university of groningen

faculty of science and engineering

Dataset Reduction Methods for Data and Energy Efficient Deep Learning

Daan Krol

January 13, 2023

university of groningen

faculty of science and engineering

University of Groningen

Dataset Reduction Methods for Data and Energy Efficient Deep Learning

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
dr. S.H. Mohades Kasaei (Artificial Intelligence, University of Groningen)
and
dr.ir. L.J. Cornelissen (AI Frameworks Engineer, Intel, Netherlands)
dr.ir. L.E. Hogeweg (Machine Learning Researcher, Intel, Netherlands)

Daan Krol (s3142221)

January 13, 2023

Contents

Acknowledgments

This project turned out to be much larger than initially expected and I would like to use this opportunity to express my appreciation for the people without whom it would not have been possible. First and foremost, I would like to thank Hamidreza Kasaei for his valuable input, support, and feedback throughout the research process. I also extend my gratitude to Intel Groningen for providing access to their high performance cluster, as well as to my supervisors and colleagues at Intel for their knowledge, feedback, and fruitful discussions.
I am especially grateful to Ludo Cornelissen and Laurens Hogeweg for sharing their expertise on research methodologies, engaging in theoretical brainstorming sessions, and offering continuous guidance throughout this project.
I would also like to thank the authors of the CORDS library for their generous access to their resources. My sincere thanks also go to my friends Julian Bruinsma and Jeroen van Brandenburg for their valuable insights and for providing much-needed distraction during the course of this project.
Finally, I am deeply grateful to my parents for their unwavering support and to my mother for her linguistic expertise. Without the help and support of all of these individuals, this thesis would not have been possible.

Abstract

Artificial intelligence models that achieve high performance in tasks such as natural language processing and computer vision often require large and complex datasets. However, the use of these datasets can be costly in terms of computational expense, training time, and energy consumption. The carbon footprint of these models can also be significant. Data-efficient machine learning, which aims to reduce the amount of data required to achieve good performance, is therefore an important area of research. Dataset reduction methods aim to select or synthesize a smaller subset of the data that still allows good performance to be achieved. These methods can be applied non-adaptively by pruning the dataset in advance or adaptively by selecting a subset of the data every X epochs while training.

In this thesis, we provide an in-depth analysis of 6 non-adaptive and 14 adaptive dataset reduction methods. Several optimizations are presented that can further increase the speedup or that can increase the performance while offering some speedup. In addition, we present a novel adaptive dataset reduction method called Supervised Contrastive Learning (Super-CL), which is based on an existing method called Contrastive Active Learning (CAL). We test these algorithms on biodiversity datasets, which are complex datasets that make a good test case for operational machine learning settings as they give a good representation of the imbalanced world. Our results show that some non-adaptive methods can effectively prune datasets when redundancy is present. However, we also find that there is no adaptive method that can effectively speed up training on all datasets, suggesting that more research is needed to define a robust adaptive method. In addition, we explore the effect of combining transfer learning with dataset reduction to further reduce energy consumption, which shows promising results and can in some cases speed up training with transfer learning.

1 Introduction

Recently, large complex models in Artificial Intelligence have shown remarkable results in areas such as Natural Language Processing and Computer Vision [1, 2, 3, 4]. The increase in model performance is said to be related to the exponential increase in the number of parameters [5]. The increase in model complexity comes with the need for massive datasets. This results in heavy computational expense, an increase in training time, and a large increase in memory and energy consumption. The hardware to perform tensor operations requires large amounts of energy and can have a large carbon footprint [6, 7]. The models are thus not only costly to train and evaluate financially but also environmentally. Optimizing and researching these models requires many trials of training and validation runs over the dataset for different hyperparameters. This can multiply these costs by thousands of times [8].
For example, the GPT-3 model [1] with 175×10^9 parameters needed around 28000 days of training time on one V100 GPU or 90 days on 310 V100 GPUs. In [7] it is estimated that this results in 84 748 kgCO_2eq which is equivalent to driving a new car for 703 808 km, almost as far as driving to the moon and back. It is around the same amount of CO_2 equivalent emissions as 12 US households over 1 year. This is only for the final training cycle and does not include R&D such as hyperparameter optimization or neural architecture search.
Therefore, data-efficient machine learning is a very important research field [9]. To reduce the amount of computation needed, the size of the dataset should be reduced without losing significant performance. Not only can a reduced dataset reduce the computing required by a large amount, in some cases it can also increase the accuracy of the model by filtering out noisy data or mislabeled samples [10].
Dataset reduction methods try to select a subset that contains the most informative samples or that represents the dataset well. Dataset reduction methods have applicability in multiple areas such as Incremental Learning in which input data is continuously used and where the number of tasks to be learned (e.g. the classes in a classification task) are not static. Training in the continuous stream of information can result in catastrophic forgetting where previously learned tasks are forgotten by the model. Dataset reduction can be used as a method to construct a memory buffer which can alleviate this problem.
Dataset reduction methods can also be used in an Active Learning setting. In this setting, there is a small pool of labeled data and a large pool of unlabeled data. The learning algorithm queries an expert (often a human annotator) to label a subset of unlabeled samples, resulting in a larger labeled dataset. Labeling is often expensive and Active Learning tries to solve this problem by querying the most informative samples. After each increase of the dataset size, a model is trained on the labeled dataset. This differs from Incremental Learning, in which the model training continues by training only on the new data subset. Reducing the size of the dataset could greatly reduce the computational need when applying it after each run or on the initial labeled dataset at the start of the Active Learning process. Furthermore, (unsupervised) dataset reduction techniques can often be used to efficiently find the subset of unlabeled samples that the expert should label. Intelligent selection of the most informative and diverse samples can result in higher model performance with fewer labeled data [11]. Thus, potentially decreasing labeling costs.
Dataset reduction techniques can be subdivided into two categories: selection- and synthesis-based methods. With selection-based methods, such as classical coresets [12], a small amount of the most relevant samples of the dataset is extracted. This can be done by simple uniform sampling or by looking at samples that the model is most uncertain about.
With synthesis-based methods such as Dataset Condensation/Distillation the information of the original dataset is condensed by synthesizing a new smaller dataset [13, 14, 15, 16, 17]. Synthesized

Figure 1: CIFAR-10 dataset condensed into one image per class [18]

images do not come from the original data distribution, but do try to reflect this, as the goal is to represent the underlying data distribution as closely as possible. For example, a class with 400 image samples can be condensed into one image. This results in an image that is more abstract but also more information dense.
This effect can be observed in Figure 1. Here, the image of the car class is an abstract representation of a car where some characteristics such as the wheels are present. Synthesis-based selection methods are still very novel and, although they show interesting results, they have not been shown to scale up to large resolution images such as the ImageNet dataset without the need for a large amount of computational resources [19].
In addition to reducing data sets to potentially save energy, there is another method that is widely used in practical machine learning, called transfer learning. Transfer learning is the process of using a pre-trained neural network as the starting point for a new task, rather than training a new network from scratch. This can be particularly useful when the new task is similar to the original task for which the pre-trained network was trained. Transfer learning can save energy in training neural networks because the pre-trained network has already learned many of the features that are relevant to the new task, so the new network does not have to learn these features from scratch. This can significantly reduce the amount of computation and data required to train the new network, which can be especially important when training large, complex neural networks. Since both dataset reduction and transfer learning can potentially save energy, it would be interesting to see if the methods can be combined to efficiently train neural networks. There has yet to be any research on the potential combination of these methods.
Dataset reduction methods differ in complexity, performance, and in the amount of computational resources needed to find a smaller set. Furthermore, some methods do not perform well with imbalanced datasets, and most methods have not been tested on fine-grained (e.g. biodiversity) datasets. Biodiversity datasets often contain a hierarchical class structure, overlapping classes, and large class imbalance. These complex datasets make a good test case for operational machine learning settings, as they give a good representation of the imbalanced world. In this research, multiple dataset reduction methods are implemented to find their advantages, limitations, possible improvements, and their trade-off between performance and computational resource need.

1.1 Research Questions

This research is focused on one main research question: *What are the optimal subset selection methods for dataset reduction?*
This broad question can be answered by dividing it into multiple subquestions. These are needed to look at the trade-off between energy efficiency and performance:

1. *How do the methods compare in speedup and performance?*
2. *How applicable are the methods on complex and fine-grained (biodiversity) datasets?*
3. *Can the methods be applied in combination with transfer learning to increase the total energy efficiency?*

1.2 Thesis Outline

The thesis is organized into seven chapters. In the second chapter, we present a review of existing dataset reduction techniques. We discuss both selection-based and synthesis-based methods, along with their potential applications and combination with transfer learning.
Chapter three focuses on the biodiversity datasets used throughout the thesis. We provide details on the characteristics and sources of these datasets, as well as the preprocessing steps applied to prepare them for the experiments.
Chapter four presents the architectures, metrics, and optimization techniques used in our study. We discuss how the dataset reduction methods can be applied adaptively and non-adaptively, and address potential issues with the metrics used to evaluate their performance. We also introduce our novel adaptive data subset selection technique SUPERVISED CONTRASTIVE LEARNING and present the optimization of the existing method CONTRASTIVE ACTIVE LEARNING. The experimental setup is described in detail in this chapter.
The results of the experiments are presented in chapter five. We provide comprehensive performance comparisons of the different dataset reduction techniques and discuss the implications of the results in chapter six. In addition to this, we try to answer the research questions as proposed earlier and present potential future research directions.
Finally, in the last chapter, we highlight the main contributions of our work to the field of data and energy efficient deep learning.

2 Background Literature

Several methods exist for subset selection such as coresets. The term coresets was first coined in [12] in which the authors discussed several (traditional) geometry based selection techniques that for a set of points can select a small subset, the coreset, on which computations can be performed. This idea was extended with data distribution approximation techniques for various geometric problems such as σ-*coresets* using kernels for point sets [20]. For a complete overview of traditional methods see [21].
With the large dimensionality of the data and high computational complexity of deep learning, these traditional geometric methods are not always applicable and their performance is doubtful [22]. With deep learning, coresets are viewed as finding a subset $S \subset T$ with $|S| < |T|$ and where T is the full training set. Training a model on S should have close generalization power to that of a model trained on T as measured by their inference performance on the test set. Dataset reduction methods can be performed merely on the training data without a relation to deep learning models. Other methods need interaction with the model to efficiently select the best subsets. In this section multiple methods that are applicable to deep learning are described.

2.1 Selection techniques

Geometry based In a large dataset, T, there can be samples that show similar attributes (properties) and thus are relatively close to each other in feature space. These samples are conjectured to be non-informative for the model and will not give a significant increase in generalizational power. For image data this can be seen as a case of Near-Duplicate Removal [23]. Multiple geometry based methods exist that can find and remove these samples.
The *K-Center Greedy* method [24] views the coreset problem as a minimax facility location problem. This comes down to choosing multiple center points in the dataset such that the largest distance between a datapoint and the nearest center point is minimized. The original K-Center problem is NP-hard. The authors present a robust greedy approximation to solve this. The algorithm iteratively adds samples from T to S such that the largest distance between a datapoint from T to the closest point in S is minimized. K-Center Greedy can be applied to image data by applying the distance metric on the feature embedding of the images, for example by using a pre-trained network as feature extractor.
A simple geometric method is *Herding* in which samples from T are iteratively added to S to minimize the distance, in feature space, between the centerpoints of S and T [25].
In [26] the authors present both a supervised and a self-supervised pruning metric that can be used to reduce dataset size. For the self-supervised metric they first perform *k-means* clustering on the embedding space of a ImageNet pre-trained self-supervised model (SWaV [27]). The difficulty of each sample can then be expressed by its distance to the nearest prototype. A prototype can be seen as a cluster center or class representation. The supervised metric of course uses the label definition. For each class, the class prototype can be computed by taking the average of the feature vectors for all samples within a class. In their research they show that when more than 70% of the data of the ImageNet dataset is kept, the metrics show similar performance. For subsets with smaller size, the supervised metric outperforms the unsupervised approach. Throughout this thesis we denote dataset reduction with the supervised metric as the *Protoypical* data subset selection method.

Uncertainty based Uncertainty based methods try to find examples that are hard for the model to make inferences on. The samples on which the model is not confident about the class membership, should be selected to improve the discriminative ability of the model. This is often used in Active

Learning as a baseline. Model dependent methods such as uncertainty sampling can be applied on an unsupervised dataset to find samples that the model is not confident about. These samples are then labeled by a human expert such that the model can be trained on them. A disadvantage of uncertainty sampling is that it does not explicitly regard the diversity of the selected subset. It is thus not guaranteed that all classes will be present in the subset and the subset is thus not always a correct representation of the full dataset. These properties do not make it a good candidate for dataset pruning.
Uncertainty can be measured with multiple metrics such as:

1. *Least confidence* can be quantified by the difference between the maximal possible confidence and the predicted probability. Selection is based on samples with high *least confidence* values: $u = 1 - \max_i p_i$

2. *Margin of confidence* is quantified by the difference of the top two predictions. Samples with high margin of confidence are selected first. The uncertainty is then defined as: $u = 1 - (\max_i p_i - \max_{j \in \{C\} - \operatorname{argmax}_i p_i} p_j)$, where C is the set of classes.

3. *Entropy* describes the average level of uncertainty inherent to the possible outcomes of a random variable. If all outcomes are similar, entropy is low. If the difference between model predictions is high, entropy is high. It is defined as: $entropy(x) = \sum_{i=1}^{C} P(\hat{y} = i|x) logP(\hat{y} = i|x)$. The uncertainty then becomes: $u = -entropy(x)$

Another approach is *Hard Negative Mining* [28] in which we rank false positives based on the confidence of the network on the wrongly predicted class or rank them (increasingly) by the probability of the actual class. The model is thus confident about the class each sample belongs to but that it is actually absolutely wrong about. This has overlap with least confidence sampling although with least confidence sampling we do not necessarily need label information. Another difference is that *Hard Negative Mining* is solely focused on the negative samples.
A potential problem with uncertainty based methods is that the selected samples that the model is uncertain about might be similar in feature space. For example, when a model can discriminate between classes A and B but is very uncertain about samples from class C. Uncertainty sampling will then select a subset which mostly represents class C and as such the subset lacks diversity. Without diversity, the problem is that models might be biased towards certain classes. One solution would be to select a subset such that the samples are as diverse as possible, but this might result in non-informative samples.
In [11] the method *Batch Active learning by Diverse Gradient Embeddings (BADGE)* is presented. This unsupervised method tries to solve this problem by sampling a subset that the model is uncertain about but in which the samples are diverse. Uncertainty is measured by looking at the loss gradients of the hypothesized labels. The hypothesized label is the label with the highest activation in the output layer, as the true label is unknown. Diversity is captured by using the *k-Means++* algorithm on the (hypothesized) gradient space of the samples and selecting samples from each cluster.

Submodularity based A submodular function can be described as follows: Let n be the number of data points in a ground set Ω and let f be the (power) set function $f : 2^{\Omega} \to \mathbb{R}$. That translates a set of random variables to a real value. This function can be used to measure how good a subset is, if properly defined. The f function is submodular if for all $S, T \subseteq \Omega$ it holds that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ [29]. Submodular functions follow the diminishing marginal returns property which

states that the difference in incremental value of f when adding an item to the input set decreases as the set size increases. In other words, adding a sample x to a subset A results in more gain in terms of f then when adding the x to A', when $A \subset A'$. A' already contains more information than A since it is a larger superset. Submodular functions can thus be used to measure the informativeness and diversity of sets, which can be very useful for efficient data subset selection.
Equation 1 defines the data subset selection algorithm as a submodular optimization function with a constraint that the subset size is no larger than b, the budget:

$$\max\{f(X) \text{ such that } |X| \leq b\} \tag{1}$$

Submodular functions can be solved by greedy functions which iteratively add samples with the highest gain, according to f. It has been shown that this results in a lower-bound performance guarantee of a factor of $1 - 1/e \approx 0.63$ of the optimal of the problem defined in Equation 1 [30].
The previously mentioned *entropy* metric is such a submodular function. In [31] multiple submodular information measures are discussed in terms of applicability on machine learning, such as *Facility Location* and *Graph Cut*. These three functions can be successfully applied as metric for subset selection and show good results with small subsets [22].
Facility Location functions are functions that, when maximized, select samples that represent the dataset well. It thus can be seen as a greedy version of the *k-medoids* algorithm [32]. The facility location function uses a pairwise similarity matrix of the dataset. It chooses samples for the subset that maximize the pairwise similarities to samples in the ground set. The general definition of a facility location problem is defined in Equation 2. This thus differs from the *K-Center Greedy* method in that it uses pairwise similarity instead of distance.
A Graph Cut function is a function that selects samples from the ground set such that the similarity matrix is split into subgraphs. The general definition of a graph cut function can be observed in Equation 3.

$$f(S,T) = \sum_{t \in T} \max_{s \in S} \phi(s,t) \tag{2}$$

$$f(S,T) = \lambda \sum_{t \in T} \sum_{s \in S} \phi(s,t) - \sum_{s,y \in S} \phi(s,y) \tag{3}$$

where:

S = the subset
T = the ground set
ϕ = the similarity function
λ = edge connectivity

In [33] the authors connect submodularity to likelihood functions of the Naive Bayes and Nearest Neighbour classifiers in a supervised setting. Additionally, they present *filtered active submodular selection (FASS)* in which they combine the uncertainty sampling method with submodularity based subset selection for an Active Learning setting. The uncertainty scores are used to get the hypothesized labels for unlabeled samples. *FASS* outperforms both random and uncertainty sampling in an digit recognition experiment in an Active Learning setting.

Margin based Datapoints that lie close to the decision boundary are harder to discriminate and thus likely more informative for the network. Selecting these samples can increase the discriminative performance of the model. This can especially be useful when working with fine-grained datasets, in which classes can be hard to distinguish as there can be overlap in feature space. In [34] these samples are called *contrastive* as they are close in feature space but the model outputs different probability distributions over the classes. The authors applied this method in an Active Learning setting where not all data is labeled. The algorithm that selects samples based on their contrastive scores is called *Contrastive Active Learning (CAL)*. Query functions for Active Learning are often based on either diversity or uncertainty sampling. With CAL, the authors try to leverage the best of both worlds and try to combine these methods into a single query method. Their method was tested on seven text datasets in multiple natural language tasks. It showed equal or better performance than a large selection of baseline active learning query methods. Although they apply it on unlabeled data, it can also be applied in a supervised setting such that label information is also used. Furthermore, CAL was applied on text data but the algorithm could also be applied on image data by using the feature space of the images. A single iteration of the CAL algorithm can be observed in Algorithm 1.
With complex data representations, the distance to the decision boundary for a sample cannot always be computed. In [35] the authors propose *Adversarial DeepFool* to approximate this distance by using adversarial examples. By adding perturbations in the labels, the predictive power of the model is influenced. Samples that require the least adversarial perturbation to cross the decision boundary are selected. This however requires a large amount of computational power.

Algorithm 1 A single iteration of CAL.

Require: labeled data $\mathcal{D}_{\textbf{lab}}$, unlabeled data $\mathcal{D}_{\textbf{pool}}$, acquisition size b, model $\mathcal{M}$, number of neighbours k, model representation (encoding) function $\Phi(.)$

for x_p in $\mathcal{D}_{\textbf{pool}}$ **do**
  $\left\{(x_l^{(i)}, y_l^{(i)})\right\}, i = 1,...,k \leftarrow KNN(\Phi(x_p), \Phi(\mathcal{D}_{\textbf{lab}}), k)$ ▷ find neighbours in $\mathcal{D}_{\textbf{lab}}$
  $p(y|x_l^{(i)}) \leftarrow \mathcal{M}(x_l^{(i)}), i = 1,...,k$ ▷ compute probabilities
  $p(y|x_p) \leftarrow \mathcal{M}(x_p)$
  $D_{KL}(p(y|x_l^{(i)})||p(y|x_p)), i = 1,...,k$ ▷ compute divergence
  $s_{x_p} = \frac{1}{k}\Sigma_{i=1}^{k} D_{KL}(p(y|x_l^i)||p(y|x_p)$ ▷ score
end for
$Q = \underset{x_p \in \mathcal{D}_{\textbf{pool}}}{\operatorname{argmax}} s_{x_p}, |Q| = b$
return Q

Loss based Not every sample has the same influence on the loss when training a network. The influence on the loss can be used as a selection method for the coreset, since samples that have large influence on the loss are informative. This approach is similar to uncertainty based methods but instead uses metrics focused on the loss contribution or error value during training. In [36] the authors present simple scoring metrics, *Gradient Normed (GraNd)* and *Error L2-Norm (EL2N)*, which can be used to prune the dataset very early in training. To compute the GraNd scores for a dataset, the authors randomly initialize ten deep neural networks. The expected score for each sample can be computed by Equation 4 using the model parameters θ at time t.

$$\mathcal{X}_t(x,y) = \mathbb{E}_{\theta_t}||\nabla_{\theta_t} l(x,y;\theta_t)||_2 \tag{4}$$

The final score for the sample is then the average sample score over all networks. The authors report that the EL2N metric can provide even better information for pruning data. To compute the EL2N scores for all samples, they again use ten deep neural networks but now they are first trained for 20 epochs on the training data. The expected EL2N score can then be computed by Equation 5 using the model parameters θ at time t.

$$\mathcal{X}_t^*(x,y) = \mathbb{E}_{\theta_t}||p(\theta_t,x) - y||_2 \tag{5}$$

Larger scores represent more informative samples which should be selected. With their method the authors are able to prune half of the CIFAR10 dataset after just a few epochs while slightly improving test accuracy.
This differs from methods such as *Forgetting Events* [37] in which samples are pruned based on how often they are forgotten during training. Forgetting is represented by measuring the number of times that a sample was misclassified in a consecutive epoch of when it was correctly classified. They note that a significant portion of the dataset can be removed from the dataset while still maintaining state-of-the-art generalization performance. This method thus needs more resources as it cannot identify these samples early on, but is simple to implement.

