fusion. There has been less attention given to fusing completed occupancy datasets together to improve model robustness.

2.2.3 Summary

Knowledge transfer allows models to start training not from scratch, but with some context to the task. Domain context allows more rapid training, reducing training cost. This thesis explores knowledge transfer in a novel way by training a model with a reduced dataset to begin with, and then introducing more data after some time. Dataset fusion is explored with occupancy detection datasets to gain understanding of how that affects performance, and if dataset fusion is useful for the Random Forest (RF) algorithm's ability to generalise across different locations.

2.3 Data Difficulty and Learning Dynamics

Several research directions have aimed to categorise data as more or less difficult for an AI model to learn. Understanding and categorising data difficulty is useful as more difficult data has been shown to be more vital to correct classification. ‘Easy to learn’ data takes fewer training loops for the model to learn how to categorise it, and therefore over the entire training process it does not effect the model much beyond the first few training epochs. On the other hand, ‘hard to learn’ causes greater changes in a model’s weights, as more changes are needed to correctly classify these data samples, and are therefore, more important to train with. The less important ‘easy’ data is less integral to training, and may be removed to reduce training cost [55].

2.3.1 Easy vs. Hard-to-Learn Samples

There have been multiple ways to categorise data as ‘easy’ or ‘hard’ to learn. One definition is to describe easy samples as ones that are correctly classified at early stages of training, while harder samples require more training loops [56]. Research by Toneva et al. [57] introduces a ‘forgetting score’ for each datapoint. Each time a single datapoint goes from being correctly classified to incorrectly classified, its forgetting score increases. Datapoints with a higher forgetting score are harder to learn, as the model is less certain about its classification boundaries. Therefore, it is more important to training, as it is more critical to defining these class boundaries. Less forgettable datapoints are ones that are repeatedly classified correctly, and do not contribute to correct convergence as much as more difficult data. Less forgettable data is therefore a prime candidate for removal, through which time and computational resources can be minimised, increasing overall efficiency.

Data pruning [58] is a technique that reduces an entire dataset by giving each datapoint a ‘parameter influence’, and then removing datapoints with the lowest influence. Through data pruning, not only is data utility optimised but also the computational load of training the model. A similar alternative is to perform preliminary training for a few iterations on a complete dataset to first identify the most impactful features. After these features are identified, only they are used to train for the full duration [59].

Understanding the most useful data in a dataset is important, but it is also important to understand which data are most useful at different stages of training. Usually, training AI models involves several iterations of computation on the full dataset. However, this thesis explores methods of dynamic data reduction. Upon identifying which data are ‘easy’ to learn and which are ‘hard’, by means of parameter influence or the forgetting score, training may be performed with only the easier data for more rapid training, and fine-tuned with harder data once the model parameters have been improved. Alternatively, by systematically removing data and observing the effects on model performance, most useful data may be identified.

2.3.2 Dimensionality Reduction and Mapping Sample Importance

For image classification tasks, the primary goal is to differentiate images between classes. One might attempt to plot each data point in a dataset, so that the borders that separates the classes might be observed. However, the number of dimensions that an image represents cause near uniformity of distance from the classes’ centroid, which is a problem known as the ‘curse of dimensionality’ [60]. Consider the handwritten digit dataset, MNIST; each image is 28 pixels square with a single colour channel, equating to 784 dimensions. Data dimensionality becomes even more of a challenge as the size of data increases, where the number of dimensions ranges in the several hundred thousand.

To counter the effects of high dataset dimensionality, more data points may be used for neural network training. However, collecting more data is difficult for some domains where data is sparse, suffers from class imbalance, or otherwise lacks variation. For example, medical

image datasets are both sparse and suffer from class imbalance [61], and Printed Circuit Board (PCB) datasets have few images per class with little difference between them, due to their highly consistent design.

Dimensionality reduction is a popular technique which transforms data into a lower-dimensional space. Dimensionality reduction aids in data visualisation and addresses challenges associated with the curse of dimensionality, which can impede data grouping and analysis [62]. Principle Component Analysis (PCA) [27], for example, identifies and analyses principle components within a dataset, which may then be used as feature sets for training, replacing raw data [63]. By using only these essential components, models experience a reduction in complexity and operational costs without sacrificing key data insights or performance. Dimensionality reduction is popular in reducing image data [64], but when reducing time-series data it is important to consider the temporal nature of the data, and that feature order must be preserved. PCA has proven to be useful as a preprocessing step before applying machine learning, but assumes linearity in the data. Kernel PCA (KPCA) applies a kernel to the data, allowing it to handle non-linearity, but it has a high computational overhead [65]. An alternative algorithm is Uniform Manifold Approximation Projection (UMAP), a dimensionality reduction algorithm that has been used widely for pre-processing in machine learning tasks [66]. It is applicable to many types of data and has been greatly successful in tasks including image classification [67], medical science [68], and electronics [69, 70]. t-distributed Stochastic Neighbour Embedding (t-SNE) is another popular dimensionality reduction to aid with visualising high-dimensional data [71]. By using UMAP or t-SNE to reduce the dimensionality of a dataset, the distance between each data point may be much more easily quantified, which allows further processing tasks such as data reduction. For example, UMAP allows datapoints to be viewed in as little as 2 or 3 dimensions, which allows data to be represented in Euclidian space, which in turn allows the data to be displayed with conventional graphing techniques. By applying the UMAP algorithm to a dataset, the centroid of each class (where all its datapoints revolve) can be calculated, and data may then be removed based on its distance from its centroid [72]. Surprisingly, by removing data closest to its class's centroid, accuracy is increased, despite fewer training evaluations being performed [72]. On top of bonus accuracy, by training with less data the runtime of the training is reduced, which correlates to less spent on electricity, reducing the carbon footprint of the training process.

There exist alternative distance metrics used in deep learning, such as Signal-to-Noise Ratio (SNR), which are not derived from dimensionality reduction [73]. SNR has been used as a distance metric to aid the classification process in a similar way to the Euclidian distance derived from dimensionality reduction, with the principle idea is that data can be ordered and then reduced to improve training efficiency.

2.3.3 Curriculum Learning

Curriculum learning is the process of training an AI model on easier data samples at earlier stages of training, and increasing the data sample difficulty over time, which shortens training time and may improve model performance [74]. Curriculum learning was first formally introduced by Elman et al. [75] in 1993. It was first used in the field of Machine Learning by Bengio et al. in 2009 [76]. Since then, it has been an often explored method of providing data to an AI model to aid the learning process in multiple domains, including image classification [77], computer vision [78], and large language models [79], generally improving model performance.

As AI model training takes multiple loops, a curriculum learning model can select which data to use based on the current training loop, or by current training progress. Figure 2.1 shows an example of how the curriculum learning process might unfold.

Curriculum learning is the process of providing previously curated and sorted data to an AI model. The quality of the data curation directly affects the performance of the curriculum learning, and therefore requires great attention for optimal results. Data curation may be performed using human annotators [80], or automated by machine; curriculum learning systems

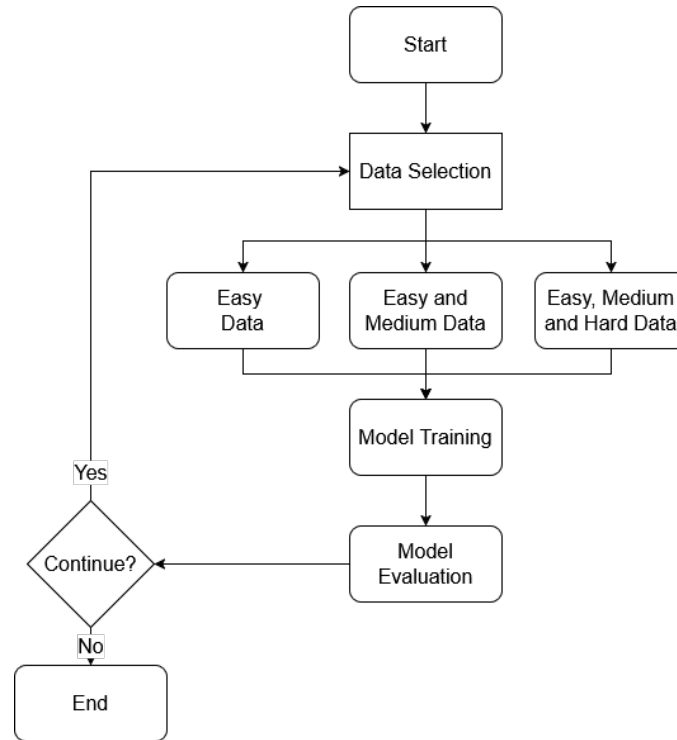


Figure 2.1: An example of the Curriculum Learning process. Data selection chooses which data subset to use, based on either the current training loop or current model performance.

may include their own data difficulty estimators, which judge which data is easy or hard before the training process [74]. The performance of such a system depends strongly on the success of the estimator, as if the estimator poorly categorises the data samples, the AI model may introduce difficult data at earlier stages and easier data later. Alternatively, if the estimator favours samples of particular classes, it will introduce class bias which can hinder performance. In such cases, training time is effectively wasted on these earlier stages, highlighting the importance of high quality data difficulty measurement.

2.3.4 Summary

Some datapoints are easier to learn than others, and some datapoints take longer to learn. Previous works attempt to identify data difficulty prior to training, to give focus to the training process. Identifying and focusing on more difficult data samples, and focusing less on easy to learn samples, can enable a model to perform better in the more difficult test cases while still performing well in easier ones. However, there is a gap in this area of research whereby a dynamic application of data may be applied after data is ordered by difficulty. This thesis explores this gap with dynamic data application.

2.4 Deployment Architectures: Cloud vs. Edge

AI solutions can generally be deployed using one of two approaches: within a cloud-based architecture or directly on-board a device through edge deployment. Each approach offers advantages and disadvantages depending on the application requirements, operational constraints, and available infrastructure [81].

Cloud-based AI deployments remain highly popular due to their scalability, flexibility and relative ease of maintenance [82]. Cloud infrastructures provide access to substantial computational resources, enabling the deployment of larger and more complex AI models that may otherwise be impractical to execute locally. However, as cloud computing grows in popularity, they face an increase in cyber security threats [83, 84].

In contrast, edge deployments require computational processing to occur on or near the data source. Edge deployment is necessary for applications where low latency, real-time decision is a priority [85, 86]. Also, by processing data directly on the device, edge AI systems can reduce reliance on continuous internet access, while also minimising the transmission of potentially sensitive information to external servers where data may be intercepted [86].

2.4.1 Cloud-Native Foundational Models

The application of AI into cloud environments allows rapid deployment and adaptation to trends and technological advancements [87]. Cloud platforms allow systems to scale dynamically in response to increased demand, making them particularly suitable for applications requiring high availability or the processing of large volumes of data [88].

A traditional AI solution is often monolithic in structure, operating as a large, single application. While such applications can be run on the cloud, a truly cloud-native solution must be developed with a unique set of principles which differentiate them from traditional AI applications which encourage scalability, performance and efficiency [89]. A monolithic application scales vertically, requiring greater resources such as memory and computational power to scale a single instance of a program; cloud computing scales horizontally, allowing more, smaller instances to be run, allowing the computational load to be spread out [90]. Containerisation isolates these cloud instances, allowing dynamic resource allocation, contrary to traditional systems [91]. Automation practices such as Continuous Integration / Continuous Deployment (CI/CD) pipelines reduce the need for manual processes, improving operational efficiency and reducing human error [92].

Outside of cloud-native design principles, techniques such as federated learning are well suited to deployment in the cloud. Federated learning is a distributed machine learning method to train a model across multiple decentralised devices, without also decentralising the data [93]. A server does not control the training, but receives information from the decentralised devices to update the model's parameters. Examples of its use include predictive text, advertising, and with Internet of Things (IoT) devices.

2.4.2 Edge AI & Resource-Constrained Learning

Edge computing allows data to be processed close to the data source with little to no data transmission, contrary to a centralised location as is common with cloud computing [94]. An edge computing device has a relatively small size and low power usage making them ideal for deployment in a multitude of environments, and excel at providing real-time inference on data received from its sensors [86]. However, machine learning, and to a greater extent deep learning, are computationally expensive, which makes edge AI challenging due to the comparatively limited computational and memory resources available on edge devices [95]. As such, smaller and more computationally efficient AI models are generally preferred, such as MobileNet for computer vision tasks [96] and the random forest algorithm for processing environmental data.

Spiking Neural Networks (SNN) are gaining interest as an alternative to the familiar artificial neural network model. Unlike traditional artificial neural networks, spiking neural networks transmit information in asynchronous discrete spikes. Activation of a neuron sends a signal to the next, with a signal strength proportional to the weight of the neuron; when the signal reaches a high enough value, it triggers the next neuron [97]. A an asynchronous With the advancements in AI which introduce additional complexities that are gradually outpacing the capabilities of edge computing, the low power, low latency nature of spiking neural networks makes them particularly attractive in edge AI [98].

Cloud AI and edge AI are often considered separate techniques, when in reality they can be used to create a single highly efficient solution. For example, a system that uses distributed computing across multiple edge devices, which each collect unique sensor data, is both a cloud and edge solution [92]. Such a system uses the best of both architectures, utilising efficient practices from cloud-native processing to compliment the low resource edge devices.

2.4.3 Measuring Power Consumption

Power consumption can be measured by treating a computer or datacentre as a black box and measuring the power draw of the entire system. Power Usage Effectiveness (PUE) is a popular metric for quantifying the efficiency of an entire datacentre by comparing the total power use of a facility against the power consumption used for computation. While it is useful to measure the efficiency of power usage in an entire system, it is essential that the efficiency of the individual component be measured [99]. For an AI application, as with any application, the power consumption may be calculated as a sum of its components. For AI specifically, the energy used to train the model, energy used at inference over its lifetime, data storage and transmission are all key components of the application, each with their own associated costs. It can be difficult to identify the power consumption of AI as there can be several overhead costs in conjunction with running an AI model. Some power measurements might consider only the GPU power draw, which accounts for most of the AI's processing, while others include CPU power draw and other application overhead [100]. Another issue in making power estimates is that assumptions about the hardware in use and model parameters may skew calculated power consumption values; while these estimations can be avoided by directly measuring the power draw of the hardware, which is a valid approach for development and small-batch testing, direct measurement becomes impractical when deployed in a larger system such as on the cloud [100].

The energy use for edge AI is generally less than that of cloud-native AI, as edge devices are inherently low power [86]. Muhoza et al. compare the same inference of a deep convolutional neural network performed on a cloud server and on an edge device, showing that energy is reduced by 21% simply by not having to transmit the data to the cloud server [101]. Therefore, data transmission is an important power consumption metric to consider.

2.4.4 Summary

How an AI system is to be deployed has challenges that affect how the system should be developed. For edge AI, a fast-acting model that runs on low-power, low-compute devices is required. Edge AI generally runs cheaper than cloud AI, which corresponds to a lower economical and environmental cost, but at a reduced capacity than a high-power cloud AI model. However, reduced resources on edge means compromised performance. This thesis explores using less data to observe the effect on performance, and if by using data reduction, resources may be freed up for allocation elsewhere. Specifically, in chapter 6, classification on the edge is explored for the task of occupancy detection, to observe how data reduction might improve performance on compute-limited edge devices.

2.5 Data Usage Dynamics

It is well established that a Deep Learning model will have increased performance if provided with more training data [3]. Data collection may be performed by manually gathering data, a process commonly achieved either as an individual or part of a group or company, by using freely available data licenced as open-source, or via methods such as crowdfunding [4, 102]. In whatever way data collection is done, it is both a time consuming and resource expensive process due to the sheer quantity of data required [103]. Another concern when collecting data is analysing data quality, or data cleanliness [104]. Data analysis includes identifying errors in collected data, ensuring appropriate class balance, or removing bias. Examples of errors might be incorrect labelling, or, in the case of image data, blurry images or insufficient lighting. For best performance in deployment, AI models need to be indifferent to conditions such as these to maximise generalisability.

2.5.1 Increasing and Decreasing Data Use

As an alternative to data collection, data augmentation may be used to increase the amount of data available to a model [105]. Instead of collecting more data samples, currently existing data samples may be modified to give new, unique datapoints for training. Data augmentation is often preferable over manual data collection, because annotated data is relatively scarce and can be difficult to obtain, as well as requiring expert knowledge in order to label and validate the data [102, 106]. Simple data augmentation techniques for image datasets include creating copies of existing images and cropping, panning, rotating and blurring them, which improves generalisability and stops overfitting by removing focus from specific features. Such techniques achieve noticeable improvements in a model despite their simplicity. Table 2.1 shows the State of the Art models for image classification across six datasets. Each model uses some form of data augmentation to artificially increase the amount and variety of data available for training, which gives the model the improved accuracy that puts the model at the top of their leaderboards.

Table 2.1: State of the Art Image Classification models with the data augmentation methods used (as of March 2022). Augmentation increases the available data, giving improved performance.

Dataset	No. training images	Model	Top Accuracy	Extra data/ Augmentation used
MNIST	60,000	Homogeneous ensemble with Simple CNN [107]	99.91%	Random rotation/ translation
CIFAR-10	50,000	ViT-H/14 [108]	99.50%	Random horizontal flip, square crop
CIFAR-100	50,000	EffNet-L2 (SAM)[109]	96.08%	Horizontal flip, padding, random crop, AutoAugment [110]
SmallNorb	24,300	Efficient-CapsNet[111]	98.77%	Downsampling, random 32x32 crop, random brightness
FlowerNet-102	1,020	CCT-14/ 7x2 [112]	99.72%	RandAugment [105], AutoAugment [110]
iNaturalist 2018	675,170	MetaFormer [113]	88.70%	RandAugment [105], AutoAugment [110]

Data augmentation may be performed in real-time, known as online augmentation, which

creates unique data each epoch. Alternatively, there is offline augmentation, where the data is augmented only once, before any training [114]. Augmentation is commonly used to combat issues such as overfitting and shortage of data [115], but for datasets with well-balanced data it can also be used to achieve higher levels of generalisability [116].

While there are well established benefits of increasing training data, there are also uses for data reduction techniques. Such practises are less common, as basic linear random data exclusion causes an exponential decrease in system performance; not only is the average accuracy decreased, but the standard deviation of accuracy increases, showing that the training outputs are inconsistent and unstable [117]. Data reduction is uncommon practice in image classification tasks, as very often more data leads to better classification. However, in cases where some data is undesired, it is beneficial to remove it; for example, data trimming is the process of removing extreme values, without compromising the dataset [118]. Some uses of data reduction techniques give an increase to a neural network model’s performance. For example, in domains where labelled data is sparse, for example medical imaging, the data often suffers from class imbalance. Class imbalance can lead to bias, where the model favours a class’s samples due to its larger sample size. To counter class imbalance artificial samples can be added to the smaller (minority) class, or removing samples from the larger (majority) class. These processes are called oversampling and undersampling, respectively [119]. Undersampling techniques may be employed to reduce the amount of data of the majority class, alleviating the effects of class imbalance [120]. Less success has been gained with random exclusion, but methods such as Cluster Centroids [121] and Tomek Links [122] select the most appropriate data to remove; they are often used with oversampling techniques to give an increase in model performance [123]. Undersampling is a technique used primarily to resolve class imbalance and is therefore unsuitable for uniformly balanced datasets, and can be considered a data reduction technique as it reduces the size of the dataset.

For balanced datasets, methods exist for identifying the ‘usefulness’ of data, which allows less useful data to be removed from training. There is a desire to be able to identify data usefulness prior to any training, before any large amounts of computation is performed. The work in [124] aims to identify redundant data in a dataset, which may be removed from training before running any AI models at all. By calculating the ‘class density’, each class is given a score which may be used to quantify how much data in that class may be removed before training. If used in conjunction with class balancing techniques such as undersampling, datasets may be refined to contain only the most relevant data to the training task at hand.

While there is a proven benefit to increasing data use on model performance, there is a cost; the more training samples are included, the more training time is required. Using extra data affects runtime and the computational cost of the model, which directly correlates to electricity use and CO₂ emissions. Also, with more data being used and collected, storage needs are increased, as well as data transfer costs. This thesis aims to investigate using as much data as available to improve model performance, but also to reduce the amount of datapoint evaluations to reduce the cost, to find the middle ground between performance and cost.

2.5.2 Dynamic Data Use

Dynamic data usage is the process of using different data throughout training. While the convention is that an AI model is trained with the same data over multiple iterations to allow the model’s weights and biases to converge to a stable state, recent research has found that modifying the training data on-the-fly can lead to performance benefits.

Data pruning [58] is a popular method to automatically remove disruptive data over the course of training. Data pruning is particularly important in semi-supervised and unsupervised learning, where some data may be mislabelled or particularly hard to learn, and would otherwise disrupt training with poor performance [125]. Data pruning can also aid training by lessening the computation required when there is a lot of data to process, such as in data mining with K-Nearest Neighbour (KNN) algorithms [126]. These methods identify which data may be removed

at runtime, live, as the program receives it.

By using less data, both the training time and computational resources may be reduced. However, data reduction often comes at the cost of classification accuracy; by excluding some data, the model may lose features vital for classification. To benefit from reduced computational resources and avoid the loss in essential features, dynamic data use may be used by selecting a fraction of the data to use for a portion of the training, and increase or decrease the amount of training data as training progresses. Dynamic data use ensures all data points of a dataset are utilised, but some data points of less importance are given less focus during training. The question is, therefore, how the importance of a single data point relative to its peers is determined. This thesis provides multiple methods to create statically defined data subsets, which may be trained on dynamically as training progresses. The methods shown aim to give a better understanding of the effects of data removed at different stages of training.

2.5.3 Summary

It is very common to increase the amount of training data to improve model performance, either by data collection or augmentation of existing data. Collecting more data is the more difficult option in many data domains, and data augmentation is relatively simple to integrate into model training. Especially with raw data collection, sometimes new data samples are misleading to an AI model, and can not only hinder the model's performance but also increase training cost. Removing these data samples through data reduction can remedy this issue, giving a boost to model performance. The experimentation in this thesis uses data reduction to remove data from a portion of the training to increase training efficiency, and uses the full dataset to ensure full data coverage. This methodology is referred to as dynamic data application.

2.6 CO₂ Emissions and Energy Efficiency in AI

With climate change being an important topic and AI currently taking an estimated 1.5% of the world's power, it is important to understand the environmental cost of AI. The emission of greenhouse gases due to human activities has caused a global increase in temperature of 1.5°C [127].

The Carbon Intensity (CI) of electricity describes how much CO₂ is emitted in relation to the electricity generated. Encouraging the use of electricity sources with low carbon intensity, such as hydro-power, geothermal, wind and solar, reduces the long-term environmental impact [128]. Also, not only is the CO₂ emissions important to consider, there are other metrics to consider in the modern application of AI that must not be ignored. For example, data centres and cloud servers consume fresh water as coolant, as well as for energy generation [129]; the Google data centre in The Dalles consumed 29% of the city's total water consumption in 2021 [130]. Also, despite the training of a model being the more energy-intensive operation, over 80% of the energy consumption comes at the inference stage, as a model is repeatedly called upon during inference at the user's request [131].

2.6.1 Green AI

Green AI [132] is a paradigm of incorporating sustainable practices and techniques to AI model design, training, and deployment so as to reduce the associated carbon footprint and environmental cost [133]. Its importance becomes apparent with the rapid expansion of AI which has brought about the substantial growth in the models and data necessary for its operation, which poses sustainability challenges for users [134]. Also, the environmental effects of AI have become an increasing concern, as training new models costs more in electricity and associated CO₂, as well as coolant and refrigeration costs [133]. As costs increase, there is a strong desire to be able to train AI more cheaply, with both more efficient AI models and training data.

Generally, an increase in data can enhance model performance; however, recent research has focused on identifying the most useful data to streamline collection efforts and model training [135]. A targeted approach by identifying most useful data can reduce the resources required for training without sacrificing accuracy or performance.

2.6.2 Carbon Accounting in AI Workflows

The power consumed by an AI application may be measured and converted into CO₂ emissions, which is a popular metric for understanding environmental cost and critical for the mitigation of the effects of climate change [136]. Not only is it useful to reduce the CO₂ cost for environmental benefit, but by reducing CO₂ usage electricity cost is also reduced, which translates to a lower economic cost.

Alternative methods to measuring only the CO₂ emissions include Environmental, Social and Governance (ESG), which also takes into account corporate earnings and the social value of a product, and is regarded as closely tied to green AI principles [137]. As AI continues to evolve, so has the need for accurate ESG reporting [138]. Traditional methods of carbon accounting rely on manual data collection from fragmented data sources, which often suffers from inconsistency and human error [139], causing inaccurate assessments. To resolve the issues of manual carbon accounting, sustainability efforts have benefited from AI-driven approaches to carbon accounting, showing promise in machine learning and natural language processing. Suri et al. report that AI-powered accounting solutions yield 32% fewer mistakes, while reducing data processing time by 70% [140]; however, AI-driven approaches are hindered by concerns over data privacy, implementation cost and algorithmic transparency [141].

2.6.3 Energy-to-Accuracy Trade-offs

Finding the balance between speed, efficiency and accuracy is a constant challenge; high-speed and efficiency is critical for real-time applications, and high accuracy is necessary for tasks such as medical diagnoses and fraud detection [142]. More broadly speaking, more efficient systems have a lower running cost, which is beneficial to all who use them. Therefore, if possible, a cheaper system with a reasonable margin of error is preferable over an expensive and slow system.

Mate et al. compare eight CNN models in terms of accuracy and energy efficiency [143], showing that different models have different running costs for the same task. Results indicate that the highest performing model, ResNet-50, achieves the highest accuracy but at the highest carbon cost. They also demonstrate that when the same model is run on different hardware, using a Raspberry Pi 4 and Nvidia Jetson Nano, the training time and carbon costs are lower for the Jetson Nano, showing the importance of using efficient hardware.

Ariyanti et al. investigate the effects of modifying model hyperparameters during model training on accuracy and energy efficiency. They show that, in the context of image classification with the MNIST dataset, using early stopping to detect when training has peaked to stop training early, can give an energy usage reduction of up to 55%, although with a slight reduction in model accuracy. By reducing the dataset by 60%, energy usage is reduced, also with a reduction in accuracy. Finally, by simply using a larger batch size, energy consumption is reduced with no noticeable loss in accuracy, concluding that there are multiple methods to reduce energy usage with varying effects on model performance [144].

2.6.4 Benchmarking Sustainability Metrics

Datacentres are a large contributor to the global carbon footprint. Gandhi et al. raise awareness to the carbon footprint issue and propose metrics to measure the end to end sustainability footprint of datacentres, arguing that common metrics such as Power Usage Effectiveness (PUE) give an incomplete picture, as it does not consider factors such as the energy source, which may be entirely green [145]. Carbon and water usage are proven to be too coarse to be mapped to an individual inference call, and Carbon Dioxide Equivalent (CO₂e) is proposed as an alternative. These efforts are expanded on by Eilam et al., who introduce Embodied Product Cost for the sustainability measurements of the software used by datacentres, which so far had been neglected [146].

Most sustainability efforts are focused on NLP and LLM tasks, due to their current relative popularity. Sustainability measurements are still very much important for edge AI and computer vision tasks. Mate et al. use Carbon Cost Per Inference (gCO₂eq), computed from the power draw and regional carbon intensity, to compare the efficiency of multiple CNN models for image classification tasks [143].

While more accurate metrics are necessary to understand the sustainability of AI and datacentres, many authors call for benchmarking procedures. Naug et al. introduce SustainDC as a software suite to benchmark datacentres' sustainability, for the task of multi-agent reinforcement learning [147]. Jegham et al. introduce a benchmarking framework to quantify the inference of an LLM, deployed in a datacentre. The methods described decouple the LLM from the rest of the cloud infrastructure, measuring the energy drawn from actively assigned GPUs [148]. Each benchmarking procedure offers different methodologies and metrics, which contradicts the point of benchmarking tools. While efforts to create benchmarks are gaining traction, there are still differing opinions of which metrics are best, which, as AI continues to expand, needs to be determined with increasing urgency.

Lifecycle Assessment (LCA) is a powerful tool for identifying the environmental impact of an application [149]. LCA assesses a product's environmental impact from all stages, starting from the collection of the raw materials used for manufacture, the manufacturing itself, upkeep of the product including electricity and data supply, and finally for disposal [150]. Since its conception

in the 1960s and growth of interest in the 1990s, it has been widely used in conjunction with concerns with the growth of carbon emissions. However, despite its established usefulness, LCA has not been adapted for modern AI until 2025 by Plociennik et al. [151], who highlight that traditional LCA methods are not inherently suited for AI, and require further adaptation.

2.6.5 Summary

Green AI is as important as ever in the current AI landscape. Reducing the carbon footprint on an AI process is important as AI continues to expand for sustainability reasons. As such, this thesis aims to find an ideal trade-off between energy use and model accuracy through relevant data reduction techniques.

In industry environments, there are many metrics that have been developed to understand the sustainability of a process. Despite this, there is still no universal metric that developers can use to understand the environmental impact of AI. For simple processes, recording the runtime and calculating the electricity and CO₂ cost is valid, but does not expand well with scale. For the sake of brevity, as the experimentation in this thesis focuses on training relatively small models, CO₂ is used as the metric to understand environmental impact.

2.7 Training Stop Methods and Early Stopping Strategies

It is difficult to predict in advance when model training will be completed. Training can be performed for a fixed amount of time, which means that even though a model is sufficiently trained, training will still carry on until the prescribed number of training epochs has passed. Alternatively, if too few training epochs are allowed, training can be cut short, causing performance loss [144]. Early stopping is a technique which detects when a model is sufficiently trained and halts training, saving time and computation resources. It is also a well established regularisation method, which can stop potential overfitting of a model [152]. A simple early stopping method, called the ‘knee’ method, involves detecting at each training loop whether a metric, such as training loss, has stagnated. Assuming a model is able fit a dataset, the loss gradually decreases as the data is learned, eventually plateauing once the global minimum is reached. At the plateau point, the early stopping metric is no longer improving, and training is halted. Also, it is possible for the training loss to continue to reduce, while test loss increases, which is a sign of overfitting; by detecting a stagnated loss metric, training can be halted to minimise the effects of overfitting.

Early stopping principles are not limited to model training, and may be applied during model inference. AI-screening tools have been used to read article abstracts, and determine whether the information is relevant to the specific task at hand and discard a paper early if deemed irrelevant [153]. Wang et al. use AI-enhanced early stopping to optimise efficiency in A/B testing frameworks by incorporating dynamic threshold adjustment, which can be used stop testing early and save computational power [154].

2.7.1 Early Stopping for Automated Hyperparameter Tuning

To ensure an AI model performs the best it can, there are several components referred to as hyperparameters that may be modified and tuned for optimal performance, including learning rate, batch size, and the loss function [155]. Identifying the best combination of so many parameters can be a time and energy consuming task, which can be shortened with early stopping to remove unnecessary processing.

AutoML is a hyperparameter tuning technique that involves setting ranges for various hyperparameters and comparing the performance of each combination to identify the best performing model. Training multiple combinations of hyperparameters can take a long time to run, especially if each combination is given the same number of training epochs to train. Long training times may be alleviated by the early stopping of configurations that show little promise at earlier stages, saving time and energy [156].

2.7.2 Dynamic Stop Criteria

Traditional early stopping methods measure the training loss and halt training after a loss value does not improve. If the loss does not improve for a fixed, consecutive number of training epochs, defined by the ‘patience’ value, training is stopped. A static patience value is popular due to its simplicity, but it can be restricting for data that is slow to learn, causing premature stopping.

Dynamic stop criteria, such as a dynamic patience value, can more optimally adapt to the data. Franchini et al. show methodology to dynamically adjust the patience value, which speeds up the process identifying of optimal learning rate and batch size [157]. Podgorelec et al. introduce a dynamic stopping criteria for hyperparameter tuning. A model is trained with different hyperparameter settings, to identify the optimal training settings. Instead of traditional early stopping for each hyperparameter combination, the Area Under Normalised Loss (AUNL) is calculated and used to compare each model with the best performing so far, dynamically [158]. By comparing each hyperparameter combination with the best performing there is no need for a static cut-off, giving each model as much time as it needs to train.

2.7.3 Resource-Aware Termination

Early stopping saves computation and energy, contributing to Green AI principles. Being inherently low-compute, edge devices benefit from early stopping, to minimise cost where possible, even to the point of understanding that a task requires more computation than available and offloading it to more powerful hardware [159]. Federated learning, which compiles a model from training performed on multiple edge devices, can benefit from resource-aware termination by identifying which edge devices contribute the least to a model and halting them. Niu et al. demonstrate how early stopping, in combination with novel methods to select the most significant edge clients, improves efficiency and data safety [160].

Resource awareness is also useful for larger AI models such as Large Language Models (LLM), which do not typically run on edge devices. Understanding the optimal Test-Time Compute (TTC) can allow smaller models to rival the performance of larger models; Amer et al. provide a novel approach to improve LLM performance by performing a check partway through training to identify an optimal Test-Time Compute (TTC) configuration, instead of at the end of training. In doing so, training may be stopped early, reducing the required training compute and improving model performance [161].

2.7.4 Summary

The experimentation performed for this thesis shows the effects of dynamic data reduction on model training. Early stopping was not used for these experiments, to grant consistency between experiments. However, as the preliminary testing has been performed and dynamic data reduction has shown promise, there is now the opportunity to explore early stopping with dynamic data reduction, to improve training efficiency by giving more intelligent dynamic data application.

2.8 Summary and Research Gap

While it is tempting to use the best algorithms and the largest datasets to gain the best performance, it is important to account for the cost of these algorithms, including not only the environmental cost, but hardware requirements and practicality of deployment. Green AI is a paradigm that aims to give awareness to environmental cost of AI, and improving sustainability and AI performance through methods other than more data and more powerful computers. Not only is using more data less sustainable, but too much data can also cause issues such as overfitting, while also increasing training cost. One method to reduce training cost is through the use of a partially trained foundational model which can give a headstart to training, by using knowledge transfer principles. Combining knowledge transfer with an understanding of data complexity can give further boosts; training a model on easier, quick to learn data and transferring this knowledge on to train on harder data is an approach explored in this thesis.

In order to minimise the training and running costs of these AI models, this thesis demonstrates a novel method to apply data to a model in a dynamic, reduced manner. This requires the model to perform fewer data evaluations, but use the full dataset available so as to not discard valid data.

This thesis begins by exploring methods of dynamic data reduction in chapter 3, where dynamic data reduction is introduced in an attempt to minimise computation while still making use of the full dataset. Methods are expanded in chapter 4 with the introduction of data ordering and informed data selection. Dynamic data reduction is further explored in 5, where the novel sliding data window method is introduced. Finally, dynamic data reduction is explored with time-series data in chapter 6, and a metric to identify suitability of dataset reduction is explored.

Chapter 3

Maintaining Performance with Less Data: Understanding Useful Data

When creating a deep learning solution, there are two main factors that determine its success: the model used, and the data used to train with. For the task of image classification, the model, often a convolutional neural network, may be programmed, optimised, and experimented with to reach an acceptable performance, but the data used to train it is fundamentally the determining factor [162]. If inappropriate data is used to train and test a model, the model will perform poorly when deployed in real-world situations and can be regarded as a bottleneck to the system's performance [102]. Therefore, consideration must be given to ensure not only an appropriate quantity of data is available, but also data of suitable quality [102].

Traditionally, a deep learning task has three stages: Data Collection [102], Data Processing [163], [164], and the training of the Network Model. These steps are most often performed sequentially, but are sometimes somewhat interlinked; for example, the Data Collection may be real-time image or text capture, or the model may use Online Augmentation, where data is augmented differently each training loop. Figure 3.1 shows the data flow of an Image Classification model with Online Augmentation. After each training loop, the model returns to the Augmentation stage.

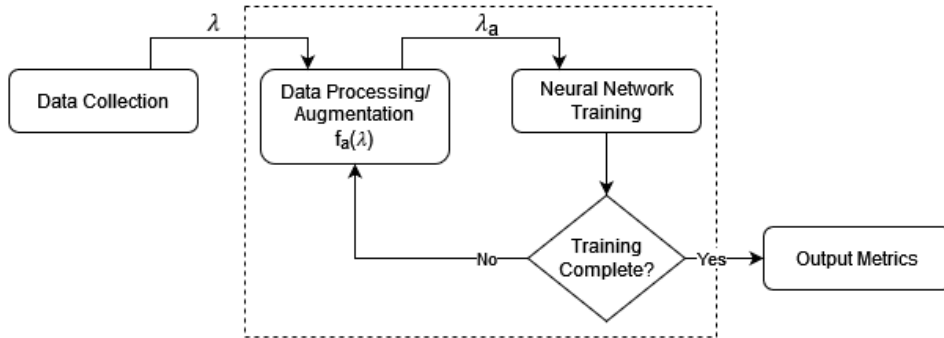


Figure 3.1: Common neural network model structure with Online Augmentation for an image dataset, where λ = initial dataset, λ_a = augmented dataset. $f_a()$ = augmentation function.

