Masterarbeit

Name: Vivek Prabhakar Chavan

Matr.-Nr.: 404543

Kurzthema: Green Incremental Learning

Betreuender Assistent: Simon Storms, M.Sc.

Externe Betreuer: Paul Koch, M.Sc.

Dipl.-Ing. Clemens Briese

Aachen, den 13th October 2022

Diese Arbeit wurde vorgelegt am Werkzeugmaschinenlabor WZL,
Lehrstuhl für Fertigungsmesstechnik und Qualitätsmanagement.

Masterarbeit

Name Vivek Prabhakar Chavan

Matrikelnummer: 404543

Thema: Green Incremental Learning

Introduction and Context:

This thesis project investigates using Machine Learning and Green Engineering principles to facilitate incremental learning in an Artificial Intelligence (AI) based framework with a continuously growing dataset. It originates from the EIBA project, currently being carried out jointly at Fraunhofer IPK, TU Berlin, and Acatech (National Academy of Science and Engineering) [1]. The research project is funded by the Federal Ministry of Education and Research of Germany (BMBF); in the framework of ReziProK (project ID 033R226). The aim is to assess the use of AI to accurately sort production parts at the end of their lifecycle and re-purpose them for re-manufacturing, thereby creating a sustainable circular economy and reducing wastage [2].

Scope of the Thesis work:

1. **Incremental Learning:**

   A traditional Supervised Machine Learning framework requires labelled training data to be available before the training. This approach is sufficient for applications where a sufficiently representational dataset is readily available for curation. However, it is not feasible for many industrial applications, where the dataset gets continuously appended as new data is acquired. This scenario is known as Incremental Learning. Related work shows that when a trained model is progressively retrained on new unseen data, its performance on the old data worsens. This project will investigate the balancing of model plasticity (learning from new data) and rigidity (retention of previous knowledge), focusing on industrial use cases. Most published literature presents a relatively static application of class incremental learning, where a fixed amount of new data (and classes) is introduced to the model incrementally. However, this does not reflect a realistic scenario where new data might be available to the model at discontinuous increments and varying rates. This thesis will explore the latter scenario and work on the state-of-the-art approaches in this context. It will also investigate the effectiveness of incremental learning on datasets of varying complexity.

2. **Dataset Sampling and Pruning:**

   As the size of the dataset increases, it becomes increasingly challenging to store and handle. This thesis will explore the dataset sampling methodologies and aim to develop an efficient and reliable approach for image datasets. The aim is to maintain the same distribution and variance as the original dataset while minimising the loss of information and redundancy.

3. **Green AI:**

   Green AI represents a paradigm shift in AI research, where an emphasis is placed on the sustainability of implementation and application in addition to minimising the prediction error. A few approaches to quantifying and comparing the energy consumption of AI operations have been discussed in published literature. However, these metrics are only a rough estimate. This thesis work will utilise a cross-referential system to compare and contrast these metrics to the actual energy consumption for the Machine Learning operations.

Project Metrics:

The following metrics shall be studied and monitored during the project.

- Accuracy of classification (Top-1 accuracy) for the dynamic incremental learning scenario.
- Efficacy of the dataset sampling approach (with regards to maintaining the original distribution and variance)
- Energy Consumption and overall sustainability of implementation.

Keywords: Machine Learning, Incremental Learning, Data Sampling, Transfer Learning, Computer Vision, Green Engineering.

Univ.-Prof. Dr.-Ing. Christian Brecher

I Contents

II Abbreviations and Acronyms

Abbreviation/Acronym	Description
AI	Artificial Intelligence
ANN	Artificial Neural Network
CIED	Circular Economy Initiaive Deutschland
CIFAR	Dataset produced by the (formerly) Canadian Institute for Advanced Research
Class-IL = CIL	Class Incremental Learning
CNN	Convolutional Neural Network
Dino	Self-distillation with no labels (developed by Meta AI)
FLOP	FLoating Point Operation
IPK	Fraunhofer Institut für Produktionsanlagen und Konstruktionstechnik
KD	Knowledge Distillation
MAC	Multiply–Accumulate Operation
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PCA	Principal Component Analysis
ResNet	Residual Neural Network
SD	Self Distillation
Top-1 Accuracy	Accuracy of classification based on the first model prediction (with the highest probability), averaged for all classes
t-SNE	t-distributed Stochastic Neighbor Embedding
ViT	Vision Transformer

III List of Figures

IV List of Tables

V List of Charts

List of Algorithms

1 Introduction

Artificial Intelligence is a pivotal technology for industrial applications and is being extensively applied in manufacturing and production. For example, it is estimated, that by 2025, all Bosch products will contain AI or at least be developed and produced using AI [1]. It is essential to not only ensure optimal integration of AI in the production of new components, but also to incorporate it for components during and after their service period.

1.1 Project Background

One way to ensure optimal use of resources is to recycle and reprocess the manufactured parts at the end of their life cycle. The Circular Economy Act [3] assigns a high priority to remanufacture. The process of collecting and sorting end-of-life parts should involve identifying and assessing the condition of each component. Such an operation would be challenging since there tends to be a large variety in terms of the appearance of the parts, which could be further augmented by dirt, soil, rust and other factors. Thus, individual expert inspection is needed, otherwise many parts could be mistakenly discarded. Each year, roughly seven per cent of such parts are misidentified and incorrectly discarded. An accurate assortment of the parts requires product expertise, which cannot be readily scaled up for an ever-increasing quantity of parts. To carry out such an operation at a large scale, an automated system is required that can sort the components with high accuracy and flexibility [5].

The thesis originates from the EIBA project [2], which is being carried out jointly by IPK, TU Berlin and Acatech (National Academy of Science and Engineering). The research work is funded by the Federal Ministry of Education and Research of Germany (BMBF) in the framework of ReziProK (project ID 033R226) [3]

The aim of the EIBA project (Sensory detection, automated identification and evaluation of end-of-life parts based on product data and information about previous deliveries) is to develop a Machine Learning based solution for identifying and assessing the condition of old components [4]. Using AI methods such as machine learning, the system should be able to see and recognise products and compare them with other available information. Humans are not to be replaced by machines, but supported by them instead. The core innovation is enabling cooperation between humans and machines to potentiate the advantages of both and to overcome the obstacles and difficulties in sorting and decision-making. Aspects of social and ecological sustainability are constantly considered in the development of the system [5]

This will make a significant contribution to closing the loop through digital technologies. By continuously expanding the data, it should also learn new products and thus adapt to new requirements. The resulting system is analysed according to aspects of sustainability, viz. the changes introduced to human operation, additional environmental changes that are initially

created by the use of machines, and the resulting sustainability and economic benefits gained through increased efficiency [3].

Figure 1.1: A Prototype of the EIBA System. The component is placed on the work desk and is visually inspected. The trained NN model then classifies the part, inspects its state and returns a confidence score. © Fraunhofer IPK/ Larissa Klassen [6]

Engineers from different disciplines are working together on the EIBA project. Fraunhofer IPK focuses on the image-assisted recognition of products. The aim is to maintain a balance between accuracy and the incurred costs. Other existing information about the products and their added value for identification is analysed by TU Berlin and transferred into a common database. C-ECO bundles the knowledge gained and implements it in a process suitable for the industry. The impact of the system on sustainability aspects is quantified by TU Berlin. Acatech makes the project results visible to other industrial sectors by asking for their requirements at the beginning of the project and is responsible for jointly discussing the results at the end [3].

The key focus of the EIBA project is:

- Use of AI for accurate part identification and assorting
- Effective integration of new technologies with existing work practices (human operation, manufacturing setup, etc.).

- Sustainability of implementation (in terms of operating time and carbon footprint)

1.2 Problem Definition

This thesis work focuses on the image classification problem for the EIBA project. Modern Deep Learning and Computer Vision approaches have proven to be highly effective at object detection and classification. When it comes to a standard deep learning project pipeline, the traditional approach expects that the data required for training the model is available before the training. This ideal scenario is referred to as batch training or joint training in established literature. In case more data becomes available after the training, the model can be subsequently trained, which is referred to as online training. In either scenario, the number of classes that the model deals with and the feature space for each class do not change. Thus, the trained model can be used for predicting images with high accuracy, as shown in 1.2.