Gradient matching based Stochastic gradient descent (SGD) methods used for training deep learning models approximate the full gradient of the dataset by calculating it iteratively only on small subsets or batches.
Methods such as CRAIG [38] select a weighted subset of which the gradient closely resembles the gradient of the full dataset by maximizing a submodular function. The authors prove that the subset S that minimizes an upper-bound on the error of estimating the full gradient, maximizes a submodular facility location function. Hence, S can be efficiently found using a fast greedy algorithm which iteratively selects a sample that reduces the upper bound on the estimation error the most. CRAIG is said to deliver a 3x speedup for training deep neural networks without losing significant performance. The authors also show that any incremental gradient method on the subset S, obtained by CRAIG, converges in the same number of epochs as it would on the full dataset T. The speedup by CRAIG is thus inversely proportional to the size of S.
A similar approach is that of [39] in which the authors present GRAD-MATCH. This gradient matching

Algorithm 2 GRAD-MATCH Algorithm, source: [39]

Require: Train set $\mathcal{U}$, validation set $\mathcal{V}$, initial subset $\mathcal{X}^{(0)}$, subset size k, tolerance ε, initial parameters θ_0, learning rate α, total epochs T, selection interval R, validation flag *isValid*, batch size B

for epochs t in $1,...,T$ **do**
  if $(t \mod R == 0)$ and $(isValid == 1)$ **then**
    $\mathcal{X}^t, \mathbf{w}^t = \text{OMP}(L_T, L_V, \theta_t, k, \varepsilon)$
  else if $(t \mod R == 0)$ and $(isValid == 0)$ **then**
    $\mathcal{X}^t, \mathbf{w}^t = \text{OMP}(L_T, L_T, \theta_t, k, \varepsilon)$
  else
    $\mathcal{X}^t = \mathcal{X}^{t-1}$
  end if
  $\theta_{t+1} = \text{BatchSGD}(\mathcal{X}^t, \mathbf{w}^t, \alpha, L_T, epochs = 1)$
end for
return final model parameters θ_T

method minimizes an error term on the gradient difference between the selected weighted subset and the gradient of either the training or validation set. Thereby directly optimizing for the gradient mismatch error and resulting in the state-of-the-art accuracy-efficiency trade-off for adaptive data subset selection. The GRAD-MATCH algorithm, which can be found in Algorithm 2, has been shown to provide better error bounds on the estimation of the gradient compared to CRAIG. The definition for this error can be found in Equation 6.

$$Err(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t) = \left\| \sum_{i \in \mathcal{X}^t} w_i^t \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t) \right\|^2 \tag{6}$$

where:

θ_t = model parameters at iteration t
L = either the training or validation loss
$\mathcal{X}^t$ = subset $\mathcal{X}$ at iteration t
w_i^t = weight for sample $i \in \mathcal{X}^t$
$L_T^i(\theta)$ = $L_T(x_i, y_i, \theta)$, the training loss at epoch i of training
$L_T(\mathcal{X}, \theta) = \sum_{i \in \mathcal{X}} L_T(x_i, y_i, \theta)$, the loss on the subset $\mathcal{X} \subseteq \mathcal{U}$ of the training samples

The adaptive subset selection algorithm results in subsets $\mathcal{X}^t$ for $t = 1, 2, ..., T$ with corresponding sample weights $\mathbf{w}^t$. So at iteration t, the model parameters are updated using the weighted loss of the model on the subset $\mathcal{X}^t$ that is weighted by w^t. The authors show that minimizing the gradient matching error can be seen as a weakly submodular maximization problem and they use a greedy orthogonal matching pursuit (OMP) algorithm [40]. OMP is an iterative greedy algorithm that comes from the field of signal processing and is used to find the sub-optimal solution to the problem of sparse signal representation. It can be seen as a facility location problem with the addition that besides samples, it also computes weights.
To prevent the algorithm from overfitting, the authors add a regularization term with coefficient λ and an L2 loss regularizer over the weight vector $\mathbf{w}$. Equation 6 then becomes Equation 7.

$$Err_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) = Err(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) + \lambda \|\mathbf{w}\|^2 \tag{7}$$

By minimizing this error term, the GRAD-MATCH algorithm can find a good representation of the gradient space of the training or validation set. The data selection optimization problem is then defined by Equation 8 which is constrained by the size of the selected subset $\mathcal{X}$.

$$\mathbf{w}^t, \mathcal{X}^t = \underset{\mathbf{w}, \mathcal{X}: |\mathcal{X}| \leq k}{\operatorname{argmin}} \; Err_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) \tag{8}$$

The GRAD-MATCH algorithm selects the weights and the subset by optimizing Equation 8. This results in Equation 9:

$$E_\lambda(\mathcal{X}) = \min_{\mathbf{w}} Err_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) \tag{9}$$

Note that solving Equation 8 is the same as solving $min_{\mathcal{X}:|\mathcal{X}| \leq k} E_\lambda(\mathcal{X})$. The OMP algorithm, which can be found in Algorithm [40], can then be used to solve this equation.
The authors present several optimizations to speed up the GRAD-MATCH algorithm. For example, they propose to apply the algorithm on individual mini-batches instead of individual samples. This could greatly decrease the computations needed to solve Equation 9 by a factor of the batch size. The mini-batch optimization problem then becomes:

$$E_{\lambda}^{B}(\mathcal{X}) = min_{\mathbf{w}} \left\| \sum_{i \in \mathcal{X}} w_t^i \nabla_{\theta} L_T^{B_i}(\theta_t) - \nabla_{\theta} L(\theta_t) \right\| + \lambda \|\mathbf{w}\|^2 \tag{10}$$

where:

B	= batch size
b_n	= the number of mini-batches
b_k	= k/B, the number of mini-batches to be selected
$\nabla_{\theta} L_T^{B_1}(\theta_t), ..., \nabla_{\theta} L_T^{B_{b_n}}(\theta_t)$	= the mini-batch gradients

When the algorithm is applied per mini-batch instead of per sample, the GRAD-MATCH algorithm is then referenced as GRAD-MATCHPB. In addition to this, they do not use all layer gradients, as this becomes intractable for large neural networks. Instead, they use the last-layer gradients. All gradients, however, still need to be saved to memory to be able to solve the optimization problem, which can result in out-of-memory errors for large datasets. To circumvent this, the authors propose per-class and per-gradient approximations. With a per-class approximation the gradient matching problem is solved per class and as such requires much less memory. Combining this with the per-gradient and last-layer approximations, the authors were able to speed up the GRAD-MATCH algorithm while reducing memory consumption. Note that the optimizations per class and per gradient do not have to be applied to the GRAD-MATCHPB algorithm.

Algorithm 3 Orthogonal Matching Persuit (OMP), source: [39]

Require: Training loss L_T, target loss L, current parameters θ, regularization coefficient λ, subset size k, tolerance ε

$\mathcal{X} \leftarrow \emptyset$
$r \leftarrow \nabla_w \text{Err}_{\lambda}(\mathcal{X}, \mathbf{w}, L, L_T, \theta)|_{w=0}$
while $|\mathcal{X}| \leq k$ and $E_{\lambda} \geq \varepsilon$ **do**
  $e = \underset{j}{\text{argmax}} |r_j|$
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{e\}$
  $\mathbf{w} \leftarrow \underset{\mathbf{w}}{\text{argmin}} \text{Err}_{\lambda}(\mathcal{X}, \mathbf{w}, L, L_T, \theta)$
  $r \leftarrow \nabla_{\mathbf{w}} \text{Err}_{\lambda}(\mathcal{X}, \mathbf{w}, L, L_T, \theta)$
end while
return $\mathcal{X}, \mathbf{w}$

Both methods work well with (artificially) imbalanced data when the validation set is kept balanced, but the methods are not tested on fine-grained (biodiversity) datasets. It would thus be interesting to see how they compare with other selection methods when the full dataset is unbalanced and there is class overlap present.

Bi-level optimization based In [41] the coreset selection problem is expressed as a discrete-continuous bi-level optimization problem. Bi-level, as it has two optimization objectives. The outer objective is to select a subset S that is a good representation of the full set, while the inner objective is to optimize the model parameters θ with respect to S. They present GLISTER, a method in which coresets are selected that maximize the log-likelihood of the validation set. The data subset selection optimization is defined by the following:

$$\mathcal{S} = \underset{\mathcal{S} \subset \mathcal{T}}{\text{argmax}} \sum_{(x,y) \in \mathcal{V}} ll \left(x, y; \underset{\theta}{\text{argmax}} \sum_{(x,y) \in \mathcal{S}} ll(x, y; \theta) \right) \tag{11}$$

where:

$\mathcal{S}$ = the subset
$\mathcal{T}$ = the training set
$\mathcal{V}$ = the validation set
ll = the log-likelihood

To save computational resource needs, greedy data subset selection is not run every epoch. The algorithm can be applied both in a supervised and unsupervised setting as it does not label information. GLISTER can get higher performance early in training than CRAIG and random uniform sampling. However, the final performance is very comparable and the question remains if the extra computational need is justifiable.

2.2 Synthesis techniques (Dataset Condensation)

With the synthesis-based dataset reduction techniques, the goal is not to select the most informative samples, but instead to learn how to synthesize new samples that can be more informative than the original images. These methods are often called Dataset Distillation or Condensation (DC) as they show properties similar to that of Model (or Knowledge) Distillation in which large trained models are compressed into smaller ones while maintaining good performance [42]. A general overview of Dataset Distillation can be found in Figure 2.

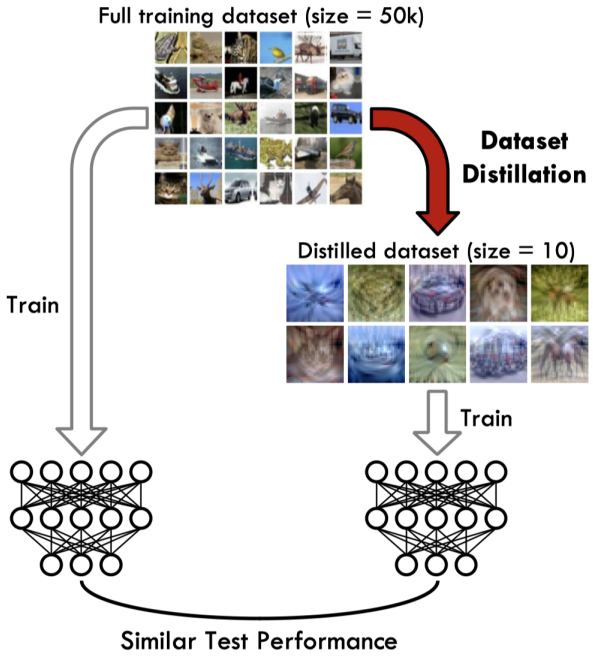

Figure 2: An overview of the dataset distillation process. Source: [18]

In [13] instead of optimizing the model parameters for an inference objective, the pixel values of the distilled images were optimized. The MNIST dataset of 60 000 images was compressed into 10 synthetic images with one image per class. Models trained on this extremely small dataset reached 94% accuracy instead of 99% when training on the whole data set. Using AlexNet [43] pre-trained on ImageNet they were able to distill the PASCAL-VOC and CUB-200 datasets into one image per class. Fine-tuning a new pre-trained AlexNet on the synthetic set showed similar performance as when training on the whole sets containing thousands of images. Fine-tuning was done with only one gradient step repeated for three epochs. While this shows that it might be possible to synthesize datasets, the authors also note that there are problems when the initialization of the synthesizing model is different from that of the latter classification model. Data sets synthesized by one architecture could not be used successfully to train a model with a different architecture.

These problems were solved by reformulating the DC problem as a gradient matching problem [15]. The idea is that the difference in gradients of two sets of network parameters, one trained on the synthetic set and one on the original set, should be minimized. Condensed sets created using one architecture can also be used to train different architectures, showing that the original initialization

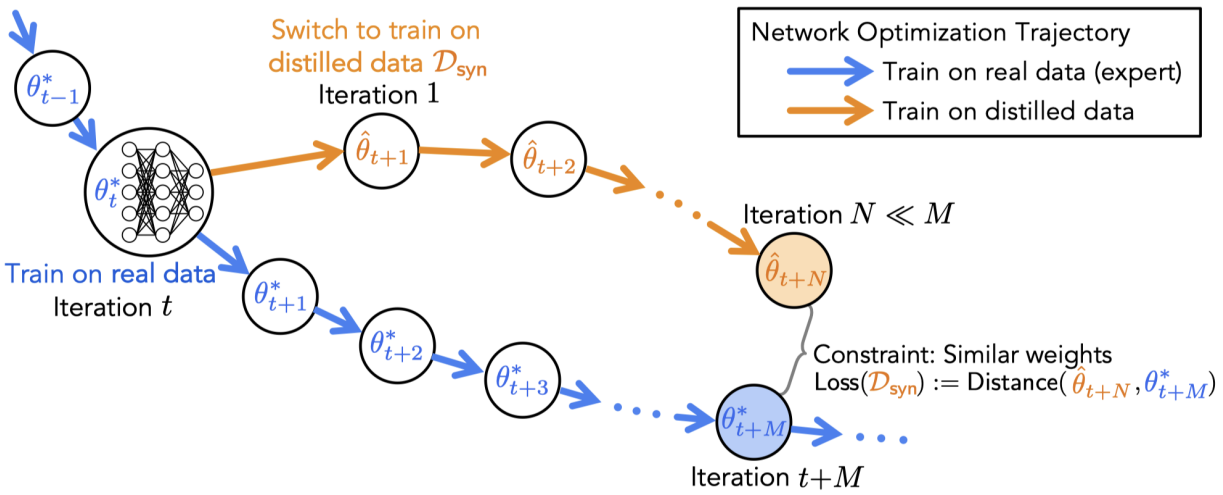

Figure 3: The trajectory matching between training on the synthetic and real data. The synthetic data $\mathcal{D}_{syn}$ is trained such that N training steps on this dataset will match the same result in parameter space as training M steps on the real data. (Source: [18]).

problems are solved. Although this method does outperform the original DC method and is more energy efficient when training from scratch, it is not tested with a pre-trained network. Additionally, it was not tested with more complex and higher resolution datasets as in [13].

In [44] the authors showed several limitations of the gradient matching approach. In the gradient matching method, images are condensed with a per-class strategy. This results in dominating task-irrelevant class-common features, and the method fails to pay attention to class-discriminative or *contrastive* features. This is especially a problem with fine-grained datasets. This is solved by using a Data Condensation method that sums up the gradients over all classes before measuring the gradient distance instead of doing this per class. The method shows much better performance on fine-grained datasets than the original DC and gradient matching methods. However, it was not tested on more complex and higher-resolution datasets, and the effect on pre-trained networks was not discussed.

The CAFE method proposed in [17] outperforms previous DC methods by using three different modules: (i) a feature alignment module that tries to preserve the original features in the synthetic set, (ii) a modified loss function that focuses on contrastive or discriminative features between classes, and (iii) a bi-level optimization technique to prevent over- and under-fitting. However, the method does not scale to high-resolution images such as ImageNet.

A recent method, called *Dataset Distillation by Matching Training Trajectories*, as proposed in [18], outperforms all previously mentioned DC methods and is the first method to scale up to large-scale datasets such as ImageNet. First, a large amount (around 200) of *expert* networks are trained on the full dataset and their parameter update trajectories are saved. Student networks are trained on the synthetic dataset, and the dataset is optimized by the difference of the student and expert training trajectories. This process can be observed in Figure 3 and the synthesized images from some of the classes of the ImageNet dataset can be observed in Figure 4. While this method is now state-of-the-art in Dataset Condensation and is able to scale up to large datasets, it does need a large amount of computing power and memory storage to construct the expert model training trajectories.

The overall problem with these synthesis-based techniques is their large demand of computational resources. The advantage of these methods is that they do construct a very small synthetic dataset, which is much smaller than coresets, as the samples are condensed with more informative information.

However, the number of GPU hours needed to synthesize a dataset is several magnitudes larger than the most demanding selection-based methods.

2.3 Applications

Dataset reduction methods can be used in multiple applications. The most basic application is to enable efficient training of machine learning models. This is especially useful for tasks such as Neural Architecture Search (NAS) or Hyperparameter Optimization (HPO) in which a large number of models have to be trained [45].

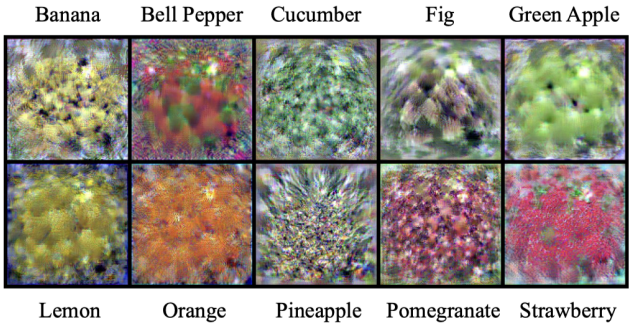

Figure 4: The distilled class images of ten classes from the ImageNet dataset. Source: [18]

In NAS, the key objective is to replace humans who design artificial neural networks. A search space can be defined with, for example, the possible types, number and resolution of layers. An algorithm can search through this space and find the best combination of architectural settings. The model has to be trained very often on the same dataset to see the performance of each architecture.
With HPO, the goal is to optimize the performance of a given architecture by optimizing the training hyperparameters such as the learning rate or batch size.
If the dataset could be safely reduced such that the model test performance is comparable when training on all data, this could save enormous amounts of energy and time for both NAS and HPO. Even if the reduced dataset results in a decreased performance, it could still be used as a proxy for NAS and HPO after which the model can be trained on the full dataset.

Active Learning Deep learning requires large amounts of labeled data. However, this can be a problem because many datasets do not have label information. Manually labeling the data can take a long time and results in large labeling costs. An overview of an active learning cycle can be found in Figure 5. An acquisition or query function selects samples from the unlabeled set. An expert then labels these samples, adds them to the labeled set, and then a new model is trained on the extended labeled set. Intelligently selecting the unlabeled samples based on their informativeness can greatly speed up this process. Good query methods could potentially save labeling costs and reduce the required computations. In addition to that, active learning can even improve model performance in some cases as the querying technique can filter out noisy data. Some of the dataset reduction methods can work with unlabeled data and could thus be used as active learning

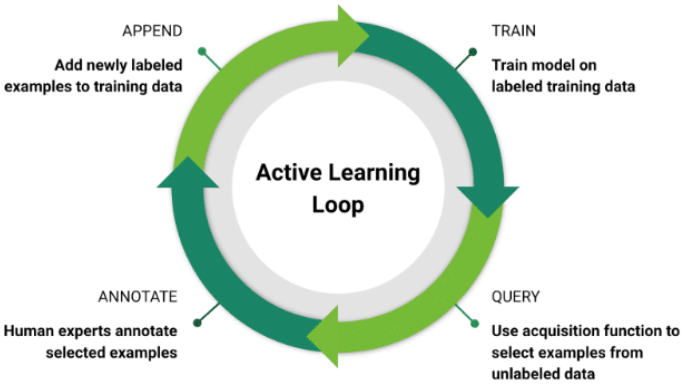

Figure 5: An overview of the Active Learning loop. Source.

querying methods. Some of the methods are actually active learning query methods, but can also be used as a dataset reduction technique. Dataset reduction methods could also be used next to the active learning querying technique. Most acquisition functions use the current model parameters to select samples that the model is uncertain about. However, this model is changed after each train step. Its confidence over the samples in the labeled set can have changed, and thus there can now be redundant samples in the labeled set. Here dataset reduction methods could potentially reduce this set.

Continual Learning In some cases not all data on which a model is trained are always available. This is called Continual Learning in which model training has to be continued when new data becomes available. A problem with continual learning is that the network can be subject to catastrophic forgetting. This is the tendency of the network to abruptly forget previously learned information. A common solution to this problem is to keep a memory buffer in which some (or all) of the previous samples are saved. With the arrival of new data, the model can be trained on some of the previous samples in the memory buffer. Thus, a memory buffer has to be a good representation of the dataset in order to be able to avoid catastrophic forgetting. Dataset reduction techniques could be a viable approach to constructing these memory buffers.

Foundation Datasets Recently, the term Foundation Models has been coined for large artificial neural networks that are trained on huge unlabeled datasets and can be individually fine-tuned for many applications. These models are often used by many people as the models are often open-source, but they require a great deal of data and energy to train. Examples of these models are BERT [46], GPT-3 [1], DALL-E [3] and now more recently Stable Diffusion [47]. Reducing the size of the datasets used to train these models could drastically reduce costs and could also benefit the innovation of these models. The costs of reducing the datasets can be repaid by the increase in efficiency in training these models.

2.4 Transfer Learning

Transfer learning refers to a situation where what has been learned in one setting (i.e., from data distribution P_1) is exploited to improve generalization in another setting (i.e., from data distribution P_2). The assumption is that many of the factors that explain the variations in P_1 are also relevant for the variations in P_2. For example, a model can be trained to discriminate between images of cats and dogs. This same model could then be used to discriminate between two new visual categories such as bees and wasps. In this case, if there are significantly more data, which are sampled from P_1, in the first setting, then this can help to quickly discriminate between samples in P_2. If the sample size and diversity (e.g. number of classes) from P_1 is large enough, this could allow for training a model that can then serve as a generic model of the visual word. This is because many visual categories share low-level features such as edges, shapes, and lighting effects. Therefore, it is important that there is some overlap of features in the low-level notions of the samples of P_1 and P_2.
Transfer learning has been shown to work in multiple domains and in different tasks such as image classification, text translation, sentiment analysis, object detection, and several others [48]. It could potentially solve the problem where there is not enough data in P_2 to effectively train a model. In addition to this, transfer learning can greatly decrease the time required to get a good generalization on P_2 because lower-level features are already learned. In practice, the model pre-trained on samples of P_1 does not have to be completely re-trained on the data of P_2. The lower-level layers of the model can already detect basic features that are useful for classifying samples. These layers can be kept *frozen*, which basically means that no training (for example, by backpropagation) will take place. The

other layers are then *fine-tuned* on the samples of P_2. In practice, this often means that several of the top-level layers are unfrozen, and the classification layer is adapted such that the number of nodes is equal to the number of classes.
Because of the potential speed-up, performance increase, and dependence on less data, it will be interesting to see if transfer learning could be combined with dataset reduction methods. Assuming that both dataset reduction methods and transfer learning can result in energy savings. Some research has shown that data set reduction can improve transfer learning performance in certain settings [26]. However, most of the methods discussed in this chapter have not been tested to see whether they can be applied in combination with transfer learning.

3 Datasets

The dataset reduction methods presented earlier have been tested on typical research datasets such as CIFAR10 [49], SVHN [50] and ImageNet [19]. These sets are designed for research purposes, and the research around the datasets is changing, not the dataset itself. However, in applied machine learning we often have to deal with more complex datasets where not every class has the same amount of samples and the class distribution is heavily imbalanced. Models trained on these datasets are often biased towards the dominant classes and do not perform well on tail classes. Biodiversity datasets are characterised by being long-tailed, having fine-grained classes, a hierarchical class structure, and having partially overlapping classes.
Class overlap can imply two things: Semantic overlap or overlap in feature space. Fine-grained class prediction can be seen as distinguishing subordinate categories within entry-level categories. For example not only recognizing if the animal is a bird but also which type of bird it is. This is thus a form of semantic overlap as the birds share the same super class. However, in practice this will also result in overlap in the feature representation of the two birds. Class overlap can become a serious problem for classification when class imbalance is also present in the dataset and vice versa [51]. The imbalance of each dataset can be defined by the *Shannon entropy*. A completely balanced dataset has a balance score of 1, where unbalanced datasets tend to 0. The balance score is defined as:

$$\begin{aligned} H &= -\sum_{i=1}^{k} \frac{c_i}{n} \log \frac{c_i}{n} \\ Balance &= \frac{H}{\log k} \end{aligned} \tag{12}$$

where:

H = Shannon entropy
k = number of classes in the dataset
c_i = number of samples in class i
n = total number of samples in the dataset

(a) Hooded Warbler (b) Western Gull (c) Acadian Flycatcher

Figure 6: Three samples of the CUB200 dataset with their class name. Note that these are the pre-processed images.

CUB200 Biodiversity datasets make for a perfect test bed for finding out if selection techniques are applicable in a realistic applied setting. The Caltech-UCSD Birds-200-2011 [52] is one of the most widely used datasets for fine-grained image classification. It is an extension of the original CUB-200 dataset [53] and contains approximately 12000 images of 200 bird subcategories, mostly North American bird species. There can be varying backgrounds and noise such as tree branches or leaves in front of the bird, which could could make it harder for the model to identify the bird. The images can vary in size and aspect ratio. When resizing and cropping this can have a negative effect when a part of the bird is outside the crop region. However, these augmentations are needed to reduce the memory consumption and to reduce training time, as the goal of this thesis is not to get the best performing model. See Figure 6 for some examples that show these difficulties. Each class has approximately 30 samples. Each image is accompanied by multiple annotations such as the bounding box and 312 binary attributes. As we want to compare the selection techniques over multiple vision datasets, we do not use these annotations except for the class label. Throughout this thesis, the Caltech-UCSD Birds-200-2011 dataset will be denoted as *CUB200*. The dataset has a predefined test set with 5,794 images out of the 11,788 images. A random stratified split is used for the validation set such that the class distribution in the training and the validation set remains equal. We make sure that there is at least one sample per class in the validation set. All images are first resized to 256×256 then center cropped to a resolution of 224×224 and then are normalized. During training, a random horizontal flip is used as augmentation with a probability of 0.5. CUB200 is a balanced fine-grained dataset which means that although each class has (almost) the same number of samples there is a lot of class-overlap. This results in a fairly complex dataset. The balance score is 0.96.

(a) Papilionidae Papilio demodocus demodocus

(b) Papilionidae Papilio lormieri

(c) Papilionidae Graphium antiphates itamputi

Figure 7: Three samples of the Papilionidae dataset with their class name to visualize the common properties of the images. Butterflies *a* and *b* are both from the Papilio species which shows the finegrainedness of this dataset. Note that these are the resized images.

Papilionidae The Papilionidae [54] dataset is a collection of butterflies of the family *Papilionidae* from the museum collection of Naturalis Biodiversity Center. The dataset contains 8243 images of butterflies where the classes have three hierarchical levels: Genus (11 classes), Species (79 classes) and Subspecies (112 classes). As we want to see how well the dataset reduction algorithms perform on biodiversity datasets, classification is based on the most fine-grained subspecies category. This has relatively more class overlap than the other categories and has the largest set of individual classes, increasing the complexity of the dataset. The images have a resolution of around 600×600 and are

resized to 224×224. During training, a random horizontal flip augmentation with a probability of 0.5 is used. The Papilionidae dataset is a heavy long tailed dataset as can be observed in the distribution plot in Figure 40. The largest class contains 1318 images, the smallest three images, and the long tail contains on average five samples per class. The balance score of the dataset is 0.69. No predefined train/test split is provided. 33% of the dataset will be used as test set. This is divided by using a stratified split, making sure that the class distribution remains equal and that there is at least one image per class in the test set. 10% of the train set is used as validation using the same splitting process.
The dataset images share several common properties. Images have comparable sizes such that resizing does not give large differences in aspect ratio, images are cropped correctly such that butterflies are always centered and all images have the same background with very little noise. This might help the model learn class features more easily, but it can also result in a model that has low generalisation power on butterflies in a realistic environment. See Figure 7 for three example images that visualize these properties.

iNaturalist The iNaturalist dataset [55] is a large biodiversity dataset containing 859,000 color images from more than 5,000 species of plants and animals. Datasets are often produced by scraping images from public websites which can result in selection bias. This dataset is instead produced by a crowd based effort. All labels were verified by multiple scientists. The classes are heavily imbalanced and very fine-grained. The images have a much higher resolution than, for example, the CIFAR10 dataset. The large size of the dataset in combination with the high resolution results in a need for much more computational power and memory.