This chapter introduces and investigates three sets of experiments where, each epoch, data is selected for training during the data selection stage. The result is a model that uses a reduced dataset, and does not use all data for each epoch. These strategies are **Data Step**, **Data Increment** and **Data Cut**. Figure 3.2 shows the model structure created for this chapter, with

a data selection stage. This allows the model to select and reduce data dynamically, at each loop of training. This in turn enables the various data reduction techniques described in this chapter.

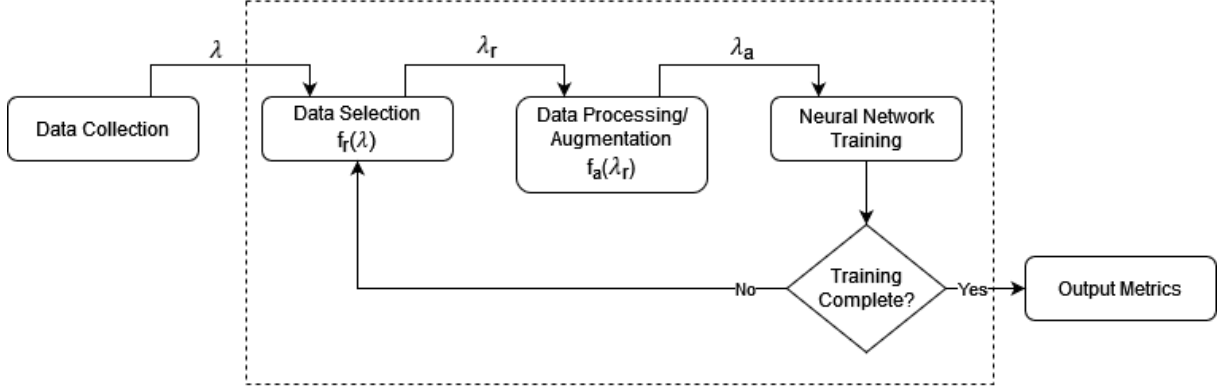


Figure 3.2: Proposed model structure with Dynamic Data Selection and Online Augmentation for an image dataset, where

λ = initial dataset, λ_r = reduced dataset,
 λ_a = augmented dataset.
 $f_r()$ = reduction function,
 $f_a()$ = augmentation function

3.1 Methodology

3.1.1 Datasets

Multiple variations of experiments were performed to observe the effect of varying amounts of data reduction on model performance. As such, datasets with small image sizes are ideal, as they required less time to train allowing rapid experimentation. Using popular and open source datasets allows easy comparison with other image classification models, which makes it possible to compare with the current state of the art.

Three datasets were used for experimentation: the handwritten digits dataset MNIST [165], CIFAR-10 [166], and the model toy dataset smallNORB [167]. Table 3.1 shows the properties of each dataset. MNIST has the most datapoints and smallest image size. CIFAR-10 has slightly fewer images than MNIST, and the images are slightly larger and in colour, as shown by having three colour channels. This requires a small change in the model input pipeline which is standard for using different datasets. The smallNORB dataset has the fewest image datapoints and one colour channel, but the largest image size, which requires a slightly different data input pipeline similarly to CIFAR-10.

Table 3.1: Dataset properties for the datasets used in this chapter.

Dataset	# Training images	# Test images	Image size (pixels)	# Colour channels
MNIST	60,000	10,000	28x28	1
CIFAR-10	50,000	10,000	32x32	3
smallNORB	24,300	24,300	96x96	1

3.1.2 Hardware

Multiple hardware were used to perform training, to make the most of resources available. Training for a dataset was carried out on a single machine, to ensure metrics were consistent across that single dataset. The average CO₂ emission for each GPU is also given, derived from the average power use by the GPU during training and statistics from the Department for Energy Security and Net Zero [168]. Table 3.2 shows the hardware used for experimentation in this chapter, for each dataset.

Table 3.2: Hardware used for the experimentation of each dataset.

Dataset	CPU	GPU	RAM	CO2 emission per hour
MNIST	AMD Ryzen 9 3950X 16-Core	nVidia GeForce RTX 2070 32GB	64GB	31.47g
CIFAR-10	Intel i7 10700	RTX 4000	64GB	35.61g
smallNORB	AMD Ryzen 9 3950X 16-Core	nVidia GeForce RTX 2070 32GB	64GB	31.47g

3.1.3 Performance Metrics

These experiments are to observe the effect of reducing data used each epoch of training. To quantify the observed data reductions, the results of the experiments show the total number of evaluations performed; each evaluation is a single image used for training. The runtime of each experiment is also recorded. The successfulness of the model is quantified with the maximum and average accuracy achieved over five runs. Accuracy is used for comparability with previous state of the art work. The runtime and accuracy for each experiment is directly compared with the baseline for each dataset, to show the increase or decrease in performance in accuracy and runtime. As the model is otherwise unchanged when run with data reduction techniques, the CO₂ reduction of each experiment is directly correlated to the runtime. Therefore, CO₂ reduction is proportional to the runtime reduction.

3.1.4 Network Architecture

The model used for testing is a Convolutional Neural Network (CNN) with a Simple Monolithic Architecture. It uses 9 Convolutional Layers, and a Primary and Secondary Capsule layer. The Capsule layer uses Homogeneous Vector Capsules (HVC), which replace the fully connected layer. Figure 3.3 shows the structure of the CNN used. The model is based on [116], which shows relatively high performance despite its few network layers. Tests were run for 300 epochs with a batch size of 120. Optimisation was performed using the Adam optimiser with an initial learning rate of 0.999, with an exponential decay rate of 0.005 per epoch. These values provide consistent initial settings across all experiments for all datasets.

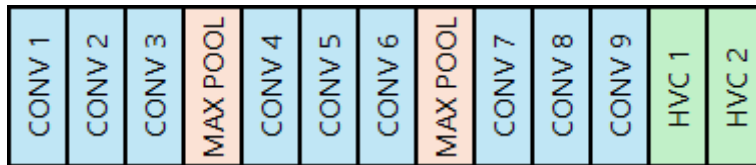


Figure 3.3: CNN model architecture used in this chapter:
a 9 layer CNN with max pooling between every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.

The datapoints to exclude for each of the strategies were selected at random. The experiments were performed to gather an understanding of the effect of the novel dynamic data reduction strategies with the simplest method of data exclusion, namely random data exclusion.

3.2 Experimental Results

Dynamic data inclusion is explored and its performance is measured through experimentation. Data is reduced in various quantities and with multiple strategies to observe the effects of each data inclusion strategy on the neural network performance.

3.2.1 Benchmark

Table 3.3 shows the benchmark runtime and accuracy for MNIST, CIFAR-10 and smallNORB. These results will be used to compare the data reduction experiments, to show the reduction in runtime and effect on performance. Each experiment was run five times to calculate the average accuracy, each for 300 epochs, to ensure convergence and give consistency between datasets, considering that some datasets require more time to fully train. The number of total evaluations is the number of training images evaluated multiplied by the number of training epochs.

Table 3.3: Benchmark results for MNIST, CIFAR-10 and smallNORB datasets

Dataset	Dataset size	#Total Image Evaluations	Runtime (hours)	CO ₂ emission (kg)	Test Accuracy		
					Best	Average	Standard deviation
MNIST	50,000	18,000,000	02:01	0.134	99.719%	99.701%	0.011%
CIFAR-10	49,920	14,976,000	02:54	0.268	88.825%	88.689%	0.096%
smallNORB	24,240	7,272,000	01:27	0.097	93.152%	92.984%	0.135%

3.2.2 Data Step

The simplest strategy of dynamically selecting data is to ‘step up’ the data usage at a given point during training. A fraction of the dataset is used for a given number of training loops, after which, the full, initial dataset is used for training. This splits the training process into two sections; Section one (S_1), which uses less data, and Section two (S_2), which uses the full, initial dataset. Below are definitions for the two sections:

- The number of epochs run in section 1: S_1^E
- The amount of data used in section 1: S_1^D
- The number of epochs run in section 2: S_2^E
- The amount of data used in section 2: S_2^D

With these definitions a formula for the total number of evaluations performed can be derived:

$$n_o \text{ total evaluations} = S_1^D * S_1^E + S_2^D * S_2^E$$

The model will only train with a fraction of the data for the initial part of training, S_1 , and there will be a ‘step up’ in data usage when transitioning to the second part of training, S_2 . Figure 3.4, Figure 3.5 and Figure 3.6 show how the data split is applied between these sections. The hypothesis is that, due to less data being processed in section S_1 , there will be a reduction

in runtime. After the step, the full dataset is used, ensuring that all features are made available at some point in training, although not at every epoch. Using all the data reduces overfitting, which is common when too little data is used. Experimentation for the data step strategy is split into three groups, with three experiments each.

Group 1: Starting with 25% of the dataset

Experiment group 1 consists of three experiments E1, E2 and E3, which start with 25% of the original dataset. Each experiment is made of two sections; the first section, S_1 , uses the reduced dataset, and the second section, S_2 , uses the full dataset. Each of the three experiments has the same percentage of data in the first section, S_1^D . The difference in each experiment is the point at which S_1 switches to S_2 , determined by S_1^E , shown in Table 3.4. The values of S^D vary between datasets as each dataset has a different amount of training images, but the percentage of the dataset used is the same for each dataset.

Table 3.4: Group 1 initial data and epoch distribution.

S_1^D, S_2^D : Data percentage in S_1 & S_2 of training.
 S_1^E, S_2^E : Percentage of training epochs in S_1 & S_2 of training.

Experiment	S_1^D	S_2^D	S_1^E	S_2^E
E1	25%	100%	25%	75%
E2	25%	100%	50%	50%
E3	25%	100%	75%	25%

Figure 3.4 shows graphs of the data use, for the MNIST dataset with 60,000 training images. The total area under the line represents the total number of evaluations.

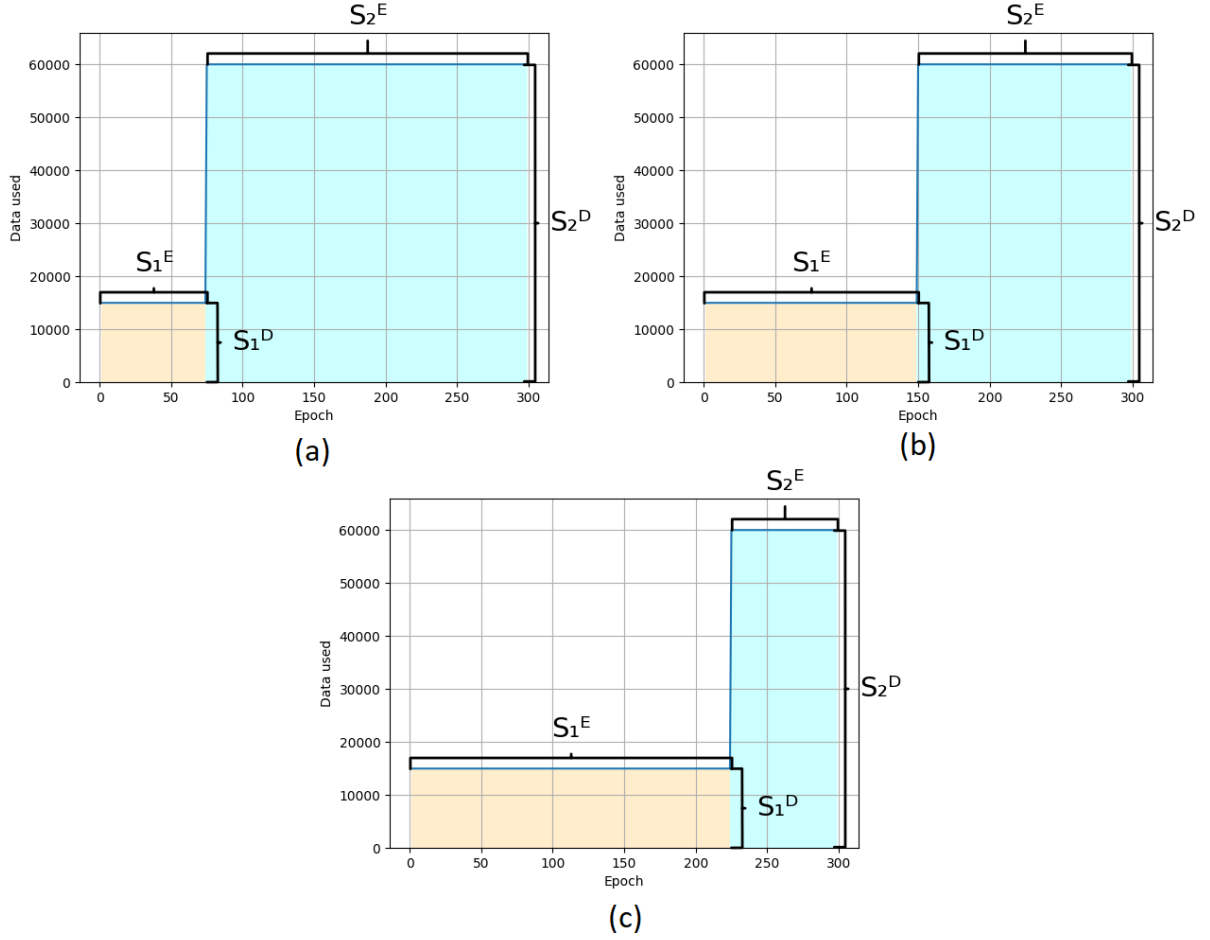


Figure 3.4: Data Step Experiments starting with 25% data.

- (a) Step at 25% epochs through training, (b) Step at 50% epochs through training,
(c) Step at 75% epochs through training.

Table 3.5 shows results of the experimentation. Values in bold show the largest decrease in runtime for each dataset, and the greatest accuracy. The first series of experiments sees the largest decrease in evaluations performed compared to each benchmark, due to the largest amount of data exclusion performed. In particular, experiment E3 has the largest decrease out of all the data step experiments. As such, experiment E3 differs the most from the benchmark. Table 3.5 also shows the percentage differences between the experimental results and their benchmark. Results of the first series of experiments have shown that when fewer evaluations are performed, runtime is reduced which is as expected.

As for the model accuracy, with the MNIST dataset, there has been a slight increase in average and best accuracy: E1, E2 and E3 show an average accuracy **increase** of **0.01%**, **0.024%** and **0.008%** respectively. For the CIFAR-10 dataset, there is a much more noticeable decrease in accuracies. E1, E2 and E3 show average accuracy **decrease** of **0.507%**, **1.141%** and **2.346%** respectively. For the smallNORB dataset, there is also a decrease in accuracy: E1, E2 and E3 show average accuracy **decrease** of **0.043%**, **0.425%** and **0.962%** respectively.

As previous research has found, reducing data from the CIFAR-10 and smallNORB datasets gives poorer accuracy than the benchmark. However, the MNIST dataset shows a slight increase in performance over the benchmark. What is interesting still is that for both MNIST and

Table 3.5: Data Step group 1 experimental results.
 Values in bold indicate best results.

Test type	# Image Evaluations	Runtime (hours)	Test Accuracy			Percentage Difference		
			Best	Average	Standard deviation	Runtime	Best Accuracy	Average Accuracy
MNIST								
Benchmark	18,000,000	02:01	99.719%	99.701%	1.10E-04			
E1	14,678,880	01:43	99.739%	99.711%	2.79E-04	-15.148%	0.020%	0.010%
E2	11,312,880	01:22	99.749%	99.725%	1.82E-04	-32.627%	0.030%	0.024%
E3	7,946,880	00:58	99.749%	99.709%	2.35E-04	-51.742%	0.030%	0.008%
CIFAR-10								
Benchmark	14,976,000	02:54	88.825%	88.689%	9.59E-04			
E1	12,214,320	02:25	88.675%	88.239%	2.78E-03	-16.467%	-0.170%	-0.507%
E2	9,415,320	01:54	87.801%	87.677%	1.49E-03	-34.293%	-1.153%	-1.141%
E3	6,616,320	01:22	86.878%	86.608%	2.55E-03	-52.555%	-2.193%	-2.346%
smallNORB								
Benchmark	7,272,000	01:27	93.152%	92.984%	1.35E-03			
E1	5,931,120	01:15	93.172%	92.944%	2.22E-03	-13.922%	0.022%	-0.043%
E2	4,572,120	01:04	93.003%	92.588%	3.70E-03	-26.674%	-0.159%	-0.425%
E3	3,213,120	00:51	92.628%	92.089%	4.59E-03	-41.191%	-0.562%	-0.962%

smallNORB datasets, the standard deviation of their accuracies is greater than their benchmarks, but for CIFAR-10 the standard deviation is lower. There seems to be no pattern between datasets as each dataset shows unique characteristics.

Group 2: Starting with 50% of the dataset

The next three experiments under the data step strategy follow the same form as in group one, but the initial data S_1^D is 50% of the full dataset. Table 3.6 shows the amounts of data used, and Figure 3.5 shows graphs that demonstrate the data use with the MNIST dataset. This set of experiments is to observe the effect of a lesser amount of data reduction than the previous experiments. Experimental results are shown in Table 3.7.

Table 3.6: Group 2 initial data and epoch distribution.

S_1^D, S_2^D : Data percentage in S_1 & S_2 of training.
 S_1^E, S_2^E : Percentage of training epochs in S_1 & S_2 of training.

Experiment	S_1^D	S_2^D	S_1^E	S_2^E
E4	50%	100%	25%	75%
E5	50%	100%	50%	50%
E6	50%	100%	75%	25%

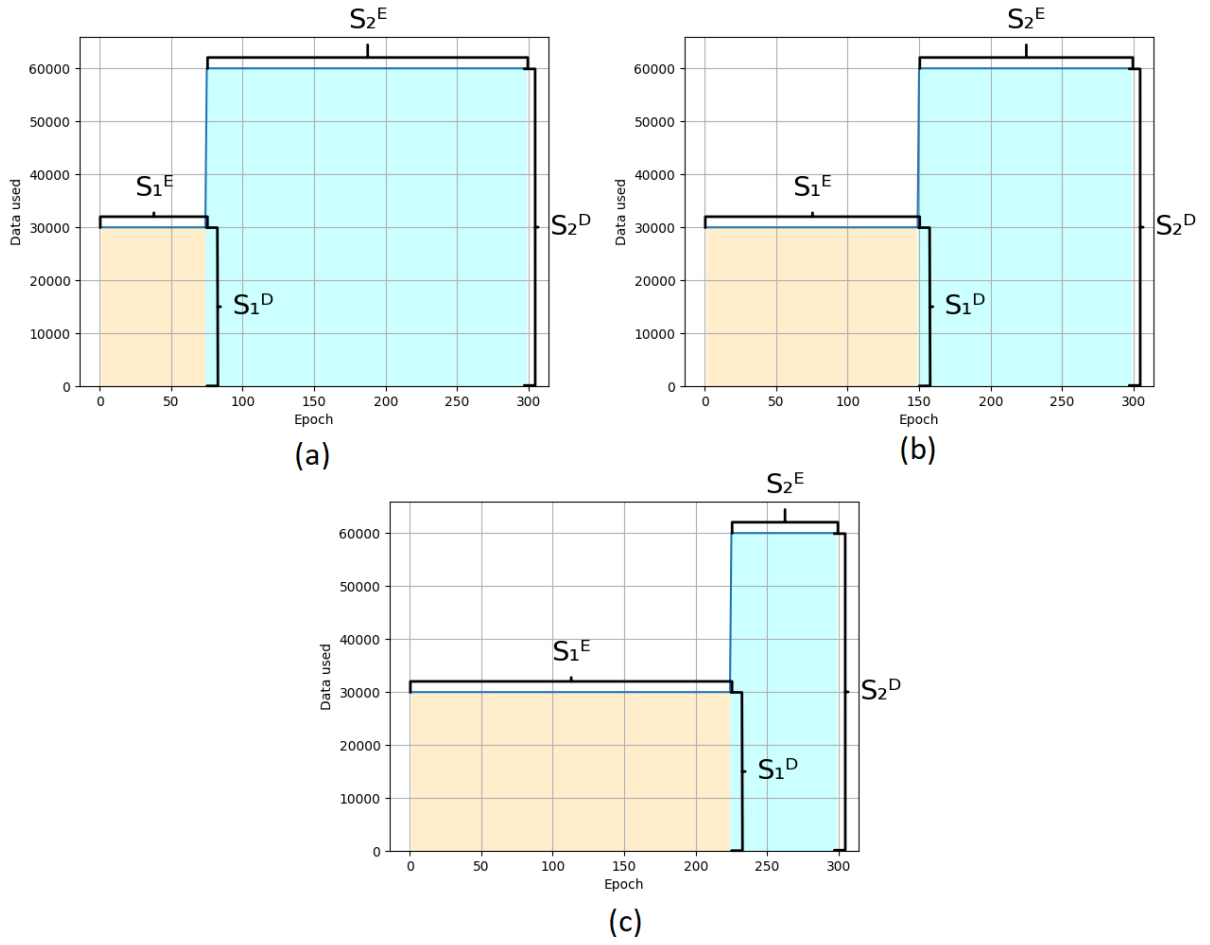


Figure 3.5: Data Step Experiments starting with 50% data.
 (a) Step at 25% epochs through training, (b) Step at 50% epochs through training,
 (c) Step at 75% epochs through training.

Table 3.7: Data Step group 2 experimentation results.
Values in bold indicate best results.

Test type	# Image Evaluations	Runtime (hours)	Test Accuracy			Runtime	Percentage Difference	
			Best	Average	Standard deviation		Best Accuracy	Average Accuracy
MNIST								
Benchmark	18,000,000	02:01	99.719%	99.701%	1.10E-04			
E4	15,788,880	01:53	99.759%	99.715%	2.52E-04	-6.638%	0.040%	0.014%
E5	13,547,880	01:39	99.719%	99.699%	1.59E-04	-18.038%	0.000%	-0.002%
E6	11,306,880	01:24	99.729%	99.695%	2.98E-04	-30.597%	0.010%	-0.006%
CIFAR-10								
Benchmark	14,976,000	02:54	88.825%	88.689%	9.59E-04			
E4	13,137,840	02:35	88.464%	88.283%	1.78E-03	-10.458%	-0.407%	-0.457%
E5	11,274,840	02:14	88.434%	87.998%	3.01E-03	-22.581%	-0.441%	-0.779%
E6	9,411,840	01:55	88.243%	87.711%	3.00E-03	-33.831%	-0.656%	-1.103%
smallNORB								
Benchmark	7,272,000	01:27	93.152%	92.984%	1.35E-03			
E4	6,384,000	01:20	93.564%	92.978%	4.45E-03	-8.475%	0.443%	-0.006%
E5	5,484,000	01:11	93.263%	92.500%	6.19E-03	-18.317%	0.120%	-0.520%
E6	4,584,000	01:04	93.544%	92.905%	4.39E-03	-26.675%	0.421%	-0.084%

The results of the second phase of experiments mirrors those of the previous phase of experiments. With fewer evaluations performed, runtime is reduced; though as more evaluations are performed compared to the previous phase, the runtime is decreased less. The model accuracies show the following patterns: With the MNIST dataset, there is a slight increase in average and best accuracy for all but one experiments: E4 shows an average accuracy **increase** of **0.014%**, while E5 and E6 show average accuracy **decrease** of **0.002%** and **0.006%** respectively. For the CIFAR-10 dataset, there is still a decrease in accuracies across all experiments, but not as steep a decline as in the previous experiments. E4, E5 and E6 show average accuracy **decrease** of **0.457%**, **0.779%** and **1.103%** respectively. For the smallNORB dataset, there is also an accuracy decrease, but less extreme than in the previous phase. E4, E5 and E6 show average accuracy **decrease** of **0.006%**, **0.520%** and **0.084%** respectively. The average accuracy of experiment E5 is also much higher than the others, and also has a higher standard deviation, which shows that the metrics are less stable.

The expectation is that as these experiments use more data than the previous set, the accuracies should be improved as more data is used. However, experiments E5 and E6 for MNIST and E5 for smallNORB show worse accuracies, showing that dynamic data reduction with this data step strategy hinders the accuracy of the model, for these particular datasets.

Group 3: Starting with 75% of the dataset

The last group of three experiments were performed, also with two sections each. Epochs in S_1^E use S_1^D , and epochs in S_1^E use the full dataset S_2^D . In the same manner as the previous two groups of experiments, each of the experiments switch between these sections at a different epoch, shown in Table 3.8. Figure 3.6 shows graphs that demonstrate the data used. The aim of these experiments is to observe the effect of a smaller data reduction than in previous tests. Table 3.9 shows the results of the experimentation.

Table 3.8: Group 2 initial data and epoch distribution.

S_1^D, S_2^D : Data percentage in S_1 & S_2 of training.
 S_1^E, S_2^E : Percentage of training epochs in S_1 & S_2 of training.

Experiment	S_1^D	S_2^D	S_1^E	S_2^E
E7	75%	100%	25%	75%
E8	75%	100%	50%	50%
E9	75%	100%	75%	25%

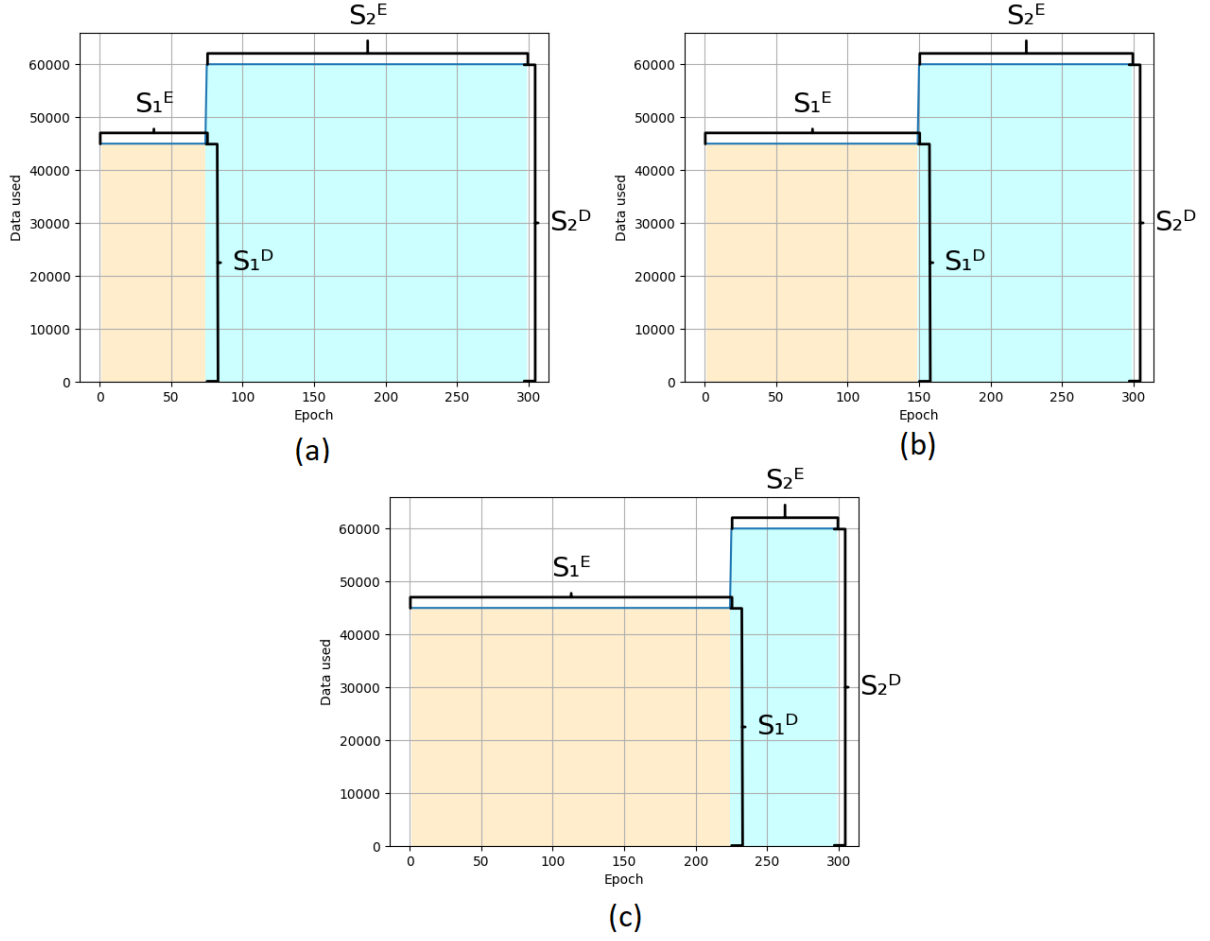


Figure 3.6: Data Step Experiments starting with 75% data.

(a) Step at 25% epochs through training, (b) Step at 50% epochs through training,
 (c) Step at 75% epochs through training.

Table 3.9: Data Step group 3 experimentation results.
 Values in bold indicate best results.

Test type	# Image Evaluations	Runtime (hours)	Test Accuracy		Standard deviation	Runtime	Percentage Difference	
			Best	Average			Best Accuracy	Average Accuracy
MNIST								
Benchmark	18,000,000	02:01	99.719%	99.701%	1.10E-04			
E7	16,898,880	01:55	99.729%	99.719%	1.00E-04	-5.412%	0.010%	0.018%
E8	15,782,880	01:54	99.729%	99.717%	1.31E-04	-5.713%	0.010%	0.016%
E9	14,666,880	01:47	99.729%	99.713%	1.52E-04	-11.948%	0.010%	0.012%
CIFAR-10								
Benchmark	14,976,000	02:54	88.825%	88.689%	9.59E-04			
E7	14,061,360	02:47	88.675%	88.625%	5.59E-04	-4.027%	-0.170%	-0.072%
E8	13,134,360	02:36	88.655%	88.402%	2.98E-03	-9.888%	-0.192%	-0.324%
E9	12,207,360	02:25	88.353%	88.239%	1.14E-03	-16.565%	-0.531%	-0.507%
smallNORB								
Benchmark	7,272,000	01:27	93.152%	92.984%	1.35E-03			
E7	6,828,000	01:23	93.705%	93.059%	5.45E-03	-5.036%	0.593%	0.081%
E8	6,378,000	01:19	93.581%	92.926%	5.19E-03	-9.528%	0.461%	-0.062%
E9	5,928,000	01:16	93.300%	92.751%	3.85E-03	-13.423%	0.159%	-0.250%

The final set of testing uses the least data reduction, and the decrease in both runtime and average accuracy is the smallest. The following patterns are observed for each dataset: With the MNIST dataset there is an increase in accuracy for all experiments: E7, E8 and E9 show an average accuracy **increase** of **0.018%**, **0.016%** and **0.012%** respectively. Experiment E7 shows a decrease in standard deviation of accuracies, compared to the benchmark. With the CIFAR-10 dataset, the results follow the same patterns as with the results of the last groups of experiments - the accuracy decrease directly correlates to the reduction in data used, although this correlation is non-linear. E7, E8 and E9 show an average accuracy **decrease** of **0.072%**, **0.324%** and **0.507%** respectively. With the smallNORB dataset, E7 shows an average accuracy **increase** of **0.081%**, and E8 and E9 show an average accuracy **decrease** of **0.062%** and **0.250%** respectively. There is an increase in performance with experiment E7; this is the only experiment on the smallNORB dataset to have an increase in performance. The average accuracies of experiments E8 and E9 are decreased, which shows how much of an effect such a slight difference in number of evaluations performed has on the model.

3.2.3 Data Increment

Like the methods described in the data step section, the data increment strategy uses a fraction of the data at the start of training. At a previously defined interval, more of the data is added incrementally, causing multiple smaller steps in data usage throughout the training. As such, the number of ‘sections’ present is equal to the number of increments.

Adding the data using the data increment strategy will decrease the time spent training for the earlier epochs, while ensuring that all training data is eventually used. Figure 3.7 shows example data usage graphs with varying increment intervals. Table 3.10 shows the results of the experimentation.

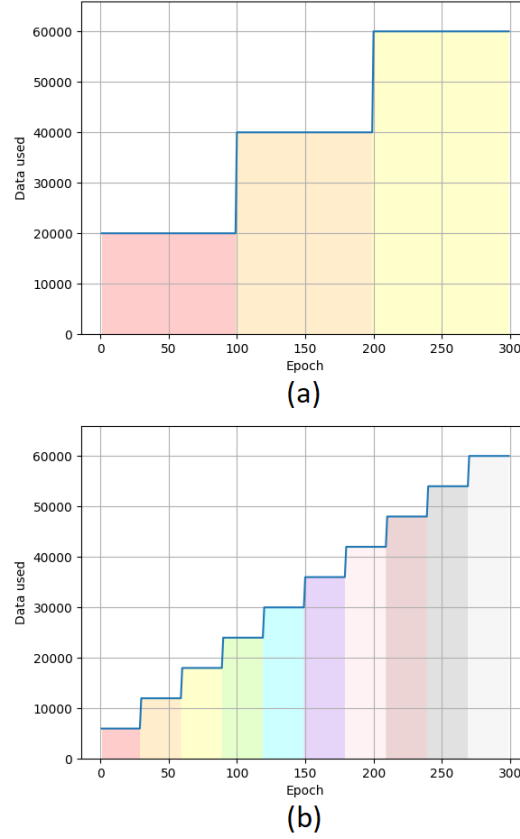


Figure 3.7: Data Increment experiments, with
(a) 33% data increments. (b) 10% data increments.

Table 3.10: Data Increment experimentation results.
Values in bold indicate best results.

Test type	# Image Evaluations	Runtime (hours)	Test Accuracy			Runtime	Percentage Difference	
			Best	Average	Standard deviation		Best Accuracy	Average Accuracy
MNIST								
Benchmark	18,000,000	02:01	99.719%	99.701%	1.10E-04			
33%	11,992,080	01:23	99.739%	99.703%	2.89E-04	-31.790%	0.020%	0.000%
25%	11,241,120	01:20	99.719%	99.699%	1.42E-04	-34.210%	0.000%	0.000%
10%	9,903,720	01:09	99.709%	99.687%	2.50E-04	-42.630%	-0.010%	-0.010%
5%	9,456,720	01:07	99.729%	99.701%	1.93E-04	-44.820%	0.010%	0.000%
1%	9,099,120	01:08	99.709%	99.677%	2.18E-04	-44.090%	-0.010%	-0.020%
0.33%	9,041,880	01:04	99.719%	99.695%	1.96E-04	-47.310%	0.000%	-0.010%
CIFAR-10								
Benchmark	14,976,000	02:54	88.825%	88.689%	9.59E-04			
33%	9,981,360	02:00	88.153%	87.723%	2.69E-03	-30.920%	-0.760%	-1.090%
25%	9,361,560	01:52	87.771%	87.582%	2.18E-03	-35.130%	-1.190%	-1.250%
10%	8,253,000	01:42	87.550%	87.317%	1.52E-03	-41.380%	-1.440%	-1.550%
5%	7,877,520	01:36	87.490%	87.275%	1.92E-03	-44.530%	-1.500%	-1.590%
1%	7,581,360	01:33	87.299%	87.106%	1.25E-03	-46.370%	-1.720%	-1.780%
0.33%	7,531,560	01:33	87.560%	87.225%	2.06E-03	-46.560%	-1.420%	-1.650%
smallNORB								
Benchmark	7,272,000	01:27	93.152%	92.984%	1.35E-03			
33%	4,840,200	01:07	92.954%	92.482%	3.61E-03	-23.040%	-0.210%	-0.540%
25%	4,536,240	01:03	92.628%	92.463%	2.21E-03	-28.180%	-0.560%	-0.560%
10%	3,999,840	01:02	92.768%	92.373%	3.51E-03	-28.850%	-0.410%	-0.660%
5%	3,821,040	00:59	92.735%	92.291%	3.45E-03	-32.180%	-0.450%	-0.740%
1%	3,676,560	00:57	92.562%	92.115%	4.91E-03	-34.520%	-0.630%	-0.930%
0.33%	3,651,600	00:56	92.525%	92.248%	2.39E-03	-35.700%	-0.670%	-0.790%

Using the data increment strategy causes much less data to be used throughout training than the data step strategy. The most extreme case of data reduction is with steps of 0.33%, where roughly half of the benchmark number of evaluations are performed, making the data increment strategy a more extreme data reduction strategy than the data step strategy. For each dataset, the observations are as follows: For the MNIST dataset, average accuracy is within a margin of **0.002%** of the baseline with experiments with 33% and 25% increments. For both of these cases, runtime is shortened by at least 30%. For the CIFAR-10 dataset, all average accuracies are worse by at least **1%**. Like MNIST, runtime is shortened by a minimum of 30%. For the smallNORB dataset, as the dataset has much fewer training images, both the runtime and accuracy reduction are not as large as with other datasets. For all experiments, average accuracy is reduced by less than **1%**, while runtime is reduced by between 20% to 40%.

Based on these results, the data increment strategy seems to have a more detrimental effect on network performance. It is also worth noting that the standard deviation of average accuracies are worse than the benchmark across all experiments, meaning the networks are less reliable for getting consistent results.

3.2.4 Data Cut

The data cut strategy splits a dataset into multiple cuts of equal size. Each cut is then used for an equal number of training epochs. The more cuts a dataset is cut into, the less data is present in each cut. As such, the more splits there are, the fewer evaluations will be performed. Figure 3.8 shows how a dataset may be divided into 6 cuts, for training over 300 epochs.