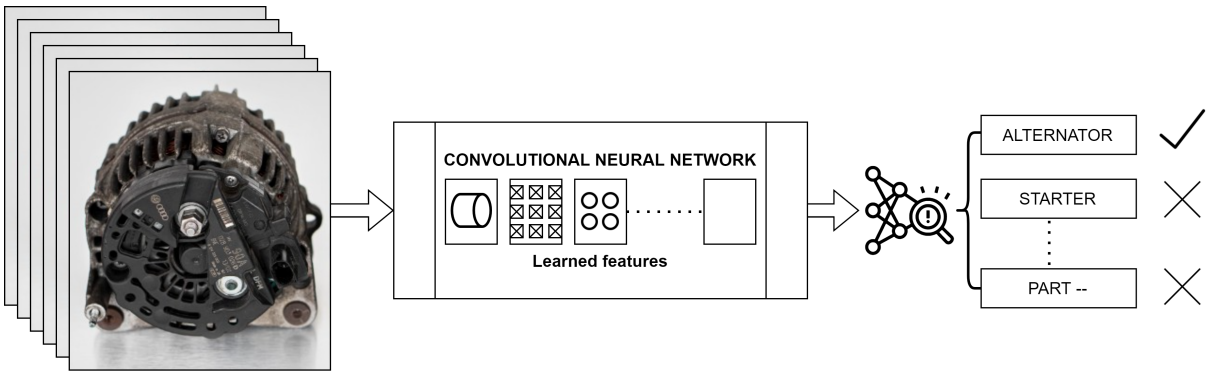
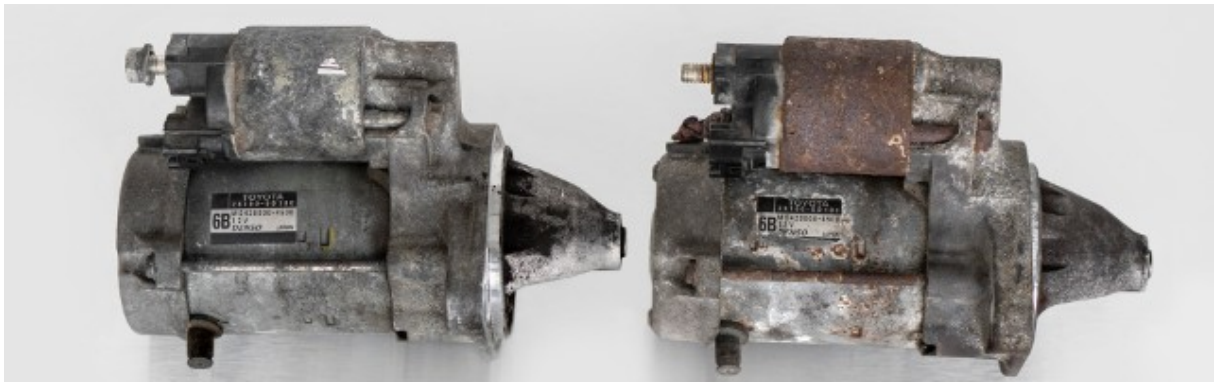
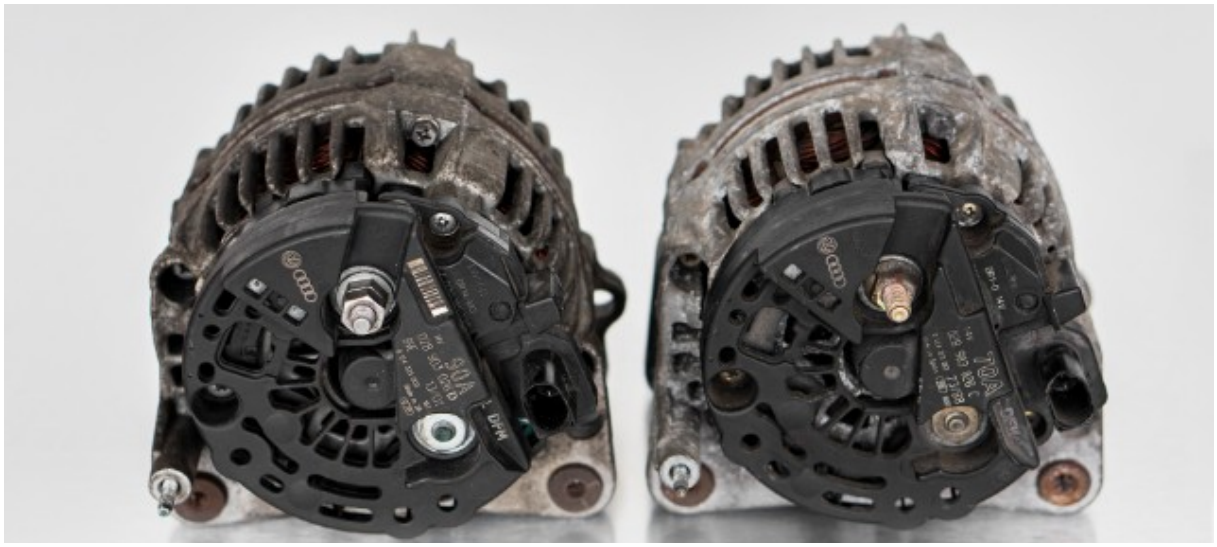


Figure 1.2: Framework of a classical ML Computer Vision project. Once a model is trained, it can be used to classify objects.

However, with long-term industrial applications, the amount of used production parts increases over time. This corresponds to an increasing number of object classes that the ML model needs to learn, as well as presents a shifting of the feature space for the different objects. This can be understood by the examples presented in Figures 1.3a and 1.3b. 1.3a represents a scenario when there is a change in the features of the object, even though the object class/label remains the same. The part to the left was digitalised first. The part to the right was received later and is a part with more external damage (rust, wear, and tear). When the two specimens are appended, the ML model must learn to include the features from both specimens to make a classification prediction on future objects. 1.3b represents a scenario when the two objects look similar, but have different object labels; hence, an ML model must incrementally learn the new object class and learn to differentiate between the two objects accurately. Researchers observed that when ML models are retrained on new unseen data, they tend to overwrite the learnt features from the previously seen data. The result is that the model can classify the newer data with much higher accuracy than the old data. This phenomenon is referred to as Catastrophic forgetting and is explored in more detail in 2.1. A robust and efficient way of incrementally adding new industrial



(a) Two components with the same label, but different appearances.



(b) The two components look visually similar, but belong to different generations and have different target labels.

Figure 1.3: Sample images captured using the EIBA prototype. © Fraunhofer IPK/ Larissa Klassen [6]

objects into the EIBA ML system needs to be developed. Incremental learning is a domain of ML that deals with such scenarios. Different Incremental Learning scenarios and approaches have been proposed by researchers, and it is incumbent to select and apply the pertinent ones to the EIBA environment. This is one of the objectives of this thesis.

Secondly, as more and more data is successively generated, it becomes likelier that the dataset will contain bad quality data (that does not represent the target object properly, for example, Figure 1.5b), or redundant data (similar images, for example, Figure 1.4). Thus, a data management approach that can allow the ML system to identify the characteristics of the data and make decisions is required. This is the second objective of the thesis.

Along the lines of the EIBA project, sustainability of implementation is an important factor when developing the approaches in the thesis work. This can broadly be labelled as **Green**

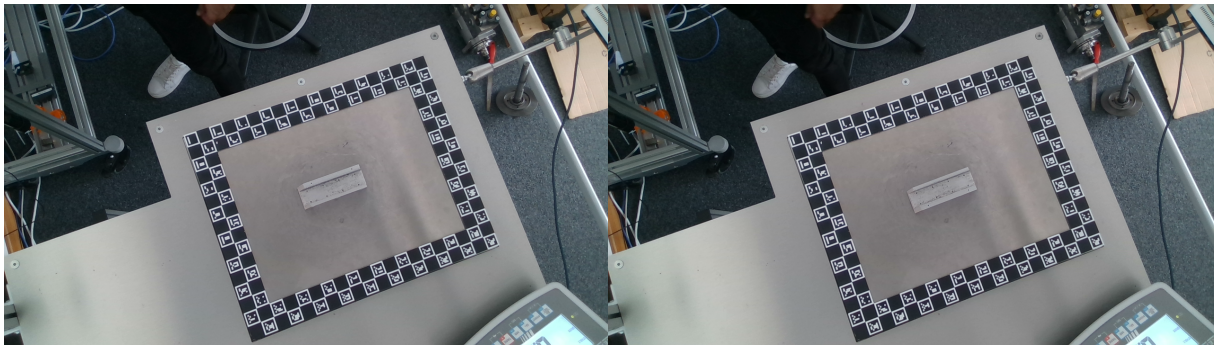


Figure 1.4: Two images captured with different object rotations; since the object is symmetric, the two images appear identical. © Fraunhofer IPK

(a) The object is satisfactorily captured.