CIFAR10 The CIFAR10 dataset [49] mentioned earlier contains 60,000 3-channel images with a resolution of 32×32. The dataset contains 10 classes that have no overlap and are completely mutually exclusive. The dataset is completely balanced, which means that all classes have the same number of samples. This is a fairly easy dataset to learn to classify with large models. It is used to test if the results of the papers of some of the selection methods [22, 38, 41, 39] can be replicated and it is used as a baseline. As the image size is small, we can train models and test selection methods rather quickly. The CIFAR10 dataset has a predefined test set with 10,000 samples. A validation split is produced using a stratified split of 10% of the training set. During training, a random horizontal flip augmentation with probability 0.5 is used. All images are normalized.

Dataset	**Total size**	**Train size**	**Test size**	**No. of classes**	**Fine-grained**	**No. of features**	**Balance score**
CIFAR10	60,000	50,000	10,000	10	✗	$32 \times 32 \times 3$	1
CUB200	11,788	5,794	5,994	200	✓	$224 \times 224 \times 3$	0.96
Papilionidae	8,243	5,440	2,803	112	✓	$224 \times 224 \times 3$	0.69

Table 1: Overview of datasets. Note: No. of features after pre-processing.

4 Methods

This chapter will first cover the architectures and training parameters that will be used to compare the performance of the data subset selection algorithms. Then, a description of the metrics used to evaluate this performance is given. It then continues with discussing how subset selection-based training differs from full training and what effect the selection frequency and fraction size have on this. Several possible optimizations are discussed such as the warm-up strategy, extended training, and combining data subset selection-based training with transfer learning. Furthermore, a new method, Supervised Contrastive Learning, is presented. The differences and possible shortcomings of the Supervised Contrastive Learning algorithm and that of Contrastive Active Learning (CAL) are discussed. An optimization technique for CAL is presented. Finally, the chapter ends with an overview of the experimental setup. All code relevant to this thesis can be found in this GitHub repository (https://github.com/Daankrol/Dataset-Reduction-IL).

4.1 Architectures and training parameters

All methods and models are implemented using the PyTorch Machine Learning framework [56]. The models in this research are optimized using the SGD optimizer with a learning rate of 0.01, a momentum of 0.9 and a weight decay of 5e-4. For the CIFAR10 dataset, we use the ResNet-18 model [57] with a custom input layer to accept the input resolution of the CIFAR10 dataset. Since some of the methods listed above also used this model, we can replicate and validate their results [39, 38, 41].

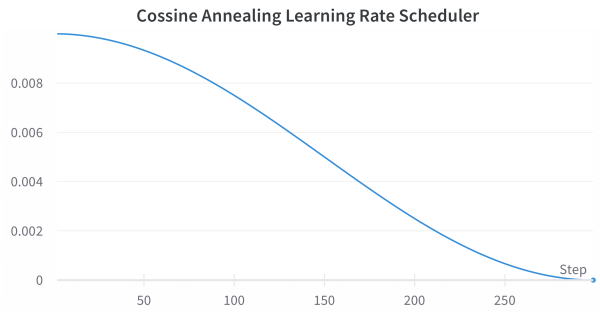

Figure 8: Learning rate graph when using Cosine Annealing over 300 epochs

Standard convolutional neural networks are often simply arbitrarily scaled in depth, width, or input resolution to obtain better generalization power. This can be a tedious process, where the end result can still be a suboptimal solution. The authors of [58] studied the effect of three scaling parameters on the depth, width, and input resolution. They presented a novel model-scaling method that is very efficient in scaling and can produce models that outperform the current state-of-the-art. Since the EfficientNet family models are small and perform well, they are perfect in a research that is focused on energy efficient training. Thus, for the Papilionidae and the CUB200 dataset, the EfficientNet-B0 model is used. This is the smallest of the EfficientNet family of models in terms of the number of parameters and it is more than twice as small as the ResNet-18 model. There are eight different models in the EfficientNet family. Starting from the smallest EfficientNet-B0, with 11M trainable parameters, to EfficientNet-B7, with 66M trainable parameters.

A batch size of 32 is used for each dataset. We train each model for 300 epochs unless otherwise specified. To avoid overfitting and to reduce model error, the cosine annealing learning rate scheduler is used [59]. An example trace of the learning rate over time can be observed in Figure 8. The T_{MAX} is set to the total number of epochs such that we do not have restarts. For all experiments, the average over five runs is reported with their standard deviation. All experiments were executed using one NVIDIA GeForce RTX 3090 GPU and 6 cores of one Intel Xeon Gold 6330 CPU with a total of 12 threads. To avoid the data loading bottleneck, we optimize the number of subprocesses used by the

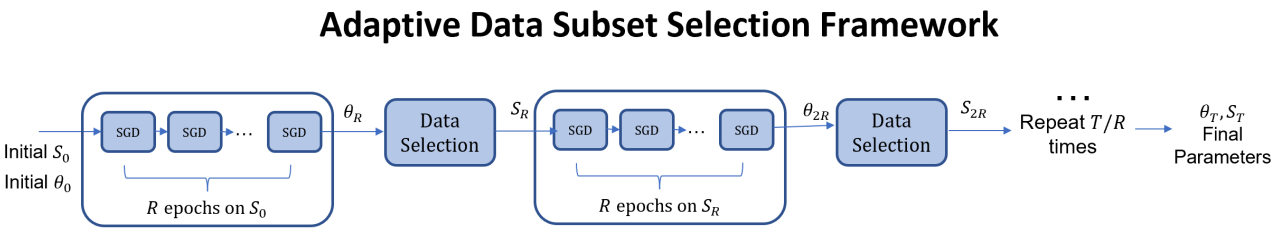

Figure 9: Overview of training with adaptive subset selection. Selection is performed every R epochs and the model parameters are updated by Stochastic Gradient Descent (SGD) on the selected subset. Source: [39]

data loader. The `num_workers` parameter of the PyTorch data loader instance is set to 8 in all cases.

The goal of this thesis is not to optimize deep learning models to achieve the best possible inference performance. Instead, we try to find the optimal selection techniques. These hyperparameters are thus not optimized but are kept fixed to allow for relative comparisons between selection methods and to save time and energy.

4.2 Adaptive and non-adaptive subset selection

The selection methods described above can be applied in two different ways; adaptively and non-adaptively. With adaptive data subset selection (DSS), the query method is applied every R epochs. The idea is that the subset of data is refined as the model parameters are updated. An overview of the adaptive subset selection process can be found in Figure 9. This differs from full normal training, in which a model is trained on all training data for T epochs. The update step for an adaptive DSS framework is defined in Equation 13. The selection technique produces both a subset $\mathcal{X}^t$ and weights w_t. Therefore, the parameters are updated with a weighted loss per sample. Note that not all methods compute loss weights. In these cases, the weights are set to 1.

With non-adaptive subset selection, the selection process is only performed once, before training. This could thus be seen as a dataset pruning mechanism or as offline dataset reduction. This means that in Equation 13 the same subset will be used for all $t \in \mathcal{T}$ thus $\mathcal{X}^t = \mathcal{X}$. For some non-adaptive methods, this can be done model-agnostic by using only the properties of the dataset itself. Adaptive DSS methods can also be applied non-adaptively. However, adaptive DSS methods are applied in conjunction with model training. Since without training the model parameters are random variables, the model inference power could be too low to select a subset that correctly represents the dataset. For example, with Contrastive Active Learning (CAL), the contrastive scores of the samples are based on the divergence of the predictive likelihoods. If used in combination with an untrained network, this would result in contrastive scores of zero for all samples. This behavior can also be observed when Least Confidence selection is used non-adaptively. The output of the model will be the same for all samples, which does not give any information about which samples the model is uncertain about.

$$\theta_{t+1} = \theta_t - \alpha \sum_{i \in \mathcal{X}^t} w_i^t \nabla_\theta L_T(x_i, y_i, \theta_t) \tag{13}$$

where:

θ_t = Model parameters in iteration t
α = Learning rate
w_i^t = The weight value for sample i at iteration t
∇_θ = Gradient with respect to the model parameters θ
L_T = Loss function

4.3 Metrics

Dataset reduction methods can be compared in several ways. Their performance is often compared by the accuracy of the model and the fraction of data used. There are several problems with these metrics. While this can be a viable measure for non-adaptive selection, this does not work for adaptive selection. In non-adaptive selection the selected subset does not change, but with adaptive selection the subset is continuously changing. Another problem is that this does not take into account the sampling duration of the method, and thus we cannot compare the energy efficiency and practicality of the method. A good adaptive selection technique is one that results in comparable or better performance in less time and energy than when training on all data. As it needs to sample every R epochs, the sampling duration should be low to reduce the total time taken.

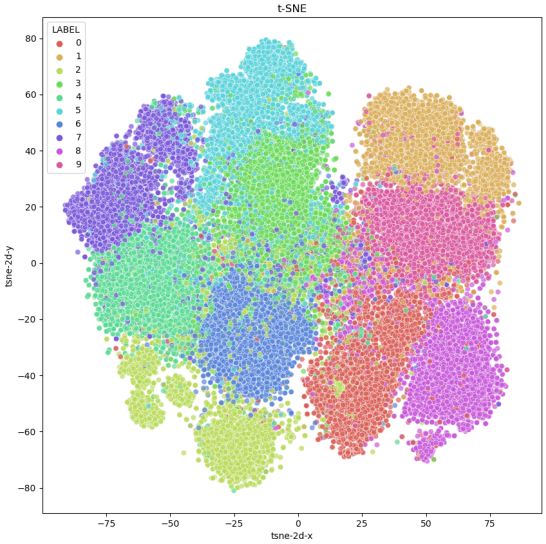

Figure 10: A t-SNE plot of the CIFAR10 dataset

Performance is often measured in model accuracy, even in situations where the dataset is made imbalanced by removing samples from classes. For example, in [39] the authors report the performance of several DSS methods using imbalanced data. They create an imbalance by removing 90% of the samples of 30% of the classes in a balanced dataset. However, they only do this on the train split. The validation and test set are kept balanced. Performance is reported by the accuracy on the test set. However, their approach is questionable, as we are now dealing with a dataset that is composed of splits that have different data distributions. The GRAD-MATCH, GLISTER and CRAIG methods try to find a subset that closely resembles the validation set. Thus, when the test set is balanced but the train set is imbalanced, they need to keep the validation set balanced to get good accuracy on the test set. This is not a correct representation of a real-world imbalanced dataset, and metrics that are often used with imbalanced datasets are missing.

Accuracy is not a good metric when using imbalanced data. As an example, let us say we have a model that is trained to detect if a person has disease X. This model is trained with a dataset that contains data on 100 people, 90 people who are healthy, and 10 people who have the disease. The model is trained and is reported to have an accuracy of 90% to detect if a person has disease X or not. But in reality, the model performs very poorly and always predicts that a person does not have disease X. Although the accuracy of the model is high, the model cannot discriminate between the two classes, and thus it has low predictive power. This phenomenon is called *Accuracy Paradox* [60].

The model simply always predicts that a person does not have disease X as it favors the dominant class.
For a better view of the performance of the multiclass classification model on imbalanced data, the metric *precision* and *recall* should be used. Precision quantifies the number of correct positive predictions made. For the disease example, precision thus focuses on the accuracy on the minority class since this class is the positive class, i.e. having disease X. See Equation 14 for the definition of precision. Precision can already be more useful than accuracy in some situations, but it does not tell the full story. It does not give information on the number of false negatives. Here, *recall* comes in. It quantifies the number of true positives out of all true positive predictions that could have been made. Its definition can be observed in Equation 15.
Individually, the metrics do not tell the entire story of the model performance. We want a metric that covers both the dominant and minority class. This is why often the F1 score is used as a metric for imbalanced multiclass classification problems. It is the harmonic mean of precision and recall and its definition is presented in Equation 16. Note that we use macro averaging for the precision, recall and F1 scores because we are dealing with multi-class classification. Macro averaging is performed by first calculating the metric for each class separately, and average the metrics across classes.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{14}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{15}$$

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 * \text{True Positives}}{2 * \text{True Positive} + \text{False Positive} + \text{False Negative}} \tag{16}$$

In addition to the final performance of the model, we also need to know how much time and how much energy it took to reach this performance. GPU energy consumption is tracked using the *pyJoules*[1] Python package. The time is used as a proxy to measure the energy consumption of the CPU.
To measure how good selection techniques are at sampling informative samples, we can measure the performance per computational load. The computational load can be described as the number of individual samples on which the model has been trained through backpropagation. To find out which samples are being selected by a technique and why it samples those, we can visualize the dataset manifold by using the t-Distributed Stochastic Neighbor Embedding (t-SNE) [61]. This is a technique for dimensionality reduction that is very useful for visualizing high-dimensional data. An example of a t-SNE visualization can be found in Figure 10. t-SNE is applied on the features of all samples in the dataset which are extracted using the EfficientNet network that is pre-trained on ImageNet. The data for t-SNE is first transformed to two-dimensional data using Principal Component Analysis (PCA) to speed up the slow t-SNE computation. Normally, model performance can be compared after each epoch of training. However, using dataset reduction techniques means that the amount of data on which the model is trained in an epoch depends on the fraction size of the selection technique. Therefore, the performance per epoch cannot be compared between experiments, and thus we track the performance per elapsed time and consumed energy.

4.4 Selection frequency and fraction size

With non-adaptive DSS, a fraction of the full dataset is selected as subset before training. This happens only once, and thus, the samples that are not part of the subset will never be seen by the model.

[1] https://pypi.org/project/pyJoules/

This assumes that there is some form of redundancy in the dataset. When there are multiple near-duplicates, these could be removed to reduce redundancy. This would then increase the average informativeness per sample across the entire subset and could effectively decrease training time and energy consumption. The general problem with non-adaptive DSS is that we do not know if the assumption holds for a dataset beforehand. In addition to that, if the assumption holds, we do not know which fraction of the data can be removed. To be able to compare the non-adaptive DSS methods, we follow the method of [22] and use a fixed set of fractions for comparison: 50%, 70% and 90%.
With adaptive DSS a fraction of the dataset is selected as subset every R epochs. These subsets can have overlap, which could potentially solve the general problem in non-adaptive selection where the fraction of to be removed data is unknown. Even if there is no redundancy in the dataset, the adaptive selection method could theoretically query all samples so that no sample would be disregarded. The goal of the methods is to be data efficient and save time per epoch. This can only be done by using fewer data than the full dataset in each epoch. But how small should this subset be? And for how many epochs should the model be trained on this subset, e.g., what should the value for R be? Selecting a small set infrequently can result in training too long on the same images, and thus not being able to select all informative samples before we reach the final training epoch. Selecting a large subset very often can result in the model not having enough parameter updates to correctly learn the underlying data properties. Selecting larger subsets generally increases the total training time as each training epoch takes longer. Smaller subsets give higher speed-up and energy gains but also have a larger accuracy drop than when using larger subsets. The fraction size and frequency can thus be fine-tuned depending on the goal of the user: high performance, small training duration, or a balance between both. Note that, with some methods, increasing the subset size also increases the sampling time. For example, with methods that use submodular optimization functions where samples are added iteratively in a greedy process. Other methods, such as CAL, Super-CL, or Uncertainty sampling, need to first compute scores for all samples before they can select the subset. With these methods, the sampling time remains the same with different subset sizes. To keep a fair comparison, we use the same fraction and frequency for all methods. For the frequency we set $R = 10$, and as a fraction, we use a default of 20% except when otherwise specified. Each selection method could be optimized per dataset and depending on the goal of the user. Due to time constraints, this was not done for all methods. The choice of fraction size and sampling frequency is based on a balance between performance and speed in all selection methods and datasets.

4.5 Warm-up strategy

As mentioned above, adaptive methods are based on the assumption that while the model is being refined, the subset is refined accordingly. Thus, subsets in early stages of the training cycle can be less informative than subsets in later stages, as the model does not yet have enough inference power to select a subset that is a good representation of the dataset. Following the results of [39] we find that training the model on the complete dataset for T_f epochs can result in good *warm-start* models and better convergence. This could allow the DSS method to select informative subsets early on. The definition of warm-up epochs can be observed in Equation 17. Throughout this thesis, methods that use a warm-up strategy will be denoted with the suffix *'-warm'*, e.g. GRAD-MATCH with warm-up is called GRAD-MATCH-WARM.

$$\begin{aligned} T &= T_s + T_f \\ T_f &= \kappa T \end{aligned} \tag{17}$$

where:

T = total number of training epochs
T_s = number of epochs with subset selection based training
T_f = number of epochs with full training
κ = warm-up epochs as a fraction of total epochs

For example, an experiment with 120 epochs and $\kappa = 0.3$ will first have 40 full training epochs. After that, the first subset will be sampled and data subset selection-based training starts. From this epoch on, every R epochs a new subset will be sampled. Note that using a value of κ of 1 will result in normal full training without data and energy-efficient training. Thus, the value of κ also has a direct influence on the total running time. The increase in κ results in an increase in the number of longer epochs. Because of time constraints no ablation study for each dataset could be performed. The value for κ is based on the ablation study in [39] and is set to 0.5.

4.6 Extended Training

With adaptive DSS, an epoch will take less time when the subset size is smaller than the data set, e.g. $|S| < |D|$. This should thus decrease the total run time. Following the results of previous research, the performance of models trained with adaptive DSS is often lower than or comparable to full training [39, 38, 41]. As mentioned previously, the hyperparameters such as fraction size and sampling frequency can be optimized depending on the goal of the user; high performance, short training time, or a balance of both. With adaptive DSS methods, models are trained on less data per epoch and thus training on the same number of epochs as FULL training can be quicker, assuming that the sampling duration is small. At the cost of some efficiency, we can do extended training to see if it might be possible to maximize performance and possibly get even better performance than with full training while still being faster. Several experiments will be performed in which the total number of epochs is increased from 300 to 350 unless otherwise specified. We set the T_{MAX} value of the cosine annealing learning rate scheduler to the same number of epochs. If T_{MAX} is not changed accordingly, the extended training would use a learning rate of zero. To limit the number of experiments, these experiments will be performed only on a selection of the best methods.

4.7 Transfer Learning

As described in subsection 2.4, transfer learning in general decreases the amount of energy and time needed to achieve good convergence. Furthermore, it can drastically increase the inference performance. To see if we can further reduce energy consumption, we study the effect of combining transfer learning with DSS and their application on biodiversity datasets. To the best of my knowledge, there has been no research on the application of DSS methods with transfer learning. The EfficientNet-B0 model is used, which is pre-trained on the ImageNet dataset. The total number of training epochs is reduced to 150 as the model can converge much faster when using transfer learning. We study the effect of using transfer learning with DSS methods on the Papilionidae and CUB200 datasets. The CIFAR10 dataset overlaps with the ImageNet dataset. This results in a very short training cycle, as the model already has very good inference power on this dataset. This does not allow for good comparisons between the DSS methods, and thus this dataset is not used. The size of the model output layer is changed based on the number of classes in each dataset. All layers are unfrozen and fine-tuned while training. The learning rate is lowered to 0.005 as we are not training from scratch, and

the cosine learning rate scheduler is used to regularize training. All images are normalized based on the mean and variance of the image color channels of the ImageNet dataset.

4.8 Supervised Contrastive Learning

In this subsection, we present *Supervised Contrastive Learning (Super-CL)*. *Super-CL* is an adaptive data subset selection method based on the algorithms and properties of *Contrastive Active Learning (CAL)* and the *Prototypical* DSS methods.

In *Prototypical* DSS, samples are selected on the basis of their distance from the class prototype. The general idea is that samples with the largest distance are hypothesized to be close to the decision boundary or form sub clusters. Note that outliers have a large distance to the prototype but do not necessarily be close to the decision boundary. Samples with large distance to the prototype may be more difficult to classify than samples that are very similar to the prototype. Therefore, it is hypothesized that the samples are more informative. Note that samples with large distance to the prototype can also be outliers. This method is best used in a non-adaptive way, as it is not model dependent and uses a pre-trained network to construct the feature space. Note that it can be made model-dependent by using the training model as a feature extractor. However, when the training model has not been trained or pre-trained it might not be able to correctly represent the feature space of the dataset.

CAL is used in Active Learning and is thus used to sample unlabeled samples to be labeled by an expert. The sampling is based on the *contrastive* score of a sample. Samples that are similar in feature space but have maximally different predictive likelihoods have high contrastive scores. In other words, data points with similar feature representations and dissimilar model outputs are contrastive. They probably lie close to the decision boundary and could therefore increase the inference performance of the model. A pre-trained network is used as a feature extractor. Probabilities are computed by the training model. As the training model continues to change, contrastive scores are a reflection of the currently most informative samples.

A disadvantage of Prototypical DSS is that it is focused only on the feature space and not on the model uncertainty. CAL does make use of this model uncertainty, but it does not use any label information since it is used in an unsupervised matter. Since it does not use label information, it also cannot sample per class. Per-class sampling can counteract the effects of an imbalanced dataset. Another disadvantage is that CAL needs a lot of memory since it has to store the entire feature space, all pairwise distances and the nearest neighbors for each sample in the entire dataset. In addition to that, sampling can take a long time because of the computation of the pairwise distance matrix over the whole dataset. CAL thus only works with small datasets or with small feature vectors and could not be used with the Papilionidae and CUB200 datasets due to out of memory errors.

We present *Supervised Contrastive Learning (Super-CL)*. Super-CL uses both model information and label information to find contrastive samples within a class. It does so by first computing the feature space of a single class. Then we calculate the pairwise distance within the class. The contrastive scores are then calculated for each sample within the class. As it samples per class, its memory footprint is much smaller, as we do not need to keep the feature vectors, distances, and nearest neighbors of all samples in memory. Because it uses label information, it is more aimed at the hardest samples within a class than at the hardest samples in the full dataset. With CAL there can be subsets that do not contain each class. It is hypothesized that this will give Super-CL an advantage when applied to biodiversity or long-tailed datasets, as it can do per-class sampling where the data distribution in the

Algorithm 4 A single iteration of the Supervised Contrastive Learning data subset selection algorithm.

Require: training set $\mathcal{U}$, number of samples to be selected (budget) b, number of neighbours k, model $\mathcal{M}$, representation (encoding) function $\Phi(.)$

$\mathcal{X} \leftarrow \emptyset$
for each class $c \in \mathcal{U}$ **do**
  $\mathcal{U}_c \leftarrow (x,y) \in \mathcal{U}$ where $y = c$ ▷ subset of $\mathcal{U}$ of class c
  $b_c \leftarrow b\frac{|\mathcal{U}_c|}{|\mathcal{U}|}$ ▷ compute class budget
  for each sample $x_p \in \mathcal{U}_c$ **do**
    $\left\{(x_l^{(i)}, y_l^{(i)})\right\}, i = 1,...,k \leftarrow KNN(\Phi(x_p), \Phi(\mathcal{U}_c), k)$ ▷ find neighbours in $\mathcal{U}_c$
    $p(y|x_l^{(i)}) \leftarrow \mathcal{M}(x_l^{(i)}), i = 1,...,k$ ▷ compute probabilities
    $p(y|x_p) \leftarrow \mathcal{M}(x_p)$
    $D_{KL}\left(p(y|x_l^{(i)})||p(y|x_p)\right), i = 1,...,k$ ▷ compute divergence
    $s_{x_p} = \frac{1}{k}\Sigma_{i=1}^{k} D_{KL}\left(p(y|x_l^{(i)})||p(y|x_p)\right)$ ▷ compute score
  end for
  $Q = \underset{x_p \in \mathcal{U}_c}{\operatorname{argmax}} s_{x_p}, |Q| = b_c$ ▷ selection
  $\mathcal{X} \leftarrow \mathcal{X} \cup Q$
end for
return $\mathcal{X}$

subset remains the same as in the full dataset.

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} \tag{18}$$

A single iteration of the Super-CL selection algorithm can be observed in Algorithm 4. The main difference between CAL and Super-CL is that Super-CL uses within-class nearest-neighbor and divergence estimation. Note that the nearest-neighbor calculation only has to be performed once. After the first iteration, the KNN result can be reused, and only the divergence calculation has to be performed. Super-CL uses the same divergence calculation as CAL. The definition of the divergence can be observed in Equation 18. Note that the pairwise distance calculation for the nearest-neighbor search still takes some time which heavily impacts the total sampling duration. There may be some edge cases in the Super-CL selection algorithm. For example, when a class has a limited number of samples such that the number of neighbors per sample is smaller than k, we then take all samples as neighbors. Furthermore, we make sure that for all classes at least one sample is present in the subset. If a sample has no neighbors, we set the divergence to a maximal value such that the sample is always picked. In addition to this, a division by zero error can occur in the divergence calculation. This is solved by replacing the zero values in the probability vectors with a small value of 10^{-6}.

It is hypothesized that Super-CL can work well with the warm-up technique. Super-CL picks hard and informative samples, and thus the subsets can miss a degree of diversity as there can be overlap in subsets. This potential problem could be solved by injecting diversity before or after the Super-CL DSS is performed. With warm-up, the model is trained on the full dataset and thus is injected with diversity early on in training. After that, Super-CL can target the more complex samples to get better model convergence.

In this thesis, the effect of diversity injection is studied by looking at the effect of warm-up and post-training on Super-CL. Post-training can be seen as a novel inverted warm-up where the model is first trained using the DSS method and diversity is injected by training with the full dataset. When post-training is applied, the κ value determines the number of epochs of DSS training. Thus, Equation 17 changes to $T_s = \kappa T$.