The data cut approach is most similar to Transfer Learning techniques, in which a network that has already been trained on some data is introduced to a new set of training data. The new input data contains the same classes as it had already been trained with, so the network does not need to adapt to new classes. However, while previous strategies in this chapter utilise the full dataset, the data cut strategy does not include the data used in previous cuts, but continues training on only new data. Normally, when introducing new data via transfer learning, a fresh network is deployed, and given the weights and biases of previously trained networks. The data cut strategy aims to give the network different data at different times, while preserving the state of the network throughout a single training cycle. Table 3.11 shows the results of the experimentation.

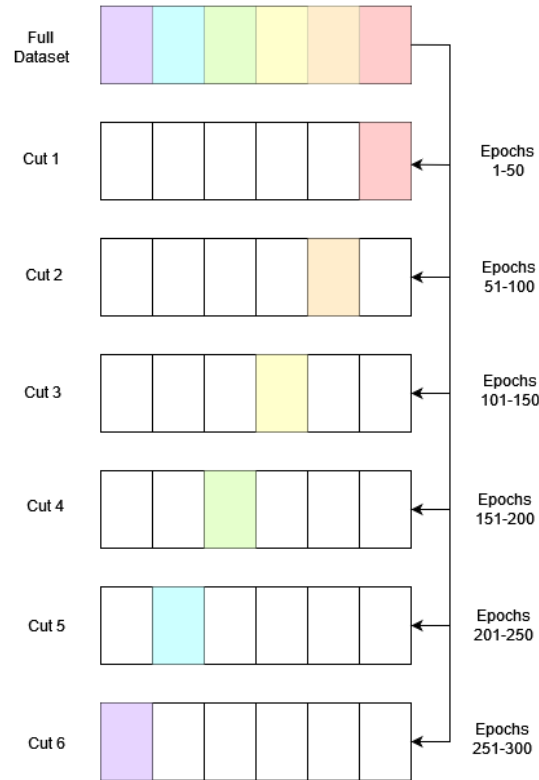


Figure 3.8: A dataset split into 6 sections (cuts).
A single cut is used for training every 50 epochs out of 300.

Table 3.11: Data Cut experimentation results.
Values in bold indicate best results.

Test type	# Image Evaluations	Runtime (hours)	Test Accuracy			Percentage difference		
			Best	Average	Standard deviation	Runtime	Best Accuracy	Average Accuracy
MNIST								
Benchmark	18,000,000	02:01	99.719%	99.701%	1.10E-04			
2 Splits	9,000,000	01:07	99.719%	99.699%	1.23E-04	-44.272%	0.000%	-0.002%
3 Splits	5,976,000	00:44	99.679%	99.655%	1.68E-04	-63.331%	-0.040%	-0.046%
6 Splits	2,988,000	00:25	99.659%	99.596%	4.10E-04	-78.800%	-0.060%	-0.105%
9 Splits	1,980,000	00:19	99.598%	99.526%	6.82E-04	-84.016%	-0.121%	-0.175%
12 Splits	1,476,000	00:16	99.508%	99.337%	1.41E-03	-86.604%	-0.211%	-0.365%
15 Splits	1,188,000	00:14	99.367%	99.102%	2.85E-03	-88.074%	-0.352%	-0.600%
18 Splits	972,000	00:13	99.297%	98.855%	3.86E-03	-89.124%	-0.423%	-0.848%
21 Splits	828,000	00:12	99.237%	97.476%	1.83E-02	-89.909%	-0.483%	-2.232%
24 Splits	720,000	00:11	96.365%	95.645%	7.33E-03	-90.501%	-3.363%	-4.068%
CIFAR-10								
Benchmark	14,976,000	02:54	88.825%	88.689%	9.59E-04			
2 Splits	7,488,000	01:33	86.777%	86.528%	1.54E-03	-46.518%	-2.306%	-2.436%
3 Splits	4,968,000	01:04	85.492%	85.143%	3.20E-03	-62.692%	-3.753%	-3.998%
6 Splits	2,484,000	00:37	82.269%	81.984%	3.27E-03	-78.491%	-7.381%	-7.560%
9 Splits	1,656,000	00:28	80.110%	79.673%	2.65E-03	-83.690%	-9.811%	-10.166%
12 Splits	1,224,000	00:23	77.791%	77.072%	6.68E-03	-86.431%	-12.422%	-13.098%
15 Splits	972,000	00:20	75.552%	72.888%	2.02E-02	-88.013%	-14.943%	-17.816%
18 Splits	828,000	00:19	72.440%	69.066%	2.70E-02	-88.971%	-18.447%	-22.125%
21 Splits	684,000	00:17	67.731%	66.030%	1.94E-02	-89.880%	-23.748%	-25.548%
24 Splits	612,000	00:16	64.518%	60.600%	4.62E-02	-90.370%	-27.365%	-31.671%
smallNORB								
Benchmark	7,272,000	01:27	93.152%	92.984%	1.35E-03			
2 Splits	3,636,000	00:59	93.106%	92.744%	3.37E-03	-32.490%	-0.049%	-0.257%
3 Splits	2,412,000	00:51	92.880%	92.616%	2.36E-03	-41.598%	-0.292%	-0.396%
6 Splits	1,188,000	00:42	92.946%	92.104%	5.30E-03	-52.148%	-0.221%	-0.946%
9 Splits	792,000	00:38	91.840%	91.408%	3.93E-03	-55.659%	-1.408%	-1.694%
12 Splits	576,000	00:36	90.924%	90.012%	7.59E-03	-58.625%	-2.392%	-3.195%
15 Splits	468,000	00:36	89.975%	87.791%	2.33E-02	-58.774%	-3.410%	-5.584%
18 Splits	396,000	00:36	88.461%	87.139%	1.05E-02	-58.445%	-5.035%	-6.286%
21 Splits	324,000	00:36	86.865%	80.511%	8.85E-02	-58.097%	-6.749%	-13.414%
24 Splits	288,000	00:37	86.894%	80.776%	7.48E-02	-57.390%	-6.718%	-13.128%

Of the three experimental strategies described in this chapter, the data cut strategy cuts the number of evaluations performed down the most. It becomes apparent quickly that the more splits are used, the lower the achieved accuracy is. Results on the MNIST dataset show that by cutting the dataset into two splits, the runtime is cut almost in half while maintaining the benchmark accuracy. With 24 splits, there is a large drop in performance, showing that too few evaluations are performed, and the model doesn't have enough iterations to allow the weights and biases to converge, causing more classifications to fail. Figure 3.9 shows the percentage difference in runtime and accuracy of the CIFAR-10 dataset, compared to its benchmark. The results of the CIFAR-10 dataset show the biggest percentage differences. The graph shows that by introducing more data cuts, the runtime decreases non-linearly, while the accuracy decreases

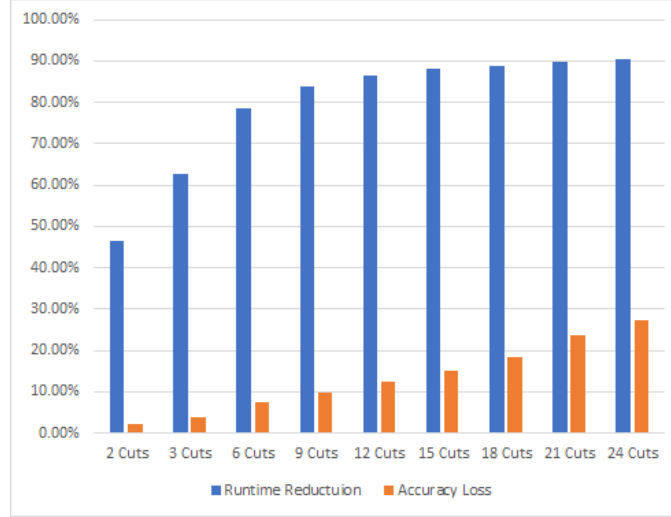


Figure 3.9: Runtime reduction and accuracy loss with Data Cut strategy, on the CIFAR-10 dataset. With more cuts, there is greater runtime reduction, but also greater accuracy loss.

more steadily. Therefore, the most appropriate number of data cuts must be chosen to benefit the most from the data cut reduction technique.

3.3 Discussion

The runtime of the experiment correlates directly with the total number of evaluations performed. Runtime reduction is expected, as the fewer evaluations are performed, the fewer calculations are performed by the neural network. Each dynamic data reduction strategy has a different effect on the model’s performance, discussed for each strategy below.

3.3.1 Data Step strategy

For the data step strategy, the most notable results are those of the MNIST dataset, and E7, E8 and E9 of smallNORB, as they show an increase in accuracy despite the fewer evaluations performed.

For the MNIST dataset, experiments E2, E4 and E7 are the best performing in terms of model accuracy. Experiment E2 has the largest amount of reduction at 37.15% fewer image evaluations, E7 has the least reduction at 6.12%, and E4 is in the middle at 12.29%. One might expect that the more data is reduced, the worse the model accuracy would be; however, the best accuracy achieved is with experiment E4 at **99.759%**, an increase in 0.04%. Experiment E4 has the largest standard deviation of the three experiments at 2.52E-04, which shows it has the lowest stability. It also has the worst average accuracy of the three at **99.715%**. Curiously, the experiment with the best average accuracy is E2 at **99.725%**, which, as previously mentioned, also has the greatest data reduction. These results show that the combination of S_1^E , S_1^D , S_2^E and S_2^D of experiment E2 offers the best combination of model performance and runtime reduction for the MNIST dataset.

For the CIFAR-10 dataset, experiments E1, E4 and E7 are the best performing. Each of these are the experiments with the least data reduction, at 18.44%, 12.27% and 6.11% respectively. Unlike with MNIST, none of these experiments perform better than the benchmark for CIFAR-10 in either best accuracy or average accuracy. Therefore, it is justifiable that these data reduction strategies are inappropriate for the CIFAR-10 dataset for the purpose of increasing model accuracy. However, the strategies may still be useful if a runtime reduction is desired at

a small cost to model accuracy. Experiments E3, E6 and E9 show the most extreme amounts of data reduction with reductions of 55.82%, 37.15% and 18.49% respectively. For these three experiments accuracy losses are, respectively, **1.947%**, **0.582%** and **0.472%**, and give runtime reductions of 52.555%, 33.831% and 16.565%. Depending on the application, if an accuracy loss as great as almost 2% is acceptable, runtime may be reduced by over a half.

For the smallNORB dataset, much like with the CIFAR-10 dataset, the best performing experiments in terms of model accuracy are experiments E1, E4 and E7. Unlike with the CIFAR-10 dataset however, each of these three experiments yield higher accuracies than the benchmark of 93.152%, with E1 achieving **93.172%**, E4 giving **93.564%** and E7 giving **93.705%** accuracy. There is a proportional relationship between the accuracies and the amount of image evaluations performed; the less data is reduced, the greater the maximum accuracy achieved, indicating a delicate balance between removing enough data without removing too much.

3.3.2 Data Increment strategy

Table 3.12: Comparison of number of image evaluations between experiment E2 of the Data Step strategy and Data Increments of 25%.

Dataset	Number of image evaluations	
	Data Step experiment E2	Data Increments of 25%
MNIST	11,312,880	11,241,120
CIFAR-10	9,415,320	9,361,560
smallNORB	4,572,120	4,536,240

As with the data step strategy, the data increment strategy shows that reducing the number of evaluations performed decreases the runtime. However, despite both strategies performing the same fundamental task of data reduction, they produce different results for roughly the same number of evaluations performed. For example, comparing experiments E2 of the data step strategy and data increments of 25%, accuracies can be observed with roughly the same number of evaluations performed. Table 3.12 shows the number of image evaluations of the experiments compared between data step and data increment strategies.

However, despite the marginal difference in number of evaluations performed, the difference in accuracy between these two experiments show that for experiment E2 for MNIST dataset gives an accuracy **increase** of **0.024%**, while data increments of 25% gives an accuracy **difference** of **0%**. Experiment E2 for CIFAR-10 dataset gives an accuracy **decrease** of **1.141%**, while data increments of 25% gives an accuracy **decrease** of 1.250%. Experiment E2 for smallNORB dataset gives an accuracy **decrease** of **0.425%**, while data increments of 25% gives an accuracy **decrease** of 0.560%.

For all cases above, the data increment strategy shows inferior accuracy. While it is true that the data increment strategy has slightly fewer evaluations performed, the difference in data (less than 1% for all datasets) amounts to less than three epochs of training. Three epochs is negligible as the top accuracy of a model settles at much earlier epochs. The results of the experimentation show that decreasing the number of evaluations with the data increment strategy has a more detrimental effect on the accuracy of the model.

3.3.3 Data Cut strategy

Finally, the data cut strategy shows the largest reduction in evaluations performed. In the same way the data step and data increment strategies were compared in section 3.3.2, the data incre-

Table 3.13: Comparison of number of image evaluations between the Data Increments strategy with increments of 0.33% and the Data Cut strategy with 2 splits.

Dataset	Number of image evaluations	
	Data Increments of 0.33%	Data Cut with 2 splits
MNIST	9,041,880	9,000,000
CIFAR-10	7,531,560	7,488,000
smallNORB	3,651,600	3,636,000

ment experimental results can be compared with data cut results that have a similar number of image evaluations performed. Table 3.13 shows the number of image evaluations of the experiments compared between data increment with increments of 0.33% and the data cut strategy with 2 splits.

Comparing the accuracies between these two experiments, these observations are made: The data increment strategy for MNIST gives an accuracy **decrease** of **0.01%**, while data cut with 2 splits gives an accuracy **decrease** of **0.02%**. Data increment for CIFAR-10 gives an accuracy **decrease** of **1.65%**, while data cut with 2 splits gives an accuracy **decrease** of **2.436%**. Data increment for smallNORB gives an accuracy **decrease** of **0.79%**, while data cut with 2 splits gives an accuracy **decrease** of **0.257%**.

Accuracy for all observed data cut experiments are worse than those of the data increment strategy. To conclude, as the data increment strategy is inferior to the data step strategy, the data cut strategy is least effective at maintaining accuracy. However, if the need is to reduce the runtime by as much as possible without hindering accuracy too greatly, a data cut of 9 splits is the best compromise between runtime and accuracy.

3.3.4 Comparing with static data reduction

Experimentation in this chapter has shown that the data cut strategy is the best performing dynamic data reduction strategy. This work was based on that by Byerly et al.[72], who use static data reduction to discover which data is most (and least) useful to training, using the MNIST and CIFAR-10 datasets. Comparisons between these two methods highlight the differences between static and dynamic data reduction methods, and data reduction as a whole.

Table 3.14 compares the average accuracies of experiments performed in this chapter and those by Byerly et al., comparing the best performing dynamic data reduction with the static data reduction with the closest amount of data removed at random. The difference in average accuracy between the benchmark accuracy is given, and used to determine the superior data reduction strategy. The best results are in bold. Due to the difference in data reduction methods, the amount of data reduced between these experimental results do not match, and so the best performing results of dynamic data reduction experimentation are compared with the static data reduction results with the closest amount of reduction.

Table 3.14: Comparison of model accuracies between static and dynamic data reduction methods, for the MNIST and CIFAR-10 datasets.

Reduction strategy	Amount of reduction	Average accuracy	Benchmark	Improvement from benchmark
MNIST				
Static	25%	99.526%	99.716%	-0.190%
Static	50%	99.236%	99.716%	-0.480%
Dynamic	37.2%	99.725%	99.701%	0.024%
CIFAR-10				
Static	5%	89.086%	89.146%	-0.060%
Static	10%	88.838%	89.146%	-0.308%
Dynamic	6.12%	88.625%	88.689%	-0.064%

For MNIST, data step experiment E2 is best performing, with a data reduction of 37.5%. The closest amounts of data reduction performed with static data reduction are 25% and 50%. When comparing the difference in average accuracies against the benchmark accuracies, dynamic data reduction yields improved results over static data reduction, despite more datapoints having been removed. Also, the benchmark results for static data reduction are slightly higher than those obtained in this chapter, likely due to hyperparameter differences in the models used. However, the average accuracy after dynamic data reduction surpasses the higher benchmark accuracy, further demonstrating the benefits of dynamic data reduction.

For CIFAR-10, data step experiment E7 was the best performing, with a data reduction of 6.12%. The closest amounts of static data reduction are 5% and 10%. Unlike with MNIST, the difference in average accuracy and the benchmark accuracy for CIFAR-10 does not surpass static data reduction outright, but is very close to the accuracy with a static data reduction of 5%.

For the MNIST dataset, dynamic data reduction is the superior strategy. However, it is worth noting the amount of data being removed in both static and dynamic data reduction approaches. MNIST has many datapoints, and as such a large amount of data can be removed and performance is improved, it is possible that the dataset has a lot of redundant data, and possibly some data that hinders performance. For the CIFAR-10 dataset, much less data can be removed before performance drops more noticeably. This suggests there is less redundant data, and data reduction, whether static or dynamic, is less suitable for this dataset.

3.4 Conclusion

As deep learning solutions become more popular, their computational complexity increases, giving cause for more intricate algorithms and models which have longer runtimes and require more training data, which causes a greater cost in time, hardware, and environmental resources. This chapter has proposed a novel strategy for training a convolutional neural network for image classification to reduce input data dynamically, in order to reduce the costs of training the model. Dynamic data use is achieved by using less data at the earlier stages of training, which are where the model parameters undergo the most change, and then introduce the rest of the dataset at later stages, allowing the model to be fine-tuned on all the available data.

The results of these experiments have shown that, contrary to the norm, reducing the data used for training has in some cases improved the performance of the model. It proves that not all data may be necessary for training, and in fact some data may hinder it. By using data reduction techniques, the amount of work performed and, therefore, the environmental impact of AI techniques may be reduced. Also, with dynamic data reduction, accuracy may be maintained while reducing runtime by up to 50%, and reducing carbon emission proportionally, and gives a way forward for the better implementation of useful data.

However, the approaches used are somewhat brutish, excluding data randomly without consideration of the value of the datapoints removed. Other works cited have shown algorithmic approaches to select which data to remove to enhance performance, used with static data reduction. The next gap to address is to apply the dynamic data reduction strategies developed in this chapter to data non-randomly, which is addressed in chapter 4. Another consideration to be addressed is to investigate why there is a performance increase for some datasets but not others. Chapter 4 investigates dynamic data reduction for larger image datasets.

Chapter 4

Dimensionality Reduction to Dynamically Reduce Data

In chapter 3, dynamic data reduction was introduced as a new method that removes some data for the initial portion of training to improve training efficiency, and provides the full dataset for the final portion of training to ensure maximum data coverage. To achieve this, data was selected at random to be removed from training for a portion of training, which gave promising results for the MNIST dataset, and achieved similar results to the model trained with static data reduction on the more difficult CIFAR-10 dataset. However, random data reduction gives no consideration to the quality of the remaining data used for the earlier stages of training, and it does not guarantee an even distribution of data is used throughout training. Also, previous work has shown that non-random data reduction yields superior results with static data reduction [72]. The aim of this chapter is to introduce and apply intelligent data selection for the data step strategy described in chapter 3. This chapter also introduces datasets of larger image sizes, to show the effects of the data reduction methods introduced in chapter 3 on more complex data.

4.1 Methodology

4.1.1 Datasets

Two datasets, Imagenette and the Micro-PCB dataset, were used for training. They were chosen in order to evaluate the effectiveness of data reduction techniques on more complex data than those used in chapter 3, as well as due to their unique characteristics, described below.

The first dataset, Imagenette, which consists 9,469 training images, and 3,925 test images. Images were scaled to 299x299x3 to increase training and testing speed. The dataset is split into 10 classes with between 858 and 993 images per class. High accuracy is more difficult to achieve with the Imagenette dataset, compared to MNIST, CIFAR-10 and smallNORB. By using more difficult to learn data, the differences between the benchmark experiments and those performed here will be highlighted.

The second dataset, the Micro-PCB dataset, which consists of 6500 training images, 1625 test images. Images are downsampled to 299x299x3 for training and testing. The data is split across 13 classes with even class distribution. Each Printed Circuit Board (PCB) image has been photographed from 25 different perspectives, and 5 unique rotations. More orientations gives images in each class much more variation, as images of PCBs from a single perspective would offer very little variation. The Micro-PCB dataset was selected due to its highly structured nature, which may make identifying which data is more useful a more intuitive process.

4.1.2 Hardware

Experimentation on both datasets was performed on the same hardware, for consistency. The CO₂ usage was computed from the average power use by the GPU during training and statistics from the Department for Energy Security and Net Zero [168].

- CPU: Intel i7 10700
- GPU: RTX A4000
- RAM: 64GB
- CO₂ emission per hour: 35.61g

4.1.3 Performance Metrics

The experiments were performed to observe the effect of reducing data used each epoch of training. Accuracy is used for comparability with previous state of the art work. To quantify the amount of data reduction used, the experimentation results provide the total number of evaluations performed, where an evaluation is a single image used for training. The runtime of each experiment is also recorded, to show how the data reduction method used affects not only the accuracy of the network, but the runtime. The runtime and accuracy for each experiment is directly compared with the baseline for each dataset, to show the increase or decrease in performance in accuracy and runtime.

The p -values of each experiment are also calculated to show the statistical significance of each resulting accuracy. The closer the p -value is to 0, the greater the statistical significance. The p -values were calculated based on Analysis Of Variance (ANOVA), using the accuracy results of the baseline experiment (where the full dataset is used, with no reduction) and each experiment with data reduction, individually.

As the model is otherwise unchanged when run with data reduction techniques, the CO₂ reduction of each experiment is directly correlated to the runtime. Therefore, CO₂ reduction is proportional to the runtime reduction.

4.1.4 Network Architecture

The model used for testing is a Convolutional Neural Network (CNN) with a Simple Monolithic Architecture. It uses 11 convolutional Layers, and a primary and secondary capsule layer. The capsule layer uses Homogeneous Vector Capsules (HVC), which replace the fully connected layer, based on the model used in [72]. This model shows relatively high performance despite its few network layers. Figure 4.1 shows the structure of the CNN used.

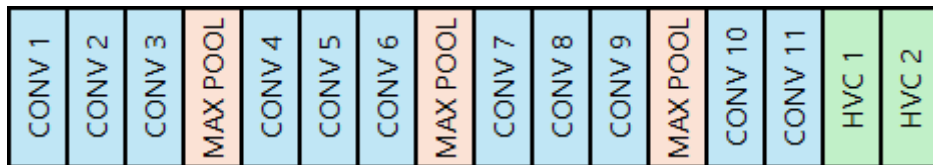


Figure 4.1: CNN model architecture used in this chapter:

11 layer CNN with max pooling after every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.

Tests were run for 250 epochs, which was enough epochs to allow complete training as rapidly as possible, which was necessary for the amount of experiments that were run. A batch size of 120 was used, and optimisation was performed using the Adam optimiser with an initial learning

rate of 0.98, and an exponential decay rate of 0.001 per epoch. These values provide consistent initial settings across all experiments for all datasets.

Online data augmentation was used during the training. The augmentation used was uniform, and the same methods were used as those performed in experiments in [72].

4.1.5 Data Selection

The main contribution of this chapter is to order image data by a universal metric which will allow for more structured data reduction than the random reduction used in chapter 3. Uniform Manifold Approximation Projection (UMAP), as used by Byerly et al. [72], allows for dimensionality reduction, through which the data points may be represented in Euclidian space. Dimensionality reduction allows each data point to be observable in relation to every other data point in the dataset, and thus ordered by distance to the classes' centroid. After ordering the dataset, data may be procedurally excluded; data closest to its classes' centroid is the 'central' data, and data furthest from its classes' centroid is the 'lateral' data.

Algorithm 1 shows the process of calculating the centroid distance for each datapoint for c classes.

Algorithm 1 Centroid distance calculation for c classes

```

1: for Each Dataset do
2:   Split Data into  $c$  classes
3:   for Each Feature in Each Class do
4:     Feature Centroids = Calculate Mean Average of Each Feature
5:     for Each Datapoint in Dataset do
6:       Feature Distances = Difference Between Feature And Feature Centroid
7:       Centroid Distance = Linear Normalisation of all Feature Distances
8:     end for
9:   end for
10: end for

```

Figure 4.2 shows the distribution of datapoints (shown in red), as well as the centroid of the class (shown in black), for a single class of the Micro-PCB dataset. By using UMAP, each datapoint can be observed by relative distance from each other datapoint.

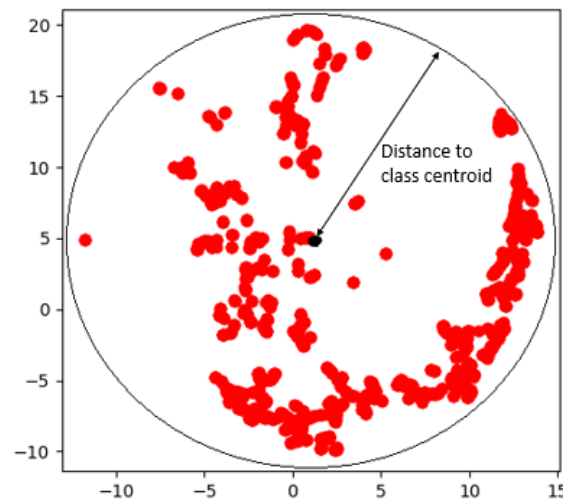


Figure 4.2: Datapoint distribution for a single class of the Micro-PCB dataset, after UMAP dimensionality reduction to 2D. © 2022 IEEE

The centroid distance is calculated and used in experiments to show the effect of exclusion of central and lateral data. Previous work has shown that some datasets perform better with data close to the centroid, and others with data further, which is investigated in this chapter. Experiments performed exclude 1%, 5% and 10% of the data.

Static Data Exclusion is defined as data exclusion that is the same throughout training. Before training is started, the data to be used for training is selected through the data selection process, and throughout each loop of training the same reduced dataset is used.

Dynamic Data Exclusion is defined as data exclusion that changes in some way during training. For the experiments performed in this paper, the Data Step method introduced in chapter 3 is used to split the training process into two parts; an initial stage in which a reduced dataset is used to train the model, and a second stage in which the full dataset is used.

The experimentation described in this chapter investigate data reduction by means of ‘central’ and ‘lateral’ data exclusion, using static data reduction. Dynamic data reduction with central and lateral data exclusion is also investigated based on methods introduced in chapter 3. While other works focus on adaptive pruning, whereby data is removed at intervals during training depending on its performance [169], this chapter defines the data reduction process prior to the start of training. The reduced dataset is used for the first half of training, and then the full, unreduced, dataset is used for the remainder of training.

4.2 Experimental Results

Dynamic data reduction methods are further investigated in this section. Based on the methods used in chapter 3, the data step method is applied to the Imagenette and Micro-PCB datasets. However, first, these datasets are ordered by centroid distance, so that data selected for reduction may be non-random. Ordering the data will allow more consistent measurements, and will provide insight into which data is more important to model training.

4.2.1 Benchmark

Table 4.1 shows the benchmark runtimes and accuracies for both datasets used. These values will be compared against the results of all experiments to give a reference to the change in runtime and accuracy of each data reduction technique.

Table 4.1: Benchmark accuracies and runtimes. © 2022 IEEE

Dataset	# Image Evaluations	Runtime (hours)	Test Accuracy	
			Average	Standard Deviation
Imagenette [170]	2,340,000	3:15	90.69%	4.05E-03
Micro-PCB [171]	1,620,000	1:55	99.96%	5.73E-04

4.2.2 Static Data Reduction

A total of six experiments for each dataset are performed to show the effect of static data reduction. Table 4.2 shows the results of the Imagenette dataset and table 4.3 shows the experimental results of the Micro-PCB dataset.

The runtimes of all experiments are reduced, as fewer evaluations being performed due to the data reduction. The amount of runtime reduction correlates proportionally to the amount of data exclusion used.

Table 4.2: Imagenette dataset results with static data reduction. © 2022 IEEE

Exclusion Type	# Image Evaluations	Data exclusion	Runtime (hours)	Test Accuracy		Percentage difference	
				Average	<i>p</i> -value	Runtime	Average Accuracy
None	2,340,000	-	3:19	90.686%	-	-	-
Lateral	2,310,000	1.00%	3:18	90.347%	0.3394	-0.962%	-0.373%
Central	2,310,000	1.00%	3:18	89.826%	0.0660	-0.996%	-0.948%
Lateral	2,220,000	5.00%	3:12	89.757%	0.0269	-3.548%	-1.024%
Central	2,220,000	5.00%	3:14	89.861%	0.0386	-2.989%	-0.909%
Lateral	2,100,000	10.00%	3:03	88.967%	0.0147	-8.034%	-1.895%
Central	2,100,000	10.00%	3:08	89.852%	0.0408	-5.910%	-0.919%

Table 4.3: Micro-PCB dataset results with static data reduction. © 2022 IEEE

Exclusion Type	# Image Evaluations	Data exclusion	Runtime (hours)	Test Accuracy		Percentage difference	
				Average	<i>p</i> -value	Runtime	Average Accuracy
None	1,620,000	-	1:55	99.962%	-	-	-
Lateral	1,590,000	1.00%	1:50	99.833%	0.1657	-4.280%	-0.128%
Central	1,590,000	1.00%	1:46	99.897%	0.3015	-8.422%	-0.064%
Lateral	1,530,000	5.00%	1:42	99.910%	0.4284	-11.781%	-0.051%
Central	1,530,000	5.00%	1:45	99.731%	0.0712	-9.024%	-0.231%
Lateral	1,440,000	10.00%	1:36	99.821%	0.1300	-16.654%	-0.141%
Central	1,440,000	10.00%	1:36	99.679%	0.0210	-16.903%	-0.282%

The average accuracies of the Imagenette experiments show a pattern of lateral exclusion being outperformed by central exclusion. Firstly, all experiments, except lateral exclusion of 1%, have a p -value of less than 0.1. These low values indicate high statistical significance, which suggests a real difference in model performance. A large p -value would indicate low statistical significance, meaning any difference in model performance is likely down to random chance.

The accuracy reductions are **1.024%** with lateral exclusion of 5%, **0.909%** with central exclusion of 5%, **0.909%** with lateral exclusion of 10%, and **0.919%** with central exclusion of 10%. The accuracy reductions of lateral exclusion experiments are worse than those of central exclusion, which correlates to work by Byerly et al. [72], where data far from the classes' centroid appears to hold more important information for classifying an image.

The percentage differences of the experiments shows a high amount of variance between lateral and central exclusion, which is an unexpected result. The expected behaviour is that the runtimes would be more similar, as the same amount of reduction is performed, and therefore the same amount of image evaluations. For example, with the Micro-PCB dataset, lateral exclusion of 1% of the dataset gives a runtime reduction of 4.280%, while central exclusion gives a runtime reduction of 8.422%, which equates to 4 minutes. Such a relatively small time could be the result of change in conditions that the computer was run under, such as increase in room temperature.

The results of the experimentation also show a decrease in average accuracy across all experiments. The observed accuracy drop is also proportional to the amount of data excluded. However, there is a moderate difference in accuracy between central and lateral exclusion. For example, with the Micro-PCB dataset and data exclusion of 10%, lateral exclusion gives a decrease of 0.141% and central exclusion gives a decrease in 0.282%. The p -value of these experiments must also be considered; lateral exclusion has a p -value of 0.13, and central exclusion has 0.02. As it is more than 0.05, the values of lateral exclusion are shown to be less statistically significant than those of central exclusion, more prone to random change and less reliable. Statistical significance may be improved by modifying the model to exclude operations that use randomness, such as those in data augmentation.

The results of the Imagenette datasets show similar patterns. The maximum reduction in runtime is 8.034%, which is half of the maximum runtime reduction with the Micro-PCB dataset. The difference in runtimes can be explained by considering that the image data of the Imagenette dataset is more complex than those of the Micro-PCB dataset, and as such, evaluating a single image takes more processing time.

4.2.3 Dynamic Data Reduction

Based on methods introduced in chapter 3, the training process is split into two sections. The first section uses a reduced dataset, and the second section uses the complete dataset. Figure 4.3 shows an example of the data step for the Micro-PCB dataset, where:

- $S_1^E = 125$ epochs,
- $S_2^E = 125$ epochs,
- $S_1^D = 90\%$ of the dataset,
- $S_2^D = 100\%$ of the dataset.

Table 4.5 shows the experimental results on the Micro-PCB dataset, and table 4.4 shows the results on the Imagenette dataset.

The first notable result is that p -values are highly variable. The closer the p -value is to 1, the less statistically significant it should be considered as. While non-statistically significant results may seem unfavourable, it tells us that the results are less distinguishable from the benchmark accuracies. As the aim of this chapter is to reduce the data and maintain performance, this is in fact ideal.

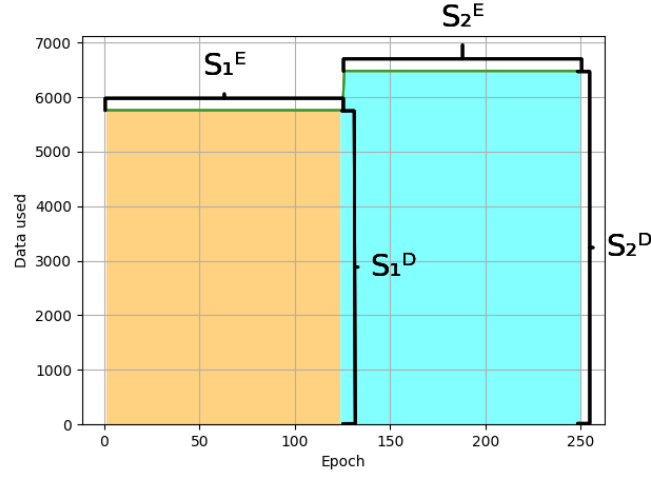


Figure 4.3: Dynamic data allocation with a step from 90% data to full dataset use, for Micro-PCB dataset. © 2022 IEEE

Table 4.4: Imagenette dataset results with dynamic data reduction. © 2022 IEEE

Exclusion Type	# Image Evaluations	Data exclusion	Runtime (hours)	Test Accuracy		Percentage difference	
				Average	<i>p</i> -value	Runtime	Average Accuracy
None	2,340,000	-	3:19	90.686%	-	-	-
Lateral	2,325,000	1.00%	3:20	90.217%	0.1516	0.041%	-0.517%
Central	2,325,000	1.00%	3:20	90.556%	0.7184	0.093%	-0.144%
Lateral	2,280,000	5.00%	3:16	90.399%	0.3929	-1.744%	-0.316%
Central	2,280,000	5.00%	3:11	90.122%	0.1394	-4.176%	-0.622%
Lateral	2,220,000	10.00%	3:12	90.182%	0.4018	-3.609%	-0.555%
Central	2,220,000	10.00%	3:12	90.512%	0.7091	-3.607%	-0.191%

Table 4.5: Micro-PCB dataset results with dynamic data reduction. © 2022 IEEE

Exclusion Type	# Image Evaluations	Data exclusion	Runtime (hours)	Test Accuracy		Percentage difference	
				Average	<i>p</i> -value	Runtime	Average Accuracy
None	1,620,000	-	1:55	99.962%	-	-	-
Lateral	1,605,000	1.00%	1:53	99.974%	0.7328	-2.150%	0.013%
Central	1,605,000	1.00%	1:51	99.897%	0.3015	-3.381%	-0.064%
Lateral	1,575,000	5.00%	1:53	99.974%	0.6810	-2.365%	0.013%
Central	1,575,000	5.00%	1:50	99.885%	0.0598	-4.926%	-0.077%
Lateral	1,530,000	10.00%	1:48	99.731%	0.2222	-6.427%	-0.231%
Central	1,530,000	10.00%	1:48	99.910%	0.3393	-6.413%	-0.051%

The results of the Imagenette dataset show average accuracy decrease across all experiments. However, due to the high p -values of all experiments (the lowest of all being a value of 0.1394), these results are statistically significant from the benchmark, and are therefore not varying enough to be used for a reliable comparison. Deeper analysis can only be considered as speculation.

The results of the Micro-PCB dataset show two experiments with an increase in average accuracy. Lateral exclusion with both 1% and 5% exclusion have an accuracy increase of 0.013%. As previously mentioned, the high p -values of 0.74 and 0.68 show that the results are not statistically significant, and highly prone to small random changes in the network model. Therefore, unfortunately, this result must be concluded as unreliable.

4.3 Discussion

The results of the dynamic data reduction experiments give much higher p -values than those of static data reduction, suggesting that the difference in accuracy compared to the benchmark is less statistically significant, and any differences are likely due to random change. A lack of statistical significance means that the results are less distinguishable from the benchmark. As the task of this chapter is to reduce the data use and maintain the accuracy as much as possible, results indistinguishable from the benchmark is an acceptable result.

There is a **0.114%** loss in accuracy with central reduction of 1% for the Imagenette dataset. As previously mentioned, the high p -value means this result is not statistically significant from the benchmark. However, central reduction yields superior accuracy compared to the lateral reduction of the same amount of data. These results correlate with the conclusions of Byerly et al. [72], where the data furthest from the centroid holds more important features necessary for distinguishing the image from other classes, compared to data closer to the centroid. The runtime of the central reduction of 1% experiment and the lateral reduction equivalent are more than the benchmark, meaning the computation to order and remove the data takes longer to process than simply training on that data. Therefore, as there is a greater cost in energy use and CO₂ emission, 1% data reduction is not beneficial for the Imagenette dataset.