(b) The object is cropped and the operator's hand is accidentally captured.

Figure 1.5: Sample images captured by workers in the lab. © Fraunhofer IPK

Engineering or **Green AI** in the context of ML projects. The exact characteristics and properties of a **Green** implementation w.r.t. an ML framework is introduced in detail in Section 2.3 and explored further in 6.3.

1.3 Motivation

Based on the areas of focus for the EIBA project, it is essential to develop AI technologies further such that they are closer to human perception. Modern Machine Learning approaches are broadening in terms of their scope and are becoming more accurate and capable. However, one of the key ways in which AI/machine capabilities remain inferior is the ability to learn and relearn information [14]. For example, a young student may learn arithmetic as a child, then later learn trigonometry and calculus, such that each of these learnings builds on top of the other. The student must have a strong foundation in arithmetic to then learn trigonometry and subsequently deal with calculus. A pertinent example in terms of vision intelligence would be learning to recognise newer objects without forgetting previously seen objects. A child might learn to identify cats and dogs and learn about elephants the next day. The child will continue

to remember cats and dogs, and will now also have added knowledge about elephants. This scenario for AI systems is discussed in more detail in the next chapter in 2.1.

The second key area in which human intelligence is far superior to machine performance is in terms of using past data to successfully identify patterns in data and make generalisations and apply them to other applications. Thus, humans can work on a wide variety of applications by transferring the knowledge gained from previous experiences onto new tasks. Transfer Learning is the domain of AI that utilises the weights from a model trained on one task for other tasks (either as weight initialisation or fine-tuning). Within the framework of this thesis, transfer learning will be useful for deriving inferences about the image dataset without needing to train an ML model from scratch first; it is discussed in detail in 2.2.

Thirdly, it is well established that the human brain is extremely energy efficient when it comes to its operation. It is estimated that the human brain consumes about 20% of human calorie consumption, which is only about 400-500 Calories per day [7]. This is the area in which AI research and development severely lacks human performance and is discussed in 2.3.

An AI system designed to help industry experts in making accurate assessments and quick decisions requires a synergy between human control and machine operation. Current state-of-the-art would result in a performance bottleneck on part of the machine system. On the other hand, the human expert (operator/researcher) cannot engage in repetitive action for large quantities of data for industrial applications, which makes using computational systems a necessity. Therefore, AI research in this domain requires addressing all three areas successfully.

2 Literature Review and Background Study

This chapter provides a technical foundation of the key areas of interest along with the existing research gap that guides the thesis work.

2.1 Incremental Learning Scenarios

Incremental Learning (also known as Continual Learning or Lifelong Learning in other contexts) is a Machine Learning Paradigm that deals with the accumulation of knowledge from previous tasks and learning incrementally [14]. While this concept may seem natural and obvious to humans, the current traditional Machine Learning projects involve learning in isolation, and expanding the scope of the project isn't straightforward. Specifically, in terms of Neural Network applications, this phenomenon is well studied and hypothesised [13]. For traditional CNN-based architectures, the number of classes that the network trains on is fixed [11]. If a new object is presented to the network, it wrongly predicts a previously learnt class; unlike humans, Neural Networks are not good at recognising the limitations of their capabilities [70].

2.1.1 Catastrophic Forgetting and the Plasticity-Rigidity Dilemma

Catastrophic Forgetting (also known as Catastrophic Inference) refers to the tendency of Neural Network architectures to forget previously learnt information upon being trained on new data [13]. This happens because the weights and biases saved from the extracted features get overwritten by newly learnt features. In the context of image classification, such an architecture tends to have much higher accuracy for the newly learnt objects compared to the previously learnt objects. Based on established literature, the drop in accuracy for the older classes tends to be significant, the more the exposure to the newer data [15]. In this case, the model can be said to be very *plastic*, in that it readily adapts to the new data and completely forgets the old data. The other extreme scenario would be a *rigid* model, where the parameters of the model are frozen to avoid forgetting older classes. However, such a model fails to learn new incoming data. All incremental learning approaches aim to resolve this dilemma and strike a balance between model plasticity and rigidity [17].

2.1.2 Joint Training and Retraining

To tackle the problem of catastrophic forgetting in Neural Networks, a naive solution might be to append all the data together and train the model from scratch on the full dataset. This is referred to as *joint training* (or batch learning in other contexts) [18]. The issue with this naive approach is that the amount of information available might increase continuously throughout the lifecycle of the project and carrying out full joint training at each step might not be feasible [25]. Another concern is that for some applications, the training data might be comprised of sensitive

and private information from customers, so storing it for future use might not always be an option [19]. In the framework of the EIBA project, this is not expected to be the case. However, as previously discussed, with a pool of continuously increasing datasets, it is necessary to prune the dataset and only retain the most important and feature-rich samples for future use. The simplest scenario for new incoming data would be one where the total number of objects to be classified (number of classes) remains the same and the feature distribution that the image data represents does not change. The new data only adds a quantitative load on the project setup, and the model can be retrained on the new data without any catastrophic forgetting. This scenario is referred to as *Online training* and can be used to progressively train the NN model to improve its overall accuracy; this scenario is, in principle, similar to transfer learning with pre-trained models [12].

2.1.3 Incremental Learning

Based on the scope of the new incoming data and the project requirements, the incremental learning scenario might be different and offer varying degrees of challenges [25]. Each set of data can be represented as a separate task. Hence, the initial (joint) training can be referred to as Task 0, the first increment thereafter as Task 1 and so on [55].

Van de Ven and Tolias (2019) [25] present three scenarios for incremental learning, which have been elaborated on below with the following example: Consider the simplified case, where 10 car parts (10 classes in total) are to be classified; the objects are introduced to the NN model at different times. Two objects are introduced into the dataset during each task.

Table 2.1: Example dataset: Car parts

Task 0	Task 1	Task 2	Task 3	Task4
Tyre	Fender	Radiator fan	Mirror	Headlights
Front Bumper	Rear Spoiler	Condenser	Fender	Tail lights

2.1.3.1 Task Incremental Learning

Task incremental learning is the simplest incremental learning scenario. The task ID is available during training as well as testing/prediction. Hence, the model would need to make an accurate prediction out of all the classes present in that particular task. E.g. given task 1, the model needs to decide whether the object is Fender or Rear Spoiler. Task-specific ML approaches such as packNet [56] determine the mask map of convolution filters, when given a task ID, the corresponding mask is selected to make the prediction. In the framework of the EIBA project, this can be represented by the example in Figure 1.3b, where the object manufacturer and type are known, and the model has to decide the Model Year of manufacture.

2.1.3.2 Domain Incremental Learning

Domain-IL adds more complexity to the learning, in that the task ID is not available during the test/prediction phase, but the model must solve the task at hand. For example, the model needs to predict whether the object was the first class (Tyre, Fender, Radiator fan, Mirror, Headlights) or the second class (Front Bumper, Rear Spoiler, Condenser, Fender, Tail lights), regardless of the task ID. Domain incremental learning often represents a shift in the input feature distribution in the dataset, even though the prediction space remains the same.

Task and Domain IL problems can be solved using parameter regularization-based methods, such as Elastic Weight Consolidation (EWC, 2017) [14]. The approach selectively slows down learning on the weights for the older tasks, such that a balance between the accuracy of prediction between tasks is obtained.