4.9 Contrastive Active Learning optimizations

One of the reasons that Super-CL was devised, was to solve the out-of-memory errors of CAL. CAL needs to compute the pairwise distance matrix for the nearest neighbor calculation over the complete dataset. This becomes intractable for medium to large dataset sizes. To solve this, CAL can be integrated with Facebook AI Similarity Search (Faiss) or similar libraries [62]. Faiss is a library for efficient similarity search and clustering of dense vectors. Faiss is based on years of research. It implements several techniques that result in great speed optimizations such as computing on the GPU, lossy compression for high-dimensional vectors, and product quantization. Besides this, it also enables to search in sets of vectors that are so large that they would normally not fit in RAM memory. To avoid out-of-memory errors and to reduce the sampling duration, the kNN calculation in the CAL algorithm can be performed by Faiss. CAL can now be used on the CIFAR10, Papilionidae, and CUB200 datasets. In the results and discussion section, the term CAL is used to denote CAL with Faiss optimizations. Although Super-CL solves the out-of-memory issues of CAL by performing per-class sampling, this could still fail for datasets with very large class sizes. Super-CL could thus also be optimized with Faiss. However, because of time constraints, this was not performed.

4.10 Experimental setup

Several experiments will be carried out to test the performance of dataset reduction methods. The non-adaptive methods that will be evaluated are: GRAND, EL2N, PROTOTYPICAL, GRAPH-CUT and GRAD-MATCH. The fraction sizes that are used are 50%, 70% and 90%. They will be evaluated on the CIFAR10, Papilionidae, and CUB200 datasets. As a baseline, RANDOM and FULL training will be performed. With RANDOM non-adaptive DSS, a fraction of the data is selected as a subset with random uniform probability. For GRAND 10 models are used to average the GRAND scores. With EL2N the same number of models are used but are now trained for 20 epochs before calculating scores. This follows the implementation of [36].
The adaptive methods that will be evaluated are: ENTROPY, LEAST-CONFIDENCE, GLISTER, CRAIG, CRAIGPB, GRAD-MATCH, GRAD-MATCHPB, CAL, SUPER-CL, GRAPH-CUT and FACILITY-LOCATION. These methods will be compared with RANDOM, FULL and FULL-ES which is full training with an early stopping mechanism with patience of 20 and a minimum delta of the validation loss of zero. All methods will be evaluated on the CIFAR10, Papilionidae, and CUB200 datasets. A selection of the best methods will be made after comparing the performance on these datasets. These methods will be tested with the warm-up strategy and extended training. All methods will be tested in combination with transfer learning. A selection of the best methods will be tested in combination with a warm-up strategy, extended training, and transfer learning.
Note that when both a warm-up strategy and extended training are used we make sure that the amount of warm-up epochs stays equal. E.g. when RANDOM-WARM is trained for 300 epochs with $\kappa = 0.5$, there is a total of 150 warm-up epochs. When both extended training and warm-up are used, we have a total of 350 epoch of which 150 should be warmup. Thus the κ value is set to 0.42857.

Uncertainty-based methods ENTROPY and LEAST-CONFIDENCE both use per-class sampling so that the same fraction of each class is present in the subset. The submodular based methods, GRAPH-CUT and FACILITY-LOCATION, are applied on the last layer gradients following the implementation of [41, 38, 39]. For GLISTER the same learning rate that is used for the training model is also used for the one-step gradient update. CRAIG is applied per-class where CRAIGPB is applied per-batch. GRAD-MATCH uses per-class-per-gradient computation. This is the same as per-class sampling, but now the OMP algorithm is applied on the gradients corresponding to the classification layer of that class only. With GRAD-MATCHPB the OMP algorithm is applied on each mini-batch of datapoints. For both versions of GRAD-MATCH the regularization constant λ for the OMP solver is set to 0.5 and the parameter ε is set to 10^{-100} after the ablation study in [39]. CAL uses the optimization discussed in subsection 4.9. Both CAL and SUPER-CL use $k = 10$ for the nearest-neighbor calculation. The submodular methods FACILITY-LOCATION and GRAPH-CUT are applied to the full training dataset. Both methods are implemented using the Apricot Python package [63] and use the two-stage optimizer, first naive greedy and then lazy optimization, to solve the submodular function.

5 Results

5.1 Non-adaptive Data Subset Selection

The test performance of models trained on subsets selected by the non-adaptive DSS methods is shown in Table 2, Table 3 and Table 4.
In Table 2 the performance of the methods on the CIFAR10 dataset can be observed. FULL training is able to reach a performance of 91.8% for both F1 and accuracy. Most methods are able to prune the dataset and show similar performance. The GRAND method showed decreased performance compared to the other methods. With a fraction size of 50% the GRAND method loses 58.6 F1 score points compared to FULL. Other methods do not show this behavior and can prune 50% of the dataset, while losing only eight relative F1 score points at maximum in the case of GRAD-MATCH.
In Table 3 the results for the methods applied on the Papilionidae dataset can be found. Training on the full dataset results in an accuracy 92.8% and an F1 score of of 70.0%. We observe that EL2N is able to reduce the dataset to 50% of its size while maintaining good performance. Furthermore, RANDOM sampling results in comparable accuracy with the EL2N method but the F1 score is lower for all fraction sizes.

Table 2: The results for the non-adaptive methods on the CIFAR10 dataset averaged over five runs. Full training accuracy and F1 score are **91.1±0.8** and **91.1±0.8** respectively. For each fraction and metric, the best performing method is marked in bold. Note that for the 50% columns there is no best method as there is overlap in the standard error.

Fraction	50%		70%		90%	
	acc.	F1	acc.	F1	acc.	F1
RANDOM	86.7±0.3	86.7±0.3	89.7±0.1	89.7±0.1	91.5±0.1	91.5±0.1
PROTOTYPICAL	86.9±0.2	86.9±0.2	89.1±0.2	89.1±0.2	91.7±0.0	91.7±0.0
GRAD-MATCH	83.1±0.1	83.1±0.1	86.5±0.3	86.4±0.3	88.5±0.1	88.5±0.1
GRAPH-CUT	84.8±0.2	84.8±0.3	88.9±0.1	88.9±0.1	91.3±0.1	91.3±0.0
GRAND	46.8±0.7	32.5±0.8	66.0±0.5	56.1±0.2	83.2±0.1	79.1±0.1
EL2N	86.6±1.5	86.6±1.5	**91.1±0.1**	**91.2±0.1**	**92.1±0.3**	**92.1±0.2**

In Table 4 the results for the non-adaptive methods on the CUB200 dataset can be found. Training on the full dataset results in an average accuracy of 48.3% and an F1 score of 48.1%. The GRAD-MATCH method shows the best performance, in terms of accuracy and F1, for each fraction size. The performance of EL2N is lower for smaller subsets than for the other methods.
The energy it took to sample the smaller subset together with model training can be observed in Figure 11. The energy needed to prune the dataset can be observed by looking at the beginning of each graph. For example, pruning the Papilionidae dataset to 50% of its original size with EL2N takes around the same energy as pruning the same dataset with PROTOTYPICAL together with training a model on this data. The non-adaptive methods can be compared with FULL training where no pruning is performed. For example, RANDOM needs almost no energy to prune the dataset as its starting point is almost the same as that of FULL. Although it has lower accuracy than FULL, the energy required for training is much lower because the dataset is smaller.
In Figure 12 and Figure 13 the performance and energy consumption of the non-adaptive methods EL2N, GRAD-MATCH and RANDOM are shown for three different fraction sizes on two datasets.

Table 3: The results for the non-adaptive methods on the Papilionidae dataset averaged over five runs. Full training accuracy and F1 score are **92.8±0.1** and **70.0±1.1** respectively. For each fraction and metric, the best performing method is marked in bold. Note that for the 70% acc. column, there is no best method as there is overlap in the standard error.

Fraction	50%		70%		90%	
	acc.	F1	acc.	F1	acc.	F1
Random	91.0±0.1	55.9±0.3	92.2±0.1	62.1±1.7	92.1±0.1	65.2±0.8
Prototypical	90.5±0.1	53.5±1.3	91.2±0.1	55.5±0.9	92.0±0.1	55.5±0.7
Grad-Match	**91.5±0.1**	62.5±1.4	92.2±0.1	67.1±1.4	92.7±0.1	68.5±1.1
Graph-Cut	90.7±0.2	55.5±1.2	90.5±0.2	54.0±0.9	90.0±0.2	52.1±0.9
GraNd	56.4±2.2	29.5±5.0	71.3±2.0	40.7±2.3	87.1±1.1	56.5±2.3
EL2N	90.2±0.5	**67.2±2.2**	92.0±0.5	**69.1±1.5**	**92.9±0.1**	**70.4±0.9**

Table 4: Results for the non-adaptive methods on the CUB200 dataset averaged over five runs. Full training accuracy and F1 score are **48.3±0.8** and **48.1±0.7** respectively. For each fraction and metric, the best performing method is marked in bold.

Fraction	50%		70%		90%	
	acc.	F1	acc.	F1	acc.	F1
Random	29.2±0.2	28.7±0.2	38.2±0.4	37.8±0.5	45.7±0.5	45.3±0.6
Prototypical	20.9±0.4	20.2±0.3	32.9±0.5	32.2±0.6	43.6±0.3	43.5±0.3
Grad-Match	**29.8±0.8**	**29.5±0.7**	**38.6±0.4**	**38.3±0.5**	**47.2±0.3**	**46.9±0.3**
Graph-Cut	26.6±0.5	26.2±0.5	36.7±0.7	36.5±0.8	44.4±0.7	44.0±0.7
GraNd	25.6±0.6	17.7±0.4	34.4±0.4	28.8±0.5	43.9±0.7	41.8±0.7
EL2N	15.0±0.6	13.3±0.5	29.0±0.6	27.5±0.7	44.4±0.7	44.4±0.7

We observe that varying the fraction size does not have a large effect for the methods applied on the Papilionidae dataset except for the RANDOM method. In addition to this, we observe that EL2N with a fraction size of 70% has a higher final F1 score than FULL training. In Figure 13 we can observe that for the CUB200 dataset, performance is heavily influenced by the fraction size. In addition to this, RANDOM and GRAD-MATCH seem to behave on par. EL2N has a lower performance for all fraction sizes. FULL out performs all methods independent of the fraction size.

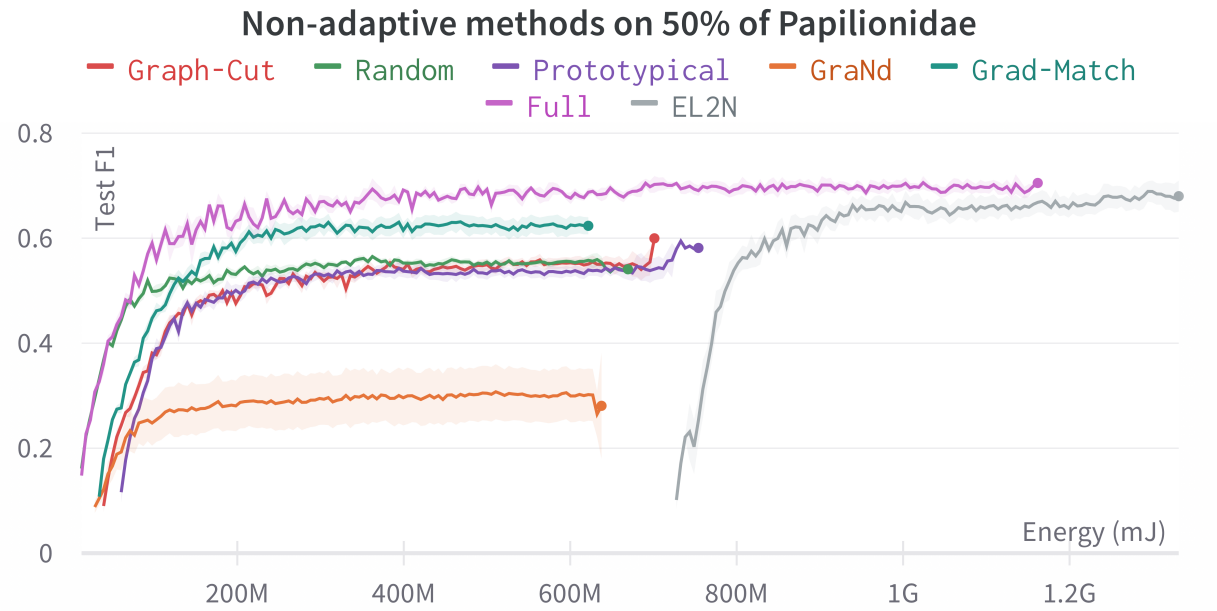

Figure 11: Test F1 score vs energy consumption for models trained on subsets provided by the non-adaptive methods. Energy needed for the non-adaptive data subset selection is included. Each method is averaged over five runs and the standard error is shown as error band. Full is a model that is trained on all data.

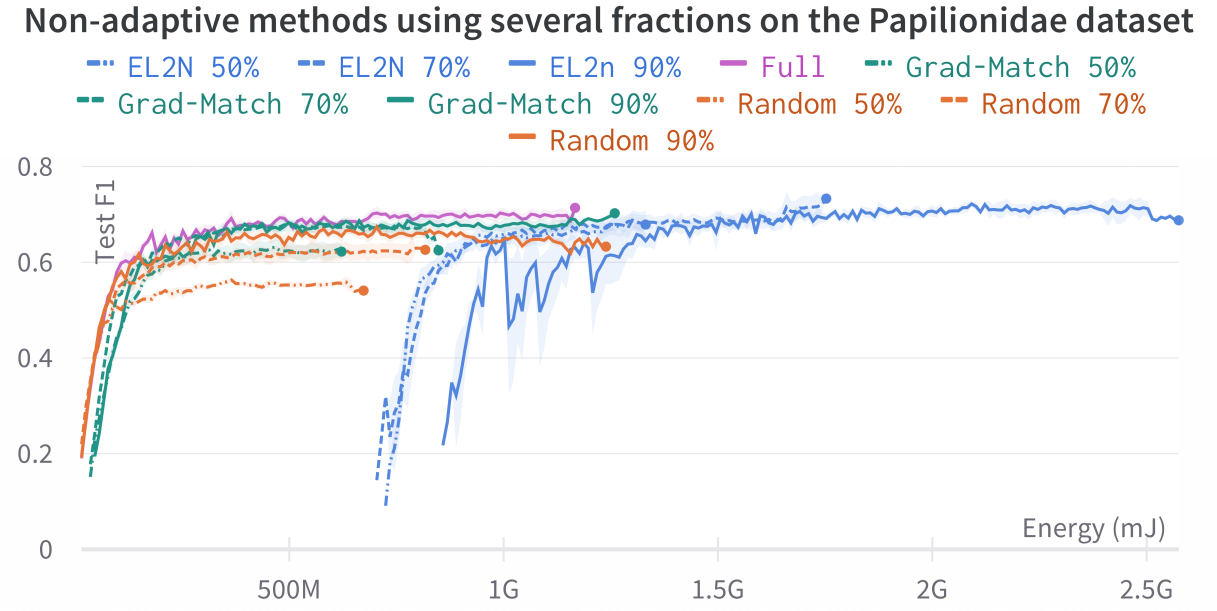

Figure 12: Test F1 vs energy consumption for models trained on subsets of the Papilionidae dataset provided by RANDOM, GRAD-MATCH and EL2N. FULL represents a model trained on all data. The effect of varying the fraction value can be observed. Note that the energy consumption of the sampling method is included and causes some methods to start shifted to the right.

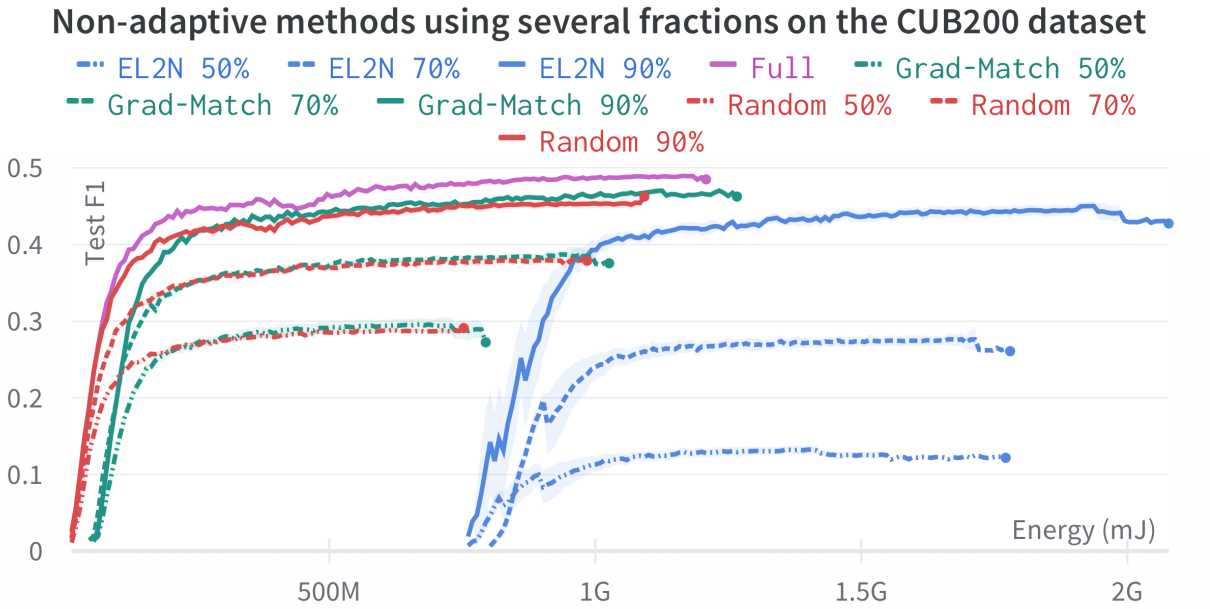

Figure 13: Test F1 vs energy consumption for models trained on subsets of the CUB200 dataset provided by RANDOM, GRAD-MATCH and EL2N. FULL represents a model trained on all data. The effect of varying the fraction value can be observed. Note that the energy consumption of the sampling method is included.

5.2 Adaptive Data Subset Selection (ADSS)

The results of the adaptive DSS methods for the three datasets can be observed in Figure 14, Figure 15. To show the performance of the methods and their optimizations in comparison to FULL training, we calculated the relative speed and relative error. These calculations are performed using the final performance reported and the total duration. The relative speed is calculated by dividing the total duration of FULL by that of each ADSS method. Note that because of this, the relative speed can never be lower than 0. For example, a relative speed of 2 means that the ADSS method is twice as fast as FULL training and a relative speed of $\frac{1}{2}$ means that the DSS method is twice as slow. The relative error is computed by the difference in accuracy or F1 score. Thus, a relative error of 2% means that the ADSS method performs 2 percentage points lower than the training of FULL.

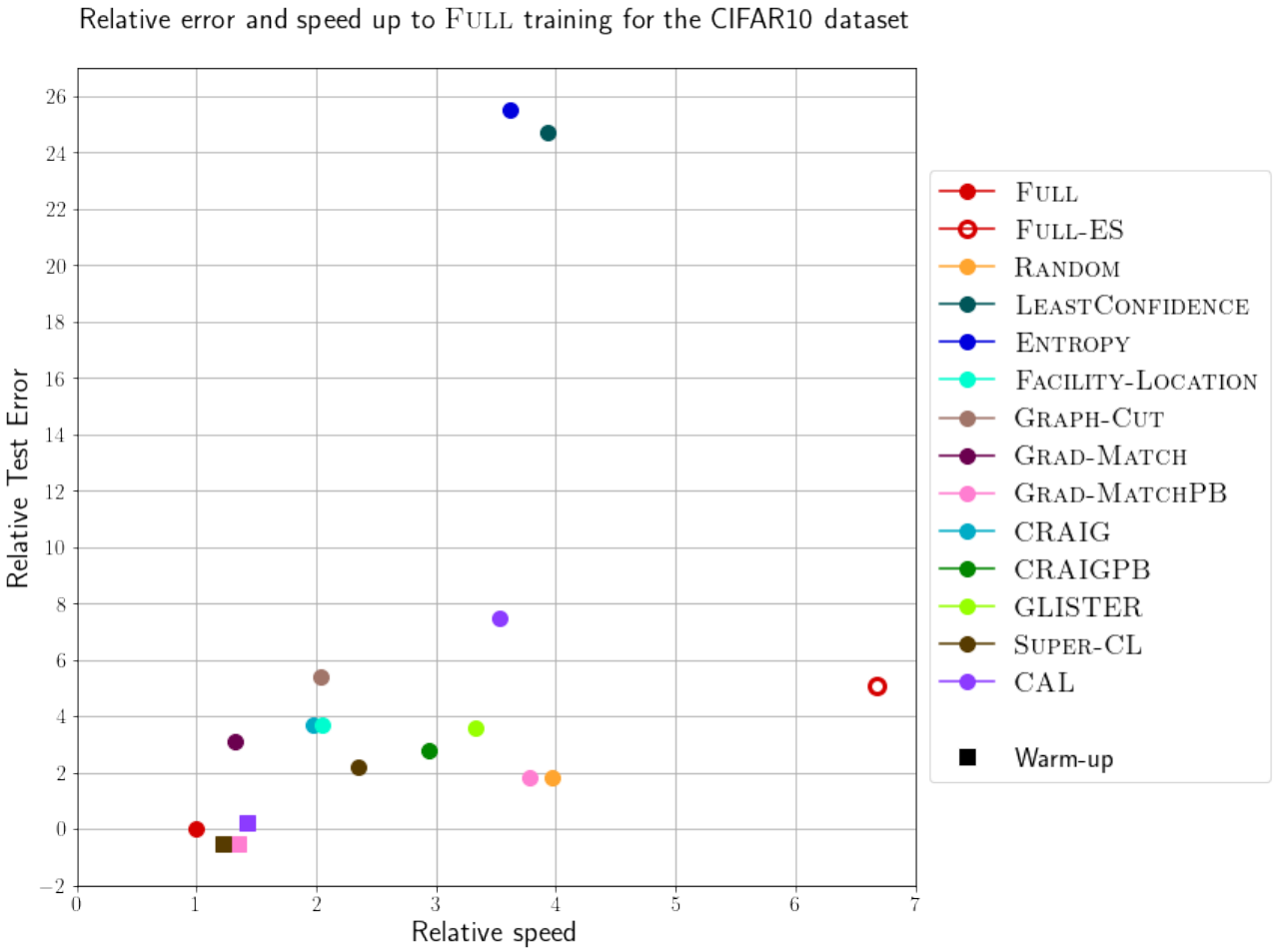

Figure 14: This plot shows the speed and error of several DSS methods relative to using FULL training for the CIFAR10 dataset. Relative speed is computed by dividing the total run time of FULL by that of each DSS method. Relative error is the difference in accuracy of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an accuracy of **91.1±0.8** and takes 3.35±0.37 hours.

The relative performance and speed of all methods on the CIFAR10 dataset can be observed in Figure 14. Methods such as GRAD-MATCHPB, SUPER-CL and RANDOM can greatly speed up training while only losing around 2 percentage points of performance. ENTROPY and LEASTCONFIDENCE showed the worst performance. Using warm-up decreases the speedup, but does improve the performance. Early stopping results in the largest speedup, but also in dropping more than 5 performance percentage points. The development of performance on the test set while training the networks can be observed in Figure 17. Here, we see that all methods that use warm-up and the FULL method show the same performance until the warm-up epochs are completed. CAL-WARM then quickly increase performance while SUPER-CL-WARM and GRAD-MATCHPB-WARM reach this performance only later on. Furthermore, we observe that the ADSS methods without warm-up start off with a lower performance than FULL but get good performance early on.