The best performing accuracy for the Micro-PCB dataset is, curiously, the same value for both 1% and 5% lateral data reduction, at **99.974%**, which is above the benchmark accuracy for the Micro-PCB dataset. However, as with the Imagenette dataset, the p -values for these experiments are too high for these results to be considered statistically significant when compared with the benchmark. As previously mentioned, marginally different accuracies and p -values that indicate no statistical significance show that the results are indistinguishable from the benchmark. The runtime of training is shorter than that of the benchmark, so the data reduction methods used in this chapter are effective for slightly lowering runtime, and therefore CO₂ use, for the Micro-PCB dataset.

The runtime reductions of the Micro-PCB dataset are greater than those of the Imagenette dataset, despite the same amount of data reduction implemented, indicating that the Imagenette dataset requires more time per image evaluation than that of the Micro-PCB dataset. The difference in runtime reduction also suggests that the images of the Micro-PCB dataset are less complex than those of Imagenette. Therefore, the Micro-PCB dataset can be described as less computationally demanding, and more suitable for data reduction methods such as the ones described in this chapter. The difference in performance across different datasets may be useful for prototyping new data reduction technologies and would benefit from further experimentation.

4.4 Conclusion

To further investigate the practicality dynamic data reduction, the need arose for non-random reduction. Chapter 3 introduced dynamic data reduction, but did not focus on any particular

data, and removed data at random. While the results from non-random data reduction showed that any data could be removed and performance maintained, it lacked the capacity to see if some data exists that does not benefit training, and could therefore be removed. By using data reduction techniques, the aim is to identify the least critical data to be excluded from training, shortening training time and cutting the environmental cost.

This chapter used dimensionality reduction to calculate centroid distances for each datapoint in two datasets. Centroid distance is used to order data, allowing us to remove data closest or furthest from the centroids, giving us a non-random data reduction method. This chapter has shown that it is possible to order image data by centroid distance, after reducing the dimensionality of the images. Once the data is ordered by the distance of each datapoint from its classes' centroid, removing data furthest from the centroid causes a drop in model accuracy. Therefore, central reduction is the preferred reduction method.

This chapter has used less reduction than that used in chapter 3. A smaller amount of data reduction allows comparison with other state-of-the-art work, to validate the effectiveness of the methods introduced. A research gap to address is to use larger amounts of non-random data reduction to compare with chapter 3. Another gap is to use smaller image datasets, like those used in chapter 3. A final gap is to develop a method that does not train with a smaller portion of each dataset and then move onto the full dataset, but use a reduced dataset throughout all training while still using the full dataset.

Chapter 5

Dynamic Data Reduction with Sliding Window

In chapter 4, a non-random method to order and select data for reduction was introduced. As with chapter 3, these methods show that data may be removed while performance maintained. This chapter aims to further investigate how data is presented to artificial neural networks through the development of new methods to dynamically present data. These new methods are compared with the standard procedure of full data inclusion. Image datasets of varying image sizes and complexities are used to gain an understanding of how the methods affect different image datasets.

5.1 Methodology

5.1.1 Data selection

The UMAP algorithm reduces the dimensionality of a dataset, allowing us to observe the datapoints relative to each other in Euclidian space. The algorithm was run for each dataset’s training set, based on work by Byerly et al. [72]. Dimensionality reduction yielded a Euclidian distance of each datapoint, relative to that datapoint’s class’s centroid. The magnitude of the Euclidian distance is then used to order the datapoints by their distance to their centroid, on a scale of ‘central’ to ‘lateral’, and used as the basis for determining whether a datapoint is included by the sliding data window.

5.1.2 Data application

The aim of this chapter is to use a portion of the full dataset every epoch. The data used in an epoch depends on the selected ‘window size’ and the epoch number. The method is described as sliding window, as the amount of data used in an epoch is encompassed in a ‘window’ that slides across the data each epoch, selecting a new data portion to train with. The direction of the window slide is defined as ‘forward’ when sliding from central data to lateral, and ‘reverse’ when sliding from lateral data to central.

Figure 5.1 show example graphs of used data against training epoch, with a dataset of 100 datapoints. Each graph shows which data are used at each epoch, depending on the predetermined ‘window size’. The top two graphs (a) and (b), shown in blue, are experiments that provide data from ‘central to lateral’; the datapoints used at the earlier stages of training are closer to their class’s centroid. The data slides in the forward direction. The bottom two graphs (c) and (d), shown in orange, are ‘lateral to central’, and the datapoints used at the earlier stages of training are furthest from their class centroids. The data slides in the reverse direction. The left two graphs (a) and (c) have a window size of 20%; each epoch only uses 20% of the training. As such, every sequential epoch sees a bigger change in data used. Oppositely, the right two

graphs (b) and (d) show data use of 80% of the full dataset. More data is used each epoch than the 20% experiments, and each epoch sees less of a change in data used. It is also important to note that when the window size is more than 50%, the data whose distance to its class's centroid is used throughout all the training. The larger the window size, the more data there is that fits this category.

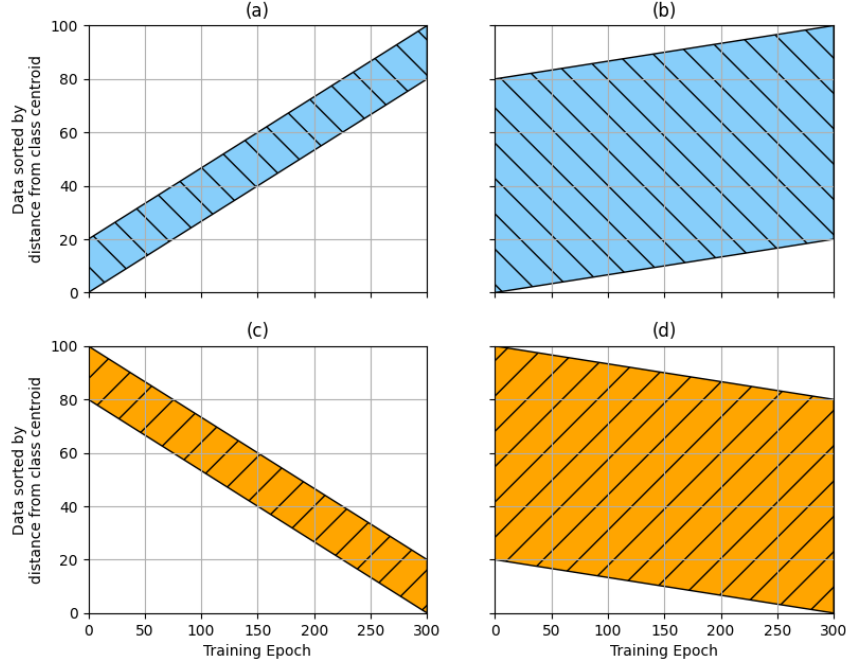


Figure 5.1: Example data usage graphs with Sliding Window method, where
 (a) shows 20% data usage, in the forward direction,
 (b) shows 80% data usage, in the forward direction,
 (c) shows 20% data usage, in the reverse direction and
 (d) shows 80% data usage, in the reverse direction.

5.1.3 Datasets

Five datasets were used for experimentation: the handwritten digits dataset MNIST [165], CIFAR-10 [166], and the model toy dataset smallNORB [167] are datasets with smaller images, but a larger quantity of them. Imagenette [170] and Micro-PCB [172] contain larger images, but fewer of them. Table 5.1 shows the properties of each dataset.

Table 5.1: Properties of each dataset used in this chapter.

Dataset	# Training images	# Test images	Image size (pixels)	# Colour channels
MNIST	60,000	10,000	28x28	1
CIFAR-10	50,000	10,000	32x32	3
smallNORB	24,300	24,300	32x32	1
Imagenette	9,469	3,925	299x299	3
Micro-PCB	6,500	1,625	299x299	3

5.1.4 Hardware

All experimentation was performed on the same hardware to maximise consistency. The CO₂ usage was computed from the average power use by the GPU during training and statistics from the Department for Energy Security and Net Zero [168]. The estimated CO₂ emission is used for emission reduction calculations.

- CPU: Intel i7 10700
- GPU: RTX A4000
- RAM: 64GB
- CO₂ emission per hour: 35.61g

5.1.5 Performance Metrics

The aim of this chapter is to reduce the overall amount of evaluations performed through training. Also, the CO₂ emissions calculator tool [173] uses time as a parameter for its calculation. Therefore, to calculate the reduction CO₂ emissions, the runtime of each experiment is needed. The runtime of each experiment is recorded and compared with the benchmark to gauge the percentage difference. While being able to reduce the CO₂ emissions is desirable, it is also important not to degrade the performance of the network. Accuracy is recorded to show the successfulness of an experiment. Likewise with runtime, the percentage difference against the benchmark is calculated for convenience.

p -values are also calculated for each experiment, with relation to each dataset's benchmark. The p -value is critical for identifying the statistical significance of the results of an experiment. For this chapter, a p -value of 0.05 or lower is chosen to determine if the results are statistically significant, which is a widely used value for this purpose. p -values are calculated based on analysis of variance (ANOVA) [174].

5.1.6 Network Architecture

A Convolutional Neural Network (CNN) with a simple monolithic architecture was employed for all experimentation, as per [116]. Due to the differing image size of the datasets used, different models were required.

For smaller image datasets MNIST, CIFAR-10 and smallNORB, a nine-layer CNN was used. The output of the final convolutional layer was fed into a HVC layer [116] in place of a dense layer, which then fed into the output classification layer. Figure 5.2 shows the structure of the CNN used for the MNIST, CIFAR-10 and smallNORB datasets.

For the larger image datasets Imagenette and Micro-PCB, an eleven-layer CNN was used with pooling after every three convolution layers. The final convolutional layer feeds into an HVC layer, and then the output classification layer. Figure 5.3 shows the structure of the CNN used for the Imagenette and Micro-PCB datasets.

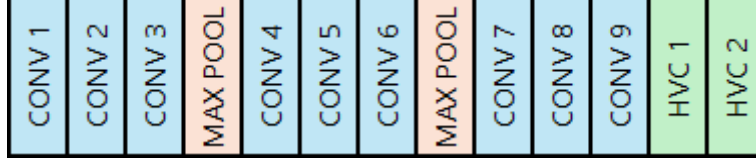


Figure 5.2: CNN model architecture used for MNIST, CIFAR-10 and smallNORB datasets: a 9 layer CNN with max pooling between every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.

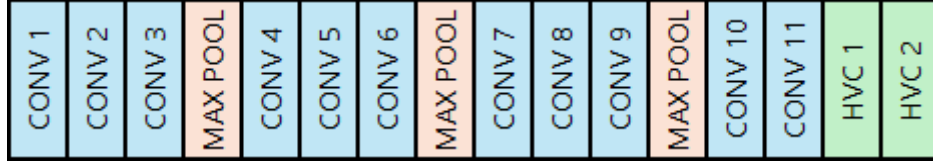


Figure 5.3: CNN model architecture used for the Imagenette and Micro-PCB datasets: 11 layer CNN with max pooling after every 3 convolutional layers, and ending with 2 HVC layers in place of dense layers.

Tests were run for 300 epochs, with a batch size of 120, for consistency with chapter 3. The Adam optimiser was used with an initial learning rate of 0.001, and a decay rate of 0.02 per epoch. These values provide consistent initial settings across all experiments for all datasets. Online augmentation was used for training, whereby training images would undergo augmentation based on [116] each epoch, including random rotation, random shift and image distort.

5.2 Experimental Results

This chapter measures the performance of the sliding window strategy through experimental work. The experimental results will be compared with methods in chapters 3 and 4 to determine whether the sliding window method is superior. Also, as with chapter 4, experimentation will help identify which data is more useful at the different stages of training, by sliding the data window in different directions.

5.2.1 Benchmark

Experiments were run with the full dataset and no data exclusion to serve as a benchmark for all other experiments. Table 5.2 shows the benchmark runtime and accuracy for all the datasets used in this chapter. These results will be used to compare the data reduction experiments, to show the reduction in runtime and effect on performance. The number of total image evaluations is the number of training images evaluated multiplied by the number of training epochs.

Table 5.2: Benchmark results for each dataset used in this chapter.

Dataset	# Training epochs	Total # Image evaluations	Runtime (hours)	CO ₂ emission (kg)	Test Accuracy
MNIST	300	18,000,000	1:58	0.130746	99.749%
CIFAR-10	300	14,976,000	2:18	0.238128	89.249%
smallNORB	300	7,272,000	1:23	0.091806	92.685%
Imagenette	300	2,808,000	3:36	0.238590	90.323%
Micro-PCB	300	1,944,000	2:24	0.158926	99.987%

5.2.2 MNIST

Table 5.3 shows the results of the experimentation for the MNIST dataset. The value marked in bold is the highest average accuracy achieved.

Table 5.3: Experimental results for the MNIST dataset.
The value in bold is the best average accuracy.

Window Size	Runtime (hours)		Average accuracy		Runtime difference		Accuracy difference	
	Forward	Reverse	Forward	Reverse	Forward	Reverse	Forward	Reverse
10%	0:22	0:22	99.353%	99.016%	-81.078%	-80.964%	-0.397%	-0.735%
25%	0:37	0:37	99.550%	99.454%	-68.816%	-68.824%	-0.199%	-0.296%
50%	1:05	1:07	99.669%	99.552%	-45.227%	-43.493%	-0.081%	-0.197%
55%	1:10	1:10	99.665%	99.606%	-40.727%	-41.047%	-0.085%	-0.143%
60%	1:15	1:17	99.659%	99.620%	-36.333%	-34.884%	-0.091%	-0.129%
65%	1:21	1:21	99.665%	99.629%	-31.409%	-31.463%	-0.085%	-0.121%
70%	1:27	1:27	99.669%	99.631%	-26.139%	-26.661%	-0.081%	-0.119%
75%	1:32	1:32	99.689%	99.643%	-22.149%	-21.864%	-0.060%	-0.107%
80%	1:38	1:41	99.691%	99.659%	-17.386%	-14.707%	-0.058%	-0.091%
85%	1:43	1:47	99.703%	99.682%	-13.108%	-9.515%	-0.046%	-0.067%
90%	1:49	1:48	99.711%	99.707%	-8.222%	-8.686%	-0.038%	-0.042%
95%	1:58	1:54	*99.719%	*99.723%	-0.149%	-3.858%	-0.030%	-0.026%

MNIST is well known for being easy to train and achieving accuracies above 99%. The results show that even with only 10% of the total evaluations performed, over 99% accuracy is achieved. Runtime can be heavily reduced by over 80% by training in this way, with a marginal detriment to the accuracy. The best performing strategy is with a window size of 95% in the reverse direction, with an accuracy decrease of 0.026% and a runtime difference of four minutes. Four minutes is a very small runtime difference relative to the benchmark runtime of almost 2 hours, which is not a very beneficial runtime reduction for the loss in model accuracy. Figure 5.4 shows the differences between experimental results for the MNIST dataset more clearly. One feature to note is that experiments in the forward direction (central to lateral data inclusion) outperform the experiments in the reverse direction (lateral to central inclusion) in all but one cases; the final experiment with a window size of 95% shows that the reverse direction beats central to lateral by 0.004%. Considering that this experiment performs 0.024% worse than the baseline for only a 4 minute runtime reduction, it would be more beneficial to simply use the full dataset.

Accuracies marked with an asterisk have associated p -values above 0.05. As such, they do not have a statistically significant difference from the baseline. Both experiments with a window size of 95% share this property, and so their validity is questionable.

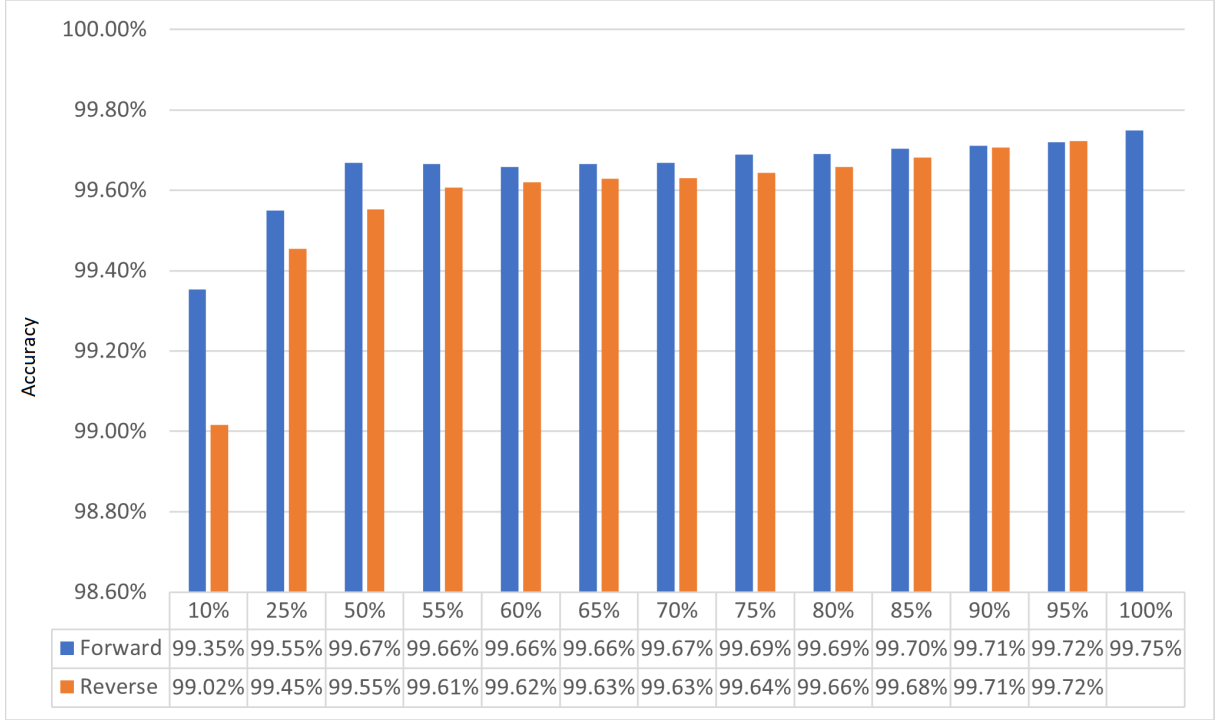


Figure 5.4: Experimental results with MNIST dataset

Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.

5.2.3 CIFAR-10

Table 5.4 shows the results of the experimentation for the CIFAR-10 dataset. The value marked in bold is the highest average accuracy achieved.

Compared to MNIST, CIFAR-10 is more difficult to train, shown by the lower accuracy achieved and increased runtime despite fewer evaluations performed. The best performing strategy is with a window size of 95% in the forward direction, with an accuracy decrease of 0.153% and a runtime difference of five minutes. As with the MNIST dataset, this is a very small runtime difference relative to the benchmark runtime of over 2 hours, which is not a very beneficial runtime reduction for the loss in model accuracy. As the window size is decreased, fewer evaluations are performed, which causes a drop in both runtime and accuracy. If an accuracy decrease of 1% is acceptable for a runtime drop of roughly 15% (24 minutes), then perhaps the 80% window size may be desirable.

The experiments in the forward direction outperform the reverse direction across the board, including the 95% window size experiment. This is shown clearly in Figure 5.5. However, as with MNIST, the p -value of the 95% lateral to central is above 0.05, and thus the network performance is unstable. Despite this, a sliding window in the forward direction is superior with all window sizes, showing that centralmost data is better used at the beginning of training and lateralmost data at the end.

Table 5.4: Experimental results for CIFAR-10 dataset.
The value in bold is the best average accuracy.

Window Size	Runtime (hours)		Average accuracy		Runtime difference		Accuracy difference	
	Forward	Reverse	Forward	Reverse	Forward	Reverse	Forward	Reverse
10%	0:23	0:23	76.114%	75.749%	-83.070%	-83.058%	-14.717%	-15.126%
25%	0:42	0:42	83.297%	81.528%	-69.084%	-69.072%	-6.669%	-8.651%
50%	1:15	1:15	86.466%	85.432%	-45.409%	-45.510%	-3.118%	-4.277%
55%	1:21	1:21	87.018%	86.100%	-41.081%	-41.188%	-2.500%	-3.528%
60%	1:28	1:28	87.191%	86.564%	-36.235%	-36.431%	-2.306%	-3.008%
65%	1:34	1:34	87.645%	86.942%	-32.045%	-31.675%	-1.798%	-2.585%
70%	1:41	1:40	87.803%	87.386%	-26.956%	-27.272%	-1.620%	-2.088%
75%	1:47	1:47	87.924%	87.703%	-22.189%	-22.418%	-1.485%	-1.732%
80%	1:54	1:54	88.315%	88.106%	-17.676%	-17.334%	-1.046%	-1.280%
85%	2:00	2:00	88.618%	88.462%	-13.106%	-12.925%	-0.706%	-0.882%
90%	2:06	2:06	88.727%	88.657%	-8.541%	-8.405%	-0.585%	-0.664%
95%	2:13	2:13	*89.096%	*88.867%	-3.269%	-3.617%	-0.171%	-0.427%

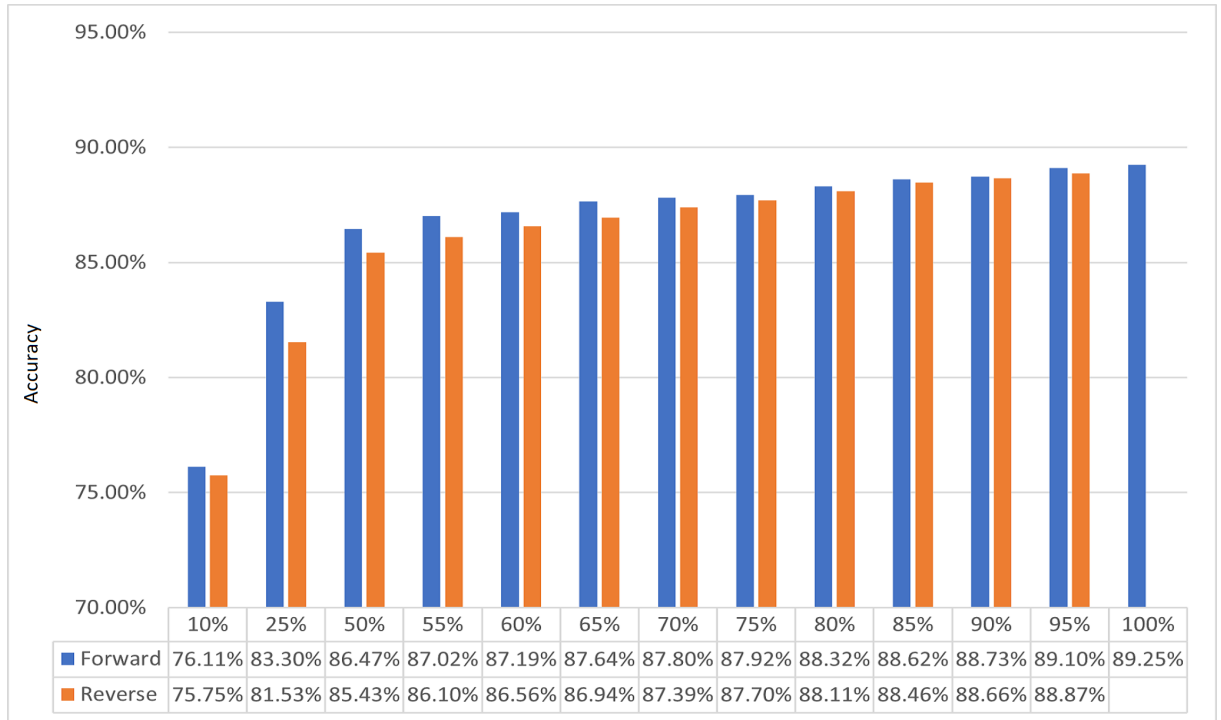


Figure 5.5: Experimental results with CIFAR-10 dataset.
Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.

5.2.4 smallNORB

Table 5.5 shows the results of the experimentation for the smallNORB dataset. The value marked in bold is the highest average accuracy achieved.

Table 5.5: Experimental results for smallNORB dataset.
The value in bold is the best average accuracy.

Window Size	Runtime (hours)		Average accuracy		Runtime difference		Accuracy difference	
	Forward	Reverse	Forward	Reverse	Forward	Reverse	Forward	Reverse
10%	0:42	0:41	81.057%	73.536%	-48.612%	-50.709%	-12.546%	-20.660%
25%	0:42	0:42	88.897%	87.849%	-48.876%	-49.629%	-4.087%	-5.217%
50%	0:52	0:52	90.801%	90.944%	-36.641%	-36.496%	-2.032%	-1.878%
55%	0:56	0:55	91.519%	91.165%	-32.515%	-32.974%	-1.258%	-1.640%
60%	0:58	0:58	91.112%	*91.647%	-29.846%	-29.689%	-1.697%	-1.120%
65%	1:01	1:02	91.341%	*92.047%	-26.303%	-25.220%	-1.450%	-0.688%
70%	1:04	1:04	*92.254%	*91.863%	-22.774%	-22.719%	-0.465%	-0.887%
75%	1:07	1:08	91.020%	*92.814%	-18.862%	-17.846%	-1.796%	0.140%
80%	1:10	1:10	*92.217%	*92.212%	-15.195%	-15.773%	-0.505%	-0.510%
85%	1:13	1:13	*92.366%	*92.593%	-11.403%	-11.558%	-0.344%	-0.099%
90%	1:17	1:16	*93.108%	*92.574%	-7.654%	-8.010%	0.457%	-0.119%
95%	1:20	1:20	*92.382%	*93.339%	-3.788%	-4.030%	-0.327%	0.706%

The results of the experimentation for the smallNORB dataset show fluctuations in accuracy as the window size is increased. For example, experiments in the forward direction show a decrease in accuracy when increasing the window size from 50% to 55% and from 70% to 75%. A decrease in model accuracy is interesting behaviour, as usually adding more data or running more evaluations results in a boost in model performance. What is even more unexpected is that experiments 90% in the forward direction and 75% in the reverse direction have a higher accuracy than the benchmark by **0.654%**, with a runtime difference of 3 minutes. Figure 5.6 also shows great fluctuation between accuracies in the forward and reverse direction, making it impossible to identify the best window direction to use for this dataset. Considering the high p -values calculated for these experiments, indicated by the asterisks on the accuracy columns, the statistical significance and validity suggest that the results are indistinguishable from the benchmark. Also, consider that the smallNORB dataset features fewer datapoints than the previously explored MNIST and CIFAR-10; fewer datapoints could cause randomness in the network, which is only amplified by initial random states of the neurons and the augmentation used, which causes greater noise in the results.

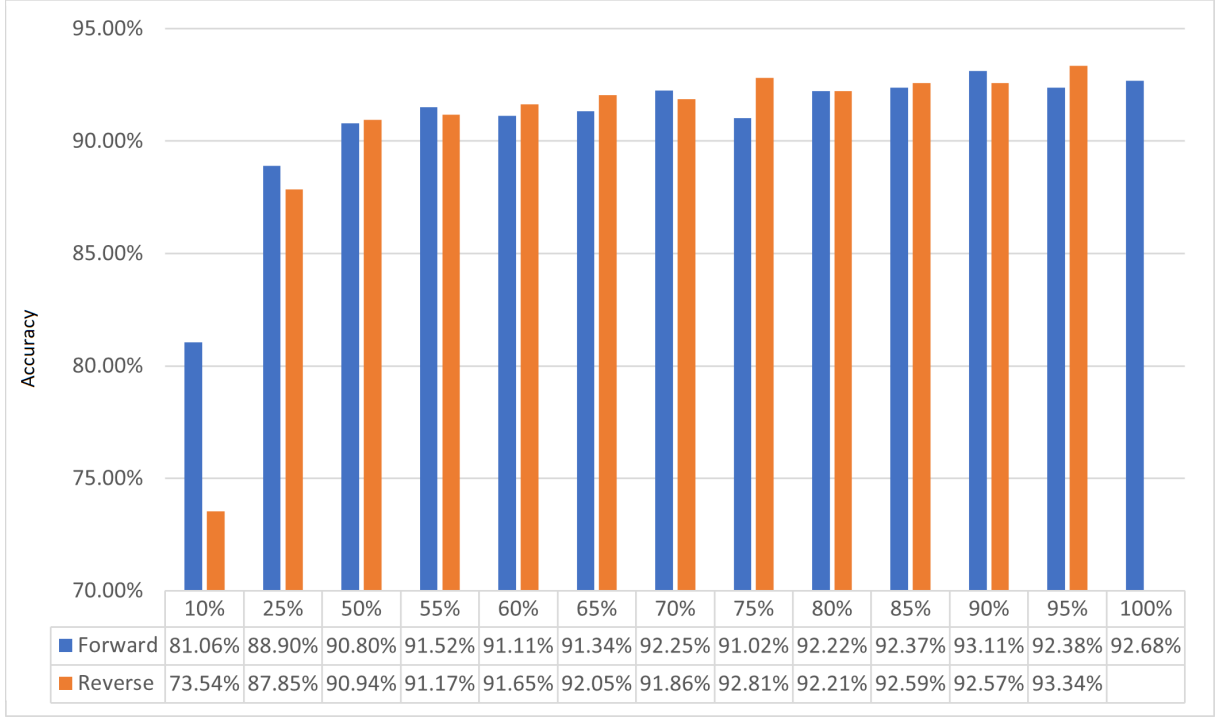


Figure 5.6: Experimental results with smallNORB dataset.

Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.

5.2.5 Imagenette

Table 5.6 shows the results of the experimentation with the Imagenette dataset. The value marked in bold is the highest average accuracy achieved.

Similarly to CIFAR-10, Imagenette is one of the more difficult datasets used in this chapter to train. However, the experimentation follows a familiar pattern, where the fewer evaluations are performed, the worse the accuracy of the testing. In particular, with a window size of 10% an accuracy of only 40% is achieved; this is unacceptable performance, and so much data reduction is unsuitable.

The best performing strategy is with the reverse direction with a window size of 95%, with an accuracy decrease of **0.115%** and a runtime reduction of 2 minutes. For such a minimal reduction in runtime, this is a fairly significant accuracy drop. As shown by Figure 5.7, experiments in the forward direction outperform those in the reverse direction in most cases for Imagenette, except with window sizes 80% and 95%, showing that when less data is available, the forward strategy is superior. As window size increases, p -values indicate the null hypothesis is not true for some strategies making it impossible to compare with the smaller window sizes with sufficient validity.

Table 5.6: Experimental results for Imagenette dataset.
The value in bold is the best average accuracy.

Window Size	Runtime (hours)		Average accuracy		Runtime difference		Accuracy difference	
	Forward	Reverse	Forward	Reverse	Forward	Reverse	Forward	Reverse
10%	0:52	0:52	41.495%	39.406%	-75.711%	-75.615%	-54.060%	-56.372%
25%	1:21	1:21	69.313%	67.833%	-62.423%	-62.228%	-23.261%	-24.899%
50%	2:07	2:08	84.990%	83.307%	-41.046%	-40.873%	-5.905%	-7.767%
55%	2:18	2:19	85.771%	85.010%	-36.107%	-35.881%	-5.040%	-5.882%
60%	2:28	2:27	87.151%	86.625%	-31.738%	-32.194%	-3.512%	-4.094%
65%	2:38	2:37	87.672%	87.026%	-27.142%	-27.517%	-2.935%	-3.650%
70%	2:46	2:46	88.693%	87.641%	-23.122%	-23.196%	-1.805%	-2.970%
75%	2:55	2:55	88.865%	88.094%	-18.955%	-19.019%	-1.615%	-2.468%
80%	3:06	3:05	88.958%	89.000%	-14.008%	-14.666%	-1.511%	-1.465%
85%	3:15	3:14	*89.687%	88.911%	-9.952%	-10.436%	-0.703%	-1.563%
90%	3:23	3:23	*90.073%	89.766%	-6.124%	-6.164%	-0.277%	-0.617%
95%	3:32	3:32	*89.932%	*90.208%	-1.866%	-1.901%	-0.432%	-0.127%

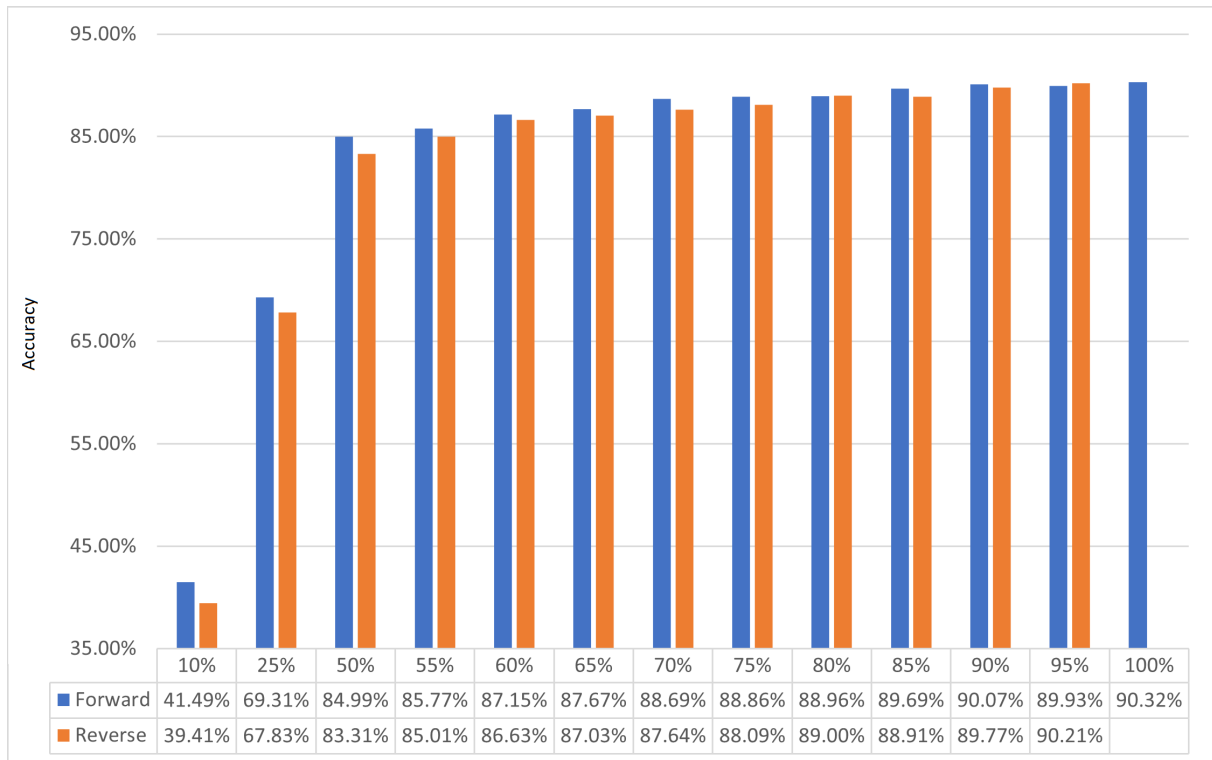


Figure 5.7: Experimental results with Imagenette dataset.

Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.

5.2.6 Micro-PCB

Table 5.7 shows the results of the experimentation with the Micro-PCB dataset. The value marked in bold is the highest average accuracy achieved.

Table 5.7: Experimental results for Micro-PCB dataset.
The value in bold is the best average accuracy.

Window Size	Runtime (hours)		Average accuracy		Runtime difference		Accuracy difference	
	Forward	Reverse	Forward	Reverse	Forward	Reverse	Forward	Reverse
10%	0:24	0:24	13.321%	15.038%	-82.827%	-82.989%	-86.678%	-84.960%
25%	0:44	0:44	69.000%	65.859%	-69.027%	-68.977%	-30.991%	-34.133%
50%	1:16	1:17	94.026%	*95.897%	-46.814%	-46.446%	-5.962%	-4.090%
55%	1:23	1:24	97.897%	98.090%	-42.491%	-41.465%	-2.090%	-1.898%
60%	1:31	1:32	99.513%	98.615%	-36.412%	-36.293%	-0.474%	-1.372%
65%	1:39	1:39	*99.782%	99.590%	-30.889%	-31.315%	-0.205%	-0.397%
70%	1:43	1:44	99.128%	*99.128%	-28.115%	-27.738%	-0.859%	-0.859%
75%	1:51	1:51	*99.590%	99.846%	-22.963%	-22.614%	-0.397%	-0.141%
80%	1:58	1:59	99.846%	*99.846%	-18.048%	-17.508%	-0.141%	-0.141%
85%	2:04	2:04	*99.744%	*99.949%	-13.867%	-13.878%	-0.244%	-0.038%
90%	2:11	2:10	*99.897%	*99.936%	-8.862%	-9.448%	-0.090%	-0.051%
95%	2:18	2:18	*99.910%	*99.936%	-3.940%	-3.855%	-0.077%	-0.051%

The Micro-PCB dataset features 13 classes of PCBs, with the focus being on capturing different perspectives. There are few images in this dataset, as there is very little variation between PCBs of the same type, and so there is little variation that would need to be represented. This is similar to the smallNORB dataset, where the images are of the same objects at different angles.