2.1.3.3 Class Incremental Learning

Class Incremental Learning (CIL) represents the most challenging scenario for continual model learning. The model is required to correctly classify the object regardless of the task ID or the sequence in which the classes were introduced to the model. For example, in the car parts dataset, the model would need to predict the object type from the 10 classes. In future, if 5 more classes are introduced, then the model is needed to correctly classify the object from 15 classes in total. CIL is a generalised case of incremental learning, in that online learning, Task-IL and Domain-IL are more specialised cases of class incremental learning.

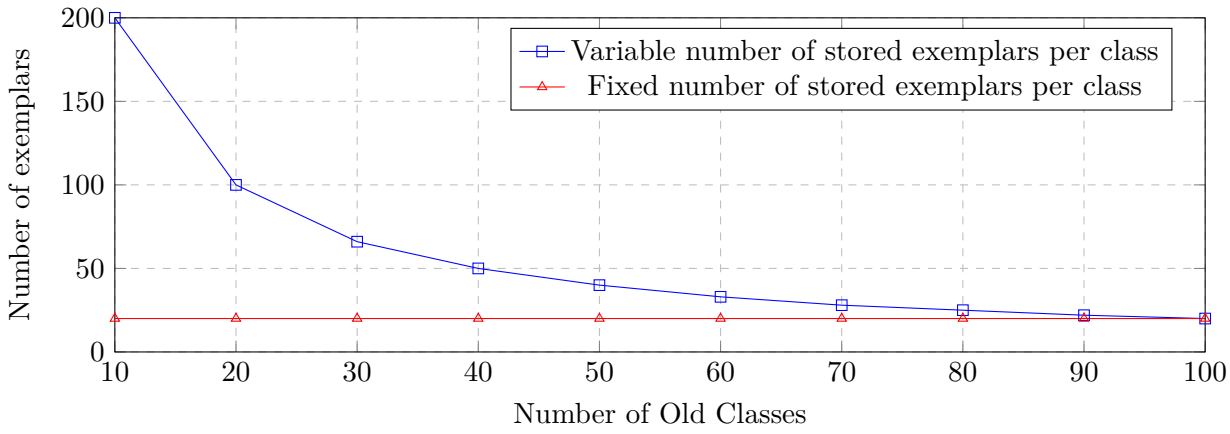
Several proposed methodologies for CIL involve knowledge distillation (KD), where the learning from a larger model (teacher) is transferred to a smaller (student) model [24]. The loss function of KD is the KL divergence that is used to minimise the model output on the teacher model and the student model for the given dataset. Learning without Forgetting (LwF, 2015) [57] was one of the pioneering works for incremental learning. The LwF method uses knowledge distillation to provide soft supervision information of the old classes to tackle catastrophic forgetting. The authors showed that this greatly improved the accuracy for the old classes even without the data from the new classes. They also compared their approach against finetuning (baseline method where the model is retrained on the new data, leading to catastrophic forgetting of the old classes), joint training and feature extraction methods. LwF uses additional memory overhead to save the old model; and during the next task, the old model is used as the teacher model. However, the performance of LwF on CIL scenarios remained poor.

Rebuffi et al. (2017) [19] introduced iCARL (Incremental Classifier and Representation Learning), which is a cornerstone for several subsequent studies on CIL. It introduced the concept of a memory buffer for storing exemplars (a fixed number of images from the previous classes that are introduced during each following task) to tackle the challenge of new incoming classes. It

also introduced *Herding*, a sampling strategy for selecting the exemplars. Herding is a greedy selection approach that uses the trained model to derive feature vectors for the dataset and make the feature mean of the selected dataset match the total mean for each class. The authors introduced an artificial limit of 2000 total exemplars with each class having an equal number of images.

For example, Chart 1 depicts a case where the initial joint training has 10 classes. Hence, during the next task, 2000/10 = a maximum of 200 images per class shall be stored. During the next task, 2000/20 = 100 images and so on. Another approach to storing data is to have a fixed number of exemplars per class, irrespective of the number of task increments. The latter approach has a disadvantage in that the earlier increments are unable to utilise the full memory capacity. On the other hand, it obviates the subsequent step to remove exemplars to make more space for new classes before every incremental training.

Chart 1: Number of exemplar images stored during incremental learning tasks



Herding is used in several other CIL approaches and remains a state-of-the-art approach. Some authors ([21], [49]) have compared herding to random exemplar selection and found no statistical difference between the two methods. Several other publications ([47], [44], [23]) report that herding provides a clear gain in accuracy over random exemplar selection. Van de Ven and Tolias (2019) [25] state that storing exemplars from old classes is necessary to ensure good performance on Class Incremental Learning tasks.

iCARL proposes a nearest-mean-of-exemplars (NME) classifier using retained exemplar data rather than using a linear classifier directly from the training phase. Since only a limited number of exemplars per class are stored, the dataset remains unbalanced during incremental tasks; which is addressed using NME. Although the work presented by Rebuffi et al. remains highly relevant to the problem of CIL, several benchmarks and survey studies ([15], [18], [55]) show that the newer approaches outperform iCARL in terms of incremental accuracy.

Masana et al. (2020) [18] present one such survey, where they classify CIL methods into three

general categories:

- Regularisation-based approaches aim to minimise the impact of the introduction of new classes by controlling the weights that are important for the previous tasks.
- Exemplar-based approaches, that store a limited number of selected samples to prevent the catastrophic forgetting of old tasks.
- Approaches that directly address the task-recency bias in incremental learning.

Most state-of-the-art methods employ multiple solutions in tandem to obtain higher accuracies. The authors noted that finetuning and storing exemplars together provide a good starting baseline for several experimental settings. They also found that the network architecture has a great impact on the performance of the CIL methods. In particular, the presence or absence of skip connections (ResNet architecture [16]) is a significant influence. The results obtained from their benchmarks are discussed in detail while developing the research methodology of the thesis in 6.

BiC ([42], 2019) follows the training paradigm of iCARL. Additionally, it uses a linear classifier for making predictions. The authors emphasize that class imbalance (or task-recency bias) has a significant impact on incremental learning performance. They introduce a bias correction phase, where a linear offset is implemented to flatten and translate the distribution between new and old classes to minimise the class imbalance.

The task-recency bias was also addressed by Hou et al. (2019) [44]. They introduced a method, LUCIR (Learning a Unified Classifier via Rebalancing). They propose to replace the standard softmax output layer with a cosine normalisation layer. They also employ a margin ranking loss to prevent new classes from occupying a similar location as classes from older tasks. The loss works by maximising the distance between the current class embeddings to the older classes that are most similar to it. According to [18] (2020), LUCIR performs better when having a larger first task followed by smaller new tasks. While BiC tends to overcompensate toward previous tasks, LUCIR shows a more gradual task recency bias and maintains a better performance.

Zhao et al. (2020) [52] also addressed class imbalance as a core problem of CIL. The authors also found that since the older classes maintain lesser exemplars, the model is automatically incentivised to predict newer classes with higher probabilities due to the high bias in the last fully connected (FC) layer. With their method, WA (Weight Alignment), they aim to achieve alignment of classification preference between old classes and new classes. Additionally, they also employ KD to minimise the discrimination within old classes (similar to iCARL), thus ensuring minimum discrimination within older classes and maximum fairness between older and newer classes.

Douillard et al. (2020) [47] proposed PODNet (Pooled Outputs Distillation for Small-Tasks Incremental Learning), specifically aimed at long runs of incremental learning tasks. They incorporate a spatial-based distillation loss. along with representation learning for dealing with catastrophic forgetting in older classes. The POD loss aims to balance reducing the forgetting of old classes and learning of new classes. They also introduce a multimode similarity classifier to introduce more robustness to shifts in distribution during incremental learning.

Mittal et al. (2021) [21] presented CCIL-SD (Compositional Class Incremental Learning: Essentials for Class Incremental Learning), where they identified underlying reasons behind catastrophic forgetting and poor learning during CIL tasks. They observe that dark knowledge (secondary information) encapsulates the semantic relationship between the target and non-target classes. They also introduce a compositional CIL system where multiple strategies are employed in tandem to ameliorate catastrophic forgetting and improve learning. They conclude that the quality of feature representation during the initial joint training has a significant impact on incremental learning in subsequent tasks. Hence, representation learning is a promising avenue for developing CIL methodologies further.