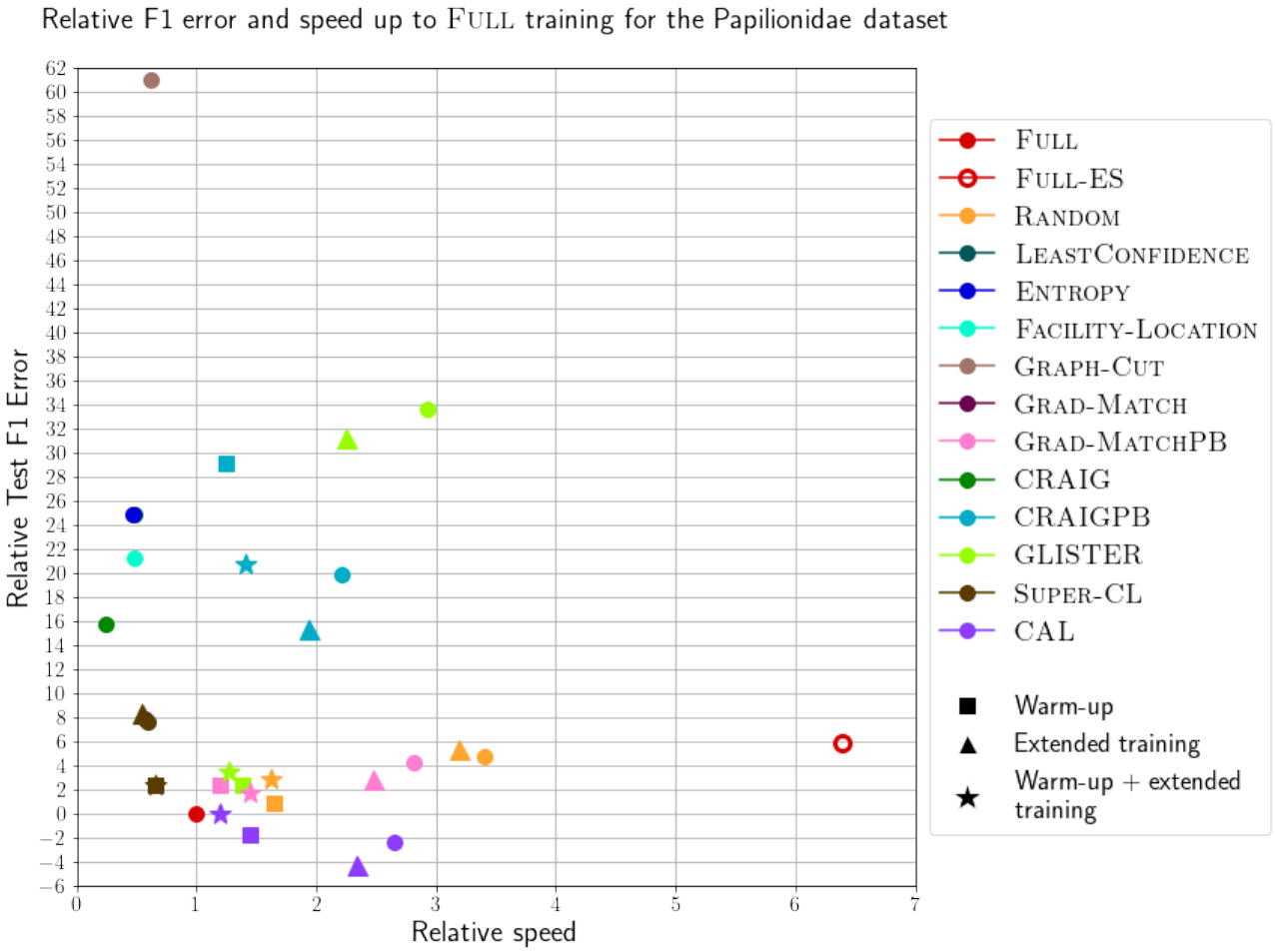

Figure 15: This plot shows the speed and error of several DSS methods relative to using FULL training for the Papilionidae dataset. Relative speed is computed by dividing the total run time of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1 score of **70±1.1** and takes 1.28±0.02 hours.

In Figure 15 the performance on the Papilionidae dataset of the adaptive DSS methods relative to that of FULL training can be observed. The RANDOM and GRAD-MATCHPB methods are both capable of speeding up training while losing limited performance. The CAL method not only can speed up training, but also showed a higher performance than FULL training. Using warm-up decreased both the speedup and the performance gain compared to normal CAL. Furthermore, the use of extended training further increases speedup and performance gain. CAL with extended training results in a performance increase of around five F1 score points while being more than twice as fast as FULL training. In Figure 18 the performance on the test set can be observed while training the networks. We see that warm-up does not have a large effect on the method performance. CAL even performs much better when not using the warm-up optimization. The energy needed to sample the first subset is much larger with SUPER-CL than for the other methods. This can be observed by looking at the starting point of each method that does not use warm-up. When comparing CAL-WARM and SUPER-CL-WARM we see that the former is faster than FULL and reaches higher accuracy. For SUPER-CL-WARM we see that it requires more time than FULL and achieves lower performance.

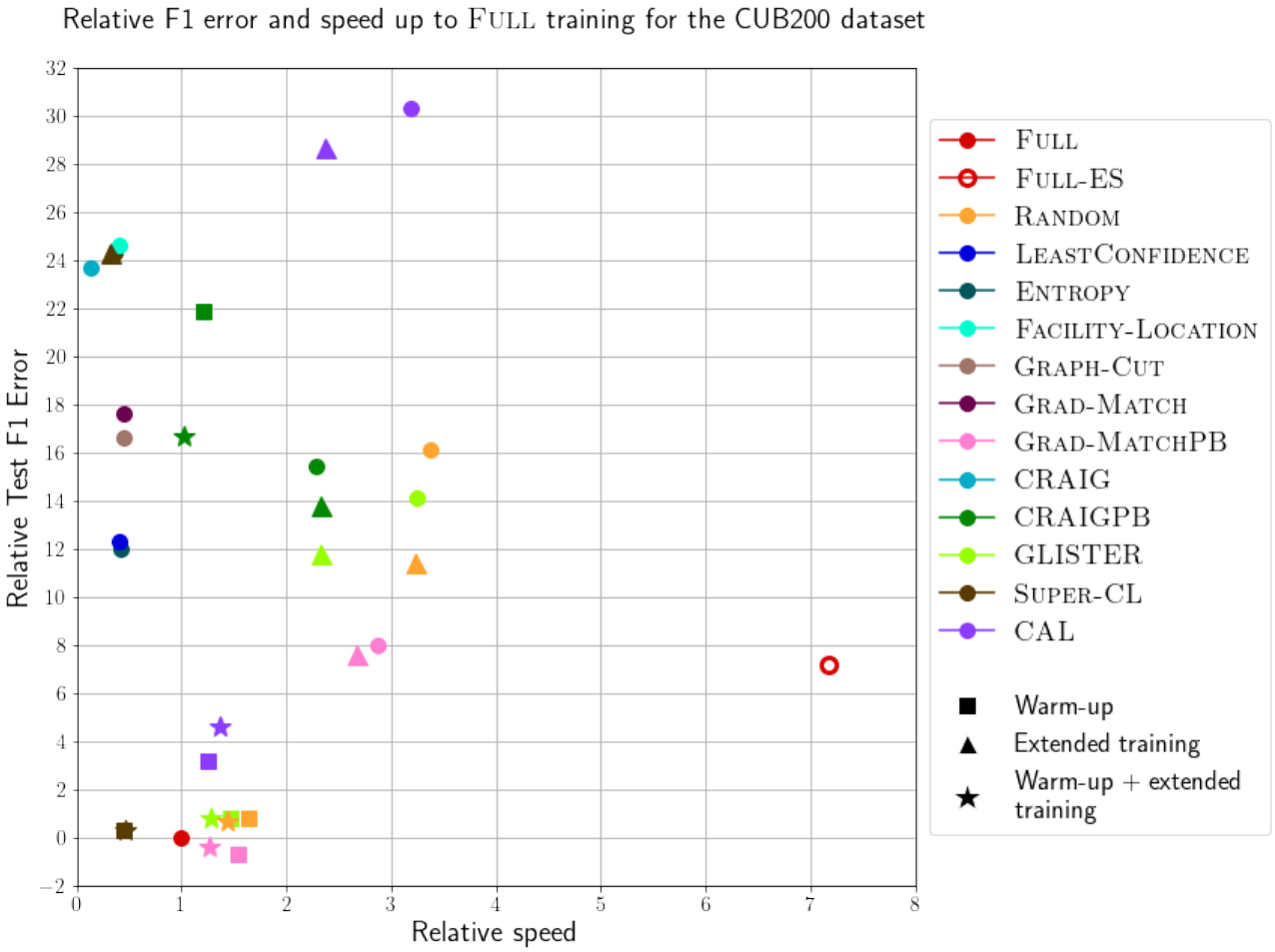

Figure 16: This plot shows the speed and error of several DSS methods relative to using FULL training for the CUB200 dataset. Relative speed is computed by dividing the total run time of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1 score of **48.8±0.2** and takes 1.34±0.01 hours.

The performance of adaptive DSS methods on the CUB200 dataset can be observed in Figure 16. Without optimizations, none of the methods show significant speedups without losing too much performance. Especially when considering that FULL only reaches a performance of 48.8 F1 score. GLISTER, GRAD-MATCHPB and RANDOM can speed up training when warm-up is used or when using both warm-up and extended training. GRAD-MATCHPB using these optimizations can even increase performance and speed. The CAL method performs very poorly. However, the effect of warm-up on the test performance of the DSS methods is the largest with this method with a difference of around 27 F1 score points. SUPER-CL with and without extended training performs poorly. However, using warm-up results in comparable performance with FULL training but takes around twice as long. In Figure 19 we can see the development of the test performance for a selection of methods with or without warm-up. GRAD-MATCHPB-WARM shows the best performance per energy consumed. CRAIGPB-WARM has a large performance drop after the warm-up epochs have been completed. The energy consumption of the SUPER-CL and GRAD-MATCH methods is much higher than that of FULL.

To see how quickly each method can sample a subset, we provide a visualization of the sampling

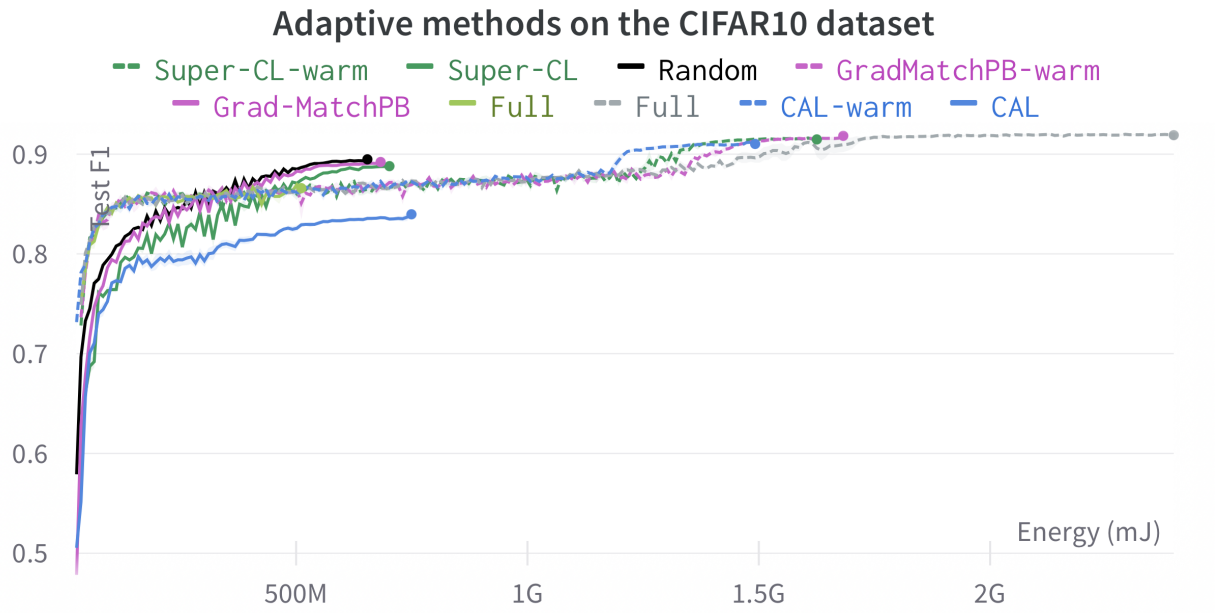

Figure 17: Test performance on the CIFAR10 dataset while training with a selection of the best adaptive DSS methods with and without the warm-up optimization. All methods are averaged over 5 runs and the standard error is shown as error band.

duration for the adaptive DSS methods. In Figure 20 we present the sampling duration for the Papil-

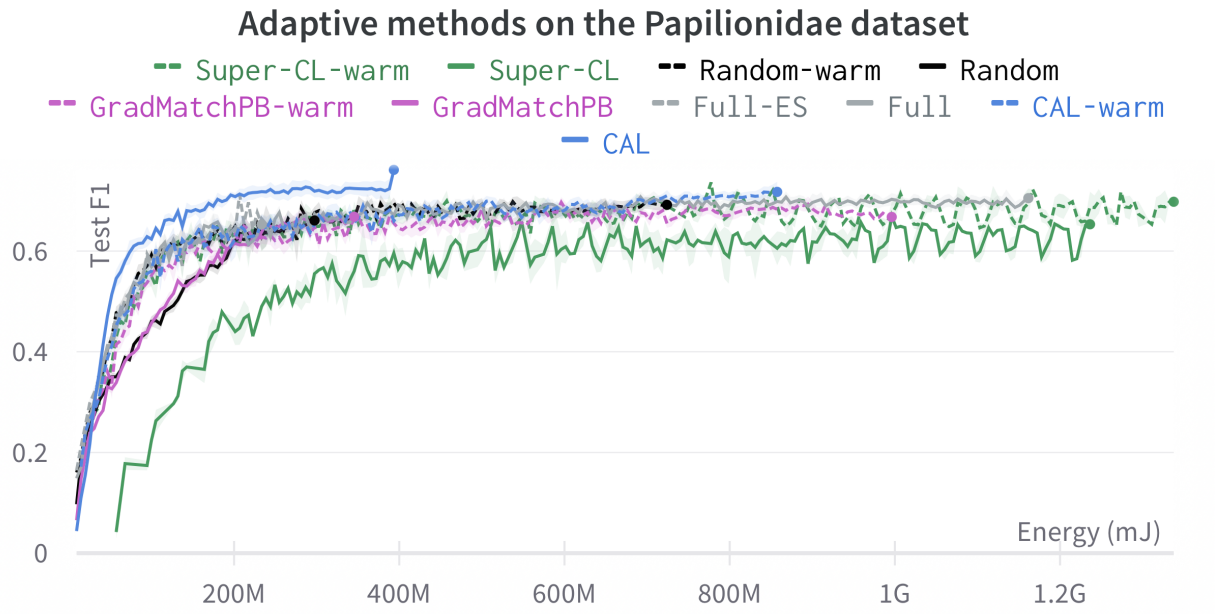

Figure 18: Test performance while training with a selection of the best adaptive DSS methods with and without the warm-up optimization. All methods are averaged over 5 runs and the standard error is shown as error band.

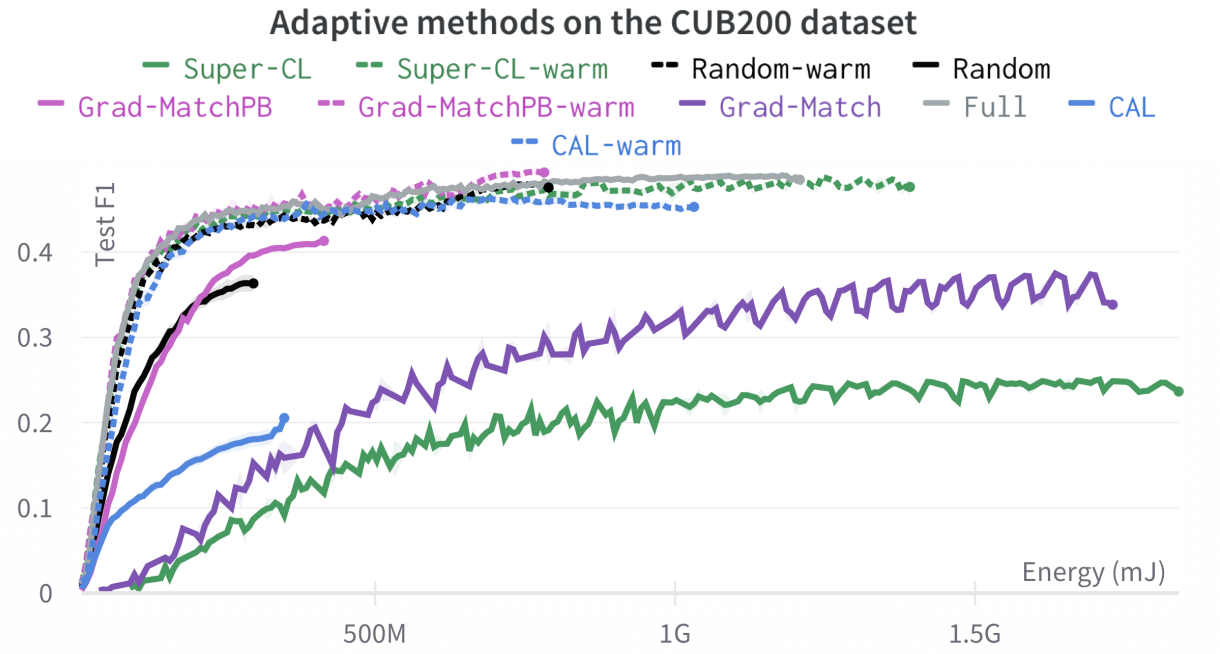

Figure 19: Test performance while training with a selection of the best adaptive DSS methods with and without the warm-up optimization. All methods are averaged over 5 runs and the standard error is shown as error band.

ionidae dataset and in Figure 21 for the CUB200 dataset. Notice that for the SUPER-CL method, the sampling time is larger at the first sampling, since it only needs to compute the nearest neighbor calculation once. In general, we observed that methods that perform per-batch selection are much faster in sampling than methods that do per-class selection. Furthermore, sampling duration for methods that do not use per-batch sampling is larger for the CUB200 dataset than for the Papilionidae dataset.

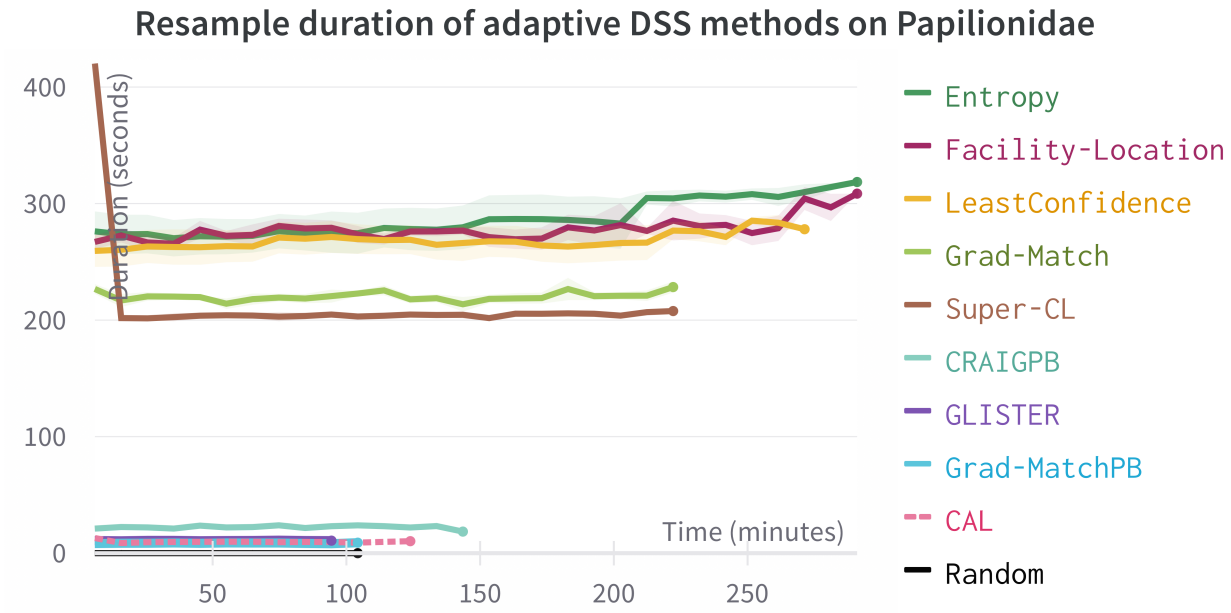

Figure 20: Resampling duration of adaptive DSS methods while training on the Papilionidae dataset.

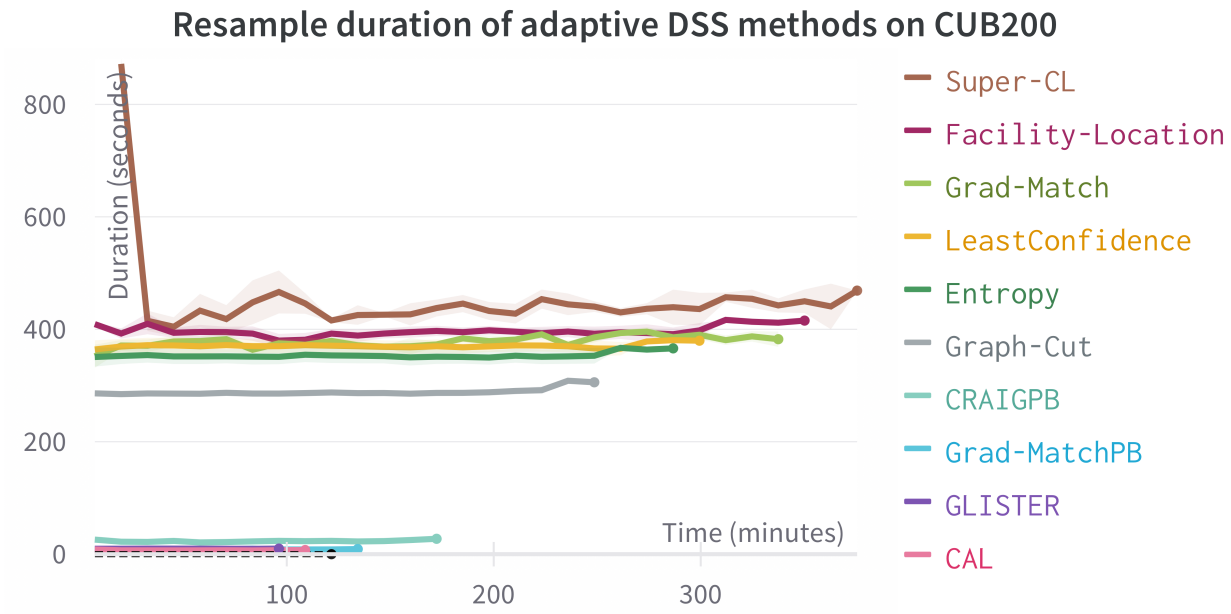

Figure 21: Resampling duration of adaptive DSS methods while training on the CUB200 dataset.

5.3 Dataset Reduction with Transfer Learning

In this subsection, we present the results for combining the adaptive DSS methods with transfer learning. All models are pre-trained on the ImageNet dataset and their final performance is compared against that of FULL training. In Figure 22 the results of the methods on the Papilionidae dataset can be found. We observe that FULL-ES, training on all data and using early stopping, results not only

in a speedup but also in increased performance. CAL is the only method not using warm-up that is able to speed up the training without losing significant performance. In all cases, warm-up is able to decrease the relative error while losing some of the speedup for each method. GLISTER-WARM, GRAD-MATCHPB-WARM, CAL-WARM and RANDOM-WARM are all able to speed up training without losing significant performance.

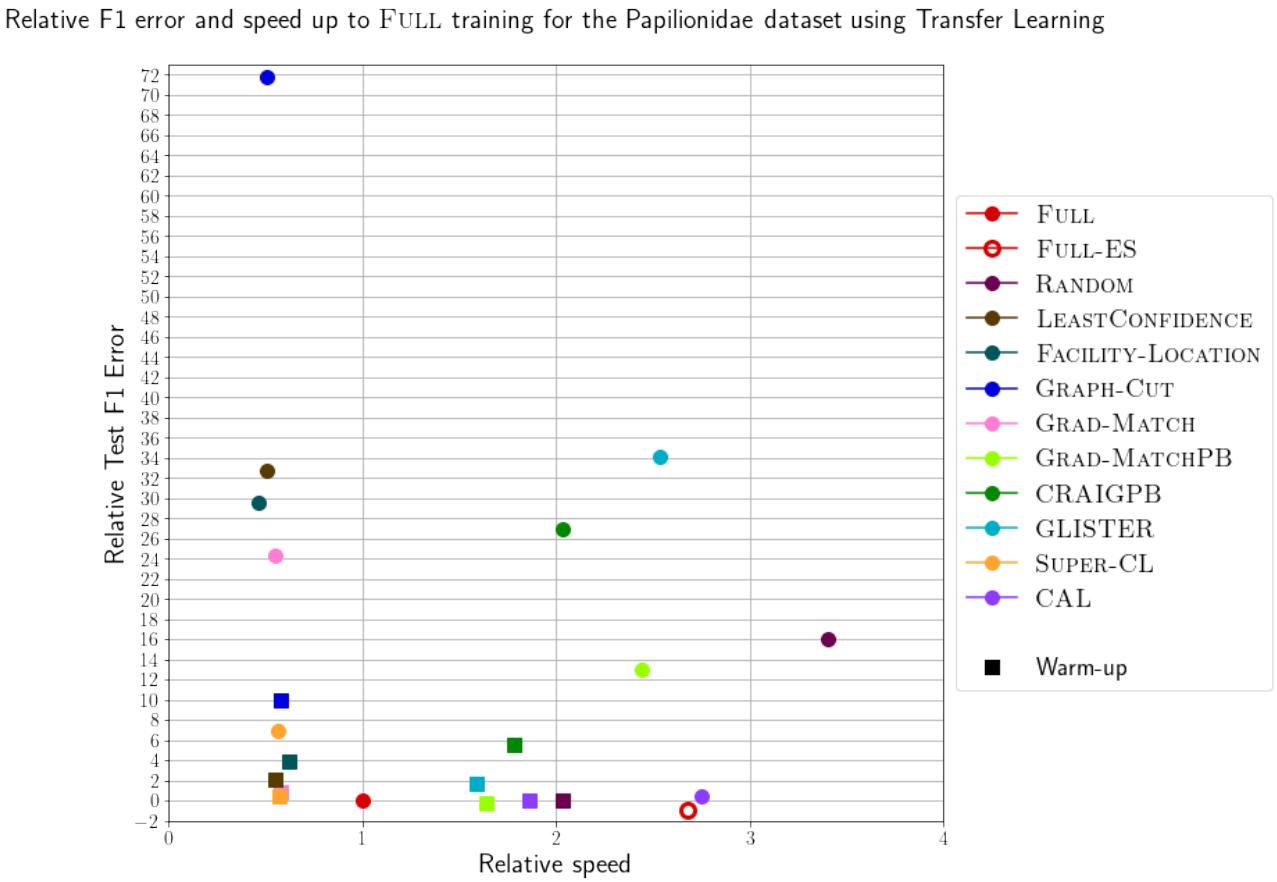

Figure 22: This plot shows the speed and error of several DSS methods relative to using FULL training for the Papilionidae dataset. All models are pre-trained on the ImageNet dataset. Relative speed is computed by dividing the total run time of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1 score of **80.4**±**1.1** and takes 0.62±0.02 hours.