The Micro-PCB dataset achieves a high accuracy (above 90%), and is relatively easy to train, similarly to MNIST. However, unlike MNIST, experiments with a window size of 10% show very poor accuracy. The performance of the 10% window size is closer to that of the CIFAR-10 and Imagenette datasets, likely due by the fewer number of total evaluations performed, as there are much fewer data in the Micro-PCB dataset compared to MNIST. Such a small window size is not appropriate for this dataset.

The best performing strategy is with the reverse direction with a window size of 85%, achieving an accuracy decrease of **0.038%** and a runtime reduction of 20 minutes. Other datasets have their highest accuracies with a larger window size of 90% or 95%; the difference between the Micro-PCB dataset and the other datasets tested so far may come down to by the high p -values of many of the experimental results. High p -values are present with a window size as low as 65% for the Micro-PCB dataset, leading to the conclusion that the results are unstable with these window sizes.

The Micro-PCB dataset shows that for a majority of window sizes, experiments in the reverse direction outperform those in the forward direction, shown in Figure 5.8. Also, a majority of experiments have p -values above the 0.05 maximum. As observed with the smallNORB dataset, high p -values with these results are due to the highly structured nature of the dataset.

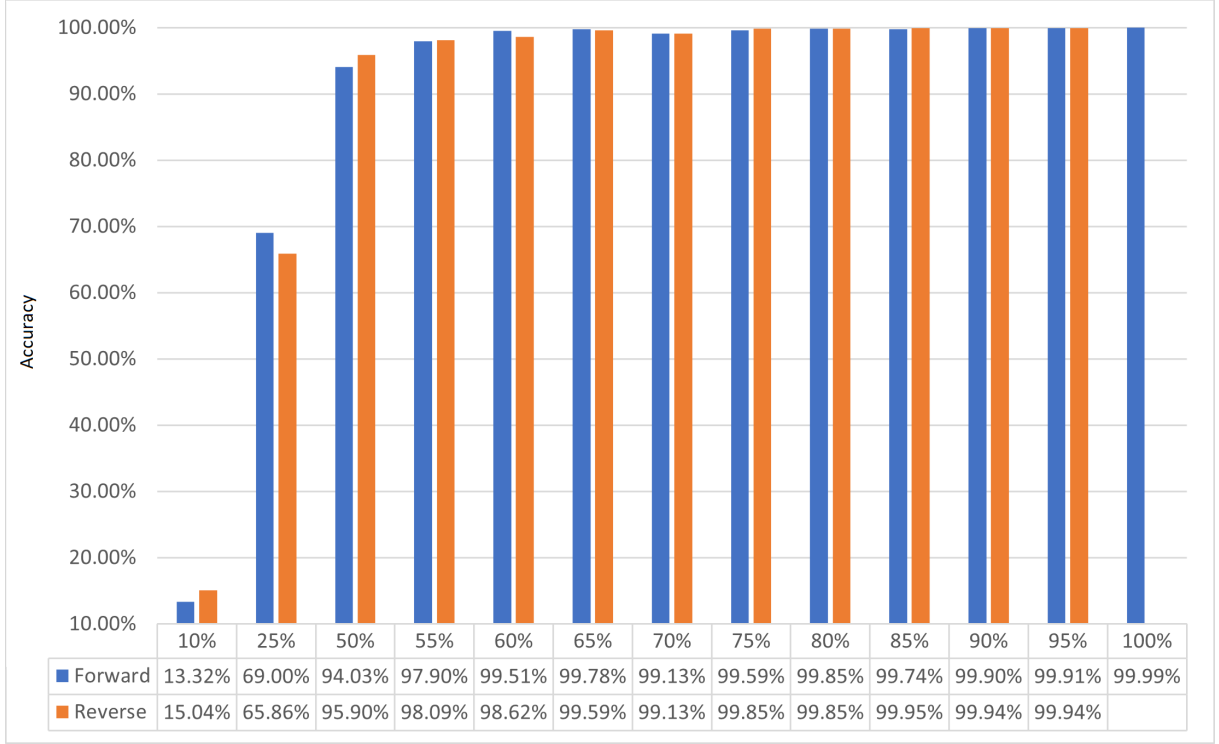


Figure 5.8: Experimental results with Micro-PCB dataset.

Blue bars show accuracy for experiments in the forward direction, and orange bars show accuracy for experiments in the reverse direction.

5.3 Discussion

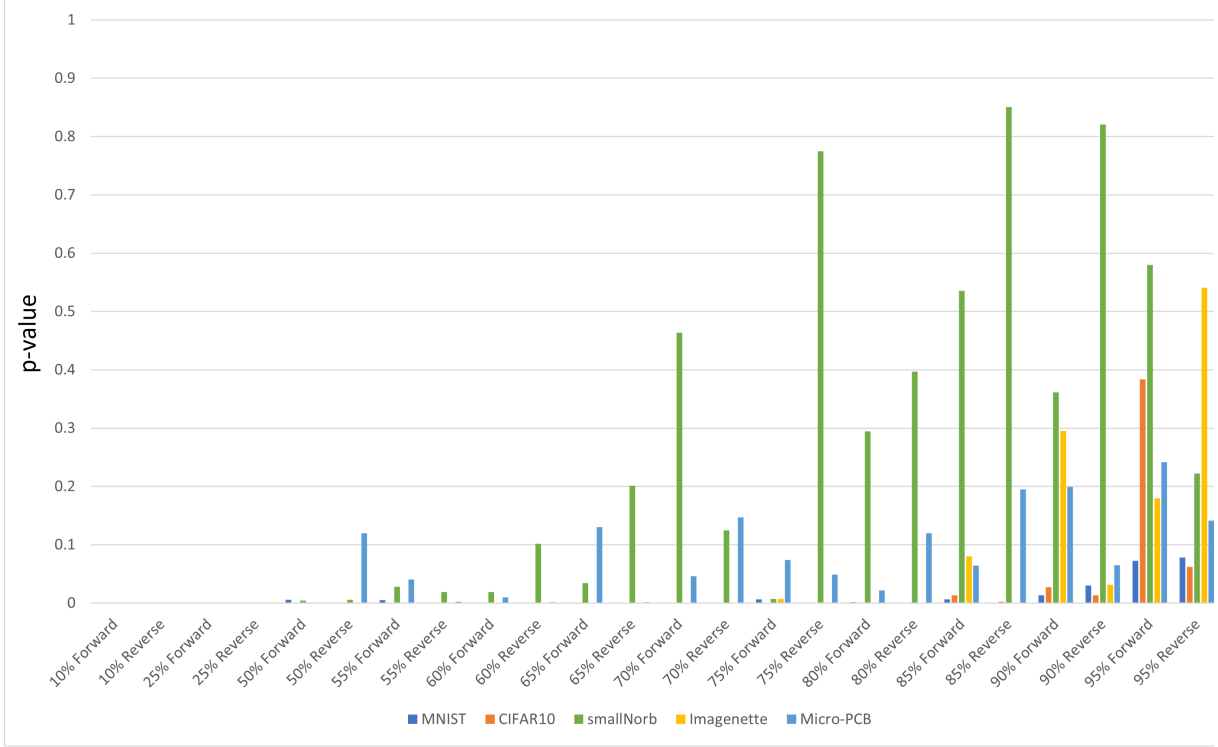
There are two results to discuss: the results of the experimentation on their own, and the results compared to those of the previous chapters of this thesis.

5.3.1 Comparing Sliding Window with Benchmark

The experimental results all show a pattern when comparing the window size’s effect on accuracy; the smaller the window size, the lower both the accuracy and training runtime. If a large reduction in runtime is required, a sacrifice in accuracy is necessary.

Before analysing and comparing the results of the experimentation more deeply, it is important that the p -values of the experiments are considered to determine statistical significance. High p -values indicate the experimental results are indistinguishable from the baseline. The tables above have shown the experiments where the p -value indicates this, and figure 5.9 shows how p -values vary between datasets. Figure 5.9 shows that while for most datasets, the p -value increases with the increasing window size. As the window size grows to the size of the full dataset, the experimental results become less distinguishable from those of the benchmark. However, for the smallNORB and Micro-PCB datasets, the p -value fluctuates much more. As previously mentioned, fluctuating p -values are possibly due to the structured nature of the data. Further experimentation is required to identify the reason for this behaviour, which may provide insight into how the different natures of data need to be handled.

For all datasets except CIFAR-10, the best performing experiments are in the reverse direction, with a minimum window size of 85%. With a window size this large, there is at least some data in the lateral half of the dataset present throughout all training. The better performance with data inclusion from lateral to central correlates with work by Byerly et al. [72], where data in the lateral half of the dataset gave better performance than that in the central half. However,


 Figure 5.9: The p -values for all experiments across all datasets.

figures 5.4, 5.5, 5.6 and 5.7 make it clear that with window sizes less than 50%, for datasets MNIST, CIFAR-10 and Imagenette, experiments in the forward direction generally outperform those in the reverse direction. This correlates with the theory that for unstructured datasets such as MNIST, CIFAR-10 and Imagenette, data closer to its class’s centroid is easier to learn, and that data further is harder, and that easier data should be used at the beginning of training.

5.3.2 Comparing Sliding Window with Data Step

Comparing the benchmarks

It is important to highlight the differences in models used between experiments using data step and sliding window in order to accurately compare them. Table 5.8 shows the hyperparameters of the models used in chapters 3 and 4 for the data step experimentation, and table 5.9 summarises the model used in this chapter for sliding window. The slight differences in these tunings were selected as a best fit for the datasets trained on each chapter of this thesis.

Table 5.8: Model hyperparameters for each dataset for experiments with the Data Step method of chapters 3 and 4.

	Model	Epochs	LR	LR decay	Batch size
MNIST	9 layer CNN, 2 HVC	300	0.999	0.005	120
CIFAR-10	9 layer CNN, 2 HVC	300	0.999	0.005	120
smallNORB	9 layer CNN, 2 HVC	300	0.999	0.005	120
Imagenette	11 layer CNN, 2 HVC	250	0.98	0.001	120
Micro-PCB	11 layer CNN, 2 HVC	250	0.98	0.001	120

All experiments use a nine-layer CNN with two HVC layers, except those performed in chapter 4, which uses an eleven-layer CNN with two HVC layers. The same goes for initial learning rate,

Table 5.9: Model hyperparameters for all dataset for experiments with the Sliding Window method of this chapter.

	Model	Epochs	LR	LR decay	Batch size
All Datasets	9 layer CNN, 2 HVC	300	0.999	0.02	120

which for all experiments is 0.999, except those in chapter 4, where it is 0.98. The learning rate and decay rates are different in between chapters 3 and 4 of this thesis, as they were chosen as the optimal values for each dataset.

The small differences in model hyperparameters may be the cause of the slight difference in model accuracies. For larger image datasets Imagenette and Micro-PCB, the same model as in chapter 4 is used; an eleven-layer CNN with two HVC layers. Unlike in chapter 4, experimentation in this chapter was run for 300 epochs instead of only 250. The batch size used is the same, at 120. The initial learning rate, and decay rate are different; chapter 4 uses values of 0.98 and 0.001, while this chapter uses values of 0.999 and 0.02.

Table 5.10 shows the differences in accuracy and runtime between the benchmark experiments run in this chapter compared with those in chapters 3 and 4. There are some slight differences in accuracy and runtime, despite there being no data reduction performed at this stage.

Table 5.10: Comparison of benchmark accuracies and runtimes between chapters 3 and 4 with this chapter.

Dataset	Model Accuracy		Model Runtime	
	Chapter 3/4 Benchmark	Chapter 5 Benchmark	Chapter 3/4 Benchmark	Chapter 5 Benchmark
MNIST	99.701%	99.749%	2:01	1:59
CIFAR-10	88.689%	89.249%	2:54	2:19
smallNORB	92.984%	92.685%	1:27	1:23
Imagenette	90.694%	90.323%	3:15	3:37
Micro-PCB	99.963%	99.987%	1:55	2:25

The slight differences in hyperparameters and changes in data caused by data augmentation have a cascading effect on the benchmark accuracies. MNIST is improved by **0.048%**, CIFAR-10 is improved by **0.56%**, and Micro-PCB is improved by **0.024%**. Imagenette is worse by **0.371%**, and smallNORB is worse by **0.299%**. These results show the benefits and detriments of universally changing hyperparameters for all datasets; a better approach is to adjust these for each dataset. These values were selected for consistency, but caused better results for some datasets but worse results for others. Also, the benchmark results for the Imagenette dataset are worse, despite there being more training epochs in this chapter’s experimentation. There is no clear answer to which model and hyperparameter combination is best universally for all datasets; the differences yield better results for some datasets, yet worse results for others. As hyperparameters and randomness introduced by augmentation can have such an effect on model performance, the differences in accuracy between experiments become less significant. Therefore, by balancing improved fine-tuning of these preprocessing methods and larger amounts of data reduction, runtime may be reduced and model accuracy maintained.

Runtime differences are similar for datasets MNIST and smallNORB. The CIFAR-10 dataset has a large difference in runtime between chapters, which was caused by improvements to the data input pipeline and data augmentation procedures. Because of this, it shows that runtime may be greatly affected by other factors, not just the amount of data used for training. The runtimes for the Imagenette and Micro-PCB datasets are lower for chapter 4, as the experimentation for

this chapter only used 250 epochs, instead of the 300 used in this chapter.

Comparing the data reduction strategies

Now, the performance of the data step methods of chapters 3 and 4 are compared with the sliding window introduced in this chapter. Table 5.11 shows the best performing results with the Data Step method, for each dataset. Table 5.12 shows the best performing results with the Sliding Window method, for each dataset. Values in bold indicate the best accuracy and runtime for that dataset, between the Data Step and Sliding Window methods.

Table 5.11: Top results for each dataset with Data Step methods of chapters 3 and 4. Data Step reduction methods: Ep is step epoch, St is data step size, the percentage of data used in the initial step. Values in bold indicate superior performance compared to best performing experiment with Sliding Window.

Dataset	Reduction	# Image Evaluations	Average Accuracy	Runtime
MNIST	Ep.50 St.25% Random	11,312,880	99.725%	01:22
CIFAR-10	Ep.75 St.25% Random	14,061,360	88.625%	02:47
smallNORB	Ep.75 St.25% Random	6,828,000	93.059%	01:23
Imagenette	Ep.125 St.95% Lateral	2,280,000	90.399%	03:16
Micro-PCB	Ep.125 St.95% Lateral	1,575,000	99.974%	01:53

Table 5.12: Top results for each dataset with Sliding Window methods of this chapter. Sliding Window reduction methods: Forward is the forward direction, Reverse is reverse direction. Percentage value is the amount of data excluded to create the sliding window. Values in bold indicate superior performance compared to best performing experiment with Data Step.

Dataset	Reduction	# Image Evaluations	Average Accuracy	Runtime
MNIST	Forward 5%	17,100,000	99.723%	01:54
CIFAR-10	Forward 5%	14,227,200	89.096%	02:13
smallNORB	Forward 5%	6,908,400	93.339%	01:20
Imagenette	Reverse 5%	2,667,600	90.208%	03:32
Micro-PCB	Reverse 15%	1,652,400	99.949%	02:04

From these results there is no clear superior method. Datasets MNIST, Imagenette and Micro-PCB yield better accuracy and runtime with the data step method, while also performing fewer image evaluations. Therefore, for these datasets, the data step methods is better in every way. Datasets CIFAR-10 and smallNORB perform better with the sliding window method, in both average accuracy and runtime, despite performing more image evaluations than its data step counterpart.

It is important to consider how the data is chosen for exclusion. For the data step method, datasets MNIST, CIFAR-10 and smallNORB use random data exclusion. Every other experiment in data step and sliding window use central or lateral exclusion. Therefore, it is possible that the superior performance from the sliding window experimentation on the CIFAR-10 and smallNORB datasets is caused by the difference in data exclusion method, instead of the data application. With this uncertainty in place, the Imagenette and Micro-PCB datasets' results can be relied upon as they use lateral reduction instead of random. Therefore, while the sliding window experimentation gave useful insight to which data is more useful at which stage of training as shown by the direction of the window, the Data Step method is superior.

5.4 Conclusion

Chapters 3 and 4 introduce dynamic data reduction, but comparisons between these chapters is hindered due to the different amounts of reduction used and different sizes of the image data. The experiments discussed in chapters 3 and 4 use a reduced dataset for a portion of the training, but use the full dataset for at least some of the training. This chapter introduces the sliding data window, which gives each epoch a different data subset each training loop, while ensuring the whole dataset is used. However, the full dataset is never used in any single training loop. The data is ordered using the centroid distance of each datapoint, derived from reduced dimensionality information as used in chapter 4.

It is shown that for some datasets, much less computational processing is required to achieve a minimal loss in model performance. The more complex the data is, the less data you may remove without negatively affecting performance. Also, in a majority of cases, a forward window direction where data is provided from centralmost to lateralmost performs better than a reverse window direction, further suggesting the theory that lateralmost data is less important at the initial stages of training but more useful at the later stages.

While this thesis so far has shown that data may be removed from the training set to maintain or improve model performance, the AI model must be fully trained to identify if the reduction methods are appropriate. A research gap, therefore, is to be able to identify suitability of data reduction for a dataset without having to train the model. Also, so far this thesis has focused on image data; another gap to address is to determine the suitability of these methods on other data formats, such as time-series data.

Chapter 6

Identifying most useful data in imbalanced time-series datasets

Chapters 3, 4 and 5 show that some data is more important to training, and by focusing on that data at different stages of training, the training process may be optimised and training time reduced. These chapters have investigated dynamic data reduction for image datasets; this chapter introduces dynamic data reduction to room occupancy datasets, a form of time-series dataset. This chapter also introduces class density, a metric that may be calculated for each class in a dataset to show suitability for data reduction and also class balancing, if the dataset suffers from class imbalance.

6.1 Methodology

The aim of this chapter is to identify the most useful data in a time-series occupancy dataset by calculating centroid distances, which consolidate all the features, the sensor values, into one variable. Having a single variable that describes all sensor values allows data to be quickly organised by a single metric, which is used to remove data by multiple data reduction techniques, giving us reduced datasets. The reduced datasets are used for training and testing, and the results are compared to show which data reduction method, and therefore which data, is most beneficial for training. However, as occupancy data commonly have an imbalanced number of datapoints in each class, the proposed method balances data by removing data from only the larger class resolving both issues of class imbalance and the abundance of less useful data.

6.1.1 Dataset Preparation and Fusion

Multiple open-source datasets have been developed for the purpose of occupancy detection with machine learning [175, 176]. The dataset selected for this experiment is the HPDMobile dataset [177], due to it having multiple sites of homogeneous environmental sensor data, making it ideal for dataset fusion. It is an open-source dataset that collects data from six sites, with each site containing the same type of sensors and using the same capture methods. Each sensor device captures temperature, humidity, and the Volatile Organic Compounds (VOC) count. Each site has either four or five of these sensors, which equates to twelve or fifteen features for each site, respectively. Each datapoint has an associated ground truth of the number of occupants, but for the sake of simplicity, the experimentation aims to differentiate between ‘some’ or ‘no’ occupants.

The HPDMobile dataset is not originally formatted by site, but by sensor; each sensor’s data are stored in a separate file. To make a complete dataset, each file of sensor data is sorted by location, time, and date and aggregated into a table for each site. The result is a csv file for each of the six sites, which are used as individual datasets. Details of the six data subsets of the HPDMobile dataset are shown in Table 6.1.

Table 6.1: HPDMMobile dataset information.

Class balance ratio is between classes ‘Not Occupied’ and ‘Occupied’. Each sensor contains three features: temperature, humidity, and VOC.

Site	Number of Datapoints	Original Number of Sensors/Derived Number of Features	Least Important Sensor	Class Balance Ratio (Not Occ:Occ)
Alpha	147,750	5:15	4	20:80
Beta	146,879	4:12	N/A	40:60
Charlie	302,399	5:15	0	22:78
Delta	146,879	5:15	4	21:79
Epsilon	129,599	5:15	4	24:76
Fazbear	328,319	4:12	N/A	47:53

The HPDMMobile dataset has on average 7% of data missing across all sites and time frames due to sensor synchronisation and duplicate dropping [177]. As gaps in data cause a loss in information, and the Random Forest (RF) algorithm does not support missing data, this issue is addressed by filling these gaps in artificially with data imputation. K-Nearest Neighbour (KNN) averaging was selected as an appropriate imputation technique for this purpose [178]. Prior to any training, each site’s dataset undergoes imputation. For each missing datapoint in each dataset, the three most similar datapoints are averaged to give the missing value. The KNNImputer package from scikit-learn is used for this.

When attempting to fuse datasets, there may be issues in mismatching data formats or missing data, which makes training difficult. Data pre-processing must be performed in order to homogenise datasets prior to their fusion. In the case of the HPDMMobile dataset, there are six individual site datasets, some containing 5 sensors, giving a total of 15 features, and with 4 sensors giving 12 features. To be able to fuse the individual site datasets together and to make comparison between each site simpler, each dataset is homogenised into the same shape with the same number of features by removing the least important sensor, and therefore the 3 least important features. The least important sensor for the larger sites is identified by classifying each of the 5 sensor datasets, and using the scikit-learn importances metric. The importances for each feature are collected and summed. The sensor with the smallest sum is then identified as the least important feature and omitted from the dataset before training. Sensor importances are highlighted in Table 6.1.

To create a fused dataset from the six individual site datasets, each site is first split into training and test sets. Then, each training set is combined into one large training dataset. The test sets are not combined but tested on individually, after the model is trained on the fused dataset. Figure 6.1 shows this procedure.

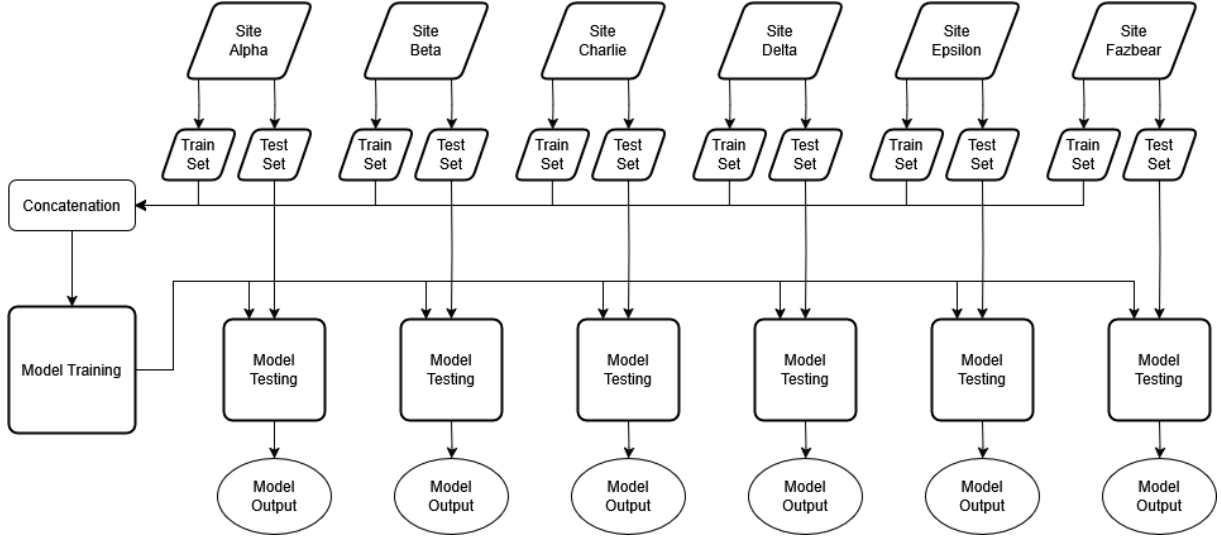


Figure 6.1: Dataset fusion procedure.

Data from each site are split into train and test sets at a ratio of 80:20. Training sets are concatenated into one datasets, and each test set is tested separately.

6.1.2 Centroid Distance Calculation

Five strategies for data reduction were developed for this paper, four of which use centroid distance as an identifier for reduction. Algorithm 2 shows the process of calculating each centroid distance.

Algorithm 2 Centroid distance calculation

```

1: for Each Dataset do
2:   Split Data into 2 classes: Occupied & Not Occupied
3:   for Each Feature in Each Class do
4:     Feature Centroids = Calculate Mean Average of Each Feature
5:     for Each Datapoint in Dataset do
6:       Feature Distances = Difference Between Feature And Feature Centroid
7:       Centroid Distance = Linear Normalisation of all Feature Distances
8:     end for
9:   end for
10: end for

```

6.1.3 Data Reduction Strategies

Once data centroids are calculated, they are used to identify which data to remove from the dataset for training. This chapter contains two experiments for data reduction: data balancing through undersampling and data reduction on both classes, or pure data reduction. Data balancing aims to set the class distribution to 50:50 to alleviate the effects of class imbalance and to reduce the amount of data used by removing data from the dominant class. For example, if a dataset has two classes with 100 and 300 datapoints per class, the data balance methods aim to reduce the majority class from 300 datapoints to 100. In total, this would be a reduction of $((300 - 100)/(100 + 300)) = 50.0\%$. As this is quite a large reduction, the experimentation described below caps the amounts of reduction by 5%, 10%, 25%, 50%, 75%, as well as the maximum. Limiting the amount of reduction performed allows us to observe the effects of varying the amount of reduction and identify the best amount to use for the HPDMobile dataset. Pure data reduction aims to identify the effects of reducing data in both classes indiscriminately. For

consistency between experiments, the same reduction caps are used as above for pure data reduction. As each site dataset has a different class balance, the maximum amount of data to remove to balance the classes differs across datasets. Datasets with greater class imbalance require more reduction to balance the classes, and vice versa.

Figure 6.2 shows the reduction methods developed. These are as follows:

- Random exclusion, where random datapoints are removed from the training set.
- Central exclusion, where datapoints with the smallest class centroid distance are removed.
- Lateral exclusion, where datapoints with the largest class centroid distance are removed.
- Data even, where datapoints from the largest density of class centroid distances are removed, effectively cutting the top off the tallest columns in the centroid distribution plots.
- Data squash, where an amount of datapoints proportional to the density of each of 10 bins of data is removed from each bin, effectively flattening all columns in the centroid distribution plots, proportionally to the size of each column.

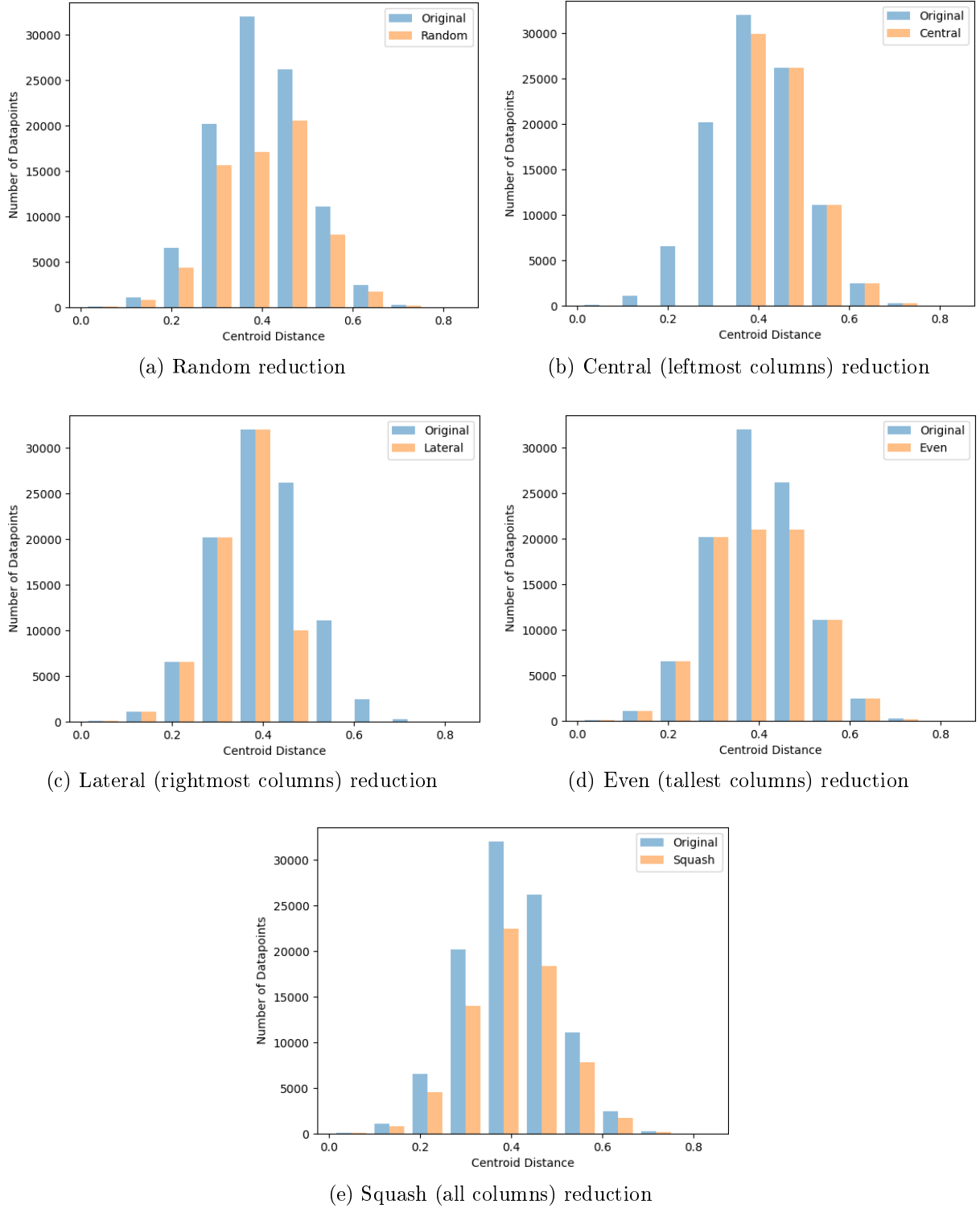


Figure 6.2: Centroid distance reduction strategies used in this chapter, showing which data is removed for each strategy, based on centroid distance.

The central and lateral exclusion methods are based on similar work on 2D image data [124], and data even and data squash were developed for this chapter.

6.1.4 Class Density Calculation

Class density, or label density [124] [179], is a measure of the aggregate similarity of datapoints within each class of a dataset. A low class density suggests the datapoints of that class are unique,

while a high class density suggests many datapoints hold the same features. For the latter case, it stands to reason that similar datapoints may be removed to reduce a dataset without taking away key features from it.

Equation 1 [124] shows how class density is calculated, where d is the density for class i for all n classes, where c_i is the count of samples for that class; σ_i is the standard deviations of the m Gaussians of the m -dimensional class i .

$$d_i = n \cdot c_i \cdot \left(\sum_j^n c_j \right)^{-1} \cdot \left(\frac{1}{m} \sum_k^m \sigma_{i_k} \right)^{-1} \quad (6.1)$$

For each experiment in this chapter, the class densities are calculated to identify the effect the different reduction strategies have on class density. With this knowledge, and the corresponding model performance, the importance of specific data to overall model performance may be observed. Class density may also be used to identify if a dataset may be reduced before attempting any data reduction; findings by [124] suggest that by reducing data from the denser classes and converging each class's density towards a value of 1, model performance may be maintained. This chapter aims to identify if this is true for the HPDMobile dataset and, by extension, other time-series datasets.

6.1.5 Metrics and Model

Accuracy is traditionally used to measure the performance of AI models, but especially in the case of unbalanced datasets, it is known to give bias to the majority class in a phenomenon known as the accuracy paradox [180]. To avoid this, model performance is also measured by the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) [181]. Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is a single value that leverages a model's sensitivity against its specificity instead of considering these metrics individually, and is considered a more descriptive metric over accuracy for biased datasets for binary classification tasks. The experimental results in this chapter show accuracy and AUC-ROC. The p -values of each experiment are also given to show the statistically significant difference between the results and the test benchmark. A p -value below 0.05 is selected to identify statistical significance.

Multiple models were considered for the experimentation, and preliminary testing was performed on each to identify the most suitable model for the rest of the experimentation. Those included were Random Forest (RF), XGBoost, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) models. The RF and XGBoost models are available as part of the scikit-learn library, and the CNN and LSTM models were created using the Pytorch library. The RF and XGBoost model configurations are shown in table 6.2, and the CNN and LSTM model configurations are shown in table 6.3.

Table 6.2: Random Forest and XGBoost model hyperparameters.

Model	Max depth	N. estimators	Tree method
Random Forest	Unlimited	100	-
XGBoost	Unlimited	100	Approx.

Table 6.3: Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network model hyperparameters.

Model	Layer configuration	Learning rate	Optimiser	Loss function	Data window size
CNN	3 conv. layers with batch norm. 2 dense layers	0.001	Adam	Binary cross-entropy	6 datapoints
LSTM	4 LSTM layers, 1 dense layer Hidden layer size: 250 Non-bidirectional	0.001	Adam	Binary cross-entropy	6 datapoints

Table 6.4 shows the results from preliminary testing on Site Alpha of the HPDMobile dataset. Preliminary results show that the RF model was the best-performing model. The RF and XGBoost models are the simpler models to train with, while the CNN and LSTM models require the input data’s sequencing to be preserved, which adds a level of complexity which may be avoided by using a model that does not require sequencing preservation. Also, as one objective of this study is for it to be deployable on edge devices with computational constraints, a computationally simpler model would be preferred. The Random Forest model was selected as a simple single-loop model for classification.

Table 6.4: Preliminary test results on RF, XGBoost, CNN, and LSTM models with HPDMobile dataset Site Alpha.

Model	Accuracy	AUC-ROC
RF	98.744%	97.143%
XGBoost	95.128%	93.054%
CNN	91.021%	89.783%
LSTM	85.470%	85.393%

6.1.6 Hardware and Power Calculation

As the focus of this chapter is to reduce the cost of operation of occupancy classification on edge devices, it is important to consider the hardware the experiments are carried out on. Due to the large number of experiments and the amount of data logging that will be performed, experimentation is performed on a desktop PC. The code is designed to be transferable to an IOT device for deployment, but in the context of this chapter the following hardware is used for experimentation:

- CPU: Intel i7-11700k;
- RAM: 16 GB DDR4;
- OS: Windows 10.

The power consumption of the CPU is measured using HWiNFO software [182] while the model is trained. For the processor used, the power consumption is 14 W when idle and 46 W when busy. The power consumption of the program is the difference between these two: 32 W. This value is consistent between experiments regardless of tree depth or dataset size, although these factors instead increase the runtime of training. In the UK, according to the Department for Energy Security and Net Zero [168], this translates to 6.623 g of CO₂ emissions per hour.

6.2 Experimental Results

6.2.1 Experiments on Individual Sites

For each experiment performed, the datasets were split into training and testing sets at a ratio of 80:20. Each experiment was run five times and results were averaged.

Experimentation was performed to observe the effects of reducing the data by varying amounts, from different areas in the data distribution. Due to the different class balances of each dataset, class balancing removes more data for more imbalanced classes, and vice versa. Table 6.5 shows the maximum percentage reduction of the larger class, and the dataset overall. It also shows the densities of the classes, derived from research in [124]. Class density will be used as a metric to identify suitability for data reduction.

Table 6.5: HPDMobile class balance and class density properties, and maximum reduction amounts after class balancing.

Site	Number of Datapoints	Class Balance (Not Occ:Occ)	Balanced Class Max Reduction	Total Dataset Reduction at Max Balancing	Class Density (Not Occ:Occ)
Alpha	147,750	20:80	74.912%	59.887%	0.674:1.585
Beta	146,879	40:60	34.599%	20.918%	1.068:1.201
Charlie	302,399	22:78	72.111%	56.386%	0.639:1.532
Delta	146,879	21:79	77.569%	63.358%	0.547:1.576
Epsilon	129,599	24:76	67.755%	51.235%	0.886:1.392
Fazbear	328,319	47:53	12.111%	6.446%	1.260:0.892

Experimental Benchmark

Before data reduction was performed, the models were trained with the full dataset to acquire the benchmark results. These results are shown in Table 6.6. The runtime of each of these experiments was less than one minute.

Table 6.6: Experimental benchmarks of accuracy and AUC-ROC with RF model.

Site	Accuracy	AUC-ROC
Alpha	98.813%	98.812%
Beta	99.613%	99.613%
Charlie	99.755%	99.612%
Delta	99.589%	99.271%
Epsilon	99.692%	99.574%
Fazbear	99.367%	99.368%

Site Alpha

Site Alpha has 147,750 datapoints and a class balance of 20:80. Figure 6.3 shows the results of varying degrees of data reduction on the single dataset. Tables 6.7 and 6.8 show the p -values of the AUC-ROC of each experiment.

Figures 6.3a,b show that model accuracy decreases steadily as more data are removed, regardless of whether the removed data are from the majority class or both. Figures 6.3c,d, show a similar drop in performance. The AUC-ROC score may be maintained with majority class reduction, up to 50%. These results are interesting, as the accuracy up to this reduction amount

has steadily decreased. By performing reduction in this way, the model’s ability to avoid false positives and false negatives may be improved. It is also important to note that the p -values of all experiments corresponding to majority class reduction indicate that the results are not statistically differentiable from the benchmark, apart from with the maximum reduction. For the maximum reduction, the performance is clearly worse, hence the differentiation. For every other case, performance is maintained while reducing the amount of data.