Dynamic feature structure is a common approach for Task-IL; Yan et al. (2021) [23] implemented it for Class Incremental Learning with a lot of success in their approach- DER (Dynamically Expandable Representation for Class Incremental Learning). The authors claim that the optimal solution to the plasticity-rigidity dilemma is to introduce separate feature extractors for each new task. The extractors from the previous classes can be kept frozen, making only the extractor of the current task plastic learn from the incremental learning. For making predictions, the different feature extractors are appended together into a single classifier. They also incorporated a pruning method with a sparse solution loss that enables parameter reduction while also maintaining model performance.

Along similar lines, Wang et al. (2022) [65] also introduce their approach – FOSTER (Feature Boosting and Compression for Class-Incremental Learning), which aims to introduce a dynamic model architecture along with feature compression. They claim that methods such as DER introduce a significant computational overhead with an increasing number of tasks and also add noise to the representation of new classes, thus harming their accuracy. FOSTER compresses the learning from all the previous tasks into a single feature extractor after each task, thus aiming to minimise the number of parameters and the computational cost.

New approaches are being introduced, that incorporate the use of Transformer architectures for CIL problems. Following the observations by Mittal et al. (2021) [21], this is a very promising avenue for getting better feature representation during training. Douillard et al. (2022) [53] and Deng et al. (2022) [54] are two such approaches that claim to match and even surpass current state-of-the-art performance. These approaches, however, require a significant initial computational investment for the training of the Transformer model. Other approaches such as

the one by van de Ven et al. (2021) [51] introduce other experimental approaches to solve the class incremental learning scenarios and are not discussed further due to their infeasibility for the project requirements of EIBA.

In addition to the aforementioned strategies, all the methods also employ a gradually reducing learning rate during each task, augmentations on the images and introduce additional forgetting loss parameters that are minimised during the training [18].

2.2 Dataset Sampling

As previously stated, the purpose of dataset sampling is to prune and manage a continuously growing dataset. In the context of Deep Learning projects, it might be appealing to gather and train with as much data as possible, however, this becomes challenging due to storage and data security concerns. Secondly, the larger datasets would also lead to an increase in training times and computational resources. Moreover, the image dataset might contain poor-quality images (where the object might be out of focus, obstructed, or cropped) or images that are very similar to each other, leading to redundancies. Unlike publicly available curated datasets like CIFAR [8] or ImageNet [9], the data in the EIBA projects are industry specific and dynamic.

The simplest and the least time-consuming solution would be a random selection of images. This approach works well in datasets like ImageNet, where the dataset consists of several publicly sourced images and each image provide a different context. This has also been found to be useful for Class-IL applications [49].

The exemplar selection process used in Class-IL solutions can be expanded and implemented for further general use. Herding is the most popular exemplar selection approach and has been reported by several researchers to provide better results than random exemplar selection. The limit of 2000 exemplars for all classes is an artificial restriction and need not apply for general data management purposes.

Liu et al. (2020) [37] propose Mnemonics training, a method that parametrizes exemplars and makes them optimizable. The authors compared their method with random selection and herding. They found that while herding selects samples in a greedy manner (exemplars are nearest neighbours of the mean for each class), the mnemonics approach tends to find samples that are located on the boundaries between the different classes. The training approach presents a Bilevel Optimization Problem (BOP) consisting of model-level and exemplar-level optimizations.

Liu et al. (2022) [64] also expounded that herding is a suboptimal and relatively static approach to select exemplars for class incremental learning. They introduce RMM (Reinforced Memory Management for Class-Incremental Learning), with a dynamic memory management strategy by utilizing reinforcement learning. They train the policy of the reinforcement learning model

on pseudo-CIL tasks built on the initial (zeroth) task and are later applied to new tasks. They use LUCIR and PODNet as baselines along with combining them with AANets [22] and add their exemplar selection method during incremental learning.

However, the sampling methodologies discussed above require the model to be trained on all the classes first before the feature vectors can be derived and statistically evaluated. As the number of classes increases, the model requires further training to be useful for exemplar selection. While this works in the context of pure Class-IL problems, it fails to provide useful insights about the data without substantial computational investment. A straightforward solution to this could be using models pre-trained on the ImageNet dataset [12] to provide a good starting point.

Recently, Vision Transformers have been shown to provide excellent results on various benchmark datasets with little or no prior fine-tuning. For example, OpenAI's CLIP (2021) [63] model outperforms pre-trained ResNet models on the ImageNet dataset and also proves to be generally resilient to domain shifts in class features. Meta's Dino, developed by Caron et al. (2021) [60] is also able to visualize attention with remarkable precision and parse general meaning and context behind image and video input. The image is divided into smaller patches, which are flattened; and the relation between the patches is encoded via a self-attention mechanism. The authors show that self-supervised ViT features contain information about the semantic segmentation of images, which outperforms supervised ViT models. The Dino model is interpreted as self-distillation with no labels [60]. The use of these vision transformer models for dataset sampling is discussed in more detail in the next chapter.

2.3 Green AI

Green AI represents a new paradigm in AI research. Most of the cutting-edge research in AI (especially Computer Vision and NLP) is carried out to maximise the accuracy metrics with no emphasis on the practical applicability and operation cost [35]. Thus, many high-end innovations (esp. those that surpass human benchmarks) are so computationally expensive that they cannot be replicated independently by research facilities. For example, GPT-3, the language model from OpenAI is estimated to have produced the equivalent of 552 metric tons of carbon dioxide during its training and cost approximately USD 20 Million in electricity bills [26].

Schwartz et al. (2020) [35] have shown that the total computational cost of training a model is proportional to the product of the cost of training a single image, the total number of images in the dataset and the number of hyperparameter tuning experiments needed. This thesis has a lesser focus on hyperparameter tuning. The other two parameters are significantly more important. The cost per image of training is largely dependent on the base network architecture. Wenninger et al. (2020) [72] also expound on the need for developing ML systems that save more energy through their application than what they consume over their training period.

Several parameters can be studied to compare different AI algorithms for their efficiency (the accuracy of classification produced per increase in computational cost). Metrics such as the number of parameters in the model provide a good generalisation, but are too simplistic. Other metrics, such as energy consumption [27] and carbon footprint [29] gives a realistic estimation of the cost but are specific to the hardware used and the source of energy. Floating Point Operations (FLOPs) and Multiply-Accumulate Operations (MACs) provide an independent metric that allows the performance to be tested across different setups (MACs = 2 * FLOPs) [35].

Conventionally, it has been challenging to accurately measure the energy consumption of machine learning algorithms. García-Martín (2019) [38] expounded on some approaches for measuring the energy requirement of some traditional ML algorithms. However, since several Class-IL approaches deal with a dynamically growing network architecture and an increasing number of model parameters, it is challenging to develop a simple algorithm-based strategy for reporting these metrics. Ideally, the power consumption of an electronic system can be measured by using a dedicated energy metering device [58], which would give the most accurate results and reflect a practical use case. The power consumption can be logged and summed up over the period of the computation to yield the total units of energy consumption (kWh) and resulting energy cost [71].

2.4 Research Gap

The established body of literature shows that class incremental learning scenarios have been studied extensively for static conditions. The implementations have not been tested thoroughly under conditions of varying increments and data availability. Moreover, it is not known whether performance on a publicly available dataset such as CIFAR-100 or ImageNet would directly translate to other application-specific data. Most published literature focuses on the accuracy metrics of CIFAR and ImageNet datasets and attempts to exceed the previously established highest incremental accuracy score for a predefined task increment sequence. Several such sequences start with 50% of the total classes during the initial joint training phase (50 for CIFAR-100 and ImageNet100, 500 for the full ImageNet dataset) and introduce a constant number of classes (5/10/100) for each new task [15]. The exemplar memory storage is also taken to be constant and static, and model performance under different data availability conditions has not been evaluated. While this allows for easy benchmarking and comparison of performance, such conditions cannot be guaranteed on practical industrial projects. Thus, a thorough study of these methods with supplementary evaluation metrics is necessary.