In Figure 23 the results of adaptive DSS methods on the CUB200 dataset using transfer learning can be found. Their performance is compared to that of FULL training. Transfer learning had a very positive effect on FULL training on the CUB200 dataset. Using a pre-trained network with FULL training increases the final F1 score from 48.8±0.2 to 79.8±0.1. FULL-ES is almost four times as fast as FULL, while losing almost no performance. Most methods that do not use warm-up cannot speed up training. Only GLISTER, GRAD-MATCHPB, RANDOM and CRAIGPB are able to, but at the cost of performance. Using warm-up for these methods results in a performance that is almost the same as FULL while sacrificing some of the speedup reached when not using warm-up. CAL showed the worst performance, which is the same behavior as when not using transfer learning on the CUB200 dataset; see Figure 16.

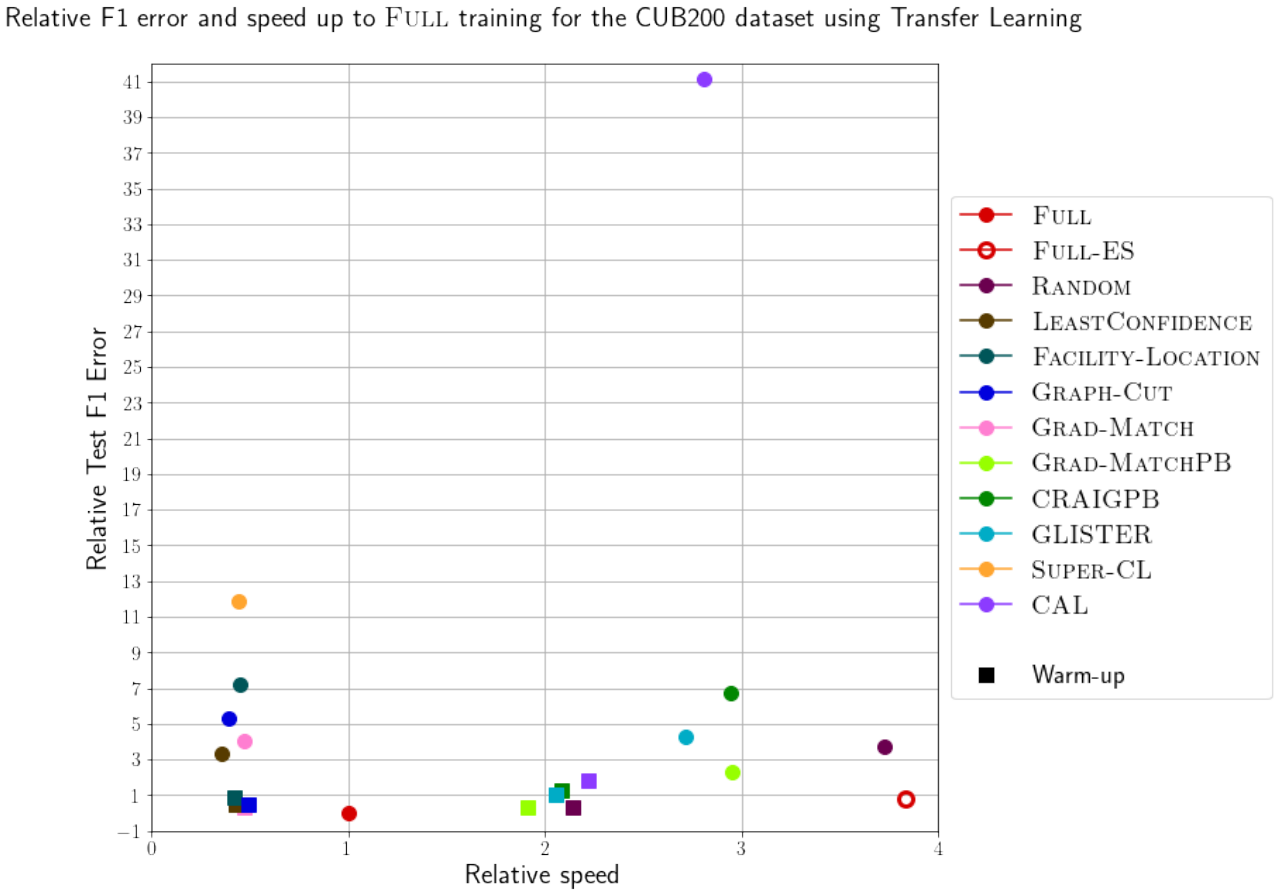

Figure 23: This plot shows the speed and error of several DSS methods relative to using FULL training for the CUB200 dataset. All models are pre-trained on the ImageNet dataset. Relative speed is computed by dividing the total run time of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1 score of **79.8±0.1** and takes 0.75±0.06 hours.

In Figure 24 the performance of the models on the Papilionidae data set can be found, while training. CAL is the only method capable of getting a performance comparable to that of training with FULL. FULL converges quickly which results in a long performance plateau. FULL-ES is able to save energy by stopping early on. This also does not sacrifice performance, as a plateau in performance has already been reached. SUPER-CL and GRAD-MATCH need more time than FULL training.

In Figure 25 the performance of the methods over time on the CUB200 dataset, while using pre-trained networks, can be found. We observe that FULL and FULL-ES converge very quickly and then reach a performance plateau. RANDOM, GRAD-MATCHPB and GLISTER are the only methods capable of achieving a performance comparable to FULL, but require much more time. SUPER-CL needs the most sampling time, as can be observed by looking at the starting point of each method graph. CAL shows the worst final performance which is the same behavior as when not using pre-trained networks; see Figure 19.

The effect of warm-up when using transfer learning can be observed in subsection D. We observe that all methods show comparable performance. Possible reasons for why this happens will be discussed in the next chapter.

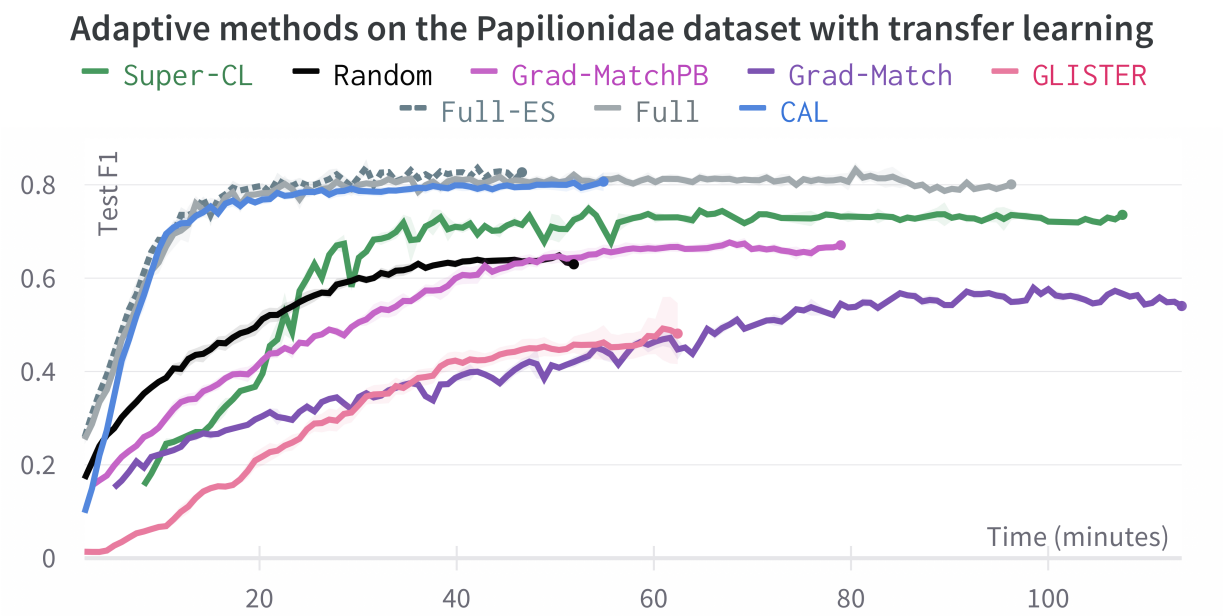

Figure 24: Test performance on the Papilionidae dataset while training with a selection of the best adaptive DSS methods and using pre-trained networks. All methods are averaged over 5 runs and the standard error is shown as error band.

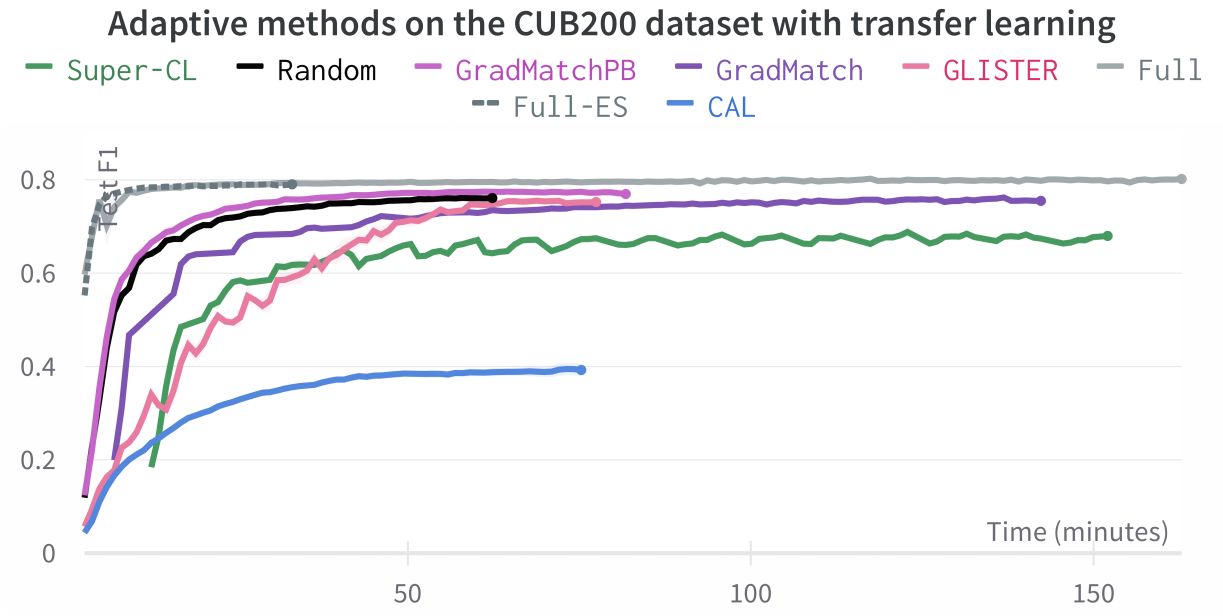

Figure 25: Test performance on the CUB200 dataset while training with a selection of the best adaptive DSS methods and using pre-trained networks. All methods are averaged over 5 runs and the standard error is shown as error band.

5.4 Supervised Contrastive Learning

In Figure 26 the performance per data sample on which the model is trained is shown. This allows us to view the performance of each method without taking the sampling duration into account. We observe that CAL shows good performance early on. The warm-up optimization results in decreased performance, and it takes longer for the model to converge. SUPER-CL shows the lowest performance. Warm-up is able to increase the final performance, but the slope of the graph is lower than without warm-up. To examine the effect of introducing diversity, we used post-training as explained in subsection 4.8. The number of epochs to use SUPER-CL before training with all data was determined by looking at the convergence point of normal SUPER-CL. Post-training results in higher performance than normal SUPER-CL. SUPER-CL-WARM reaches a higher final performance than SUPER-CL-POST-TRAINING, but post-training could be used to get good performance with fewer data.

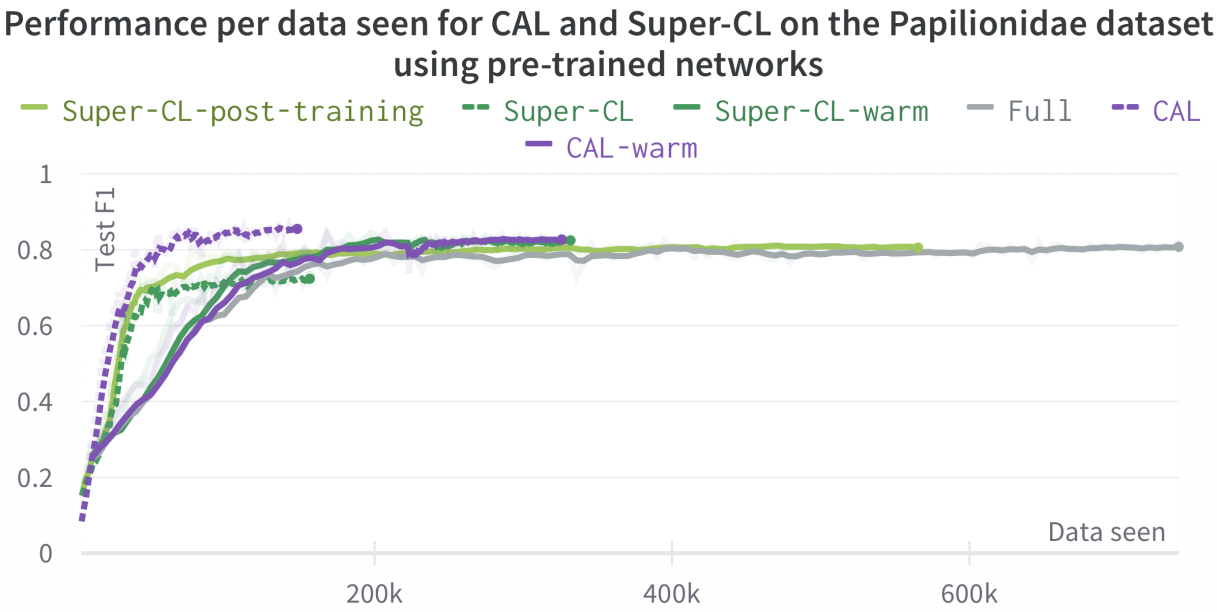

Figure 26: Test performance per amount of data samples that the model has been trained on for CAL, SUPER-CL and FULL. For CAL and SUPER-CL we also show the performance of the warm-up optimization. SUPER-CL-POST-TRAINING first uses the SUPER-CL method for κ epochs and then continues training with all data. Note that because of time constraints these results are not averaged over 5 runs but only show 1 run. This explains why CAL reaches a higher final performance than FULL and why this is not the case in Figure 24

To be able to view the samples selected by both CAL and SUPER-CL, we provide t-SNE plots for both the Papilionidae and CIFAR10 datasets in Figure 29 and Figure 32. For both methods, no transfer learning is used since the t-SNE plots of the methods with and without transfer learning are similar. We observe that SUPER-CL uses per-class sampling, and thus each class is present in the subset. CAL is not constrained by the sampling per class and is therefore able to sample more samples from the same class.

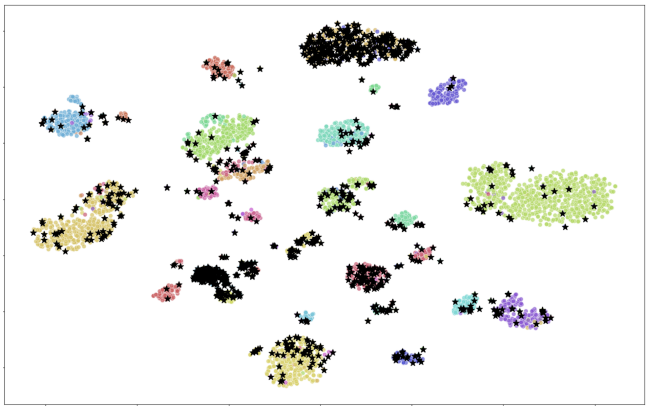

Figure 27: **CAL**

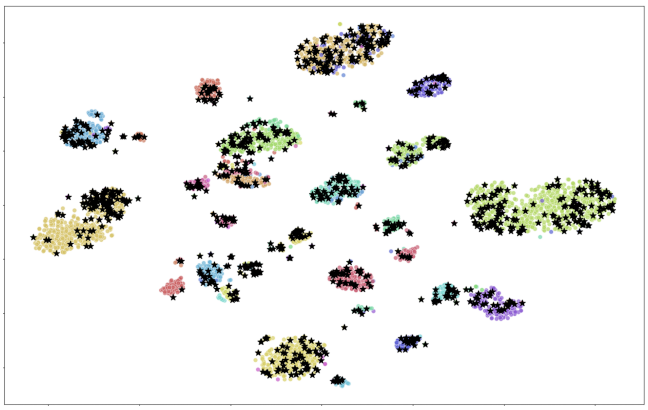

Figure 28: **SUPER-CL**

Figure 29: Two t-SNE plots of the Papilionidae dataset to visualize which samples are selected for the CAL and SUPER-CL methods. Selected samples are marked with black points. Each class is denoted by a unique color. These plots visualize the first subset that is sampled by the methods.

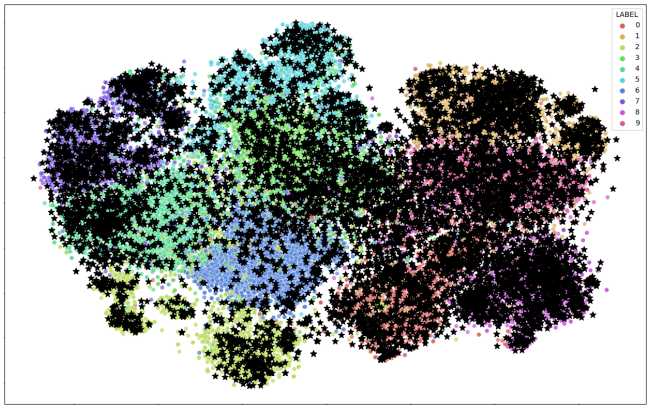

Figure 30: **CAL**

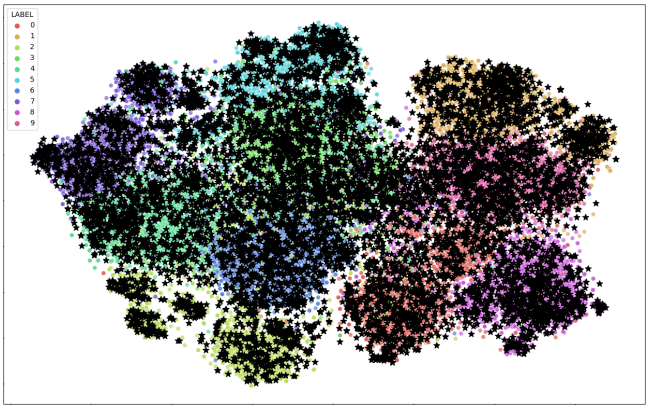

Figure 31: **SUPER-CL**

Figure 32: Two t-SNE plots of the CIFAR10 dataset to visualize which samples are selected for the CAL and SUPER-CL methods. Selected samples are marked with black points. Each class is denoted by a unique color. These plots visualize the first subset that is sampled by the methods.

6 Discussion

In this chapter, we will discuss the results of our study on optimal subset selection methods for dataset reduction. We focused on three main research questions:

1. *How do the methods compare in speedup and performance?*
2. *How applicable are the methods on complex and fine-grained datasets?*
3. *Can the methods be applied in combination with transfer learning to increase total energy efficiency?*

By answering these questions, we aim to provide insights into the trade-off between energy efficiency and performance in the context of dataset reduction.
Throughout this chapter, we will provide an in-depth analysis of the various subset selection methods that we evaluated, and compare their performance and energy efficiency. We will also discuss the implications of our findings for the use of these methods in complex and fine-grained datasets, and explore the potential for combining them with transfer learning to increase energy efficiency. By addressing these research questions, we hope to contribute to the understanding of optimal subset selection methods for dataset reduction and their potential for improving the energy efficiency of deep learning.

6.1 Non-adaptive Dataset Reduction without Transfer Learning

All non-adaptive dataset reduction methods are applied without pre-trained networks except for the PROTOTYPICAL method. This method uses a network pre-trained on ImageNet to extract the features of the samples. The models that are trained on the selected subset are not pre-trained. This allows for a fair comparison to the results of [22].
For both the CIFAR10 and Papilionidae datasets, all methods, except GRAND, are able to successfully prune the datasets without losing a lot of performance.
The CIFAR10 dataset could be pruned to 50% of its original size, while only losing (at best) around four performance percentage points. EL2N is even able to increase performance when pruning 10% of the dataset. This confirms the findings of previous research in which non-adaptive methods can increase performance [36]. However, in this paper the authors report that they can prune 50% of the CIFAR10 dataset while gaining some test performance. This could not be replicated in this research, but we confirm that EL2N can effectively prune the CIFAR10 dataset with a large fraction. The increase in performance can be the result of the removal of outliers or noisy data.
The Papilionidae dataset can also be effectively pruned without losing much performance by all methods, except the GRAND method. Pruning 10% of the dataset with EL2N can even slightly increase the performance. Although most methods can prune the dataset while maintaining high accuracy, only the EL2N and GRAD-MATCH methods can also retain a high F1 score. This shows that these methods also work well with imbalanced data.
In Figure 11 we observe the energy consumption of the non-adaptive methods. EL2N needs substantially more energy than the other methods to prune the dataset. This can be explained by the way the EL2N scores are computed. Ten neural networks must be trained for 20 epochs before they can be used to compute the scores for all samples. Although this requires much more energy, the performance of EL2N is high. The costs of reducing the dataset can be repaid by the increase in efficiency of training models on these datasets. Further research is needed to look at the effect of

using pre-trained networks with the EL2N method. We hypothesize that because of the pre-training the networks already have strong inference power and this could reduce the number of epochs the networks in the EL2N need to be trained. Because of this inference power we also hypothesize that it might lower the variance in EL2N scores and thus less networks are needed to average over. This could potentially reduce the required energy to prune a dataset. Due to time constraints this was not tested.

GRAND fails to find informative samples and is outperformed by RANDOM for the CIFAR10 and Papilionidae datasets. This was not expected, as the authors of this method report that GRAND could be used to select a subset of the CIFAR10 dataset, at initialization, that achieves test accuracy significantly better than RANDOM. In all cases, RANDOM outperforms GRAND in this research. We conclude that ranking samples based on gradient information over multiple random networks at initialization time is not a viable metric for dataset reduction. GRAD-MATCH also uses gradient information but uses this to match the gradients of the test set to that of a subset of the train set. The performance of the non-adaptive GRAD-MATCH method shows that this is a much better approach. Since GRAD-MATCH and GRAND require around the same amount of energy to prune a dataset, we would not recommend using GRAND.

For both the CIFAR10 and the Papilionidae datasets, most of the methods can successfully decrease the dataset with decreasing fraction sizes, while not losing a lot of performance. This shows that there is still redundancy in the dataset after removing the first 10% and 30%. This is a completely different story for the CUB200 dataset where for all methods the performance degrades with the amount of pruning that is performed. One potential explanation is that after a while, removing redundancy will transform into reducing diversity as there are almost no redundant samples left. The assumption is that if there is redundancy and the methods can effectively prune this from a dataset, then the performance should be comparable for the full and slightly pruned dataset. As this is not the case for the CUB200 dataset, our speculation is that there is not much redundancy present in the CUB200 dataset. This could explain the large differences in performance between various fraction sizes, as visualized in Figure 13.

The CUB200 dataset is a difficult dataset because of the finegrainedness, the large number of classes, the presence of different domain sources (e.g. paintings and photographs), and the presence of noise and overlap (e.g. leaves and tree branches in front of the birds).

Training with all data without transfer learning results in a model that is only able to correctly classify about half of the data points correctly, which is a relatively low level of performance. The model thus cannot fully understand the underlying data distribution. Therefore, it is questionable to use the model information in EL2N to successfully prune the dataset. In EL2N the model error is used to determine the difficulty of all samples. It is assumed that difficult samples are more informative and thus the samples are ranked and chosen according to their EL2N score. However, what if the error is high because the image is too difficult due to a lot of noise or because of the absence of class-specific features? For example, an image of a bird that is almost completely covered by a branch. The assumption that difficult images are informative does not hold in that case. Therefore, the complexity of the dataset could potentially explain why EL2N performs worse than RANDOM.