Tables 6.9 and 6.10 show the class densities for each experiment. Table 6.9 shows that, for each reduction method, the densities of each class become closer, up to a reduction cap of 50%. At maximum reduction, the minority class, class 0, has a greater class density than class 1. At the same time, both the accuracy and the AUC-ROC of the model decrease by a relatively large amount, supporting the theory that data may be reduced in order to balance the density of each class towards a value of 1, but further reduction that leads to an imbalance causes the model performance to deteriorate. Also, Table 6.10 shows that by performing data reduction on both classes, the difference in class density between classes 0 and 1 does not change by a significant amount. The AUC-ROC decreases as data is reduced, as expected when reducing data, without benefiting from the performance boost given by class balancing.

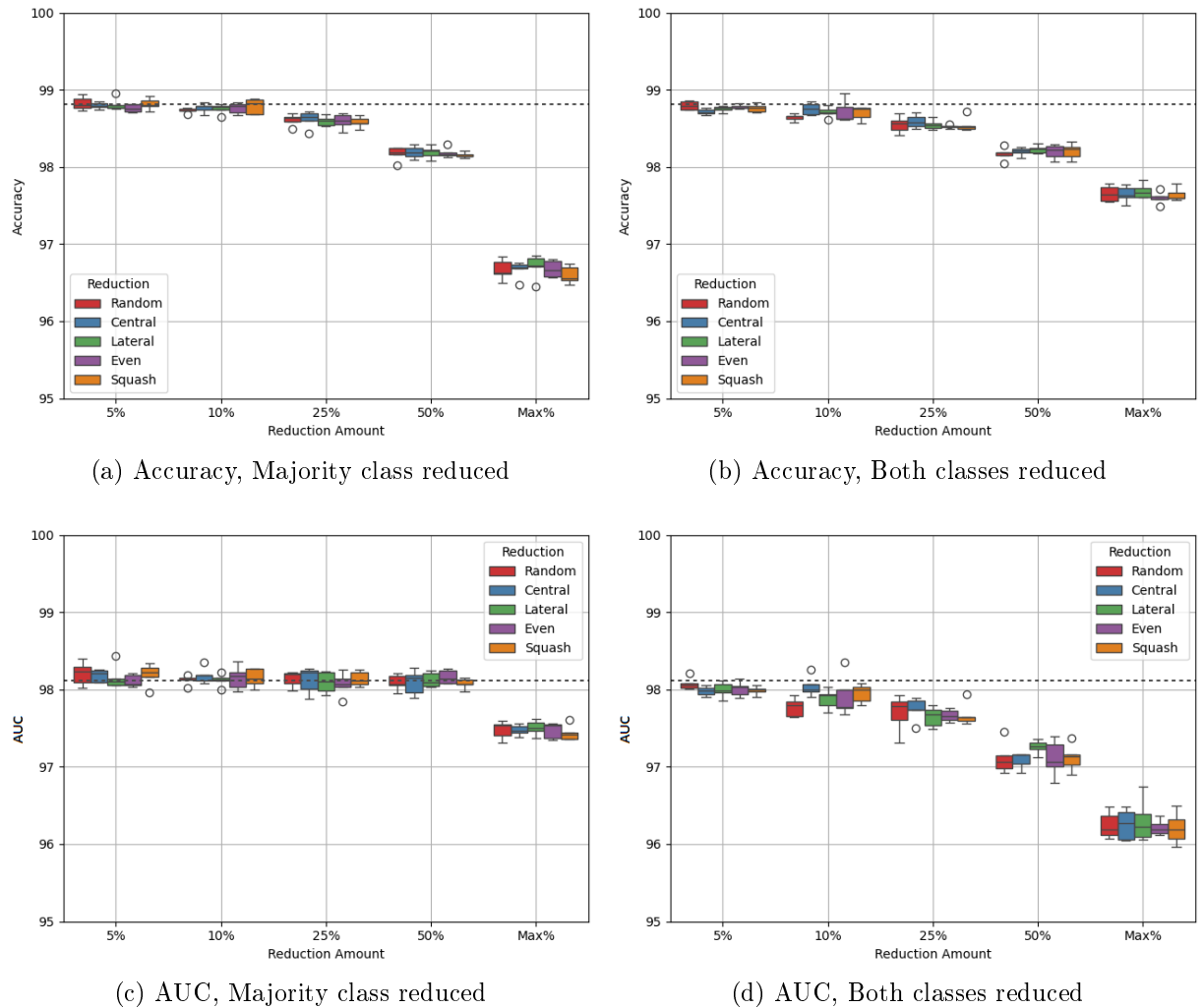


Figure 6.3: Experimental results for Site Alpha test set.

Table 6.7: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Alpha test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.268	0.182	0.616	0.794	0.329
10%	0.971	0.255	0.906	0.652	0.625
25%	0.74	0.972	0.688	0.482	0.673
50%	0.605	0.731	0.948	0.341	0.239
74%	$2.25 \times 10^{-4} *$	$2.63 \times 10^{-5} *$	$1.49 \times 10^{-4} *$	$1.59 \times 10^{-4} *$	$1.18 \times 10^{-4} *$

Table 6.8: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Alpha test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.273	$7.04 \times 10^{-3} *$	$4.42 \times 10^{-2} *$	5.92×10^{-2}	$5.56 \times 10^{-3} *$
10%	$3.51 \times 10^{-3} *$	0.267	$1.44 \times 10^{-2} *$	0.16	$3.48 \times 10^{-2} *$
25%	$1.71 \times 10^{-2} *$	$5.21 \times 10^{-3} *$	$1.21 \times 10^{-3} *$	$1.98 \times 10^{-4} *$	$2.83 \times 10^{-3} *$
50%	$3.96 \times 10^{-4} *$	$2.56 \times 10^{-5} *$	$2.64 \times 10^{-5} *$	$7.09 \times 10^{-4} *$	$2.11 \times 10^{-4} *$
74%	$1.85 \times 10^{-5} *$	$3.14 \times 10^{-5} *$	$1.25 \times 10^{-4} *$	$1.73 \times 10^{-6} *$	$3.52 \times 10^{-5} *$

Table 6.9: Class density of Site Alpha dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.707	0.712	0.670	0.687	0.704	1.565	1.565	1.582	1.565	1.551
10%	0.739	0.721	0.722	0.756	0.740	1.565	1.562	1.552	1.511	1.550
25%	0.812	0.875	0.848	0.848	0.882	1.507	1.485	1.482	1.389	1.447
50%	1.112	1.185	1.176	1.161	1.133	1.324	1.308	1.315	1.112	1.311
74%	1.782	1.734	1.725	1.796	1.621	1.008	1.010	0.977	0.680	0.968

Table 6.10: Class density of Site Alpha dataset after data reduction across both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.720	0.706	0.719	0.643	0.707	1.549	1.555	1.579	1.591	1.572
10%	0.782	0.669	0.634	0.704	0.709	1.565	1.579	1.598	1.532	1.557
25%	0.792	0.752	0.781	0.635	0.622	1.562	1.573	1.580	1.531	1.597
50%	0.707	0.716	0.735	0.595	0.633	1.571	1.569	1.582	1.458	1.579
74%	0.766	0.804	0.850	0.554	0.541	1.556	1.571	1.622	1.340	1.594

Site Beta

Site Beta has 146,879 datapoints and a class balance of 40:60. Site Beta has relatively few datapoints and less class imbalance than the others in the HPDMobile dataset, causing a lower maximum reduction of 34%. Figure 6.4 shows the results of varying degrees of data reduction on the Site Beta dataset, and Tables 6.11 and 6.12 show the p -values of the AUC-ROC of each experiment.

Both the accuracy and AUC-ROC values change by less than 0.2% as the amount of data is reduced, due to much less reduction being required to balance the classes, compared to the reduction performed for Site Alpha. However, there is still a very slight drop in both accuracy and AUC-ROC. Most of the p -values show no statistically significant difference from the benchmark, except for some of the more extreme reduction amounts. These results show a larger difference from the benchmark, which raises questions about the stability of the model after reducing the dataset.

Tables 6.13 and 6.14 show that the densities of both classes are values above 1 for all experiments except the maximum reduction of the majority class, where class 1's density is between 0.9 and 1. These results suggest that the reduction methods performed are not enough to shift the densities towards 1. Alternative methods may be required to optimise datasets like Site Beta, where classes are already closely balanced.

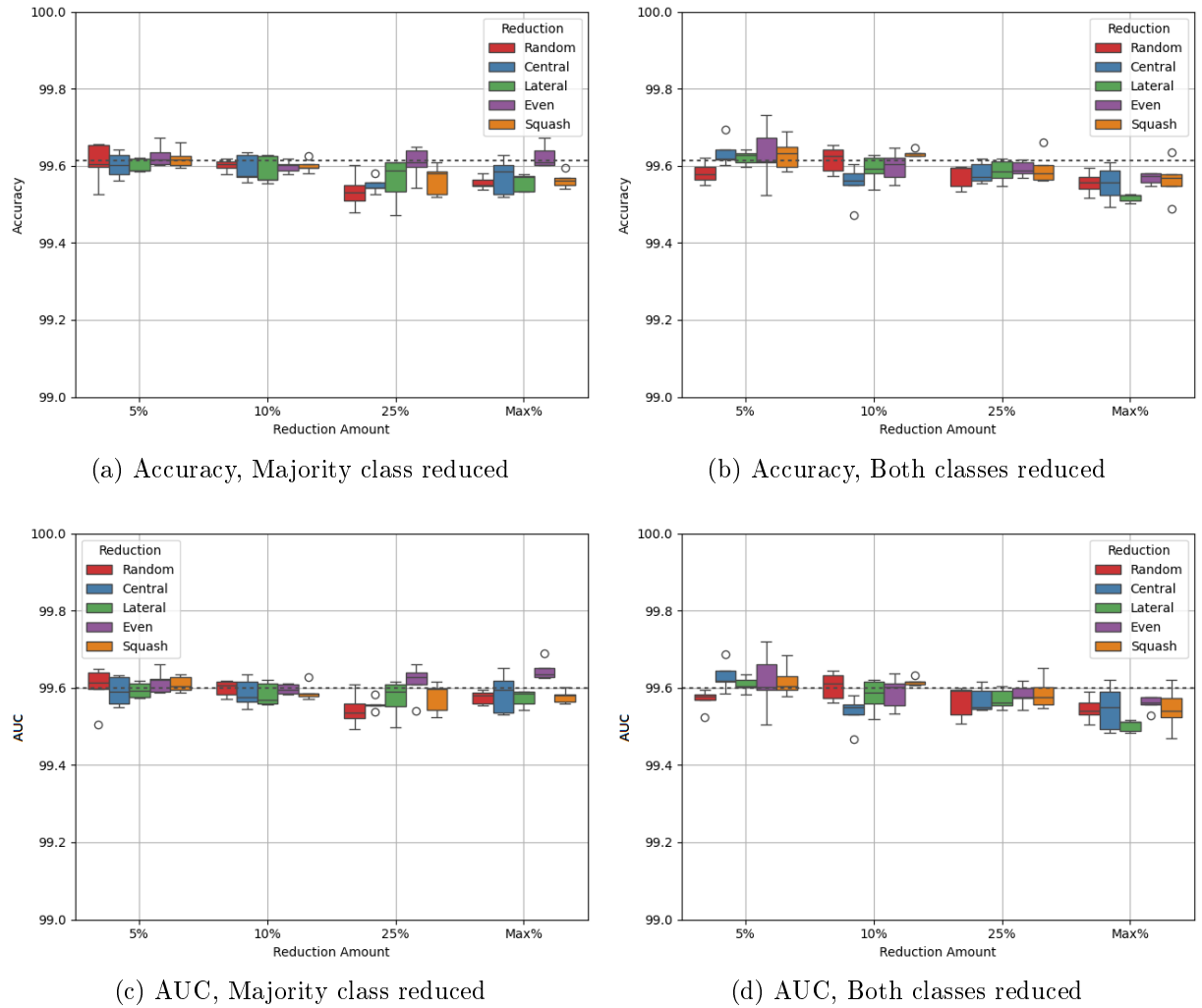


Figure 6.4: Experimental results for Site Beta test set.

Table 6.11: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Beta test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.955	0.684	0.591	0.258	0.325
10%	0.929	0.486	0.32	0.672	0.376
25%	4.65×10^{-2} *	4.82×10^{-3} *	0.297	0.483	0.251
34%	3.47×10^{-2} *	0.589	5.31×10^{-2}	1.71×10^{-2} *	4.05×10^{-2} *

Table 6.12: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Beta test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	7.27×10^{-2}	0.151	0.349	0.657	0.368
10%	0.784	3.04×10^{-2} *	0.389	0.564	2.12×10^{-2} *
25%	0.142	0.111	7.13×10^{-2}	0.247	0.523
34%	1.80×10^{-2} *	0.124	1.38×10^{-4} *	1.00×10^{-2} *	9.75×10^{-2}

Table 6.13: Class density of Site Beta dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	1.111	1.132	1.112	1.114	1.126	1.172	1.153	1.164	1.157	1.149
10%	1.144	1.157	1.162	1.149	1.158	1.137	1.146	1.146	1.119	1.120
25%	1.277	1.276	1.279	1.263	1.298	1.055	1.047	1.035	1.004	1.021
34%	1.383	1.345	1.371	1.345	1.387	0.963	1.002	0.970	0.917	0.952

Table 6.14: Class density of Site Beta dataset after each data reduction method, with reduction on both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	1.083	1.097	1.087	1.091	1.071	1.195	1.180	1.198	1.180	1.189
10%	1.087	1.088	1.094	1.062	1.079	1.191	1.194	1.192	1.187	1.177
25%	1.057	1.078	1.085	1.047	1.061	1.205	1.198	1.186	1.156	1.185
34%	1.085	1.138	1.114	1.052	1.068	1.190	1.184	1.218	1.126	1.174

Site Charlie

Site Charlie has one of the larger class imbalances, and therefore larger maximum reduction caps, with a maximum reduction of 72%. It is also one of the larger datasets, with over 300,000

datapoints. Figure 6.5 shows the experimental results and Tables 6.15 and 6.16 show the p -values of the AUC-ROC of each experiment.

Figure 6.5a shows that up to a reduction cap of 50%, accuracy is above the benchmark, with the lateral data reduction method with a reduction cap of 10% performing best. With the maximum reduction cap (at 72.111%), however, the performance is below the benchmark for all strategies, indicating a delicate balance is needed for data reduction to ensure that too much data is not removed. The performance is further explained by the class densities; Table 6.17 shows that as more of the majority class is reduced, both classes' densities converge around a value of 1. At a reduction cap of 50%, the combined difference between each density and 1 is smallest, which is where the AUC-ROC is greatest across all strategies. However, the results are less stable, as shown by the size of each box. Furthermore, Table 6.18 shows that by reducing data across both classes, the model performance is worse.

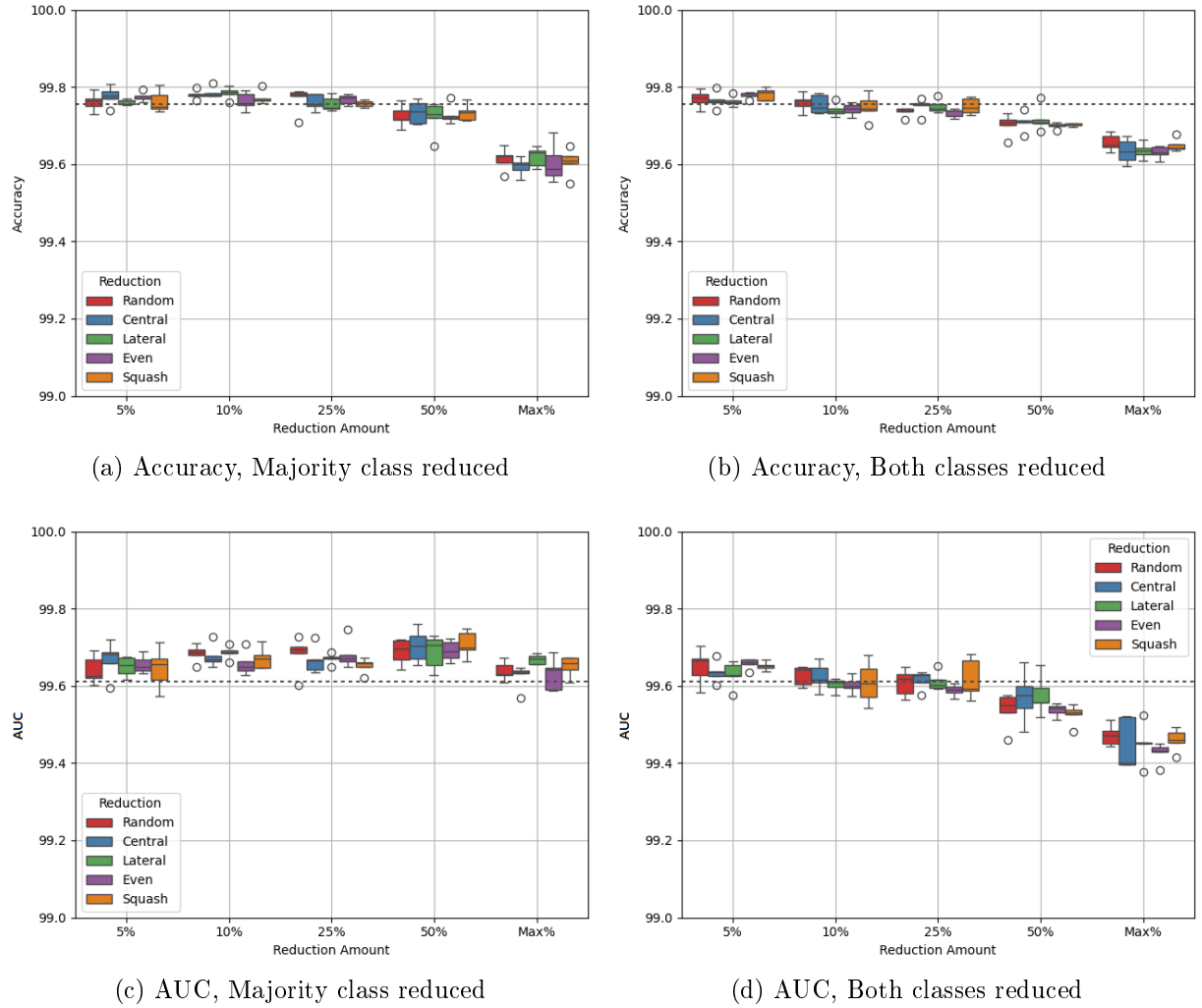


Figure 6.5: Experimental results for Site Charlie test set.

Table 6.15: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Charlie test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.14	4.96×10^{-2} *	3.08×10^{-2} *	1.18×10^{-2} *	0.215
10%	2.05×10^{-3} *	7.93×10^{-3} *	6.49×10^{-4} *	2.93×10^{-2} *	8.29×10^{-3} *
25%	2.96×10^{-2} *	2.23×10^{-2} *	6.12×10^{-4} *	1.41×10^{-2} *	8.71×10^{-3} *
50%	6.60×10^{-3} *	9.01×10^{-3} *	1.74×10^{-2} *	2.63×10^{-3} *	3.41×10^{-3} *
72%	7.02×10^{-2}	0.389	4.31×10^{-4} *	0.365	2.89×10^{-2} *

Table 6.16: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Charlie test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.138	0.116	0.299	1.71×10^{-3} *	1.23×10^{-3} *
10%	0.233	0.461	0.3	0.396	0.924
25%	0.863	0.705	0.994	2.66×10^{-2} *	0.804
50%	2.42×10^{-2} *	0.255	0.194	6.01×10^{-4} *	2.02×10^{-3} *
72%	3.48×10^{-4} *	5.56×10^{-3} *	2.32×10^{-3} *	9.48×10^{-5} *	3.73×10^{-4} *

Table 6.17: Class density of Site Charlie dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.663	0.662	0.664	0.663	0.661	1.512	1.524	1.519	1.511	1.513
10%	0.689	0.689	0.690	0.693	0.695	1.492	1.493	1.491	1.487	1.489
25%	0.794	0.793	0.793	0.791	0.791	1.449	1.423	1.421	1.403	1.414
50%	1.046	1.033	1.046	1.048	1.042	1.260	1.271	1.222	1.202	1.251
72%	1.458	1.466	1.460	1.457	1.467	0.985	0.981	0.991	0.891	0.957

Table 6.18: Class density of Site Charlie dataset after data reduction across both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.636	0.637	0.640	0.637	0.636	1.531	1.541	1.531	1.529	1.530
10%	0.638	0.639	0.639	0.632	0.637	1.534	1.532	1.531	1.525	1.528
25%	0.635	0.635	0.638	0.636	0.635	1.538	1.537	1.526	1.512	1.522
50%	0.639	0.636	0.635	0.633	0.633	1.533	1.538	1.526	1.489	1.512
72%	0.634	0.646	0.638	0.631	0.631	1.556	1.563	1.554	1.445	1.509

Site Delta

Site Delta has similar properties to Site Alpha, with 146,879 datapoints and a class balance of 21:79. Figure 6.6 shows the experimental results, and Tables 6.19 and 6.20 show the p -values of the AUC-ROC of each experiment.

Figures 6.6a,b show similar behaviour to each other, where accuracy drops as greater reduction is performed. Figure 6.6c shows that the AUC-ROC is maintained up to a 75% reduction for class balance. However, Figure 6.6d shows that the AUC-ROC drops by a larger amount with reduction across both classes. Site Delta has a large class imbalance, which suggests that some balancing is important for optimal results. The p -values of the results for all but three experiments (data squash at 10% and maximum reduction and lateral reduction at 25% reduction) show no statistically significant difference from the benchmark, suggesting once again that through these methods of data reduction, optimal model performance is maintained.

Tables 6.21 and 6.22 show the class densities after each reduction method for Site Delta. As with the other sites, reduction on both classes does little to bring the class densities towards a value of 1. For majority class reduction, an optimal amount of reduction to achieve balanced class densities is between reduction caps of 50% and 75%, again demonstrating the delicate balance needed to find the optimal amount of data to remove for the best performance.

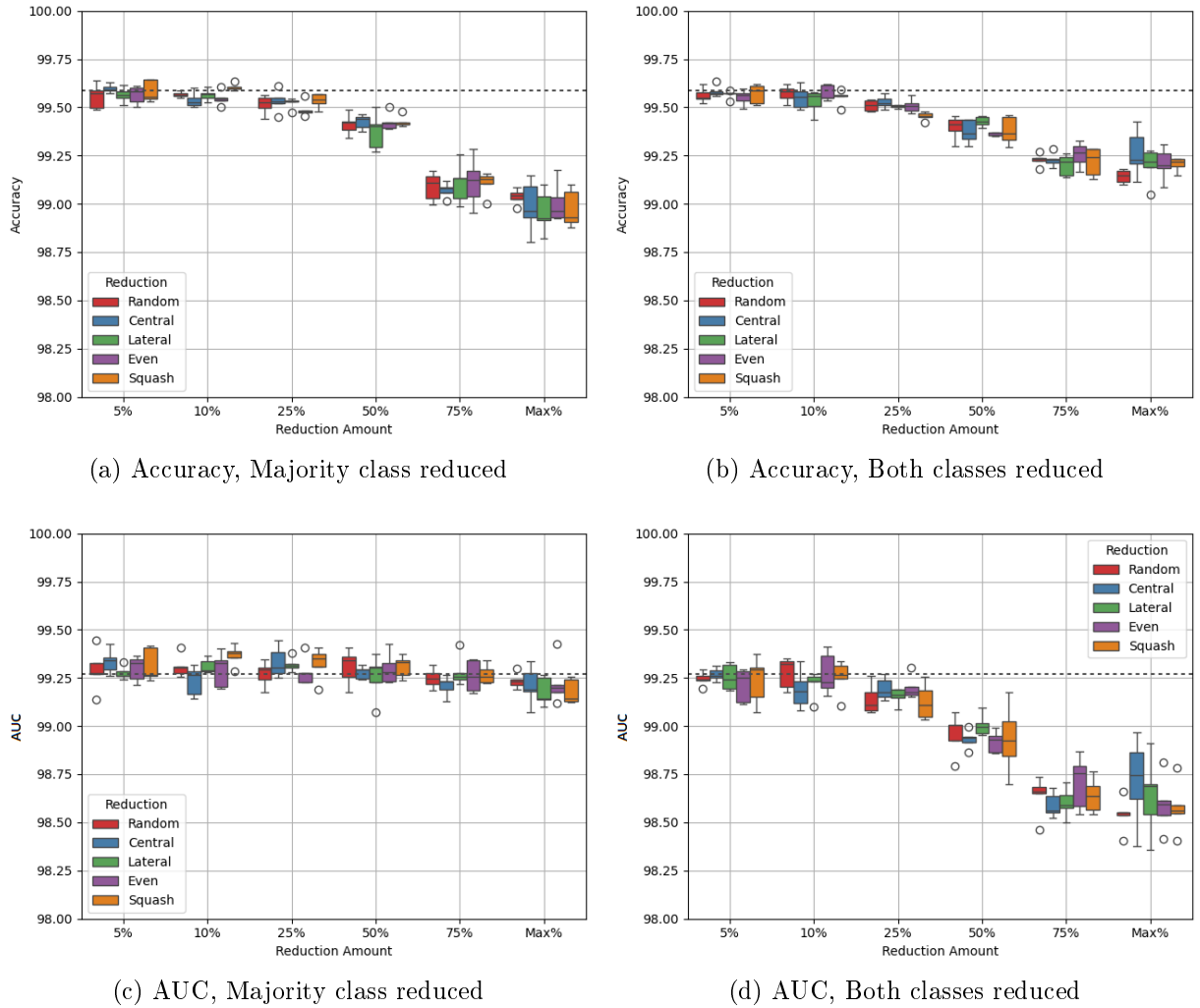


Figure 6.6: Experimental results for Site Delta test set.

Table 6.19: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Delta test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.704	8.17×10^{-2}	0.711	0.396	0.287
10%	0.199	0.35	7.70×10^{-2}	0.582	1.59×10^{-2} *
25%	0.949	0.168	4.32×10^{-2} *	0.757	0.212
50%	0.415	0.553	0.807	0.473	0.184
75%	0.375	5.83×10^{-2}	0.778	0.802	0.891
77%	0.113	0.254	5.27×10^{-2}	0.444	3.14×10^{-2} *

Table 6.20: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Delta test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.198	0.918	0.625	0.226	0.591
10%	0.869	0.144	0.155	0.967	0.686
25%	2.07×10^{-2} *	3.73×10^{-2} *	3.20×10^{-3} *	5.53×10^{-2}	2.61×10^{-2} *
50%	2.98×10^{-3} *	1.00×10^{-4} *	4.39×10^{-4} *	1.59×10^{-4} *	1.42×10^{-2} *
75%	1.70×10^{-4} *	2.11×10^{-5} *	4.50×10^{-5} *	8.12×10^{-4} *	1.02×10^{-4} *
77%	5.72×10^{-5} *	5.59×10^{-3} *	2.34×10^{-3} *	4.75×10^{-4} *	3.29×10^{-4} *

Table 6.21: Class density of Site Delta dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.578	0.561	0.586	0.556	0.559	1.566	1.573	1.553	1.549	1.550
10%	0.633	0.586	0.603	0.633	0.590	1.534	1.559	1.544	1.504	1.523
25%	0.671	0.679	0.662	0.675	0.645	1.475	1.520	1.503	1.414	1.468
50%	0.932	0.969	0.938	0.954	0.914	1.334	1.412	1.352	1.152	1.294
75%	1.354	1.494	1.484	1.493	1.420	1.052	1.052	1.103	0.737	0.957
77%	1.506	1.456	1.583	1.531	1.509	1.074	1.062	0.969	0.685	0.909

Table 6.22: Class density of Site Delta dataset after data reduction across both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.533	0.544	0.558	0.577	0.578	1.577	1.572	1.573	1.553	1.555
10%	0.563	0.516	0.561	0.542	0.557	1.605	1.600	1.579	1.549	1.557
25%	0.627	0.545	0.546	0.518	0.567	1.581	1.577	1.572	1.512	1.540
50%	0.602	0.634	0.559	0.515	0.537	1.663	1.591	1.573	1.416	1.538
75%	0.615	0.569	0.581	0.464	0.632	1.639	1.704	1.634	1.299	1.498
77%	0.573	0.562	0.663	0.445	0.629	1.688	1.593	1.615	1.278	1.474

Site Epsilon

Site Epsilon is the smallest site in the dataset, with only 129,599 datapoints. It is also among the most imbalanced of the site datasets, with a class balance of 24:76. Figure 6.7 shows the experimental results, and Tables 6.23 and 6.24 show the p -values of the AUC-ROC of each experiment.

Much like with Site Delta, Figure 6.7a,b show that data reduction causes a drop in performance of up to 0.4%. Like the experiments for sites Alpha, Beta, and Delta, the AUC-ROC does not decrease with accuracy until the maximum reduction.

Table 6.25 shows that with each reduction cap, the density of class 0 increases while the density of class 1 decreases, as with the other sites. However, class 0's density passes through a value of 1 between reduction caps of 10% and 25%, while class 1 passes through a value of 1 between a reduction cap of 50% and the maximum (for all reduction methods except random). Such variation in the class density across experiments makes it difficult to identify the best reduction cap for this dataset. Perhaps an alternative method of data reduction would be more appropriate. Table 6.26 shows that, similarly to sites Charlie and Delta, as data is reduced the densities move further from a value of 1, while both accuracy and AUC-ROC drop at the same time.

Table 6.23: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Epsilon test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.761	0.618	2.51×10^{-2} *	0.376	0.919
10%	6.61×10^{-2}	0.373	0.541	0.844	0.522
25%	0.745	0.765	0.207	0.173	0.991
50%	0.186	1.91×10^{-2} *	8.94×10^{-2}	1.57×10^{-2} *	0.282
67%	1.03×10^{-3} *	8.73×10^{-3} *	1.79×10^{-3} *	1.47×10^{-2} *	1.37×10^{-3} *

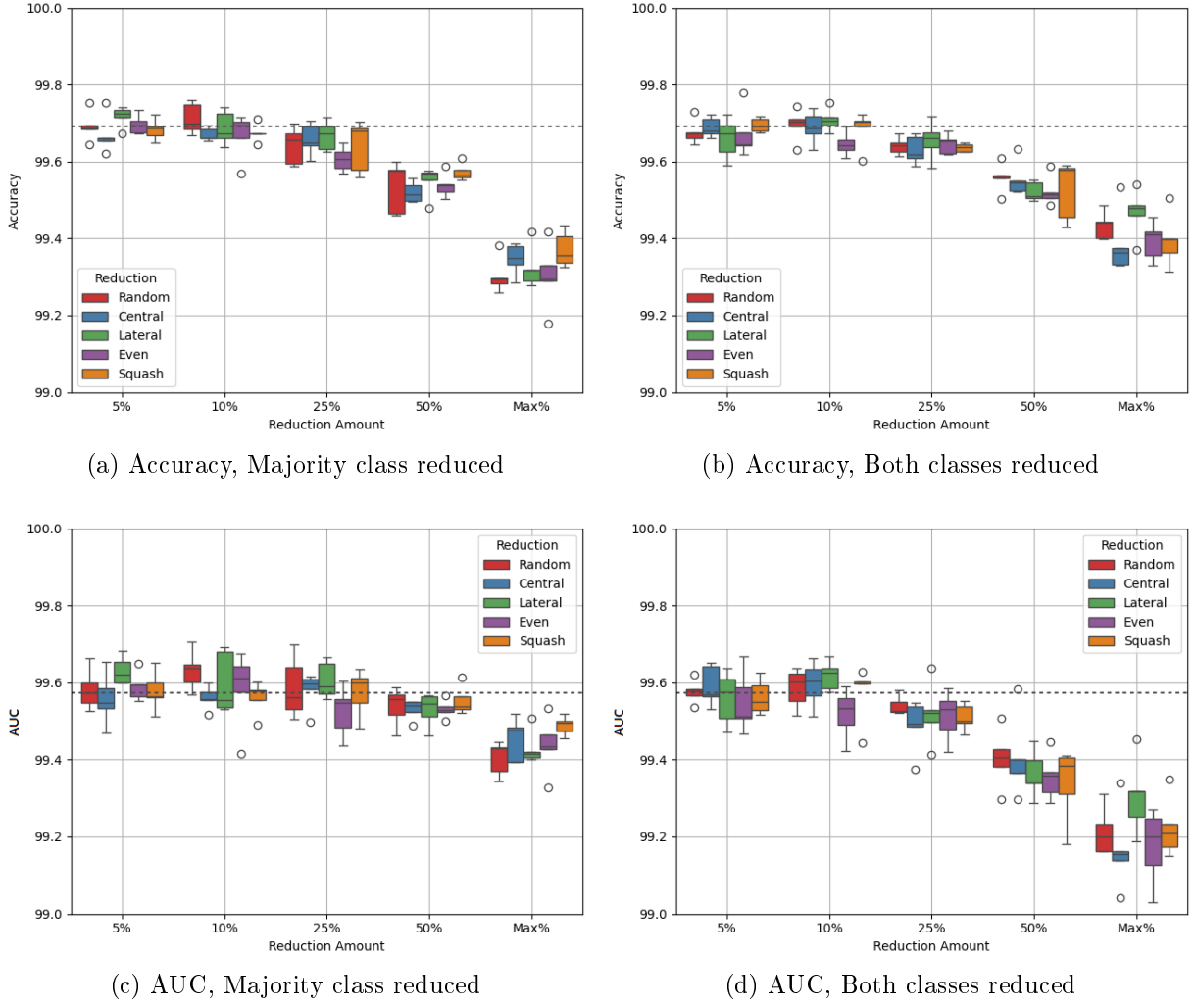


Figure 6.7: Experimental results for Site Epsilon test set.

 Table 6.24: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Epsilon test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.856	0.513	0.676	0.513	0.593
10%	0.639	0.447	6.03×10^{-2}	0.127	0.997
25%	3.89×10^{-2} *	4.75×10^{-2} *	0.203	0.105	1.51×10^{-2} *
50%	7.18×10^{-3} *	2.48×10^{-2} *	1.58×10^{-3} *	1.21×10^{-3} *	5.37×10^{-3} *
67%	2.01×10^{-4} *	1.07×10^{-3} *	3.55×10^{-3} *	8.09×10^{-4} *	5.19×10^{-4} *

Table 6.25: Class density of Site Epsilon dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.862	0.932	0.878	0.909	0.907	1.383	1.378	1.391	1.363	1.370
10%	0.944	0.946	0.956	0.969	0.954	1.377	1.341	1.369	1.325	1.340
25%	1.061	1.041	1.079	1.115	1.050	1.319	1.302	1.279	1.207	1.265
50%	1.406	1.400	1.458	1.406	1.456	1.200	1.124	1.168	0.963	1.097
67%	1.834	1.790	1.813	1.836	1.814	1.008	0.977	0.973	0.692	0.935

Table 6.26: Class density of Site Epsilon dataset after data reduction across both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	0.906	0.898	0.876	0.884	0.871	1.394	1.386	1.403	1.382	1.387
10%	0.850	0.874	0.873	0.877	0.877	1.406	1.386	1.386	1.372	1.381
25%	0.886	0.883	0.882	0.883	0.875	1.392	1.379	1.417	1.335	1.367
50%	0.858	0.863	0.883	0.878	0.874	1.404	1.443	1.422	1.262	1.365
67%	0.864	0.873	0.883	0.884	0.876	1.349	1.446	1.417	1.183	1.371

Site Fazbear

Site Fazbear is the largest of the sites, with 328,319 datapoints, and the only site to feature more datapoints in the ‘Occupied’ class, class 1. It is also the most balanced site, with a class balance of 47:53, giving a maximum reduction of 12.111%. Figure 6.8 shows the experimental results, and Tables 6.27 and 6.28 show the p -values of the AUC-ROC of each experiment.

Before analysing the box and whisker plots, the p -values indicate no statistically significant difference from the benchmark, except with 5% reduction on the data squash method. For this experiment, the AUC-ROC is slightly better than the benchmark. For all other experiments, there is no statistical significance. The lack of statistical significance can be explained by the very small amount of reduction performed for this dataset, due to the natural class balance of 47:53. Considering the class densities, shown by Tables 6.29 and 6.30, class densities reach values closer to 1 with experiments with reduction of both classes, unlike for the previous sites. The values reach just under 1.2 for class 0 and just above 0.95 for class 1 at the maximum reduction on both classes. There is very little change in class densities, like with AUC-ROC, because of the small amount of reduction performed, indicating that the reduction method of balancing datasets is not aggressive enough for already closely balanced datasets.