The other major research gap is the energy consumption analysis for incremental learning. The authors of the original implementations do not publish any data on the energy consumption; and the training time/computational cost for other methods. A further deeper dive into the computational cost change with an increasing number of class increments being appended is

also lacking. Since saving in training time and computational cost is the key reason behind the use of incremental learning approaches, this addressing this research gap is crucial. This also becomes increasingly important in light of the growing emphasis on reducing carbon emissions for industrial applications and volatility of the energy sector [71].

The use of sampling strategies for dataset pruning and analysis is a somewhat less explored area based on the body of published work. Exemplar selection is a crucial aspect of Class IL methods, however, they do not lend themselves to effective use in other areas. An efficient way of managing large quantities of image data is needed, which remains unaddressed. Particularly, when the amount of data exceeds a certain storage limit, the data needs to be pruned. In particular, bad-quality image data and redundant data (as elaborated with Figure 1.5) need to be flagged. It is not well-established how state-of-the-art approaches like herding and mnemonics deal with redundancies or outliers when selecting exemplars for incremental training. Secondly, for conducting a preliminary analysis of the data, it is essential to encode the data as feature vectors for further computation. Rather than being an academic research topic, this is a unique requirement for large projects on an Industrial scale.

Lastly, given that the EIBA project is an industrial research project, its emphasis lies in maximising the use of AI technologies for improving industrial processes instead of focusing on the mechanisms behind the Machine Learning implementations. In addition to the standard testing settings of Class-IL research (discussed further in Chapter 5), this thesis will also elaborate on its incorporation in a project environment (with pretrained models, project timelines and resource utilisation).

3 Research Questions and Thesis Structure

As discussed, the thesis work has three key areas of focus. Pertinent research questions relating to each of them are presented as follows. Questions that are natural offshoots/extensions of each other are clubbed together.

3.1 Incremental Learning

The class incremental learning implementations (and broadly all classification problems) generally focus on the Top-1 accuracy as a metric for comparison with other implementations. This makes it a suitable and agnostic metric for formulating the research questions. They also employ different loss functions, depending on the architecture changes; making it an infeasible parameter for independent research.

Question 1: How is the performance of the Class-IL implementations over a large number of increments? Additionally, how is the performance on previously learnt data (old classes) and incoming data (new classes)?

Question 2: How do the implementations perform when the size of the class increments is varied?

Question 3: What impact does the visual complexity of the data (such as obscuring of the object, presence of other similar objects, background clutter, dirt, consistent presence of hands/gloves) have on incremental learning performance?

Question 4: How can the exemplar data be managed to ensure optimal performance? Furthermore, can Class-IL implementations match joint training performance if the full dataset is provided during incremental training?

Question 5: What is the optimal number of increments that ought to be executed before warranting joint training?

3.2 Dataset Pruning and Analysis

In the case of a continuously growing dataset, the data is desired to be investigated. The ideal approach should allow use in various contexts and must be easy enough to understand for any operator. The most straightforward solution is to develop an approach that allows the user to visualise the distribution of data.

Question 6: What is the ideal approach for encoding the features from the image data? How can these feature vectors be projected into lower dimensions for visualisation?

Question 7: How can similar/redundant images be accurately flagged from a dataset? How can outliers in data be identified?

Question 8: For incremental learning, how can the original distribution of the entire class be distilled into the limited number of representative exemplars?

3.3 Green Engineering

For Green Engineering, energy consumption (kWh) is the most recognisable and easy-to-interpret metric. MACs/FLOPs values can also be used to understand computational operations.

Question 9: What factors does the energy consumption of the different Class-IL implementations depend on?

Question 10: How does the energy consumption change with an increase in incremental tasks and model parameters?

Question 11: How does the incremental learning compare to joint training w.r.t. energy consumption?

Question 12: What would the energy consumption of the dataset pruning strategy be? Does it result in performance gain to compensate for the increase in the computational cost?

3.4 Thesis Structure

The thesis will aim to answer all the above-stated questions and develop a comprehensive green environment for the ML framework to incrementally learn new information. Chapter 4 provides a quick overview of the data and the source code implementations used during the thesis. A thorough analysis of the currently established literature is necessary for further testing and implementation, which is presented in Chapter 5. The research questions discussed above will guide the design of the experiments and methodologies for the thesis, which have been presented in Chapter 6. The results obtained from the various experiments are discussed in Chapter 7. Based on the results and the established requirements, conclusions of the thesis work are presented in Chapter 8 and suggestions for further research work are presented in Chapter 9.

4 Data Assimilation

This chapter provides an overview of the base code implementations and the data used for training and validation during this thesis.

4.1 Open Source Code Implementations

Several authors have made their class incremental learning approaches discussed in Chapter 2 available for further research. These implementations serve as an excellent starting point for this thesis. The CIFAR100 and ImageNet datasets are often used for benchmarking performance. In addition to the individual publications, an independent account of several benchmarks for CIL problems can be found in [15].

Masana et al. (2020) [18] present FACIL (Framework for Analysis of Class-Incremental Learning), a survey of the available CIL implementations at the time. The authors provided an in-depth review of the class incremental learning as well as a framework for training, testing and comparing the CIL methods. However, the framework was published over two years ago (at the time of the commencement of the thesis work), and several new CIL implementations have been proposed since then. Researchers can independently edit and expand the setup to add the newer approaches, but this comes at the cost of initial time investment and commitment to the framework.

PyCIL (Python Toolbox for Class-Incremental Learning) is another such framework recently published by Zhou et al. (2021) [55]. The toolbox is also open source, available for other researchers and regularly updated with the latest implementations. The work is a part of a large-scale survey about class-incremental learning in China. The implementations are already discussed briefly in Chapter 2 and follow the same standard setup that was first implemented by Rebuffi et al. (2017) [19]. As previously stated, the newer implementations contain iterative improvements over the older ones and are often built on top of the previous work. This toolbox shall be used for the experiments discussed later in this chapter. Some other implementations, that are not a part of the PyCIL toolbox, were also studied and have been used.

The use of Vision Transformers for deriving feature vectors and for dataset pruning is also discussed in detail in the next section. In particular, the code for self-supervised vision transformers using Dino [60] (made available by MetaAI) has been used for further development in this thesis.

All used code was available with either an MIT license or an Apache license at the time of this work.

4.2 Datasets

Three datasets were considered for this project, two of out of which were used for the development and experimentation work.

CIFAR-100: The CIFAR-100 dataset [8] consists of 60000 images, 32×32 in size in 100 total classes. It contains approx. 50000 training and 10000 test images. The CIFAR-100 dataset is extensively used in the benchmarking for image classification work. Class-IL benchmarks are also available for this dataset with various task configurations. The advantage of using this dataset is that it has a small size, so it can be trained and tested relatively quickly with less time and computational effort. However, this dataset does not reflect an industrial project scenario. The images captured in industrial setups tend to be of a higher resolution and have other issues, as already seen in Figures 1.5 and 1.4. The CIFAR dataset does not sufficiently reflect the visual complexity necessary to apply the findings to the EIBA environment. This can also be inferred from the benchmark results on the CIFAR dataset [15]; the reported accuracies are well below the expectations from a fully functional industrial project setup. Secondly, per the research questions stated in Section 3, a thorough investigation of the energy consumption metrics must be an important part of the experimental setup. Since the size of the images does not reflect the computational load of the industrial datasets, CIFAR was not used further during the thesis.

ImageNet: ImageNet [9] is an image database organised according to the WordNet hierarchy; several nouns in the hierarchy are represented by hundreds/thousands of publicly sourced images. The full dataset contains 1000 classes, mostly of size 224×224. The ImageNet project has been pivotal for advancing computer vision and deep learning research. For example, most pre-trained computer vision models contain parameters tuned for the ImageNet dataset and serve as a basis for transfer learning.

Class incremental learning benchmarks exist for the full dataset as well as for a subset (100 classes) with various task and increment configurations. A few sample images from the dataset are shown in Figure 4.1. The ImageNet dataset covers an extensive range of objects, backgrounds and image contexts, making it a good starting point for this thesis work. Furthermore, the results from this dataset can be used to draw inferences regarding energy consumption and computational cost.