We conclude that the performance of non-adaptive dataset reduction methods is heavily influenced by the presence or absence of redundancy. If redundancy is present, methods such as EL2N or GRAD-MATCH can be used to safely prune the dataset. If redundancy is not present, no dataset reduction should be performed, as we need all the diversity that we can get. Therefore, it is important to know the amount of redundancy before hand. However, to the best of my knowledge, there is no method that can measure this in advance without doing substantial computation. A solution would be to use a non-adaptive dataset reduction method with several different fraction sizes. By comparing the

performance on each fraction of the dataset, we can determine if there is redundancy present and how much. However, this results in a lot of computation and might not be efficient.
Another approach would be to devise an algorithm that can dynamically compute the redundancy by using a submodular function. Submodular functions can be solved by an iterative greedy process and follow the property of diminishing returns. Samples can be added iteratively to a subset until the gain is below a certain threshold. When a sample has a very low gain, it can be discarded as redundant. When the threshold is reached, we stop sampling and can determine that the remaining samples are redundant. Further research is needed to devise a function that can measure the informativeness and to devise the threshold. This greedy approach might fail to find the global optimum. A potential solution would be to repeat the process and to select samples that are present in most of the subsets. In addition to this, the robustness of the EL2N method should be investigated by testing it on more datasets. Experiments with manual redundancy injection could be performed by replicating samples and using augmentations on the copies.

6.2 Adaptive Dataset Reduction without Transfer Learning

CIFAR10 and Papilionidae The GRAD-MATCHPB, RANDOM and SUPER-CL methods are able to speed up training on the CIFAR10 dataset while losing limited accuracy. Because of the non-adaptive results, we know that there is a lot of redundancy in both the CIFAR10 and Papilionidae datasets. For good performance, we thus need subsets that are diverse and represent the dataset well. This explains why RANDOM performs so well on the CIFAR10 dataset, since every subset is random, there is much less overlap between subsets, and thus there is a lot of diversity.

The uncertainty-based methods, ENTROPY and LEAST-CONFIDENCE, and the submodularity methods, FACILITY-LOCATION and GRAPH-CUT, do not perform well in general. For the uncertainty-based methods, this could be explained by how they use model uncertainty to find informative samples. An untrained model does not have good inference power, the error of the model might thus not reflect how informative samples are. In addition to that, because the uncertainty-based methods are not applied per class, the subsets can be very imbalanced and thus not very diverse. For the submodularity methods, the low performance could be because the submodular functions are applied on the gradient space and not on the feature space. If applied to the feature space, the facility location method could potentially select diverse subsets due to the coverage property of the facility location function. The gradient space might not be a good representation, and more research is needed to examine the effect of applying submodular functions to the feature space.

Observing the t-SNE plot of SUPER-CL and CAL in Figure 32, we see that SUPER-CL is capable of selecting a very diverse and representative subset of the dataset as the selected samples are well distributed across the classes. This is because SUPER-CL uses per-class sampling. This differs from CAL which undersamples some classes while oversampling others. Since CIFAR10 is a balanced dataset, this results in training on imbalanced data, which can explain the low performance.
For the Papilionidae dataset, SUPER-CL does not perform well, while CAL does. Papilionidae is an imbalanced dataset. SUPER-CL uses per-class sampling, selecting the same fraction of samples from each class. This keeps the imbalance in the subset and can explain the lower performance. It thus seems that the class budget calculation is incorrect and should be changed. Instead of taking the same fraction of each class, the method should balance the subset such that each class is represented equally. However, this does not allow the method to focus on hard samples. Since it might not be able to select more samples of that class if the budget already has been reached. Next to that, classes

on which the model already performs really well are still selected and this might be inefficient. This problem could potentially be solved by dividing the subset in two parts. One part can be used to select hard samples within classes and that is balanced such that each class is represented equally. The other part could be used to target samples that have not yet been selected but that have high contrastive scores globally. This thus essentially combines SUPER-CL with CAL and balances diversity and informativeness.

Observing Figure 29, we see that SUPER-CL indeed chooses a diverse subset, since all classes are present in the subset. However, CAL is capable of selecting difficult samples. For example, the cluster at the top of the t-SNE plots actually consists of two classes. The t-SNE algorithm has difficulties in clustering the two classes, which could be explained by feature overlap. These samples are very informative since they are close to the decision boundary of these classes. CAL samples these points more effectively than SUPER-CL since it is not constrained by a per-class budget and calculates contrastive scores over the entire dataset and not within classes. CAL is even able to reach a higher performance on the Papilionidae dataset than normal training.

CUB200 No method is able to speed up training on the CUB200 data set without losing significant performance. Based on the results of the non-adaptive methods, we speculate that there is little to no redundancy in the dataset. If there is almost no redundancy, the adaptive method should visit all samples over the whole training cycle such that there is as much diversity as possible. This also means that selecting the same samples in every subset reduces the total diversity. This explains why RANDOM performs so well, since the probability that a sample is selected is the same for all samples, and thus all samples can be visited over the complete training run. This differs from SUPER-CL and CAL. With these methods, it can happen that some samples with low contrastive scores might never be selected. This could result in a lot of overlap in the subsets, e.g. a large number of the same samples in every subset. This reduces diversity and can explain the low performance of both SUPER-CL and CAL. By tracking the selected samples in each subset the overlap could be calculated. Future research should be directed into the effect of overlap in subsets. GRAD-MATCHPB is able to select subsets and loss weights that together make a good representation of the validation set. GRAD-MATCHPB seems to be the most robust method as it performs well on all datasets. It is not able to speed up training on CUB200 without losing performance. This might also be explained by the fact that not all samples have been visited. A possible solution might be to always randomly select a part of the subset for all methods. This might allow for a balance between diverse and informative subsets.

Sampling duration The number of classes has a large influence on the sampling speed of per-class sampling methods. In Figure 20 and Figure 21 we observe the sampling duration of some of the methods on the Papilionidae and CUB200 datasets. The datasets have 112 and 200 classes, respectively. The sampling duration for the per-class sampling methods to sample 20% of the train set is much higher with the CUB200 dataset. Methods that sample per-batch are not affected by the increase in the number of classes and are only influenced by the dataset size and batch size. This explains the large difference in sampling duration between Grad-Match and Grad-Match-PB. SUPER-CL has a long time, which also explains the later start of the SUPER-CL method in Figure 18 and Figure 19. The long sampling duration for the first iteration of SUPER-CL is due to the kNN computation. This could be optimized using the FAISS library, which was performed with CAL. This was not done because of time constraints.

Warm-up In general, warm-up is able to give most methods a good starting point where the model already has some inference power. In some cases, the use of warm-up can even result in higher performance than normal training. However, we do notice that too much warm-up has a negative effect on the performance and energy trade-off. This is also expected, as training on all data in an epoch takes longer than training on a subset. It might thus be possible to use fewer warm-up epochs and get the same final performance early on. Thus saving even more energy.
Besides that, we notice that too much warm-up results in a performance plateau while training. Continuing with training on all data when already on a plateau is inefficient as this does not improve performance. For example, in Figure 17 we see that the warm-up methods follow the same performance as FULL, which is also expected. However, it is only when the warm-up is finished that the methods show an increased performance. More research is needed to optimize the number of warm-up epochs and to find a robust value for κ across datasets.

Extended training In general, extended training can increase the performance of adaptive DSS methods while sacrificing some speed-up. However, extended training can still be faster than FULL training since the subsets are smaller. If the DSS method with the normal number of epochs has already reached a performance plateau, then extended training will not give increased performance and will only require more energy. The number of extra epochs that can be trained thus depends on the speed-up of the DSS method. If this is known and a performance plateau has not been reached, it is advised to use extended training to improve performance. Of course, this all depends on the goal, i.e. the speed-performance trade-off.

Warm-up and extended training The combination of both warm-up and extended training does not appear to be a good method. This is mostly affected by the large number of warm-up epochs that are used. The model has already reached a performance plateau because of this. Extended training will not increase performance and will only cost more energy. Thus, more research is first needed on the effect on varying the κ values.

Subset size The fraction parameter determines the size of the subset that the DSS method should select from the training set. In general, using smaller subsets means higher speed up but with higher error. This could potentially be explained by the fact that the subsets have overlap in which samples are selected. This not only decreases the total diversity over all subsets but it could also result in overfitting on specific samples. Using larger subsets improves performance but since now more data is used in each epoch, training will take longer. For some methods increasing the subset size does not result in longer sampling duration. These methods have to first compute scores to rank the samples before selection can be performed. Larger subsets also contain more diversity which can improve the DSS method performance. More research is needed to find a fraction size that is a good balance between performance and speed up.

6.3 Adaptive Dataset Reduction with Transfer Learning

Papilionidae The CAL and FULL-ES methods are the only methods able to accelerate training on the Papilionidae dataset. In the relative performance plot in Figure 22 we observe that FULL-ES is able to get a higher performance while being almost three times as fast as FULL. However, this is a small increase in performance. FULL and FULL-ES both train on all data and the only difference

in the methods is the early stopping mechanism. A potential explanation for the difference in performance is the low number of runs on which the experiments are averaged. Increasing this number will increase the variance and averaging these runs might result in the same performance in FULL and FULL-ES. Transfer learning can be seen as a form of warm-up but then on a different data distribution than with normal warm-up. The network is assumed to have already learned to recognize basic features. The network now has to be fine-tuned on the data distribution of the Papilionidae and CUB200 datasets to learn to recognize high-level features and to be able to discriminate between classes. Following the results of the non-adaptive methods on the Papilionidae dataset we assume that there is redundancy in this dataset. With transfer learning, the network needs less data to reach good performance. If there is redundancy in a dataset, we do not have to train on all data. We need some diversity to be able to learn the underlying data distribution, and we need to target informative samples to get discriminative power. Methods such as GRAD-MATCH, GRAD-MATCHPB, GLISTER, CRAIGPB and FACILITY-LOCATION try to find a subset that represents the train set well. The performance of these methods with pre-trained networks on the Papilionidae is comparable as to when not using transfer learning. They do reach this performance faster, which could be explained by the fact that the pre-trained networks already know how to recognize basic features. However, these methods do not target informative samples, and their subsets can overlap, which results in not having selected all samples, as viewed over the complete training run. This could explain why these methods have lower performance than FULL. This is different for methods that use the current inferential power of the model to determine hard samples. Methods such as CAL and SUPER-CL do target these samples and thus get both diversity and higher performance. SUPER-CL has lower performance than CAL since it is constrained by the per-class sampling class budgets. Because of this constraint, it can not target specific classes and under sample less informative classes. This behavior can also be observed in Figure 26. Here we see that both CAL and SUPER-CL are able to sample more informative samples and can get higher performance with less data than FULL. After a while, FULL catches up with SUPER-CL and even surpasses its performance. Since SUPER-CL is constrained, we hypothesize that it cannot visit all samples and thus misses some diversity. To confirm this hypothesis, we inject diversity using post-training with SUPER-CL when it reaches the performance plateau. In Figure 26 we see that post-training indeed increases performance and allows the method to escape the plateau.

CUB200 For the CUB200 dataset we observe similar results when using transfer learning as when not using transfer learning. The use of pre-trained networks results in higher final performance which is to be expected since pre-trained models already have knowledge about features. As mentioned before, based on the non-adaptive results we assume that there is almost no redundancy in the CUB200 dataset. This means that we need subsets that are very representative of the train set. We need as much diversity as we can to learn the complete underlying data distribution of the CUB200 dataset. Without redundancy, the difference in informativeness of samples is lower than when there is redundancy. These properties explain why the methods that sample representative subsets perform so well. For example, the RANDOM method. It samples diverse subsets and because of the uniform sampling, every sample has the same probability to be selected. Thus, there is much less overlap in the subsets, and there is a high probability that all samples will be visited. However, there can still be overlap, and thus some samples might be selected more often than others, and it might still be possible that some samples are never selected. This can explain why RANDOM gets a final performance lower than that of FULL.
Targeted sampling with CAL does not work as there is much more overlap in subsets, less diversity, and not all samples are visited. SUPER-CL works better than CAL, which might be surprising. This can be explained by the per-class sampling it uses. It samples the same percentage of samples in a

class for all classes. Since CUB200 is a relatively balanced dataset, this means that the subsets always contain all classes and therefore are diverse.

Warm-up The performance of the methods while using both transfer learning and a warm-up strategy can be found in Figure 38 and Figure 39. All methods can converge while training on the full data in the warm-up epochs. For all methods, it seems that the number of warm-up epochs is too large as this results in a performance plateau. The GRAD-MATCHPB-WARM method seems to be able to even increase the performance after having completed the warm-up epochs. More research is needed to look at the effect of using smaller values of κ when using pre-trained networks.

Extended training Extended training was not performed with methods that use transfer learning. The reason for this is that all methods can converge and extended training would only decrease energy efficiency without gaining performance.

6.4 Limitations

Eventually, the iNaturalist dataset was not used, as its size makes it impractical to compare all methods. Some methods need to compute metrics, such as the pairwise distance between all data samples. This cannot be done batch-wise, so these metrics of the whole dataset have to be kept in memory, which is not possible for large datasets. Although this does not allow for a practical and fair comparison between methods, it does show the impracticality of some methods on large datasets.

One of the reasons the SUPER-CL method was devised was to solve the out-of-memory errors of the CAL method. In a later stage of the research we were able to solve this with Faiss. Not only does this solve the memory errors, it also greatly speeds up sampling. The sampling duration of SUPER-CL is relatively large and this could be reduced by using Faiss. However, this library was found in a late stage of the research and could not be used in SUPER-CL because of time constraints.

The fraction size of the dataset used in the subset, selection frequency and sampling duration have large influence on training time and speed-up for adaptive data subset selection (ADSS) based training. When the fraction size is too large, ADSS-based training will be almost the same as training on all data and no speedup is reached. If the selection frequency is high and a method has a relatively long sample duration this will heavily influence the potential speedup and could even result in longer training than FULL. We needed to do a generalization in terms of the subset size and frequency of the methods in order to compare the performance. In this research the frequency and fraction size of the ADSS methods is based on the ablation study in [39]. The assumption was that the selection frequency and subset size would not be method dependent. However, this actually can be method dependent since some methods need to calculate a metric for all samples in the dataset before being able to rank and select based on these scores. Increasing the subset size thus does not influence sampling duration. The frequency and subset size should be optimized to allow for a better performance comparison.

7 Conclusion

The recent increase in model performance for tasks in Natural Language Processing and Computer Vision is said to be related to the exponential increase in the number of parameters. With the increase in model size comes the need for for massive datasets. This is coupled with large financial and environmental costs. In this research we focus on dataset reduction methods that potentially save energy and time when training deep learning models. These methods aim to reduce training time by removing redundant samples, for instance, by cleverly selecting samples during training or by synthesizing a new (smaller) distilled dataset. As synthesis techniques are still very energy inefficient, this research mainly focused on selection techniques. Multiple non-adaptive methods were implemented to see if they can effectively prune datasets. In addition to that, several methods were implemented that can adaptively select subsets during training. We applied these algorithms on biodiversity datasets, which are fine-grained and imbalanced datasets that make a good test case for operational machine learning settings in real-world problems. Furthermore, the effect of combining transfer learning with dataset reduction was studied as this could potentially further increase energy efficiency.

In this research we set out to answer the following research question: *"What are the optimal subset selection methods for dataset reduction?"*. The following three subquestions were defined:

1. *How do the methods compare in speedup and performance?*

The non-adaptive dataset reduction methods EL2N and GRAD-MATCH are able to prune datasets when redundancy is present. EL2N was even able to increase the performance which might be explained by the removal of noisy samples or outliers.
For adaptive dataset reduction methods, we hypothesize that the performance of the methods is highly influenced by the presence or absence of redundancy. However, we do not yet have an objective measure of redundancy to confirm this assumption. We do find a strong dependency on the dataset to which the method is applied. When redundancy is present the results show that CAL not only speeds up training but also increases performance. When there is little to no redundancy in a dataset the results show that representative-based methods such as GRAD-MATCHPB perform the best. However, while these methods do result in a speedup, they also result in lower performance. Future research in objective measurements for redundancy are needed to confirm the assumption.

2. *How applicable are the methods on complex and fine-grained (biodiversity) datasets?*

Dataset properties, such as long-tail class distribution and fine-grainedness, can make dataset reduction difficult for some of the methods. The calculation of the class budget of SUPER-CL does not work well with imbalanced datasets and should be adjusted to be a balance between diversity and informativeness. For biodiversity datasets that do not contain much redundancy, such as the CUB200 dataset, we find that most adaptive dataset reduction methods fail to efficiently speed up training while maintaining good accuracy. If there is almost no redundancy, the adaptive method should visit all samples over the whole training cycle such that there is as much diversity as possible. This also means that selecting the same samples in every subset reduces the total diversity.
Only GRAD-MATCHPB-WARM was able to speed up training while reaching higher performance. This method selects subsets on which the gradient is similar to that of the test set and is the most robust. RANDOM adaptive data subset selection also works with these datasets since it selects diverse subsets that do not have much overlap.

The CAL method is capable of speeding up training and improving performance by targeting informative samples. This works well with datasets that contain redundancy such as the Papilionidae dataset. It however fails with the CUB200 dataset. This can be explained by the absence of diversity, as the method does not select samples with low contrastive scores.

3. *Can the methods be applied in combination with Transfer Learning to increase the total energy efficiency?*

The results show that when using pre-trained networks, it is best to not use dataset reduction methods. Normal training gives the best performance and gets there faster than when using dataset reduction. Warm-up optimization could give the method a good starting point to increase the potential speedup and performance. However, too much warmup was used in the experiments and the model has already reached a performance plateau before doing data subset selection. More research is needed to look at the effect of using very little warmup in combination with transfer learning.
When the sampling duration for each method is disregarded, the CAL and SUPER-CL methods result in a higher performance while using less data early on in training. This shows that dataset reduction could effectively be combined with transfer learning, provided that sampling can be done efficiently.

Bibliography

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[4] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," 2022.

[5] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," 2021.

[6] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016.

[7] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," 2020.

[8] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," 2019.

[9] H. Asi and J. C. Duchi, "The importance of better models in stochastic optimization," 2019.

[10] F. U. Nuha and Afiahayati, "Training dataset reduction on generative adversarial network," *Procedia Computer Science*, vol. 144, pp. 133–139, 2018. INNS Conference on Big Data and Deep Learning.

[11] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," *arXiv preprint arXiv:1906.03671*, 2019.

[12] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Geometric approximation via coresets," in *COMBINATORIAL AND COMPUTATIONAL GEOMETRY, MSRI*, pp. 1–30, University Press, 2005.

[13] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," 2018.

[14] F. P. Such, A. Rawal, J. Lehman, K. O. Stanley, and J. Clune, "Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data," 2019.

[15] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," 2020.

[16] B. Zhao and H. Bilen, "Dataset condensation with differentiable siamese augmentation," 2021.

[17] K. Wang, B. Zhao, X. Peng, Z. Zhu, S. Yang, S. Wang, G. Huang, H. Bilen, X. Wang, and Y. You, "Cafe: Learning to condense dataset by aligning features," 2022.

[18] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," *arXiv preprint arXiv:2203.11932*, 2022.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[20] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Approximating extent measures of points," *J. ACM*, vol. 51, p. 606–635, jul 2004.

[21] D. Feldman, "Introduction to core-sets: an updated survey," 2020.

[22] C. Guo, B. Zhao, and Y. Bai, "Deepcore: A comprehensive library for coreset selection in deep learning," 2022.

[23] K. Thyagharajan and G. Kalaiarasi, "A review on near-duplicate detection of images using computer vision techniques," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 897–916, 2021.

[24] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[25] Y. Chen, M. Welling, and A. Smola, "Super-samples from kernel herding," *arXiv preprint arXiv:1203.3472*, 2012.

[26] B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli, and A. S. Morcos, "Beyond neural scaling laws: beating power law scaling via data pruning," *arXiv preprint arXiv:2206.14486*, 2022.

[27] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9912–9924, 2020.

[28] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[29] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005.

[30] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[31] R. Iyer, N. Khargoankar, J. Bilmes, and H. Asanani, "Submodular combinatorial information measures with applications in machine learning," in *Algorithmic Learning Theory*, pp. 722–754, PMLR, 2021.

[32] L. Kaufman and P. J. Rousseeuw, "Partitioning around medoids (program pam)," *Finding groups in data: an introduction to cluster analysis*, vol. 344, pp. 68–125, 1990.

[33] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in data subset selection and active learning," in *International Conference on Machine Learning*, pp. 1954–1963, PMLR, 2015.

[34] K. Margatina, G. Vernikos, L. Barrault, and N. Aletras, "Active learning by acquiring contrastive examples," *arXiv preprint arXiv:2109.03764*, 2021.

[35] M. Ducoffe and F. Precioso, "Adversarial active learning for deep networks: a margin based approach," *arXiv preprint arXiv:1802.09841*, 2018.

[36] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," *arXiv preprint arXiv:1812.05159*, 2018.

[38] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *International Conference on Machine Learning*, pp. 6950–6960, PMLR, 2020.

[39] K. Killamsetty, S. Durga, G. Ramakrishnan, A. De, and R. Iyer, "Grad-match: Gradient matching based data subset selection for efficient deep model training," in *International Conference on Machine Learning*, pp. 5464–5474, PMLR, 2021.

[40] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44 vol.1, 1993.

[41] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, and R. Iyer, "Glister: Generalization based data subset selection for efficient and robust learning," *arXiv preprint arXiv:2012.10630*, 2020.

[42] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[44] S. Lee, S. Chun, S. Jung, S. Yun, and S. Yoon, "Dataset condensation with contrastive signals," *arXiv preprint arXiv:2202.02916*, 2022.

[45] J.-h. Shim, K. Kong, and S.-J. Kang, "Core-set sampling for efficient neural architecture search," 2021.

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

[47] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

[48] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *CoRR*, vol. abs/1911.02685, 2019.

[49] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

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

[51] V. García, R. A. Mollineda, and J. S. Sánchez, "On the k-nn performance in a challenging scenario of imbalance and overlapping," *Pattern Analysis and Applications*, vol. 11, no. 3, pp. 269–280, 2008.

[52] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[53] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-ucsd birds 200," 2010.

[54] L. Hogeweg, "Papilionidae dataset," 9 2021. Data retrieved from Figshare, `https://figshare.com/articles/figure/figshare_zip/16627324`.

[55] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.

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

[58] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[59] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[60] F. J. Valverde-Albacete and C. Peláez-Moreno, "100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox," *PloS one*, vol. 9, no. 1, p. e84217, 2014.

[61] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[62] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[63] J. M. Schreiber, J. A. Bilmes, and W. S. Noble, "apricot: Submodular selection for data summarization in python.," *J. Mach. Learn. Res.*, vol. 21, pp. 161–1, 2020.

Appendices

A Non-adaptive DSS performance over time on Papilionidae

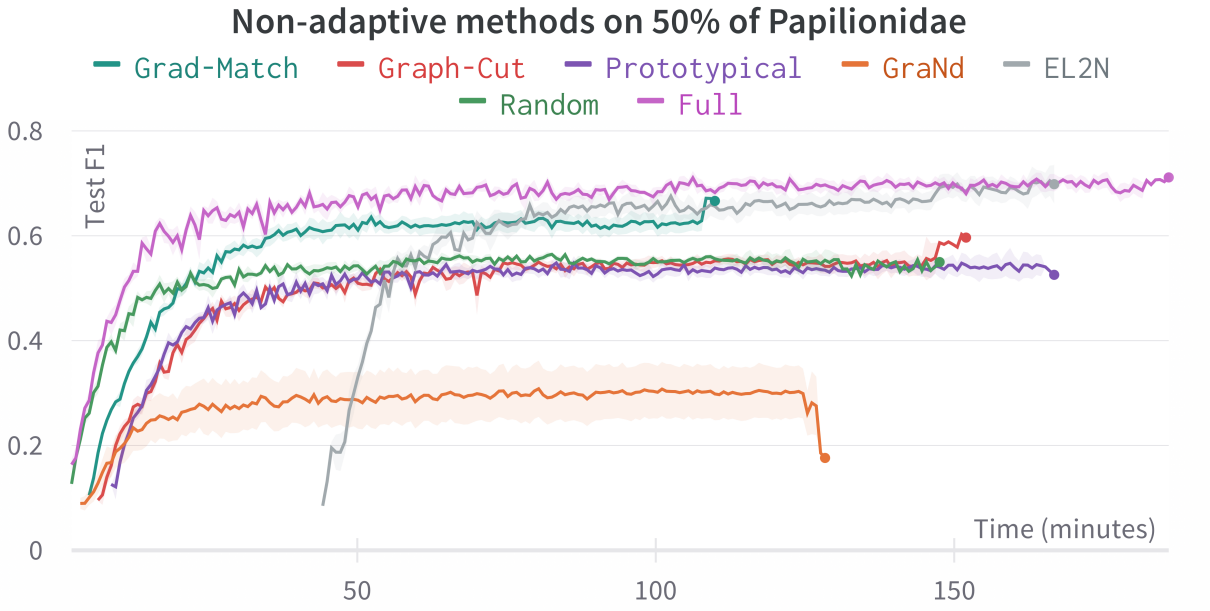

Figure 33: Test F1 over time for models trained on the subsets provided by the non-adaptive methods. Each method is averaged over five runs and the std. error is shown as error band. FULL represents the performance of a model trained on all data. The sampling duration of each method is included in the total time. Note the difference in sampling duration of each method by observing the start time.