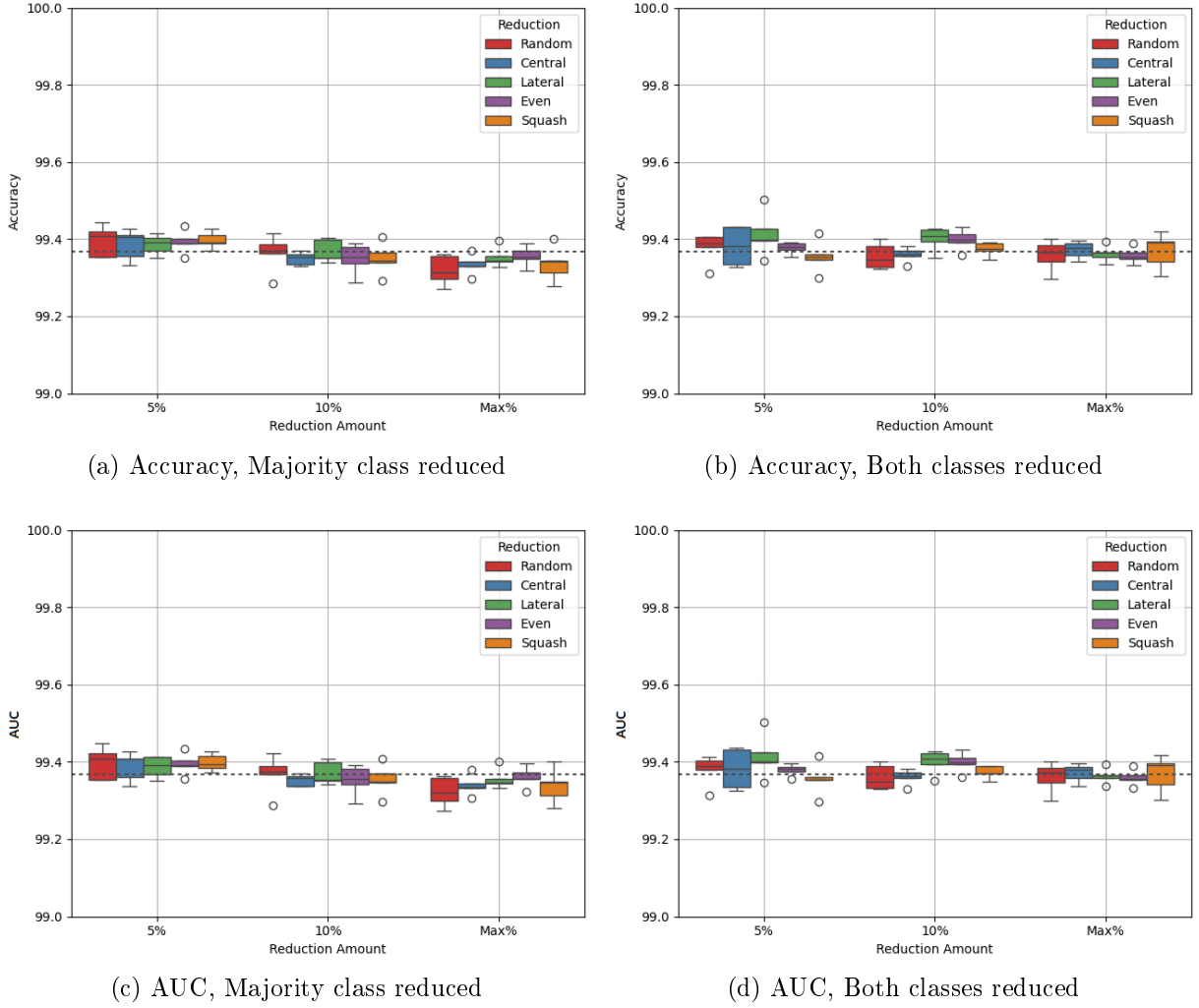


Figure 6.8: Experimental results for Site Fazbear test set.

 Table 6.27: p -values of the AUC metrics of each experiment of reduction on the majority class, on Site Fazbear test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.191	0.297	0.182	0.101	3.72×10^{-2} *
10%	0.993	9.94×10^{-2}	0.834	0.435	0.489
12%	6.08×10^{-2}	7.90×10^{-2}	0.384	0.594	0.207

 Table 6.28: p -values of the AUC metrics of each experiment of reduction on both classes, on Site Fazbear test set. Values marked with an * have a statistically significant difference from the benchmark.

Reduction	Random	Central	Lateral	Even	Squash
5%	0.534	0.574	0.138	0.18	0.568
10%	0.596	0.476	6.91×10^{-2}	5.22×10^{-2}	0.47
12%	0.727	0.742	0.669	0.442	0.929

Table 6.29: Class density of Site Fazbear dataset after data reduction on the majority class only. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	1.221	1.216	1.208	1.190	1.213	0.918	0.925	0.926	0.931	0.918
10%	1.208	1.172	1.194	1.158	1.169	0.945	0.952	0.944	0.950	0.949
12%	1.174	1.163	1.153	1.154	1.151	0.956	0.958	0.966	0.952	0.959

Table 6.30: Class density of Site Fazbear dataset after data reduction across both classes. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Data Reduced	Class 0 (Not Occupied)					Class 1 (Occupied)				
	R	C	L	E	S	R	C	L	E	S
5%	1.263	1.249	1.256	1.239	1.242	0.896	0.896	0.896	0.893	0.893
10%	1.247	1.268	1.269	1.233	1.229	0.893	0.885	0.900	0.885	0.898
12%	1.240	1.257	1.232	1.249	1.230	0.897	0.893	0.903	0.877	0.895

6.2.2 Experiments on Fused Dataset

By fusing all of the sites into one large dataset, the ability of a single model to generalise on multiple different test sets from different environments may be observed. It may then be possible to see if data reduction will benefit the model further, as it can with the individual site datasets. Table 6.31 shows the details of the fused dataset. It has 1,201,825 datapoints, whereas the largest individual site has just under 330,000.

Table 6.31: Properties of the fused dataset created in this chapter.

Total Number of Datapoints	Number of Training Datapoints	Class Balance Not Occ:Occ	Class Density Not Occ:Occ
1,201,825	961,464	32:68	0.761:1.294

Experimental Benchmark

Table 6.32 shows the benchmark results of experimentation on the fused dataset, with no reduction. The model is trained on a fused training set and tested on individual site test sets.

Table 6.32: Fused dataset benchmark experimental results.

Site	Accuracy	AUC-ROC
Alpha	79.825%	49.920%
Beta	77.936%	76.426%
Charlie	84.226%	64.675%
Delta	81.818%	50.495%
Epsilon	75.237%	51.993%
Fazbear	68.796%	70.613%

For all sites, the accuracy and AUC-ROC are lower than their non-fused counterparts. While the non-fused testing yielded accuracies and AUC-ROC scores around 98–99%, testing on the fused datasets gives accuracies between 68–84% and AUC-ROC values from 49–76%. The low AUC-ROC scores indicate that the model is no better at classification than random chance, and the disparity between low AUC-ROC and high accuracy is caused by the large class imbalance of the data; the sites with the most balanced classes (sites Beta and Fazbear) have the highest AUC-ROC scores. Such a significant decrease in performance shows that despite the training sets having more data, the model is unable to classify well on the unique test sets. The runtime of training the fused dataset is 2.3 minutes, a relatively large increase from the sub-minute runtime of individual training data subsets.

Reduced fused dataset

Table 6.33 shows the properties of the reduced fused dataset. Over half of the majority class was reduced to balance the classes, giving a total dataset reduction of 38.9%.

Table 6.34 shows the accuracy of the RF model trained on the reduced fused dataset. Table 6.35 shows the AUC-ROC. Table 6.36 shows the class densities of the fused dataset after each reduction method.

Table 6.33: Properties of the reduced fused dataset created in this chapter.

Number of Datapoints	Balanced Class Reduction	Total Dataset Reduction
618,950	52.534%	35.642%

Table 6.34: Reduced fused dataset accuracies. Values in bold indicate best-performing reduction strategy

Test Site	Random	Central	Lateral	Even	Squash
Alpha	78.075%	78.498%	69.470%	78.495%	69.443%
Beta	78.312%	78.139%	78.194%	70.282%	69.835%
Charlie	70.424%	69.351%	69.548%	69.673%	69.216%
Delta	69.290%	70.062%	78.204%	78.522%	68.678%
Epsilon	69.612%	71.202%	69.707%	78.305%	77.943%
Fazbear	59.375%	59.457%	59.304%	59.661%	58.953%

Table 6.35: Reduced fused dataset AUC-ROC scores. Values in bold indicate best-performing reduction strategy

Test Site	Random	Central	Lateral	Even	Squash
Alpha	68.134%	68.165%	73.898%	68.567%	73.823%
Beta	68.029%	67.667%	67.946%	73.507%	74.004%
Charlie	73.736%	73.546%	73.736%	74.043%	73.538%
Delta	73.553%	73.100%	67.480%	67.665%	74.341%
Epsilon	74.012%	71.115%	74.051%	68.034%	67.932%
Fazbear	65.652%	65.427%	65.159%	65.626%	65.273%

Table 6.36: Class density of reduced fused dataset. R: random reduction, C: central reduction, L: lateral reduction, E: even reduction, S: squash reduction.

Class 0 (Not Occupied)					Class 1 (Occupied)				
R	C	L	E	S	R	C	L	E	S
1.187	1.207	1.202	1.193	1.190	0.969	0.942	0.944	0.830	0.937

Individual site analysis shows the effect of different reduction strategies on the performance with each test set. Site Alpha has a decrease in accuracy by a minimum of 1.3% with the Random, Central and Even strategies, while AUC-ROC is increased by 18.5%. For the Lateral and Squash strategies, accuracy decreases by just over 10%, but AUC-ROC increased by 28.5%. Site Beta's accuracy increases by less than 1% for the Random, Central and Lateral strategies (with Random being the best), but 7-8% worse with the Even and Squash strategies. AUC-ROC decreases by 8% for the Random, Central and Lateral strategies, and 2.5-3% for the Even and Squash strategies. Site Charlie sees a drop in accuracy by at least 15% across all strategies, with the Random strategy performing slightly better with a 14% loss. AUC-ROC is increased by an average of 9%, with the Even strategy performing best. Site Delta shows a drop in accuracy by 11% for the Random, Central and Squash strategies, and a drop in 6% for the remaining Lateral and Even strategies. AUC-ROC is improved by 23% for the Random, Central and Squash strategies, and only 17% for the Lateral and Even strategies. Site Epsilon shows a decrease in

accuracy by 4–6% for the Random, Central and Lateral strategies, but an increase in 3% for the Even and Squash strategies. The AUC-ROC is increased by 21–24% for the Random, Central and Lateral strategies, but only 18% for Even and Squash. Finally, site Fazbear shows a total loss in performance, with a 9–10% drop in accuracy across all strategies, and a loss of 5% AUC-ROC as well.

For many sites, the reduction strategy used gives better results. Sites Beta and Epsilon have slightly less improved accuracy for the Random, Central and Lateral strategies, but better AUC-ROC scores. These strategies are particularly beneficial for site Epsilon, which has a benchmark AUC-ROC of 51%. The sacrifice of 2% in accuracy for a gain of 3–6% AUC-ROC score may be a good trade-off. Sites Alpha and Delta also share this behaviour, but with different reduction strategies: site Alpha has a 5% better AUC-ROC score with the Lateral and Squash strategies, and site Delta has 6% better AUC-ROC score with the Random, Central and Squash strategies at the cost of 5% accuracy. For sites Charlie and Fazbear, the reduction strategy has no effect; both the accuracies and AUC-ROC scores are consistent across all strategies. These are the two largest sites prior to fusion, which may cause this effect.

6.3 Discussion

6.3.1 Individual Site Datasets

First, the performance of each site individually are discussed. Datasets Alpha, Charlie, and Delta show promising results when performing reduction on the majority class. The AUC-ROC may be maintained, so long as class density shifts towards 1 through data reduction. The class densities of sites Beta, Epsilon, and Fazbear do not converge around 1 as the amount of data is reduced, and the AUC-ROC decreases slightly as a result. Site Epsilon has one of the largest class imbalances, but still fails to improve after class balancing through data reduction. Site Epsilon also has the fewest number of datapoints, which may explain the poor performance after data reduction is performed. Therefore, class imbalance can not be relied on solely as a criterion for data reduction. Instead, class density offers insight into a dataset that might not be immediately apparent. Class density shows more than class imbalance, it also shows whether a dataset has sufficient data for class balancing operations through data reduction. Class density may therefore be considered as a metric that encompasses class balance and sufficient data, and can be used to determine if data reduction is applicable.

Table 6.37 shows the runtimes for all the experiments. The experimental runtimes show the benefits data reduction may bring to energy and CO₂ cost reduction, as runtime, energy use, and CO₂ emissions are directly correlated. For most experiments, by increasing the amount of reduction performed, runtime is reduced. Sites Alpha, Charlie, Delta, and Epsilon have relatively large class imbalance, meaning they remove more data to balance the classes. However, sites Beta and Fazbear have less imbalance, and therefore remove less data. For example, this is especially apparent for Site Fazbear, where in experiments to balance classes, by performing reduction there is an increase in runtime. The increase in runtime is caused by the overhead of identifying which data to remove takes longer for this site, and due to its size and because so few data are removed, the runtime is similar to that of its benchmark. Also, experiments that do not balance the classes are in most cases slightly faster, as the datasets do not have to be split between classes before reduction, which is an additional overhead. Because of the overhead introduced by the proposed data reduction methods, the data reduction strategies introduced in this paper are not applicable to all datasets, such as Site Fazbear. It does however show that for datasets such as Site Charlie, the runtime may be nearly halved (from 44s to 26s) by reducing with a cap of 50%, which also improves model performance.

Table 6.37: Runtimes for experiments on each individual site at each reduction amount for experiments, for both class balancing and without.
A: Site Alpha, B: Site Beta, C: Site Charlie, D: Site Delta, E: Site Epsilon, F: Site Fazbear.
Max reduction amount is the maximum data reduced to achieve class balance for each site.

Reduction Amount	Class Balancing Runtimes (s)						No Class Balancing Runtimes (s)					
	A	B	C	D	E	F	A	B	C	D	E	F
None	15	15	44	18	15	42	15	15	44	18	15	42
5%	15	14	43	18	18	49	15	15	41	18	15	42
10%	14	14	41	18	17	49	14	14	43	18	17	40
25%	12	12	37	15	13	-	12	11	36	15	16	-
50%	10	-	26	10	10	-	9	-	27	10	9	-
75%	-	-	-	6	-	-	-	-	-	6	-	-
Max%	7	11	20	6	8	48	6	12	18	6	7	40

By focusing on the best performing results, as well as the considering the statistical significance of these results, the best data reduction strategy may be identified.

For example, Site Alpha maintains performance in AUC-ROC up to 50% reduction, but none of the reduction strategies show significant difference from the benchmark. Therefore, no strategy can definitively be identified as superior. Site Beta with the random and squash strategies performs marginally better than the benchmark with 5% reduction, although the p -values associated with these experiments show the results do not have statistical significance from the benchmark. The lateral strategy is the worst performing for this site. Site Charlie shows an increase in AUC-ROC that is statistically different from the benchmark. With a 50% reduction, the data squash method is performs best. Lateral reduction with this site is the worst performing. Site Delta has two results that stand out: the data squash method at 10% reduction and the lateral reduction at 25%. The data squash method is superior, with an average increase in AUC-ROC of 0.137%. For this site, central reduction is the worst performing, differently than previous sites. For Site Epsilon, a 5% lateral reduction performs best, with an average AUC-ROC increase of 0.0301%, while central reduction performs the worst. For Site Fazbear, a 5% reduction with the data squash method has a better AUC-ROC by an average of 0.0272%, and random reduction performs the worst. To conclude, the lateral and squash reduction methods are most often the best performing, and lateral reduction performs the worst. However, it is important to note that each site shows very different behaviour, and there is no clear best or worst performing strategy.

6.3.2 Fused Dataset

The fused data benchmark results show very poor performance. All results are below 85%, which is substantially lower than the 98–99% scores observed in the non-fused dataset results. Some of the test sets show an AUC-ROC no better at classifying than random chance; for these sites, relatively high accuracy only reflects the class imbalance. In the reduced data experiments, class imbalanced is alleviated, but performance is still considerably worse than in the non-fused data experimentation. However, the results have given insight into the importance of using the correct data; despite each site dataset containing the same types of data (temperature, humidity, and VOC), there are differences between sites that mean datasets cannot be fused in this way.

The runtime of training the fused dataset is 140 seconds, and 80 seconds for the reduced fused dataset. These runtimes are less than the sum of all individual site training times and individual reduced training times which are 179 seconds and 100, respectively. These are the

sums of the values shown in table 6.37. These runtimes indicate that by fusing the datasets together, training time is less than the time it takes to train the datasets individually. As the runtime of the experimentation is directly proportional to energy cost and CO₂ usage, dataset fusion leads towards green AI. If the model performance with the fused dataset can be improved and the lower runtime retained, a fused dataset approach would be ideal for energy saving and running on low-compute devices. But until that is achieved, the poor performance is simply too detrimental.

6.4 Conclusion

Occupancy detection for large buildings enables optimised control of indoor systems based on occupant presence, reducing the energy costs of heating and cooling. Through machine learning models, occupancy detection can be achieved with high accuracy. However, as with datasets of other formats, large amounts of data are collected with little consideration of which of the collected data are most useful to the task. The use of unnecessary data contradicts the purpose of using AI to reduce costs, giving the need for a way to identify and use only the most useful data.

This chapter has aimed to find the benefits or detriments of data reduction prior to training a Random Forest model, using the open source HPDMobile occupancy dataset, that consists of 6 individual site datasets. After performing data reduction and model training, the performance of the models have been compared with the class densities of each dataset, to observe any correlation and to find the suitability of different datasets for data reduction.

Experimentation performed in this chapter has identified that class density may be used as a metric to qualify a dataset for reduction. The results have shown that for datasets like Site Charlie, which are abundant in data and heavily imbalanced, data reduction on the majority class may be used to improve model performance and reduce the computation required to train the model, due to there being less data. For other datasets such as sites Alpha and Delta that are either abundant in data or highly imbalanced, data reduction may be used to at least maintain performance. For highly balanced or less abundant datasets such as sites Beta, Epsilon, and Fazbear, data reduction is not as beneficial. By calculating the class densities of a dataset and gradually reducing the data, if the class densities converge towards a value of 1 then the dataset may be reduced while performance is maintained.

A direction for future work might be to first perform data reduction on each individual site dataset, and then attempt dataset fusion. Individual site data reduction should ensure that the most important data are retained at the reduction stage. Alternatively, one might reduce the datasets to a much smaller set, with more data reduction than needed to balance the classes. Dataset fusion on heavily reduced data may contain insightful data that could improve model reliability in new domains.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

Convolutional Neural Networks (CNNs) have revolutionized automation of a wide array of tasks by enabling machines to achieve performance levels that rival or even surpass human accuracy in certain domains, including image classification and occupancy detection. In the wake of this success, much of the machine learning community has embraced a ‘more is better’ philosophy of feeding massive datasets into increasingly complex models, often with substantial computational power behind them. While this approach seems fruitful at first glance, it has led to an attitude with data where the assumption is that adding more data will simply lead to better results. However, a growing body of research has begun to challenge this notion. This thesis has identified that not all data contributes equally to a model’s learning; in fact, some data points are redundant or irrelevant, offering no real benefit to the training process. Worse still, some data can actually degrade performance, leading the model to learn noise or misleading patterns.

First, this thesis has shown that by removing data at random from oversaturated datasets such as MNIST, one can maintain or even improve model accuracy while reducing computational cost of training. This insight has led to the development of techniques that selectively use subsets of the training data to guide model learning more efficiently. Greater efficiency is achieved by beginning the training process with a reduced dataset, created by randomly removing a portion of the training samples, and then reintroducing the full dataset at a later stage. The dynamic application of data allows the model to establish a solid foundation by more quickly converging on meaningful weight patterns in the early training epochs, and then refine its understanding through exposure to the complete dataset in the later stages. When applied to the MNIST dataset, this strategy, named the Data Step method, has shown improvements in model accuracy compared to training on the full dataset from the outset. By contrast, alternative methods such as Data Increment and Data Cut, which aim to reduce the total number of image evaluations, succeed in lowering energy consumption and associated CO₂ emissions, but often at the cost of reduced model performance. These trade-offs underscore the importance of balancing data efficiency with model effectiveness in modern machine learning workflows.

Building upon this methodology, a more deliberate approach to data selection can be employed through the use of a distance metric. Rather than relying on random sampling to create the reduced dataset used in the early stages of training, a distance-based strategy enables a more informed and potentially more effective subset to be selected. The distance-based metric is made possible through dimensionality reduction techniques, which project high-dimensional data into a lower-dimensional Euclidean space. Once in this form, individual datapoints can be meaningfully compared and ranked by their spatial proximity, allowing for a more structured and principled method of determining which samples are most representative or diverse. The concept of ordering data using dimensionality reduction is introduced by Byerly et al. [72], where the authors explore the impact of integrating non-random, distance-informed data selection, and

expanded upon in this thesis by combining the distance metric with dynamic training procedures. The result is a novel perspective on data utilisation in CNNs that acknowledges that the sequence and composition of data exposure can significantly influence the learning trajectory of a model. By choosing datapoints that are strategically spaced apart, the reduced dataset may capture a broader range of features at the earlier training stages, allowing the model to train more quickly. However, while dynamic data reduction has yielded promising results on relatively simple datasets such as MNIST, where accuracy can even be improved over baseline training, its effectiveness diminishes when applied to more complex image datasets. Experiments with Imagenette and Micro-PCB datasets reveal a noticeable drop in performance when data reduction is applied, suggesting that as image complexity and intra-class variation increase, the benefits of selective data reduction become harder to realise. These findings highlight the importance of dataset characteristics in determining the viability of data reduction methods.

Up to this point, the data reduction techniques explored so far have primarily relied on statically defined reduced datasets, in which a fixed subset of the training data is selected and used across multiple epochs. In order to explore alternative methods of dynamic data reduction, the Sliding Window data application method is introduced. Instead of using a single, unchanging subset of data throughout the early training process and switching to the entire dataset partway through, the Sliding Window approach feeds the model a new, distinct subset of data at each training epoch, while keeping the total number of samples per epoch constant. The selection and ordering of these data subsets are guided by the previously explored distance metric, derived through dimensionality reduction techniques. The distance metric enables the data to be ordered based on proximity to the dataset’s centre or edges, allowing the ‘window’ to slide either from the centralmost datapoints outward toward the lateralmost, or in the reverse direction. The sliding data window creates a structured and interpretable way to expose the model to different types of samples across training, enabling investigations into how the positioning of data in feature space correlates with its value to the learning process. Experimental results suggest that the direction of the sliding window has a significant impact on model performance, and that this impact varies depending on the nature of the dataset. For three out of the five datasets tested, MNIST, CIFAR-10, and Imagenette, the best-performing configuration involved starting with the most central datapoints and gradually progressing toward the lateral. Compared to the two remaining datasets tested, smallNORB and the Micro-PCB dataset, these datasets are relatively unstructured, and central datapoints likely represent prototypical examples that facilitate robust early-stage learning. Conversely, smallNORB and Micro-PCB datasets are more highly structured, with images captured within controlled environments and with consistent object poses. In these cases, the best results yielded are in the reverse direction, beginning with lateralmost datapoints and moving inward. These results suggest that in more structured data domains, atypical or edge-case samples may play a more significant role early in training, possibly by helping the model escape local minima or encouraging broader feature representations. When comparing the Sliding Window method with the earlier Data Step approach, the latter generally outperforms the former in both accuracy and training efficiency across most datasets. Data Step’s success is likely due to its balance of early convergence with later-stage refinement, while Sliding Window, despite offering richer insights into dataset structure and training dynamics, introduces greater variance and longer convergence times.

Classification is a task not limited to image data. It is useful in the time-series domain to observe trends and status, for example, in occupancy detection of indoor space. When training on such kinds of data, which is captured from real world environments, the data is often noisy, repetitive and imbalanced. Data cleaning is often used as a preprocessing technique to ease training on this data, resolving these issues. As with image data, data reduction based on centroid distance may provide insight to data usefulness for occupancy datasets. Experimentation has shown that, similarly to with image data, data may be removed and performance enhanced. By performing data reduction as a means of class balancing, a model may be made more reliable.

Class density is also introduced as a metric to identify the suitability of data reduction prior to training, enabling a quick method to gauge the quality of a dataset without the relatively expensive model training. Dataset fusion is also explored as a method of generalising to be able to classify on multiple different environments. While the results of experimentation yielded disappointing results, they provided useful insight into how the class balances and sizes of multiple data subsets affect a model when combined.

Classification tasks are not exclusive to the domain of image data; they are equally vital in other fields, such as time-series analysis, where the objective often revolves around tracking temporal patterns or detecting discrete states. A notable example is occupancy detection in indoor environments, which involves classifying whether a space is occupied or vacant based on sensor readings collected over time. Unlike curated image datasets, time-series data captured from real-world settings tends to be noisy, repetitive, and class-imbalanced, which poses significant challenges to model training and generalisation. To address these issues, data cleaning is commonly employed as a preprocessing step, helping to remove outliers, reduce redundancy, and correct class imbalances. However, beyond traditional data cleaning, more sophisticated techniques such as data reduction based on centroid distance, as previously explored in image classification, can also be applied to time-series classification tasks. By once again projecting the data into a lower-dimensional Euclidean space and measuring distances from class centroids, data may be primed for a number of data reduction strategies, to observe which samples are most representative of each class and which may be extraneous or detrimental to training. Experiments with occupancy datasets demonstrate that this form of data reduction, much like with image data, can not only maintain but even enhance model performance. In particular, using data reduction as a tool for class balancing by selectively removing overrepresented samples based on their distance to the class centroid can result in more robust and reliable models. The data reduction approach introduced offers a principled alternative to oversampling or naive downsampling, which may otherwise introduce bias or omit critical variation. To support this process, the concept of class density is introduced as a pretraining diagnostic tool. Class density captures the distribution and compactness of data within each class and can be used to assess the potential effectiveness of data reduction strategies without the need to perform full model training. As such, it provides a fast and relatively computationally inexpensive way to evaluate dataset quality and balance ahead of training. Finally, the idea of dataset fusion is explored as a means of improving a model’s ability to generalise across multiple environments. By combining data from various settings, each with its own class distributions and noise characteristics, the goal is to create a more versatile model capable of robust classification in heterogeneous conditions. While initial results from this approach were underwhelming in terms of accuracy, they offered valuable insights into how the interplay between class balance, class overlap, and dataset scale influences performance in fused datasets. These findings highlight the need for more adaptive fusion strategies and pave the way for future work on cross-domain generalisation in real-world classification problems.

In summary, this thesis has introduced novel data reduction methodologies aimed at uncovering the varying impacts individual data samples have on the training dynamics of neural networks. These findings encourage a more deliberate and informed approach to data selection, prompting practitioners to consider not just the quantity but the quality and relevance of data during both collection and training. By promoting more efficient data usage, this work contributes to the broader goals of Green AI, advocating for reduced computational overhead and a smaller carbon footprint in the development and deployment of machine learning models.

7.2 Future Work

Large Language Models (LLMs) and the underlying transformer networks have boomed in popularity in recent years. They inherently require much more data than the previously popular CNNs, but perform much better and in a wider scope. Lots of work has been performed to optimise training datasets to ensure only the most useful data is trained on, to squeeze as much potential out of these models as possible. The methods in this thesis may be applied to these new models and may provide information to which data is better for training. As this work has focused largely on image data, investigation into image transformers may be a logical next step.

The datasets used in this work are relatively small images, up to 300x300 pixels. The use of larger images has an exponential increase on training time, which is all the more reason to use data reduction to lessen training time. A direction for future research may be to test on larger images, for example in medical imaging.

Using class density to determine suitability for data reduction was investigated on imbalanced time-series datasets. Class imbalance is not limited to time-series occupancy datasets, and is an issue in multiple other domains. Calculating class density for data in other domains would be a good direction for future research, to prove its ability to generalise across multiple domains. Perhaps there are metrics for these domains which are more suitable, awaiting discovery. The benefits of class density may be explained through information theory concepts, which may explain its benefits more solidly than through experiments such as those shown in this thesis. Further work linking information theory to the experimentation performed may also open new avenues for research.

A practical use for this work would be to integrate a method to determine data usefulness into a process which collects data. If, at the data collection stage, data which is input can be observed and determined to be useful or not useful, it can be saved or discarded at the source. Identifying useful data could reduce the cost of data transmission at the earliest stage possible, reducing the overall cost of the AI system. Through careful planning and understanding of the data required, future work could streamline data collection through to model training in a more efficient manner.

Early stopping, as mentioned in the literature review, is a powerful tool to reduce training time and improve efficiency. Early stopping also has uses in identifying when training starts poorly and is unlikely to improve, due to bad hyperparameter initialisation. The experimentation in this thesis has not used early stopping, to ensure consistency between different experiments with different amounts of data reduction and make comparing results less complicated. However, as the experimentation in this thesis has shown the usefulness of dynamic data reduction, early stopping may be a good avenue to explore in the future. In the case of the data step dynamic data reduction method introduced in chapter 3 and expanded on in chapter 4, early stopping may be useful in identifying when the step should happen, instead of it being decided beforehand. After training on the ‘easy’ data has plateaued, training may automatically move onto training with the ‘harder’ data, potentially greatly improving training efficiency.

Once again for the sake of consistency and better comparability with similar works, the experimentation presented in chapters 4 and 5 used UMAP dimensionality reduction to order data the image data for use with dynamic data reduction. Other dimensionality reduction methods mentioned in the literature review, such as t-SNE and PCA, were not explored; these alternative dimensionality reduction methods may yield new insights if used as alternatives for ordering data. Other methods of understanding data difficulty are also mentioned in the literature review that do not use dimensionality reduction, such as forgetting score and Signal-to-Noise Ratio (SNR), may also be explored as alternatives for data ordering as well.

Bibliography

- [1] Adib Bin Rashid and MD Ashfakul Karim Kausik. “AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications”. In: *Hybrid Advances* 7 (2024), p. 100277. ISSN: 2773-207X. DOI: <https://doi.org/10.1016/j.hybadv.2024.100277>. URL: <https://www.sciencedirect.com/science/article/pii/S2773207X24001386>.
- [2] Leslie Citrome. “Artificial Intelligence and the potential for Garbage In, Garbage Out”. In: *Current Medical Research and Opinion* 40 (Nov. 2023), pp. 1–2. DOI: 10.1080/03007995.2023.2286785.
- [3] Trond Linjordet and Krisztian Balog. “Impact of training dataset size on neural answer selection models”. In: *European Conference on Information Retrieval*. Springer. 2019, pp. 828–835.
- [4] Neil C Thompson et al. “The computational limits of deep learning”. In: *arXiv preprint arXiv:2007.05558* (2020).
- [5] Yuting Lu et al. “A Comprehensive Survey of Datasets for Large Language Model Evaluation”. In: *2024 5th Information Communication Technologies Conference (ICTC)*. 2024, pp. 330–336. DOI: 10.1109/ICTC61510.2024.10601918.
- [6] Dominic Sanderson and Tatiana Kalganova. “Maintaining Performance with Less Data: Understanding Useful Data”. In: *Proceedings of Eighth International Congress on Information and Communication Technology*. Ed. by Xin-She Yang et al. Singapore: Springer Nature Singapore, 2023, pp. 1105–1127. ISBN: 978-981-99-3091-3.
- [7] Dominic Sanderson et al. “Dimensionality Reduction to Dynamically Reduce Data”. In: *2022 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. 2022, pp. 1–5. DOI: 10.1109/PAINE56030.2022.10014786.
- [8] Dominic Sanderson and Tatiana Kalganova. “Dynamic Data Inclusion with Sliding Window”. In: *Intelligent Sustainable Systems*. Ed. by Atulya K. Nagar et al. Singapore: Springer Nature Singapore, 2024, pp. 525–544. ISBN: 978-981-99-7886-1.
- [9] Dominic Sanderson and Tatiana Kalganova. “Identifying Suitability for Data Reduction in Imbalanced Time-Series Datasets”. In: *AI* 6.5 (2025). ISSN: 2673-2688. DOI: 10.3390/ai6050098. URL: <https://www.mdpi.com/2673-2688/6/5/98>.
- [10] Alfredo Cuzzocrea. “Data Processing and Management in the Big Data Era: Advanced Methods, Techniques and Algorithms”. In: *SN Computer Science* 7.2 (Feb. 2026), p. 161. ISSN: 2661-8907. DOI: 10.1007/s42979-025-04666-y. URL: <https://doi.org/10.1007/s42979-025-04666-y>.
- [11] Sudipta Hazra and Arindam Mondal. “Fundamentals of Data Analysis and Preprocessing”. In: *Data-Driven Modeling*. Ed. by Souvik Ganguli Arindam Mondal. John Wiley & Sons, Ltd, 2026. Chap. 1, pp. 1–21. ISBN: 9781394287925. DOI: <https://doi.org/10.1002/9781394287925.ch1>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781394287925.ch1>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781394287925.ch1>.

- [12] Pawan Whig, Balaram Yadav Kasula, and Nikhitha Yathiraju. “Navigating the Essential Fundamentals of Data Processing for Modern Enterprises”. In: *Advances in Systems Analysis, Software Engineering, and High Performance Computing* (June 2024), pp. 181–189. DOI: <https://doi.org/10.4018/979-8-3693-2909-2.ch014>. URL: <https://www.igi-global.com/chapter/navigating-the-essential-fundamentals-of-data-processing-for-modern-enterprises/345809>.
- [13] Senlin Wang et al. “Field-driven data processing paradigm for multi-information additive manufacturing”. In: *Additive Manufacturing* 61 (2023), p. 103352.
- [14] Oussama H. Hamid. “From Model-Centric to Data-Centric AI: A Paradigm Shift or Rather a Complementary Approach?” In: *2022 8th International Conference on Information Technology Trends (ITT)*. 2022, pp. 196–199. DOI: 10.1109/ITT56123.2022.9863935.
- [15] Peter McCullagh. “What is a statistical model?” In: *The Annals of Statistics* 30.5 (2002), pp. 1225–1310.
- [16] Barry de Ville. “Decision trees”. In: *WIREs Computational Statistics* 5.6 (2013), pp. 448–455. DOI: <https://doi.org/10.1002/wics.1278>. eprint: <https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1278>. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1278>.
- [17] Eduardo F. Morales and Hugo Jair Escalante. “Chapter 6 - A brief introduction to supervised, unsupervised, and reinforcement learning”. In: *Biosignal Processing and Classification Using Computational Learning and Intelligence*. Ed. by Alejandro A. Torres-García et al. Academic Press, 2022, pp. 111–129. ISBN: 978-0-12-820125-1. DOI: <https://doi.org/10.1016/B978-0-12-820125-1.00017-8>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128201251000178>.
- [18] Vishal Goar and Nagendra Singh Yadav. *Foundations of Machine Learning*. 2024. URL: <https://www.igi-global.com/chapter/foundations-of-machine-learning/344516>.
- [19] K. Meena and Santhi Suriya. “A Survey on Supervised and Unsupervised Learning Techniques”. In: Mar. 2020, pp. 627–644. ISBN: 978-3-030-24050-9. DOI: 10.1007/978-3-030-24051-6_58.
- [20] Gareth James et al. “Linear regression”. In: *An introduction to statistical learning: With applications in python*. Springer, 2023, pp. 69–134.
- [21] Shan Suthaharan. “Support Vector Machine”. In: *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Boston, MA: Springer US, 2016, pp. 207–235. ISBN: 978-1-4899-7641-3. DOI: 10.1007/978-1-4899-7641-3_9. URL: https://doi.org/10.1007/978-1-4899-7641-3_9.
- [22] Steven J Rigatti. “Random Forest”. In: *Journal of Insurance Medicine* 47.1 (2017), pp. 31–39.
- [23] Irina Rish et al. “An empirical study of the naive Bayes classifier”. In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. Seattle, USA. 2001, pp. 41–46.
- [24] Samreen Naeem et al. “An unsupervised machine learning algorithms: Comprehensive review”. In: *International Journal of Computing and Digital Systems* (2023).
- [25] Kristina P. Sinaga and Miin-Shen Yang. “Unsupervised K-Means Clustering Algorithm”. In: *IEEE Access* 8 (2020), pp. 80716–80727. DOI: 10.1109/ACCESS.2020.2988796.
- [26] Pengzhi Li, Yan Pei, and Jianqiang Li. “A comprehensive survey on design and application of autoencoder in deep learning”. In: *Applied Soft Computing* 138 (2023), p. 110176. ISSN: 1568-4946. DOI: <https://doi.org/10.1016/j.asoc.2023.110176>. URL: <https://www.sciencedirect.com/science/article/pii/S1568494623001941>.