A smaller subset was constructed for training and experimentation by selecting 100 random classes (with the seed 404543). This dataset shall henceforth be referenced as **ImageNet Subset.**

Industrial-100: The Industrial-100 dataset was internally produced at Fraunhofer IPK and contains 100 objects in 20800 total images (208 images per class). The classes consist of objects that are commonly found in a production/robotics lab. Each class contains an equal number of

Figure 4.1: A few examples from the ImageNet Dataset. The images are arranged according to t-SNE embedding [9] [10]

images of the following sub-categories:

- White background: These images are taken with good lighting, with a clean white background and the object is clear, in focus and takes up most of the image.
- Clean background: These images are also taken against a clean background, however, the image resolution is purposefully low and represents downgraded and compressed data quality.
- Handheld: These images are taken with the user holding the objects, sometimes the hand also ends up obscuring a portion of the object. The background may be clean or may contain other irrelevant objects.
- Messy background: These images are taken with the object placed along with other objects from the lab in the background. In most cases, the other object does not obscure the target object.

These four categories simulate the different visual contexts in which images taken by the lab operator may capture the dataset during operation. This dataset is ideal for this thesis and to answer the research questions about the performance of different Class-IL methods and energy consumption.

Industrial-100 Imbalanced: This is a derivative of the Industrial-100 dataset with 50 classes containing only images with a clean or white background, and the other 50 classes consisting of images with a messy or handheld background. This creates an imbalance in the feature distribution and the complexity with which a CNN architecture can learn and classify the objects. Duplicate copies of the images are purposefully added to the classes to simulate a scenario where the digitalisation in a machine lab leads to redundant data.

EIBA Photo Studio Dataset (© Fraunhofer IPK): This dataset was generated by digitalising several industrial parts using the studio setup developed for the EIBA project. The object is placed on the transparent stand and images are captured against a white background with varying object rotations. This is an example of a growing industry-specific dataset; at the time of the thesis, it contains 884 objects. It will be used as the model dataset for showing a commercial application of Green Incremental Learning.

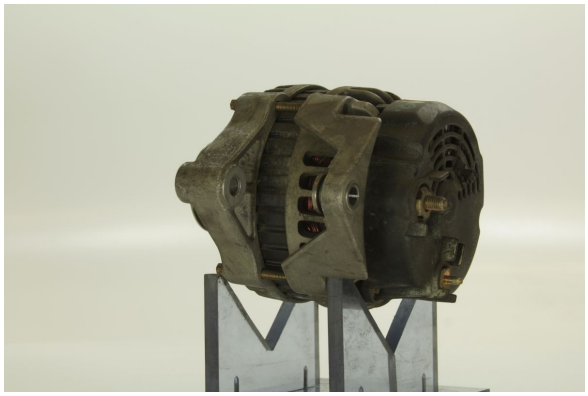
(a) White Background

(b) Clean Background

(c) Handheld

(d) Messy Background

Figure 4.2: An example of a component from the Industrial-100 dataset



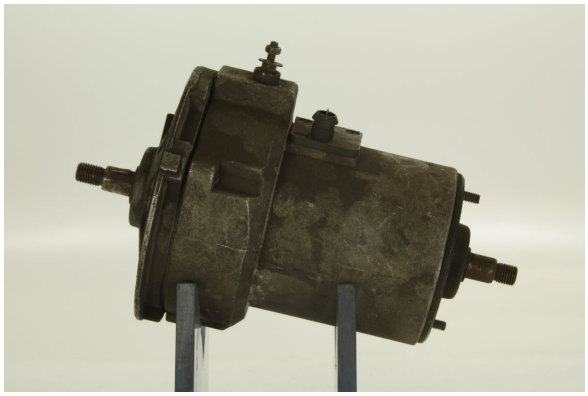


Figure 4.3: Sample images captured at the EIBA Photostudio © Fraunhofer IPK

5 Analysis

This chapter presents a deeper analysis of the state-of-the-art, which will assist in the development of the methodology and original contributions presented in the next chapter.

5.1 Incremental Learning Research

As previously discussed, class incremental learning is the most challenging problem in incremental learning, and addressing it will enable other forms of incremental learning to also be resolved with minor adjustments. The general algorithm for all state-of-the-art class incremental learning implementations is given in Algorithm 1 in the form of pseudocode. Each training starts with initial joint training, which needs the backbone network architecture and the dataset.

It is well established that a pre-trained ResNet model would provide much better and faster results than a model trained completely from scratch. Moreover, a deeper network (with more parameters) would perform differently than a more shallow architecture for the same dataset and hyperparameters. A result of a small test for training on randomly selected images from the ImageNet Subset for different ResNet architectures is given in Table 5.1.

Table 5.1: Top-1 accuracy on ImageNet Subset dataset. Models were trained on 2000 randomly selected images in total (20 images per class).

Model Architecture	Top-1 Accuracy
ResNet-18	81.24%
ResNet-34	84.56%
ResNet-50	87.81%
ResNet-101	89.62%
ResNet-152	90.91%

Hence, to study and compare different Class-IL implementations, it is necessary to use a common baseline architecture. For the CIL experiments, ResNet-18 is generally used as the backbone architecture for ImageNet and other comparable datasets. Secondly, using pre-trained weights makes it difficult to gauge the influence of the CIL implementation on test accuracy. Experiments involving incremental tasks use Kaiming/He initialisation [69] for the weights. It is a commonly used method that initialises weights with zero mean Gaussian with a standard deviation of $\sqrt{2/n_i}$

$$w_i = N(0, \frac{2}{n_i})$$

With these constraints, it is ensured that when experimenting with different class incremental learning approaches, the model learns the object features from scratch during the initial training

(Task 0) and the subsequent impact of the plasticity-rigidity trade-off is captured.

The next logical step is to review the Class-IL implementations discussed in Chapter 2 and select the suitable ones for this project. Several state-of-the-art implementations were tested during this thesis. While each approach offers distinct advantages, it is not feasible to test all approaches within the project time frame. On the other hand, it is also not ideal to select one approach by looking at the test benchmarks such as [15] due to the following reasons:

- None of the benchmarks or original publications test their implementations on real-world scenarios with variable class increments. Based on the experience with industrial machine learning projects, it is clear that new components are received by the operator at varying points of time and are introduced into the ML pipeline at an uneven rate.

- It is not clear how the implementations would perform when training on datasets such as the Industrial-100, with its specific subcategories and characteristics.

- No data exists on how datasets of different feature complexity would fare under incremental learning. The Industrial-100 dataset would be ideal for this investigation.

Based on the results achieved by other Class-IL surveys, particularly [15], [18] and [55], four recent implementations were selected for further research. The peculiarities of each method are discussed below.

PODNet [47] (2020): This approach uses all three avenues of catastrophic forgetting mitigation categorised by [18] (2020), viz., rehearsal learning, knowledge distillation and architectural change to address the class imbalance. It employs a Local Similarity Classifier to balance the old class embeddings and the new class embeddings using cosine normalisation. The authors of the original work [47] used a multimodal approach to resist catastrophic forgetting and increase incremental performance. The approach also constrains the spatial features after each ResNet block of the model architecture. The result of spatial pooling on the ResNet-18 architecture can be seen in Figure A.2. This corresponds with a slight increase in the model parameters as the model adds more incremental tasks. The authors prove this distillation approach can outperform other implementations when using limited exemplar memory and a large number of class increments.

CCIL-SD [21] (2021): This approach employs rehearsal learning, a forgetting constraint (for KD) and a compositional system with bias correction and fine-tuning. Separate activation is used for old and new classes. The authors argued that using a combined softmax output for all the classes together results in a large class imbalance, which is resolved by their compositional system. The effect of this system on the base ResNet-18 architecture can be seen in Figure A.3. There is a gradual increase in the number of model parameters during incremental learning.

DER [23] (2021): This approach is an extension of Task-IL solutions implemented to Class-IL problems. It also employs rehearsal learning and knowledge distillation. However, the model uses a dynamic expansion of the model architecture by adding a new feature extractor for each task. As seen in Figure A.4, at task 3, the model has one original extractor from the backbone ResNet-18 and other extractors are appended from tasks 1, 2 and 3. This results in a linear increase in the number of parameters and the model size with more incremental tasks. However, only the current feature extractor has trainable parameters and the other extractors are frozen. The outputs from each extractor are pooled together to give a model output.