B Relative energy consumption and performance for adaptive DSS methods

Only the energy consumption of the GPU could be correctly measured. CPU energy consumption is unknown. Because of this, we use time as a proxy for comparing the methods. Although the time and energy plots are very comparable, the energy plots are provided here for the sake of completeness.

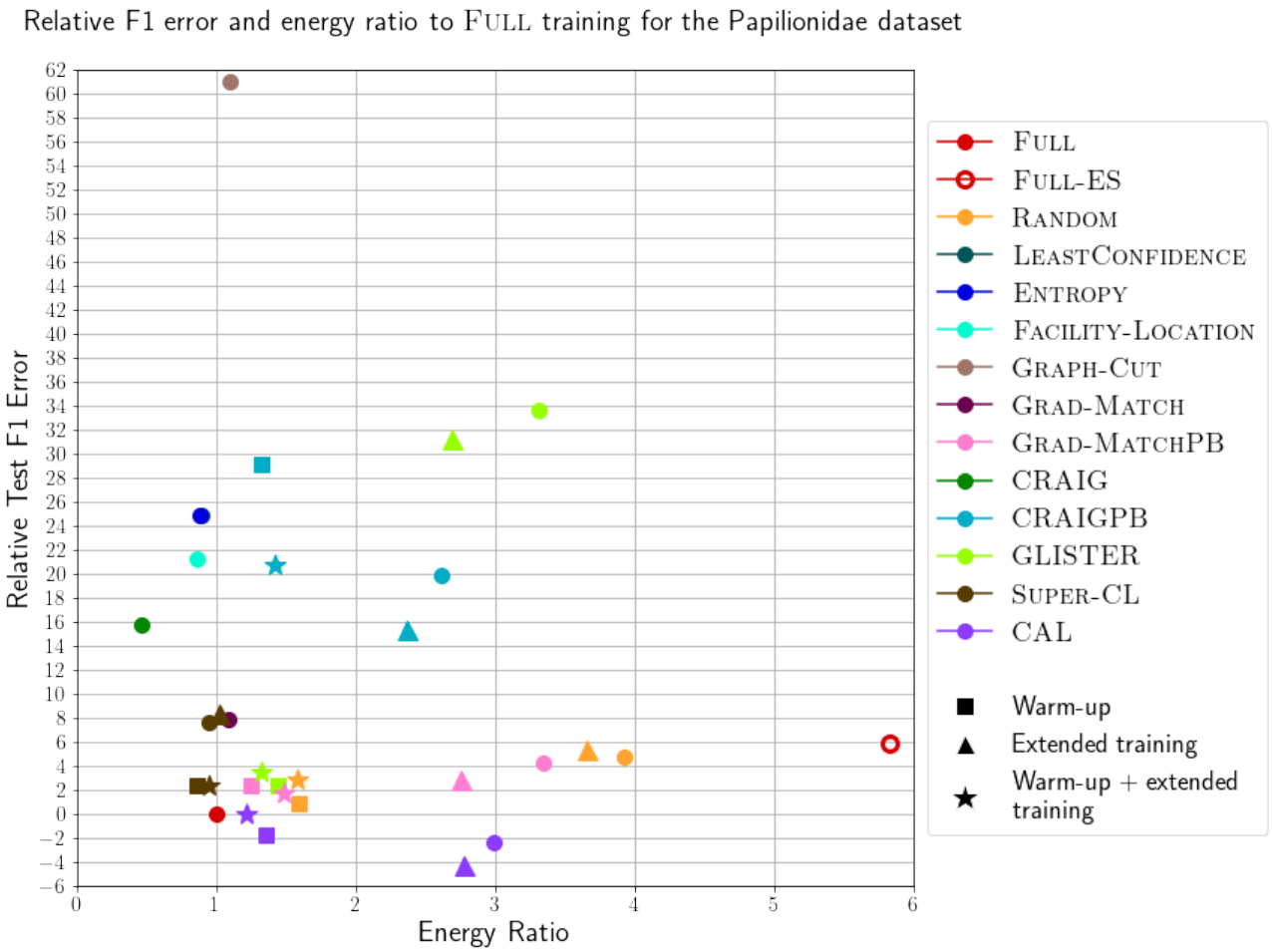

Figure 34: This plot shows the energy consumption and error of several DSS methods relative to using FULL training for the Papilionidae dataset. The energy ratio is computed by dividing the total energy consumption of FULL by that of each DSS method. Relative F1-score error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1-score of **70±1.1**

.

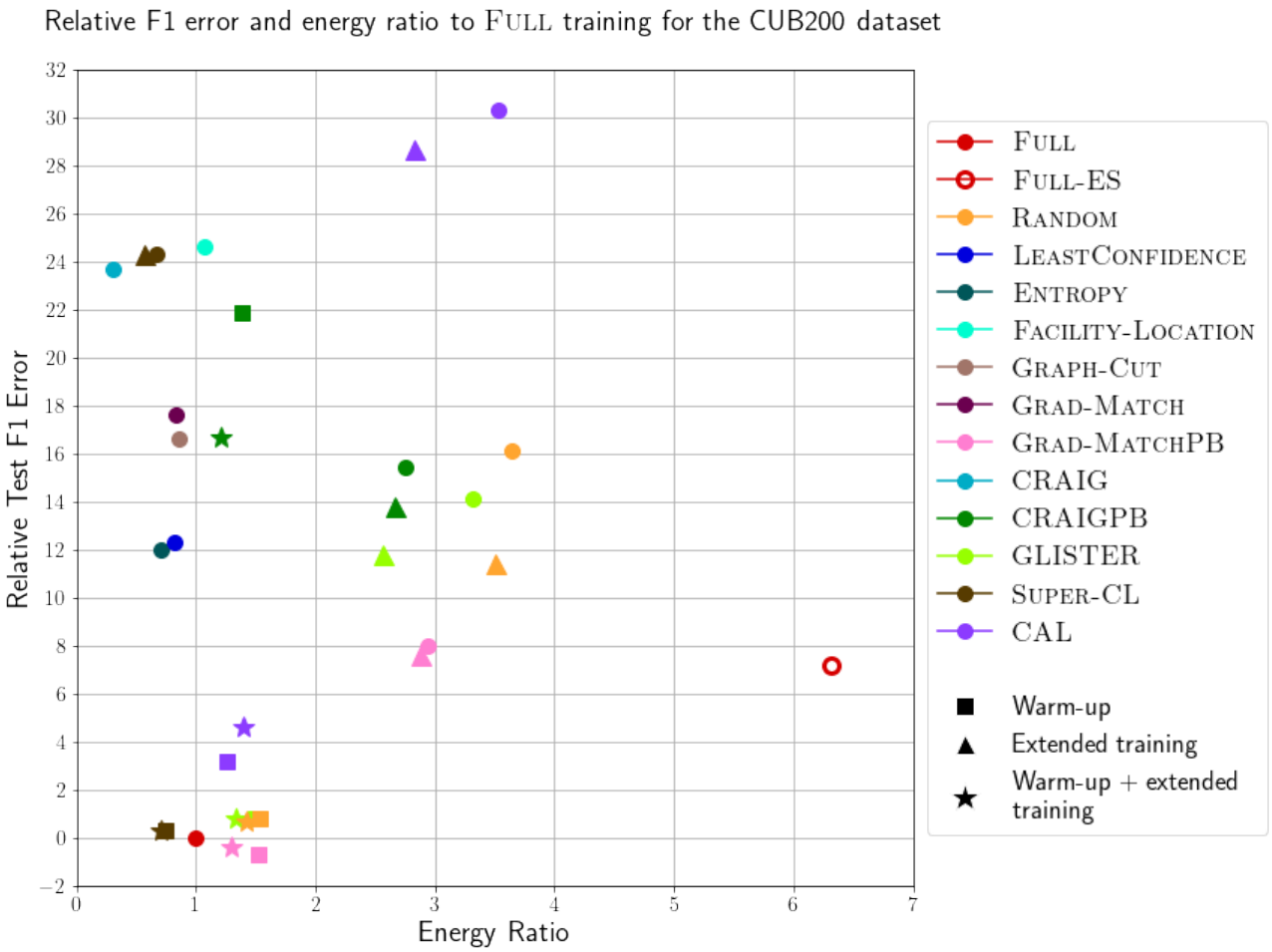

Figure 35: This plot shows the energy consumption and error of several DSS methods relative to using FULL training for the CUB200 dataset. The energy ratio is computed by dividing the total energy consumption of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1-score of **48.8±0.2**

C Relative energy consumption and performance for adaptive DSS methods with transfer learning

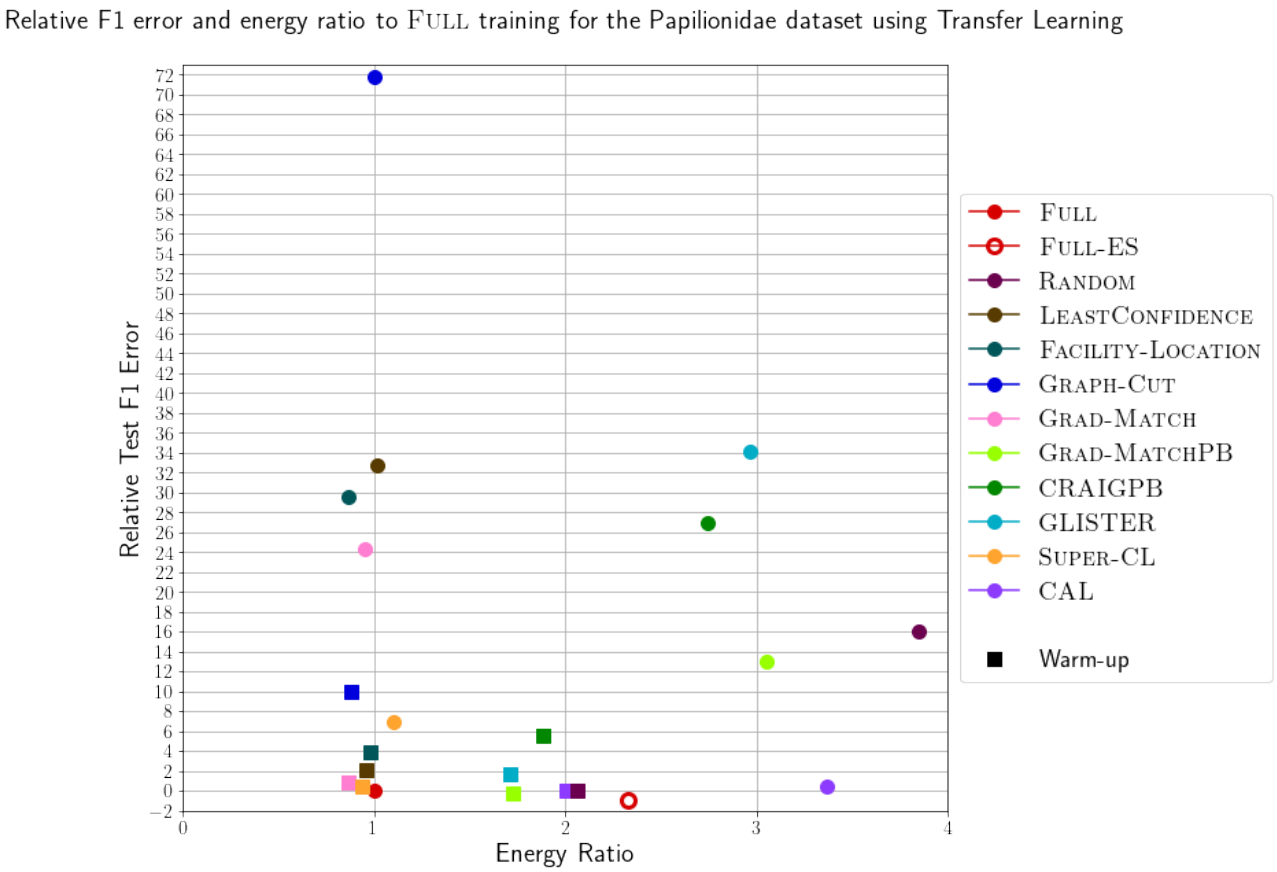

Figure 36: This plot shows the energy consumption and error of several DSS methods relative to using FULL training for the Papilionidae dataset. All models are pre-trained on the ImageNet dataset. The energy ratio is computed by dividing the total energy consumption of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1-score of **80.4±1.1**.

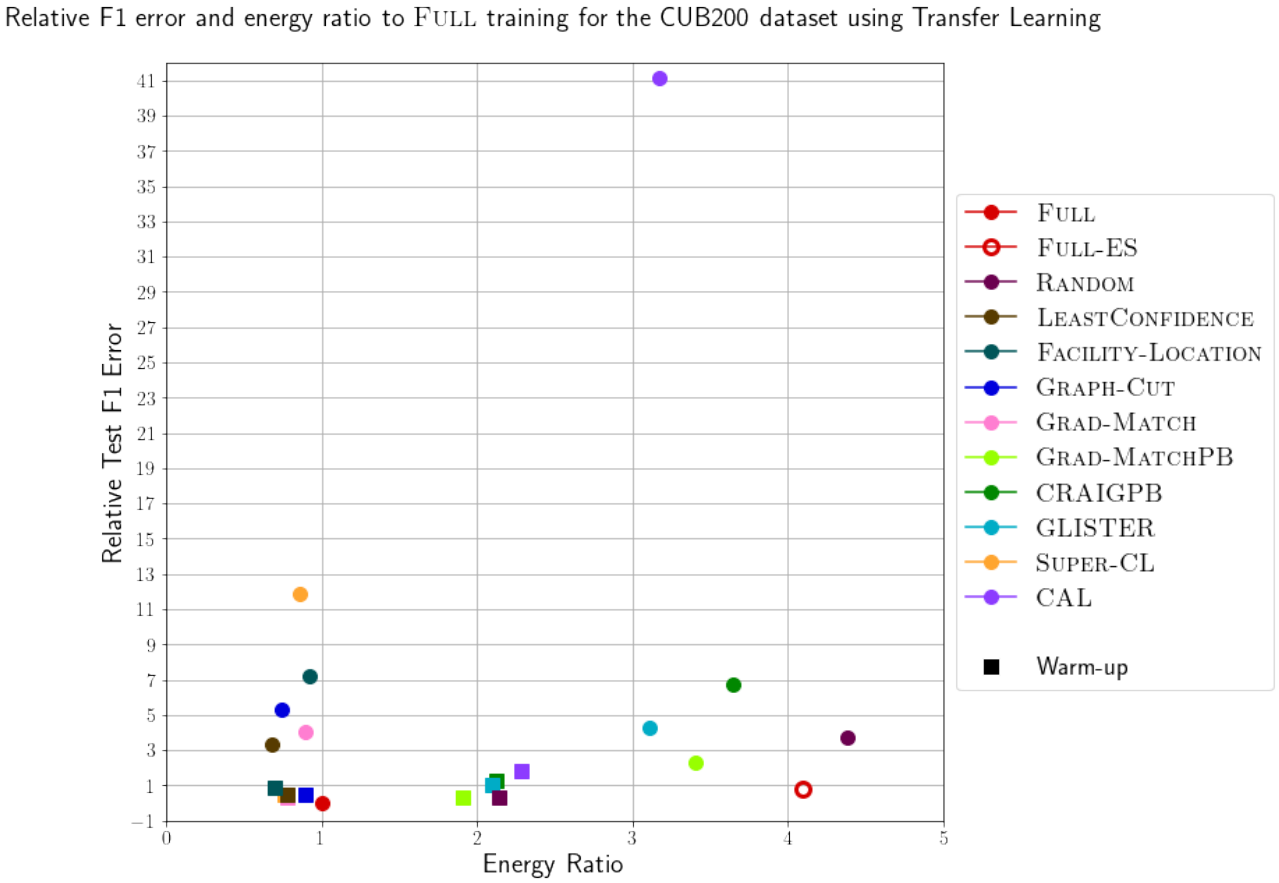

Figure 37: This plot shows the energy consumption and error of several DSS methods relative to using FULL training for the CUB200 dataset. All models are pre-trained on the ImageNet dataset. The energy ratio is computed by dividing the total energy consumption of FULL by that of each DSS method. Relative error is the difference in F1 score of the FULL and DSS methods. FULL-ES is FULL training with early stopping. FULL has an F1-score of **79.8±0.1**

D The effect of warm-up and transfer learning on the adaptive DSS methods

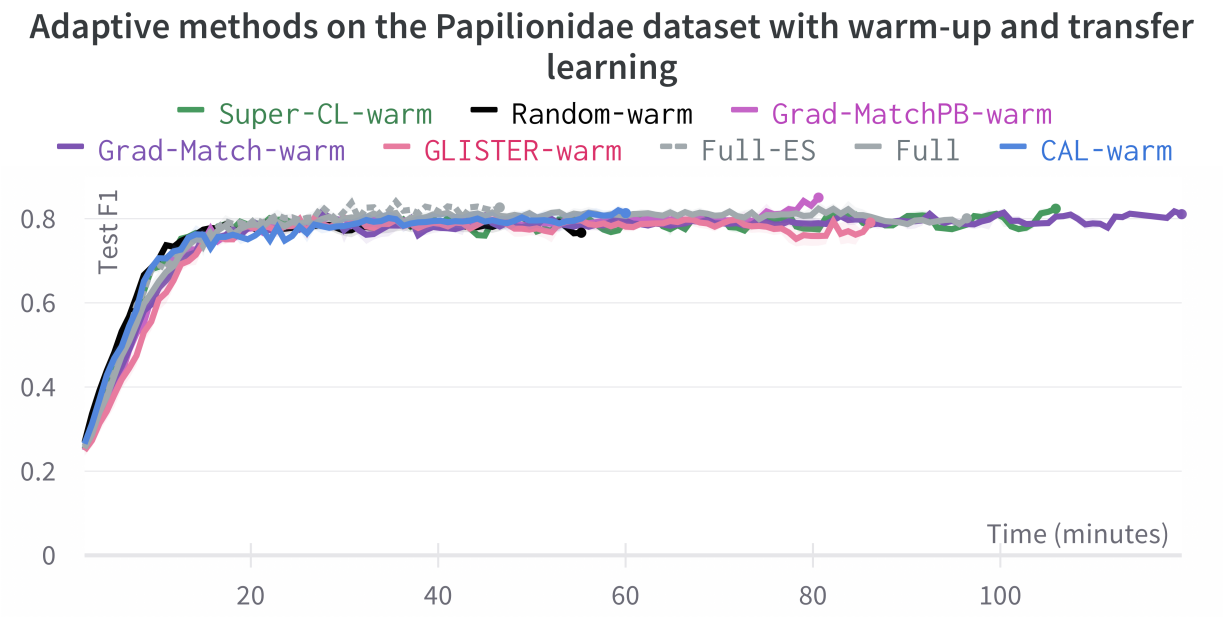

Figure 38: Test performance on the Papilionidae dataset while training with a selection of adaptive DSS methods, using pre-trained networks and warm-up. All methods are averaged over 5 runs and the standard error is shown as error band.

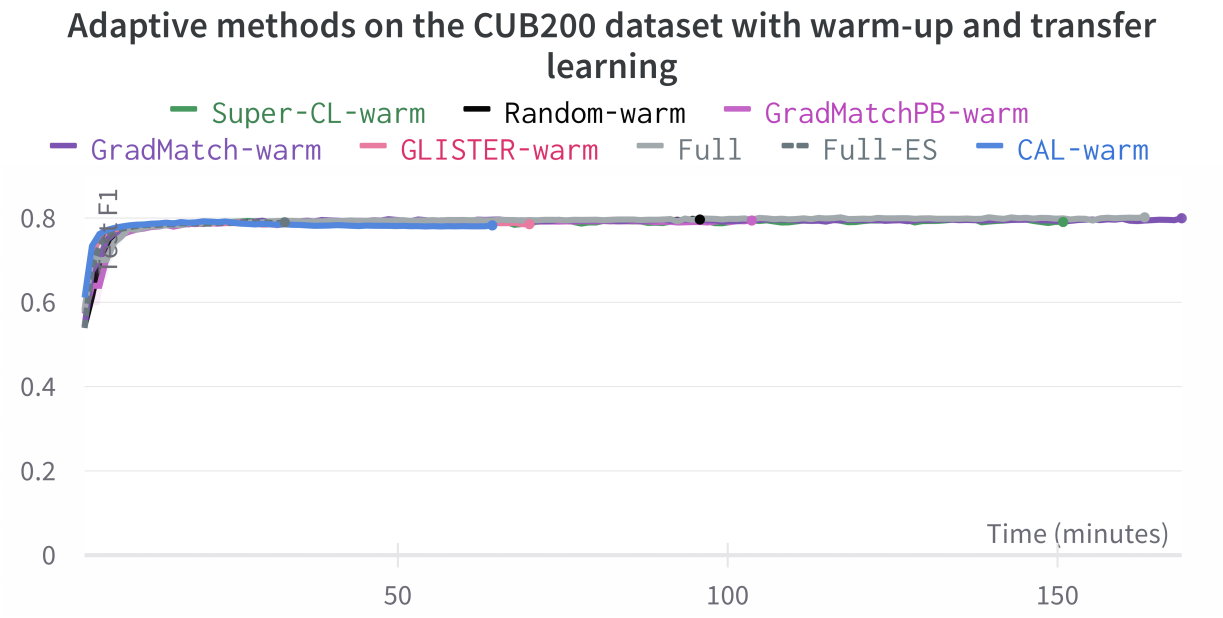

Figure 39: Test performance on the CUB200 dataset while training with a selection of adaptive DSS methods, using pre-trained networks and warm-up. All methods are averaged over 5 runs and the standard error is shown as error band.

E Papilionidae Dataset Class Distribution

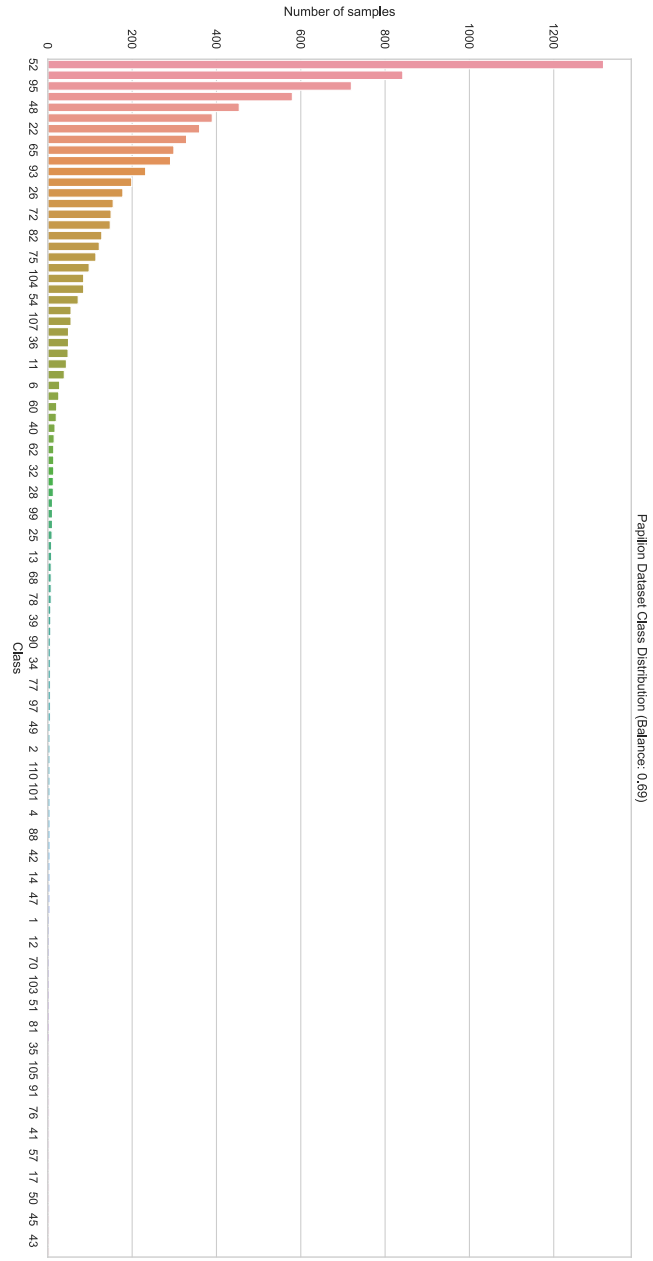

Figure 40: Class distribution of the Papilionidae dataset, showing the long-tailed distribution property of the dataset.