- [27] Michael Greenacre et al. “Principal component analysis”. In: *Nature Reviews Methods Primers* 2.1 (Dec. 2022), p. 100. ISSN: 2662-8449. DOI: [10.1038/s43586-022-00184-w](https://doi.org/10.1038/s43586-022-00184-w). URL: <https://doi.org/10.1038/s43586-022-00184-w>.
- [28] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. “Reinforcement learning algorithms: A brief survey”. In: *Expert Systems with Applications* 231 (2023), p. 120495. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2023.120495>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417423009971>.
- [29] Vinícius G. Costa and Carlos E. Pedreira. “Recent advances in decision trees: an updated survey”. In: *Artificial Intelligence Review* 56.5 (May 2023), pp. 4765–4800. ISSN: 1573-7462. DOI: [10.1007/s10462-022-10275-5](https://doi.org/10.1007/s10462-022-10275-5). URL: <https://doi.org/10.1007/s10462-022-10275-5>.
- [30] Hasan Ahmed Salman, Ali Kalakech, and Amani Steiti. “Random forest algorithm overview”. In: *Babylonian Journal of Machine Learning* 2024 (2024), pp. 69–79.
- [31] Aya Nabil Sayed, Yassine Himeur, and Faycal Bensaali. “Deep and transfer learning for building occupancy detection: A review and comparative analysis”. In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105254. ISSN: 0952-1976. DOI: <https://doi.org/10.1016/j.engappai.2022.105254>. URL: <https://www.sciencedirect.com/science/article/pii/S0952197622003207>.
- [32] Shubham Chitnis, Nivethitha Somu, and Anupama Kowli. “Occupancy estimation with environmental sensors: The possibilities and limitations”. In: *Energy and Built Environment* 6.1 (2025), pp. 96–108. ISSN: 2666-1233. DOI: <https://doi.org/10.1016/j.enbenv.2023.09.003>. URL: <https://www.sciencedirect.com/science/article/pii/S2666123323000831>.
- [33] Sofiane Z., Yiannis Gkoufas, and John Murphy. “A Machine Learning Approach to Indoor Occupancy Detection Using Non-Intrusive Environmental Sensor Data”. In: Aug. 2019, pp. 70–74. ISBN: 978-1-4503-7246-6. DOI: [10.1145/3361758.3361775](https://doi.org/10.1145/3361758.3361775).
- [34] Abolfazl Mohammadabadi, Samira Rahnama, and Alireza Afshari. “Indoor Occupancy Detection Based on Environmental Data Using CNN-XGboost Model: Experimental Validation in a Residential Building”. In: *Sustainability* 14.21 (2022). ISSN: 2071-1050. DOI: [10.3390/su142114644](https://doi.org/10.3390/su142114644). URL: <https://www.mdpi.com/2071-1050/14/21/14644>.
- [35] Andree Vela et al. “Estimating Occupancy Levels in Enclosed Spaces Using Environmental Variables: A Fitness Gym and Living Room as Evaluation Scenarios”. In: *Sensors* 20.22 (2020). ISSN: 1424-8220. DOI: [10.3390/s20226579](https://doi.org/10.3390/s20226579). URL: <https://www.mdpi.com/1424-8220/20/22/6579>.
- [36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539). URL: <https://doi.org/10.1038/nature14539>.
- [37] Koosha Sharifani and Mahyar Amini. “Machine learning and deep learning: A review of methods and applications”. In: *World Information Technology and Engineering Journal* 10.07 (2023), pp. 3897–3904.
- [38] Hossein Shahinzadeh et al. “Deep Learning: A Overview of Theory and Architectures”. In: *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*. 2024, pp. 1–11. DOI: [10.1109/AISP61396.2024.10475265](https://doi.org/10.1109/AISP61396.2024.10475265).
- [39] Ibomoie Domor Mienye, Theo G. Swart, and George Obaido. “Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications”. In: *Information* 15.9 (2024). ISSN: 2078-2489. DOI: [10.3390/info15090517](https://doi.org/10.3390/info15090517). URL: <https://www.mdpi.com/2078-2489/15/9/517>.

- [40] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.
- [41] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical Image Analysis* 42 (2017), pp. 60–88. ISSN: 1361-8415. DOI: <https://doi.org/10.1016/j.media.2017.07.005>. URL: <https://www.sciencedirect.com/science/article/pii/S1361841517301135>.
- [42] Chenyi Chen et al. “DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2722–2730. DOI: 10.1109/ICCV.2015.312.
- [43] Yaniv Taigman et al. “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708. DOI: 10.1109/CVPR.2014.220.
- [44] Walter Dean. “Computational Complexity Theory”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.
- [45] H Mistry. “Mastering Model Selection for AI/ML Models”. In: *European Journal of Computer Science and Information Technology* 13.14 (2025), pp. 55–67.
- [46] Fuzhen Zhuang et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.
- [47] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.
- [48] Lingzhen Chen and Alessandro Moschitti. “Transfer learning for sequence labeling using source model and target data”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6260–6267.
- [49] L. Y. Pratt. “Discriminability-Based Transfer between Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Hanson, J. Cowan, and C. Giles. Vol. 5. Morgan-Kaufmann, 1992. URL: https://proceedings.neurips.cc/paper_files/paper/1992/file/67e103b0761e60683e83c559be18d40c-Paper.pdf.
- [50] German I Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* 113 (2019), pp. 54–71.
- [51] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML]. URL: <https://arxiv.org/abs/1503.02531>.
- [52] Luis Manuel Pereira, Addisson Salazar, and Luis Vergara. “On Comparing Early and Late Fusion Methods”. In: *Advances in Computational Intelligence*. Ed. by Ignacio Rojas, Gonzalo Joya, and Andreu Catala. Cham: Springer Nature Switzerland, 2023, pp. 365–378. ISBN: 978-3-031-43085-5.
- [53] Athina Tsanousa et al. “Fusion of Environmental Sensors for Occupancy Detection in a Real Construction Site”. In: *Sensors* 23.23 (2023). ISSN: 1424-8220. DOI: 10.3390/s23239596. URL: <https://www.mdpi.com/1424-8220/23/23/9596>.
- [54] Thu Nguyen et al. *Combining datasets to increase the number of samples and improve model fitting*. 2023. arXiv: 2210.05165 [stat.ML]. URL: <https://arxiv.org/abs/2210.05165>.
- [55] Ben Sorscher et al. “Beyond neural scaling laws: beating power law scaling via data pruning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19523–19536.

- [56] Devansh Arpit et al. “A closer look at memorization in deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 233–242.
- [57] Mariya Toneva et al. “An Empirical Study of Example Forgetting during Deep Neural Network Learning”. In: *CoRR* abs/1812.05159 (2018). arXiv: 1812.05159. URL: <http://arxiv.org/abs/1812.05159>.
- [58] Brian B. Moser, Federico Raue, and Andreas Dengel. “A Study in Dataset Pruning for Image Super-Resolution”. In: *Artificial Neural Networks and Machine Learning – ICANN 2024*. Ed. by Michael Wand et al. Cham: Springer Nature Switzerland, 2024, pp. 351–363. ISBN: 978-3-031-72335-3.
- [59] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. “Deep Learning on a Data Diet: Finding Important Examples Early in Training”. In: *CoRR* abs/2107.07075 (2021). arXiv: 2107.07075. URL: <https://arxiv.org/abs/2107.07075>.
- [60] Lei Chen. “Curse of Dimensionality”. In: *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, pp. 545–546. ISBN: 978-0-387-39940-9.
- [61] Adrian Galdran, Gustavo Carneiro, and Miguel Ángel González Ballester. “Balanced-MixUp for Highly Imbalanced Medical Image Classification”. In: *CoRR* abs/2109.09850 (2021). arXiv: 2109.09850. URL: <https://arxiv.org/abs/2109.09850>.
- [62] M.A. Bessa et al. “A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality”. In: *Computer Methods in Applied Mechanics and Engineering* 320 (2017), pp. 633–667. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2017.03.037>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782516314803>.
- [63] Mohsena Ashraf et al. “A Survey on Dimensionality Reduction Techniques for Time-Series Data”. In: *IEEE Access* 11 (2023), pp. 42909–42923. DOI: 10.1109/ACCESS.2023.3269693.
- [64] Ji Ma and Yuyu Yuan. “Dimension reduction of image deep feature using PCA”. In: *Journal of Visual Communication and Image Representation* 63 (2019), p. 102578. ISSN: 1047-3203. DOI: <https://doi.org/10.1016/j.jvcir.2019.102578>. URL: <https://www.sciencedirect.com/science/article/pii/S1047320319301932>.
- [65] Rida Zaheer et al. “Evaluating the Effectiveness of Dimensionality Reduction on Machine Learning Algorithms in Time Series Forecasting”. In: *IEEE Access* 13 (2025), pp. 50493–50510. DOI: 10.1109/ACCESS.2025.3551741.
- [66] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).
- [67] Geoffrey E Hinton, Peter Dayan, and Michael Revow. “Modeling the manifolds of images of handwritten digits”. In: *IEEE transactions on Neural Networks* 8.1 (1997), pp. 65–74.
- [68] Anna Breger et al. “Supervised learning and dimension reduction techniques for quantification of retinal fluid in optical coherence tomography images”. In: *Eye* 31.8 (2017), pp. 1212–1220.
- [69] Shinhyun Choi et al. “Experimental demonstration of feature extraction and dimensionality reduction using memristor networks”. In: *Nano letters* 17.5 (2017), pp. 3113–3118.
- [70] Yashar Kiarashinejad, Sajjad Abdollahramezani, and Ali Adibi. “Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures”. In: *npj Computational Materials* 6.1 (2020), pp. 1–12.
- [71] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

- [72] Adam Byerly and Tatiana Kalganova. “Towards an Analytical Definition of Sufficient Data”. In: *SN Computer Science* 4.2 (Jan. 2023), p. 144. ISSN: 2661-8907. DOI: 10.1007/s42979-022-01549-4. URL: <https://doi.org/10.1007/s42979-022-01549-4>.
- [73] Tongtong Yuan et al. *Signal-to-Noise Ratio: A Robust Distance Metric for Deep Metric Learning*. 2019. arXiv: 1904.02616 [cs.CV]. URL: <https://arxiv.org/abs/1904.02616>.
- [74] Fengchun Liu et al. “A Review of the Evaluation System for Curriculum Learning”. In: *Electronics* 12.7 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12071676. URL: <https://www.mdpi.com/2079-9292/12/7/1676>.
- [75] Jeffrey L. Elman. “Learning and development in neural networks: the importance of starting small”. In: *Cognition* 48.1 (1993), pp. 71–99. ISSN: 0010-0277. DOI: [https://doi.org/10.1016/0010-0277\(93\)90058-4](https://doi.org/10.1016/0010-0277(93)90058-4). URL: <https://www.sciencedirect.com/science/article/pii/0010027793900584>.
- [76] Yoshua Bengio et al. “Curriculum learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning. ICML '09*. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: <https://doi.org/10.1145/1553374.1553380>.
- [77] Petru Soviany et al. “Curriculum Learning: A Survey”. In: *International Journal of Computer Vision* 130.6 (June 2022), pp. 1526–1565. ISSN: 1573-1405. DOI: 10.1007/s11263-022-01611-x. URL: <https://doi.org/10.1007/s11263-022-01611-x>.
- [78] Luca Anzalone et al. “An End-to-End Curriculum Learning Approach for Autonomous Driving Scenarios”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 19817–19826. DOI: 10.1109/TITS.2022.3160673.
- [79] Taiwei Shi et al. *Efficient Reinforcement Finetuning via Adaptive Curriculum Learning*. 2026. arXiv: 2504.05520 [cs.LG]. URL: <https://arxiv.org/abs/2504.05520>.
- [80] Jerry Wei et al. *Learn like a Pathologist: Curriculum Learning by Annotator Agreement for Histopathology Image Classification*. 2020. arXiv: 2009.13698 [cs.CV]. URL: <https://arxiv.org/abs/2009.13698>.
- [81] Anis Koubaa et al. “Cloud Versus Edge Deployment Strategies of Real-Time Face Recognition Inference”. In: *IEEE Transactions on Network Science and Engineering* 9.1 (2022), pp. 143–160. DOI: 10.1109/TNSE.2021.3055835.
- [82] Michael Armbrust et al. “A view of cloud computing”. In: *Communications of the ACM* 53.4 (2010), pp. 50–58.
- [83] Latifa Albshaier, Seetah Almarri, and Abdullah Albuali. “Federated Learning for Cloud and Edge Security: A Systematic Review of Challenges and AI Opportunities”. In: *Electronics* 14.5 (2025). ISSN: 2079-9292. DOI: 10.3390/electronics14051019. URL: <https://www.mdpi.com/2079-9292/14/5/1019>.
- [84] Uttama Reddy Sanepalli. “Cybersecurity Framework for Multi-Cloud Deployment Pipelines: A Zero-Trust Architecture for Inter-Platform Data Protection”. In: *International Journal of Research in Computer Applications and Information Technology (IJRCAIT)* 6.1 (2023), pp. 191–206.
- [85] Ilenia Ficili et al. “From Sensors to Data Intelligence: Leveraging IoT, Cloud, and Edge Computing with AI”. In: *Sensors* 25.6 (2025). ISSN: 1424-8220. DOI: 10.3390/s25061763. URL: <https://www.mdpi.com/1424-8220/25/6/1763>.
- [86] Haozhou Zhai et al. “Edge Deployment of Deep Networks for Visual Detection: A Review”. In: *IEEE Sensors Journal* 25.11 (2025), pp. 18662–18683. DOI: 10.1109/JSEN.2024.3502539.

- [87] Vijay Sikha. “Cloud-Native Application Development for AI- Conducive Architectures”. In: *International Journal on Recent and Innovation Trends in Computing and Communication* 11 (Oct. 2023), p. 1241. DOI: 10.5281/zenodo.14662301.
- [88] Muhammad Talha Tahir Bajwa et al. “Cloud-native architectures for large-scale AI-based predictive modeling”. In: *Journal of Emerging Technology and Digital Transformation* 4.2 (2025), pp. 207–221.
- [89] Vincent Uchenna Ugwueze. “Cloud native application development: Best practices and challenges”. In: *International Journal of Research Publication and Reviews* 5.12 (2024), pp. 2399–2412.
- [90] Victor Velepucha and Pamela Flores. “A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges”. In: *IEEE Access* 11 (2023), pp. 88339–88358. DOI: 10.1109/ACCESS.2023.3305687.
- [91] Poonam Mangwani, Niti Mangwani, and Sachin Motwani. “Evaluation of a Multitenant SaaS Using Monolithic and Microservice Architectures”. In: *SN Computer Science* 4.2 (Jan. 2023), p. 185. ISSN: 2661-8907. DOI: 10.1007/s42979-022-01610-2. URL: <https://doi.org/10.1007/s42979-022-01610-2>.
- [92] Vincent Uchenna Ugwueze and Joseph Nnaemeka Chukwunweike. “Continuous integration and deployment strategies for streamlined DevOps in software engineering and application delivery”. In: *Int J Comput Appl Technol Res* 14.1 (2024), pp. 1–24.
- [93] Jie Wen et al. “A survey on federated learning: challenges and applications”. In: *International Journal of Machine Learning and Cybernetics* 14.2 (Feb. 2023), pp. 513–535. ISSN: 1868-808X. DOI: 10.1007/s13042-022-01647-y. URL: <https://doi.org/10.1007/s13042-022-01647-y>.
- [94] Raghubir Singh and Sukhpal Singh Gill. “Edge AI: A survey”. In: *Internet of Things and Cyber-Physical Systems* 3 (2023), pp. 71–92. ISSN: 2667-3452. DOI: <https://doi.org/10.1016/j.iotcps.2023.02.004>. URL: <https://www.sciencedirect.com/science/article/pii/S2667345223000196>.
- [95] Rajesh Kumar and Anjali Sharma. “Edge AI: A Review of Machine Learning Models for Resource-Constrained Devices”. In: *Artificial Intelligence and Machine Learning Review* 5.3 (July 2024), pp. 1–11. DOI: 10.69987/. URL: <https://scipublication.com/index.php/AIMLR/article/view/89>.
- [96] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: <https://arxiv.org/abs/1704.04861>.
- [97] João D. Nunes et al. “Spiking Neural Networks: A Survey”. In: *IEEE Access* 10 (2022), pp. 60738–60764. DOI: 10.1109/ACCESS.2022.3179968.
- [98] Guanlei Zhang et al. “Spiking Neural Networks in Intelligent Edge Computing”. In: *IEEE Consumer Electronics Magazine* 14.4 (2025), pp. 66–75. DOI: 10.1109/MCE.2024.3506502.
- [99] Vlad C. Coroamă. *Temperature Optimisation in Data Centres*. 2025. arXiv: 2509.26163 [cs.OH]. URL: <https://arxiv.org/abs/2509.26163>.
- [100] Felipe Oviedo et al. “Energy Use of AI Inference: Efficiency Pathways and Test-Time Compute”. In: *arXiv preprint arXiv:2509.20241* (2025).
- [101] Aimé Cedric Muhoza et al. “Power consumption reduction for IoT devices thanks to Edge-AI: Application to human activity recognition”. In: *Internet of Things* 24 (2023), p. 100930. ISSN: 2542-6605. DOI: <https://doi.org/10.1016/j.iot.2023.100930>. URL: <https://www.sciencedirect.com/science/article/pii/S2542660523002536>.

- [102] Yuji Roh, Geon Heo, and Steven Euijong Whang. “A survey on data collection for machine learning: a big data-ai integration perspective”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2019), pp. 1328–1347.
- [103] Alhassan Mumuni and Fuseini Mumuni. “Data augmentation: A comprehensive survey of modern approaches”. In: *Array* 16 (2022), p. 100258. ISSN: 2590-0056. DOI: <https://doi.org/10.1016/j.array.2022.100258>. URL: <https://www.sciencedirect.com/science/article/pii/S2590005622000911>.
- [104] Fakhitah Ridzuan and Wan Mohd Nazmee Wan Zainon. “A Review on Data Cleansing Methods for Big Data”. In: *Procedia Computer Science* 161 (2019). The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia, pp. 731–738. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2019.11.177>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050919318885>.
- [105] Ekin D Cubuk et al. “Randaugment: Practical automated data augmentation with a reduced search space”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.
- [106] Rosa L Figueroa et al. “Predicting sample size required for classification performance”. In: *BMC medical informatics and decision making* 12.1 (2012), pp. 1–10.
- [107] Sanghyeon An et al. “An ensemble of simple convolutional neural network models for mnist digit recognition”. In: *arXiv preprint arXiv:2008.10400* (2020).
- [108] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).
- [109] Pierre Foret et al. “Sharpness-aware minimization for efficiently improving generalization”. In: *arXiv preprint arXiv:2010.01412* (2020).
- [110] Ekin D Cubuk et al. “Autoaugment: Learning augmentation strategies from data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 113–123.
- [111] Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. “Efficient-capsnet: Capsule network with self-attention routing”. In: *Scientific reports* 11.1 (2021), pp. 1–13.
- [112] Haiping Wu et al. “Cvt: Introducing convolutions to vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 22–31.
- [113] Qishuai Diao et al. “MetaFormer: A Unified Meta Framework for Fine-Grained Recognition”. In: *arXiv preprint arXiv:2203.02751* (2022).
- [114] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.
- [115] Rishi Rai and Dilip Singh Sisodia. “Real-time data augmentation based transfer learning model for breast cancer diagnosis using histopathological images”. In: *Advances in Biomedical Engineering and Technology*. Springer, 2021, pp. 473–488.
- [116] Adam Byerly, Tatiana Kalganova, and Ian Dear. “No routing needed between capsules”. In: *Neurocomputing* 463 (2021), pp. 545–553.
- [117] Junghwan Cho et al. “How much data is needed to train a medical image deep learning system to achieve necessary high accuracy”. In: *arXiv: Learning* (2015). URL: <https://api.semanticscholar.org/CorpusID:10322664>.
- [118] Katie L. Turkiewicz. “Data Trimming”. In: *The SAGE Encyclopedia of Communication Research Methods*. Ed. by Mike Allen. SAGE Publications, Inc, 2017, pp. 347–348.

- [119] Prabhjot Kaur and Anjana Gosain. “Comparing the Behavior of Oversampling and Under-sampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise”. In: *ICT Based Innovations*. Ed. by A. K. Saini, A. K. Nayak, and Ram Krishna Vyas. Singapore: Springer Singapore, 2018, pp. 23–30. ISBN: 978-981-10-6602-3.
- [120] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. “Exploratory undersampling for class-imbalance learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2 (2008), pp. 539–550.
- [121] M Mostafizur Rahman and D Davis. “Cluster based under-sampling for unbalanced cardiovascular data”. In: *Proceedings of the world congress on engineering*. Vol. 3. 2013, pp. 3–5.
- [122] Ivan Tomek. “A generalization of the k-NN rule”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 2 (1976), pp. 121–126.
- [123] M Mostafizur Rahman and Darryl N Davis. “Addressing the class imbalance problem in medical datasets”. In: *International Journal of Machine Learning and Computing* 3.2 (2013), p. 224.
- [124] Adam Byerly and Tatiana Kalganova. *Class Density and Dataset Quality in High-Dimensional, Unstructured Data*. 2022. arXiv: 2202.03856 [cs.LG]. URL: <https://arxiv.org/abs/2202.03856>.
- [125] A. Angelova, Y. Abu-Mostafam, and P. Perona. “Pruning training sets for learning of object categories”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 494–501 vol. 1. DOI: 10.1109/CVPR.2005.283.
- [126] Hamid Saadatfar et al. “A New K-Nearest Neighbors Classifier for Big Data Based on Efficient Data Pruning”. In: *Mathematics* 8.2 (2020). ISSN: 2227-7390. DOI: 10.3390/math8020286. URL: <https://www.mdpi.com/2227-7390/8/2/286>.
- [127] C. Hicks et al. “Carbon accounting: review and case study on the development of a carbon baseline for military infrastructure”. In: *Renewable and Sustainable Energy Reviews* 227 (2026), p. 116511. ISSN: 1364-0321. DOI: <https://doi.org/10.1016/j.rser.2025.116511>. URL: <https://www.sciencedirect.com/science/article/pii/S1364032125011840>.
- [128] Diptyaroop Maji et al. “The Green Mirage: Impact of Location- and Market-based Carbon Intensity Estimation on Carbon Optimization Efficacy”. In: *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems. e-Energy ’24*. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 256–267. ISBN: 9798400704802. DOI: 10.1145/3632775.3639587. URL: <https://doi.org/10.1145/3632775.3639587>.
- [129] Md Abu Bakar Siddik et al. “Spatially and Temporally Detailed Water and Carbon Footprints of U.S. Electricity Generation and Use”. In: *Water Resources Research* 60.12 (2024). e2024WR038350 2024WR038350, e2024WR038350. DOI: <https://doi.org/10.1029/2024WR038350>. eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024WR038350>. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2024WR038350>.
- [130] Nuo Lei et al. “The water use of data center workloads: A review and assessment of key determinants”. In: *Resources, Conservation and Recycling* 219 (2025), p. 108310. ISSN: 0921-3449. DOI: <https://doi.org/10.1016/j.resconrec.2025.108310>. URL: <https://www.sciencedirect.com/science/article/pii/S0921344925001892>.

- [131] Ralf Gitzel. “ESTIMATING THE SUSTAINABILITY OF AI MODELS BASED ON THEORETICAL MODELS AND EXPERIMENTAL DATA”. In: *TechRxiv* 2023.0425 (2023). DOI: 10.36227/techrxiv.22649725.v1. eprint: <https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.22649725.v1>. URL: <https://www.techrxiv.org/doi/abs/10.36227/techrxiv.22649725.v1>.
- [132] Roy Schwartz et al. “Green AI”. In: *Commun. ACM* 63.12 (Nov. 2020), pp. 54–63. ISSN: 0001-0782. DOI: 10.1145/3381831. URL: <https://doi.org/10.1145/3381831>.
- [133] Verónica Bolón-Canedo et al. “A review of green artificial intelligence: Towards a more sustainable future”. In: *Neurocomputing* 599 (2024), p. 128096. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2024.128096>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231224008671>.
- [134] Muhammad Habib ur Rehman et al. “Big Data Reduction Methods: A Survey”. In: *Data Science and Engineering* 1.4 (Dec. 2016), pp. 265–284. ISSN: 2364-1541. DOI: 10.1007/s41019-016-0022-0. URL: <https://doi.org/10.1007/s41019-016-0022-0>.
- [135] Steven Euijong Whang et al. “Data collection and quality challenges in deep learning: a data-centric AI perspective”. In: *The VLDB Journal* 32.4 (July 2023), pp. 791–813. ISSN: 0949-877X. DOI: 10.1007/s00778-022-00775-9. URL: <https://doi.org/10.1007/s00778-022-00775-9>.
- [136] Long Yin et al. “Urban carbon accounting: An overview”. In: *Urban Climate* 44 (2022), p. 101195. ISSN: 2212-0955. DOI: <https://doi.org/10.1016/j.uclim.2022.101195>. URL: <https://www.sciencedirect.com/science/article/pii/S2212095522001134>.
- [137] Xiaojing Liu et al. “Green innovation and corporate ESG performance: Evidence from Chinese listed companies”. In: *International Review of Economics & Finance* 95 (2024), p. 103461. ISSN: 1059-0560. DOI: <https://doi.org/10.1016/j.iref.2024.103461>. URL: <https://www.sciencedirect.com/science/article/pii/S1059056024004532>.
- [138] Yangyan Liu et al. “Artificial intelligence applications and corporate ESG performance”. In: *International Review of Economics & Finance* 104 (2025), p. 104559. ISSN: 1059-0560. DOI: <https://doi.org/10.1016/j.iref.2025.104559>. URL: <https://www.sciencedirect.com/science/article/pii/S1059056025007221>.
- [139] Feyisayo Michael Ogunyemi. “Carbon accounting automation through machine learning and natural language processing: Reducing compliance costs and enhancing data quality in enterprise ESG reporting”. In: *International Journal of Science and Research Archive* 4.1 (2021), pp. 490–501.
- [140] Gurneet Kaur Suri and Nuzhat Sadriwala. “Using Artificial Intelligence for Sustainable Accounting Practices: A Data-Driven Approach”. In: *Intelligent Systems for Sustainable Infrastructure: AI Solutions Shaping a Green Future: Leveraging AI Innovations for Eco Friendly Infrastructure and Environmental Resilience*. Ed. by Ramesh Chandra Poonia, Divyansh Thakur, and Ina Thakur. Cham: Springer Nature Switzerland, 2026, pp. 171–186. ISBN: 978-3-031-95240-1. DOI: 10.1007/978-3-031-95240-1_9. URL: https://doi.org/10.1007/978-3-031-95240-1_9.
- [141] F Ogunyemi. “AI-powered carbon accounting: transforming ESG reporting standards for a sustainable global economy”. In: *International Journal of Multidisciplinary Research in Science, Engineering and Technology* 6.4 (2023).
- [142] Jaho Daniel. ‘Speed vs. accuracy: Trade-offs in data analysis by AI models. Tech. rep. Tech. Rep, 2025.
- [143] Jennifer Mate and Thomas Philli. “Green AI: Benchmarking Energy-Accuracy Trade-offs in Edge Image Classification”. In: (Feb. 2026).

- [144] Sri Ariyanti et al. “Trade-Off Between Energy Consumption and Three Configuration Parameters in Artificial Intelligence (AI) Training: Lessons for Environmental Policy”. In: *Sustainability* 17.12 (2025). ISSN: 2071-1050. DOI: 10.3390/su17125359. URL: <https://www.mdpi.com/2071-1050/17/12/5359>.
- [145] Anshul Gandhi et al. “Metrics for Sustainability in Data Centers”. In: *SIGENERGY Energy Inform. Rev.* 3.3 (Oct. 2023), pp. 40–46. DOI: 10.1145/3630614.3630622. URL: <https://doi.org/10.1145/3630614.3630622>.
- [146] Tamar Eilam et al. “Towards a Methodology and Framework for AI Sustainability Metrics”. In: *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. HotCarbon ’23. Boston, MA, USA: Association for Computing Machinery, 2023. ISBN: 9798400702426. DOI: 10.1145/3604930.3605715. URL: <https://doi.org/10.1145/3604930.3605715>.
- [147] Avisek Naug et al. “SustainDC: Benchmarking for Sustainable Data Center Control”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 100630–100669. DOI: 10.52202/079017-3192. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/b6676756f8a935e208f394a1ba47f0bc-Paper-Datasets_and_Benchmarks_Track.pdf.
- [148] Nidhal Jegham et al. *How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference*. 2025. arXiv: 2505.09598 [cs.CY]. URL: <https://arxiv.org/abs/2505.09598>.
- [149] Michael Z Hauschild, Ralph K Rosenbaum, Stig Irving Olsen, et al. *Life cycle assessment*. Vol. 2018. Springer, 2018.
- [150] Dusan Dokic, Groen In’T Woud, Wolfgang Maass, et al. “Towards Sustainability of AI: A Systematic Review of Existing Life Cycle Assessment Approaches and Key Environmental Impact Parameters of Artificial Intelligence”. In: (2024).
- [151] Christiane Plociennik et al. “Life Cycle Assessment of Artificial Intelligence Applications: Research Gaps and Opportunities”. In: *Procedia CIRP* 135 (2025). 32nd CIRP Conference on Life Cycle Engineering (LCE2025), pp. 924–929. ISSN: 2212-8271. DOI: <https://doi.org/10.1016/j.procir.2025.01.079>. URL: <https://www.sciencedirect.com/science/article/pii/S2212827125003749>.
- [152] Richard M. Zur et al. “Noise injection for training artificial neural networks: A comparison with weight decay and early stopping”. In: *Medical Physics* 36.10 (2009), pp. 4810–4818. DOI: <https://doi.org/10.1118/1.3213517>. eprint: <https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.3213517>. URL: <https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3213517>.
- [153] Lars König et al. “An evaluation of the performance of stopping rules in AI-aided screening for psychological meta-analytical research”. In: *Research Synthesis Methods* 15.6 (2024), pp. 1120–1146. DOI: <https://doi.org/10.1002/jrsm.1762>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jrsm.1762>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jrsm.1762>.
- [154] Yi Wang. “AI-Enhanced Early Stopping Decision Framework for A/B Testing: A Machine Learning Approach to Optimize Experimental Efficiency”. In: *Journal of Sustainability, Policy, and Practice* 1.2 (Aug. 2025), pp. 86–97. URL: <https://schoalrx.com/index.php/jspp/article/view/15>.
- [155] Lizhi Liao et al. “An Empirical Study of the Impact of Hyperparameter Tuning and Model Optimization on the Performance Properties of Deep Neural Networks”. In: *ACM Trans. Softw. Eng. Methodol.* 31.3 (Apr. 2022). ISSN: 1049-331X. DOI: 10.1145/3506695. URL: <https://doi.org/10.1145/3506695>.

- [156] Edward Bergman, Lennart Purucker, and Frank Hutter. *Don't Waste Your Time: Early Stopping Cross-Validation*. 2024. arXiv: 2405.03389 [cs.LG]. URL: <https://arxiv.org/abs/2405.03389>.
- [157] Giorgia Franchini et al. “Biomedical Image Classification via Dynamically Early Stopped Artificial Neural Network”. In: *Algorithms* 15.10 (2022). ISSN: 1999-4893. DOI: 10.3390/a15100386. URL: <https://www.mdpi.com/1999-4893/15/10/386>.
- [158] Vili Podgorelec, Iztok Fister, and Grega Vrbančič. “Towards Green AI by Reducing Training Effort of Recurrent Neural Networks Using Hyper-Parameter Optimization with Dynamic Stopping Criteria”. In: *2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*. 2024, pp. 000161–000166. DOI: 10.1109/SISY62279.2024.10737555.
- [159] Tao Ouyang et al. “Cost-Aware Edge Resource Probing for Infrastructure-Free Edge Computing: From Optimal Stopping to Layered Learning”. In: *2019 IEEE Real-Time Systems Symposium (RTSS)*. 2019, pp. 380–391. DOI: 10.1109/RTSS46320.2019.00041.
- [160] Ziru Niu et al. “FLrce: Resource-Efficient Federated Learning With Early-Stopping Strategy”. In: *IEEE Transactions on Mobile Computing* 23.12 (2024), pp. 14514–14529. DOI: 10.1109/TMC.2024.3447000.
- [161] Hossam Amer et al. *FLOP-Efficient Training: Early Stopping Based on Test-Time Compute Awareness*. 2026. arXiv: 2601.01332 [cs.CL]. URL: <https://arxiv.org/abs/2601.01332>.
- [162] Mohammad Motamedi, Nikolay Sakharnykh, and Tim Kaldewey. “A data-centric approach for training deep neural networks with less data”. In: *arXiv preprint arXiv:2110.03613* (2021).
- [163] Fanjie Kong and Ricardo Henao. “Efficient Classification of Very Large Images with Tiny Objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2384–2394.
- [164] Chia-Wen Kuo et al. “Data-efficient graph embedding learning for PCB component detection”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 551–560.
- [165] Yann LeCun. “The MNIST database of handwritten digits”. In: <http://yann.lecun.com/exdb/mnist/> (1998).
- [166] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).
- [167] Fu Jie Huang and Yann LeCun. *The Small NORB Dataset, V1. 0, 2005*. 2009.
- [168] The Department for Energy Security and Net Zero. *Greenhouse Gas Reporting: Conversion Factors*. <https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2024/>. Accessed: 11 April 2025.
- [169] Ravi S. Raju, Kyle Daruwalla, and Mikko H. Lipasti. “Accelerating Deep Learning with Dynamic Data Pruning”. In: *CoRR* abs/2111.12621 (2021). arXiv: 2111.12621. URL: <https://arxiv.org/abs/2111.12621>.
- [170] Jeremy Howard. “Imagenette dataset”. In: <https://github.com/fastai/imagenette/tree/master> (2018).
- [171] Adam Byerly, Tatiana Kalganova, and Anthony J. Grichnik. “On the Importance of Capturing a Sufficient Diversity of Perspective for the Classification of Micro-PCBs”. In: *Intelligent Decision Technologies*. Ed. by Ireneusz Czarnowski, Robert J. Howlett, and Lakhmi C. Jain. Singapore: Springer Singapore, 2021, pp. 209–219. ISBN: 978-981-16-2765-1.

- [172] Adam Byerly. “Micro-PCB dataset”. In: <https://www.kaggle.com/datasets/frettapper/micropcb-images> (2021).
- [173] Alexandre Lacoste et al. “Quantifying the carbon emissions of machine learning”. In: *arXiv preprint arXiv:1910.09700* (2019).
- [174] Lars Ståhle and Svante Wold. “Analysis of variance (ANOVA)”. In: *Chemometrics and Intelligent Laboratory Systems* 6.4 (1989), pp. 259–272. ISSN: 0169-7439. DOI: [https://doi.org/10.1016/0169-7439\(89\)80095-4](https://doi.org/10.1016/0169-7439(89)80095-4). URL: <https://www.sciencedirect.com/science/article/pii/0169743989800954>.
- [175] André Vela, Joanna Alvarado-Urbe, and Hector G. Ceballos. “Indoor Environment Dataset to Estimate Room Occupancy”. In: *Data* 6.12 (2021). ISSN: 2306-5729. DOI: 10.3390/data6120133. URL: <https://www.mdpi.com/2306-5729/6/12/133>.
- [176] Jens Hjort Schwee et al. “Room-level occupant counts and environmental quality from heterogeneous sensing modalities in a smart building”. In: *Scientific Data* 6.1 (Nov. 2019), p. 287. ISSN: 2052-4463. DOI: 10.1038/s41597-019-0274-4. URL: <https://doi.org/10.1038/s41597-019-0274-4>.
- [177] Margarite Jacoby et al. “A high-fidelity residential building occupancy detection dataset”. In: *Scientific Data* 8.1 (Oct. 2021), p. 280. ISSN: 2052-4463. DOI: 10.1038/s41597-021-01055-x. URL: <https://doi.org/10.1038/s41597-021-01055-x>.
- [178] Dhanya Pramod Anil Jadhav and Krishnan Ramanathan. “Comparison of Performance of Data Imputation Methods for Numeric Dataset”. In: *Applied Artificial Intelligence* 33.10 (2019), pp. 913–933. DOI: 10.1080/08839514.2019.1637138. eprint: <https://doi.org/10.1080/08839514.2019.1637138>. URL: <https://doi.org/10.1080/08839514.2019.1637138>.
- [179] Panagiotis Filippakis, Stefanos Ougiaroglou, and Georgios Evangelidis. “Prototype Selection for Multilabel Instance-Based Learning”. In: *Information* 14.10 (2023). ISSN: 2078-2489. DOI: 10.3390/info14100572. URL: <https://www.mdpi.com/2078-2489/14/10/572>.
- [180] Muhammad Fahim Uddin. “Addressing Accuracy Paradox Using Enhanced Weighted Performance Metric in Machine Learning”. In: *2019 Sixth HCT Information Technology Trends (ITT)*. 2019, pp. 319–324. DOI: 10.1109/ITT48889.2019.9075071.
- [181] Andrew P. Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159. ISSN: 0031-3203. DOI: [https://doi.org/10.1016/S0031-3203\(96\)00142-2](https://doi.org/10.1016/S0031-3203(96)00142-2). URL: <https://www.sciencedirect.com/science/article/pii/S0031320396001422>.
- [182] s.r.o. Malík M.; REALiX. *HWiNFO*. <https://www.hwinfo.com/>. Accessed: 10 April 2025.