FOSTER [65] (2022): This approach directly addresses the issues of DER Net by adding a distillation and feature-boosting step after each incremental task. As a result, the model always maintains two feature extractors: one with the learnings of all the previous classes and the other one with trainable parameters to learn in the new classes. It aims to avoid the problem of linearly increasing model size. However, the implementation also adds an extra step for distilling the extractors from previous classes into a single feature extractor. This, in turn, increases the computational effort and adds additional loss and accuracy metrics.

RMM [64] (2022): This is not a separate Class-IL implementation, but rather a strategy for a memory management approach that can be appended to different implementations. The authors published their results using several architectures, including PODet + AANet [22]. The model architecture includes stable (rigid) and plastic blocks at each residual level of the base architecture. The memory management approach uses reinforcement learning concepts to create pseudo tasks and assign variable memory storage for different classes. The policy first decides the optimal distribution of memory between old and new classes and then decides on the distribution of memory amongst the old classes.

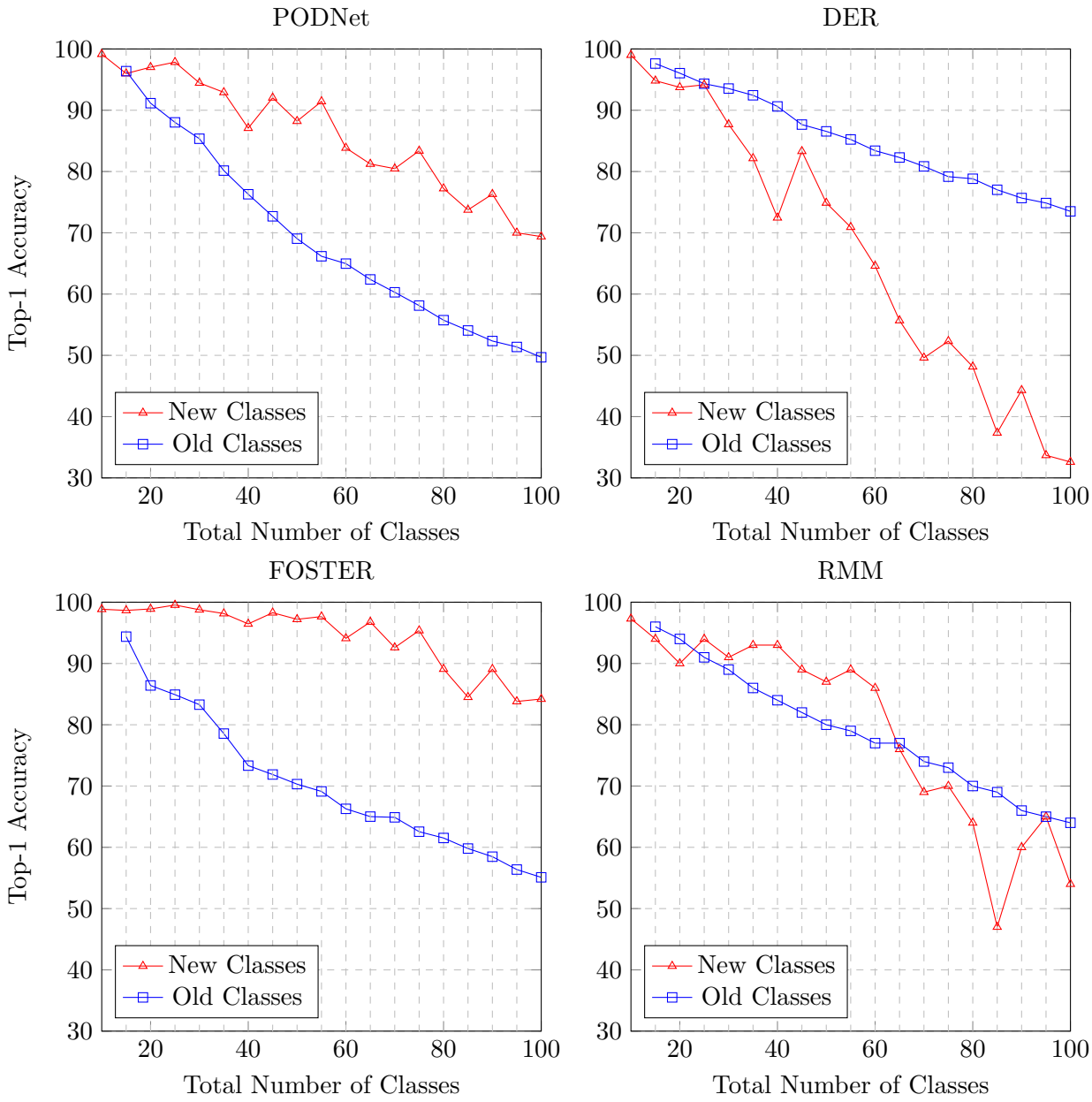
The plots in Chart 2 show the performance of the four strategies on old (previous tasks) classes and new (current task) classes for the ImageNet-Subset. This provides a more complete picture regarding the efficaciousness of the implementation. It can be seen that the PODNet and FOSTER implementations can learn and perform well on the newer classes, even after several task increments. For DERNet, the model can maintain excellent performance on the older classes, however, the accuracy and learning on the newer classes decrease catastrophically. This raises questions regarding the use of task increments beyond a certain threshold, which will have to be answered in the thesis. RMM approach is also able to maintain a good accuracy on the older classes, even though a significant drop in the learning of new classes can be seen. This data is not available for CCIL-SD with the originally published code. [1] Please note that the particular class incremental learning scenario is described later in detail in 6 and is denoted as Experiment 1.

[1] The extensive experiments presented later in this thesis show that this approach had the poorest performance under all scenarios. It was not considered for subsequent developmental work further. However, the results are still shown in this report.

Algorithm 1: General Class Incremental Learning approach

Data: Dataset, K (Total Classes), S (Start Classes), N (Incremental step), Hyperparameters
Result: Accuracy for each task, Average incremental accuracy
Accuracy List = [ ]
Number of tasks: $n_t = (K - S)/N$;
while *task* $T \leftarrow 0$ **do**
    Initial Joint Training;
    Model $M \leftarrow M_0$
        Knowledge Distillation // train student from teacher
    // Exemplar Selection
    for *i in range(0, S)* **do**
        Get θ from M_0 features from the model;
        $\mu \leftarrow$ class mean for i;
        // Select exemplars to match subset mean to μ
    end for
    Model = M_T
    Exemplar set = E_T
    Accuracy List $\leftarrow$*Accuracy*
end while
for T *in range (1,* n_t*)* **do**
    Get Model M_T
    Training Set = New Classes + Exemplar set E_T
    Incremental training loop for T
    Model $M \leftarrow M_T$
        Knowledge Distillation // train student
    // Exemplar Selection:
        Reduce exemplars from old class;
    for *i in range(0, S + T*N)* **do**
        Get θ from M_T features from the model;
        $\mu \leftarrow$ class mean for i;
        // Select exemplars to match subset mean to μ
    end for
    // Update Model Representation and Architecture:
        Model = M_T
    Exemplar set = E_T
    Accuracy List $\leftarrow$*Accuracy*
end for
Result $\leftarrow$Accuracy List

Chart 2: Performance on old and new classes for Class-IL implementations



5.2 Incremental Learning Application

The focus of incremental learning research is on exploring the mechanisms of the Machine Learning algorithms and underlying reasons behind catastrophic forgetting. The Incremental Learning implementations are used under a stricter frame, with no access to pretrained ImageNet weights. This is partly necessary because the results need to be published on the ImageNet dataset itself to compare them against other state-of-the-art implementations. Secondly, the implementations are also commonly presented with a common ResNet backbone (ResNet-34 for CIFAR100 and ResNet-18 for ImageNet Dataset). However, in a commercial project environment, the emphasis of the ML framework is on getting the most accurate results with minimal expenses. In this regard, using pretrained weights and a larger ResNet backbone (or other architecture of choice) is more apt. The ultimate goal of the Incremental Learning research presented in this thesis is to derive conclusions that will guide the use of Class Incremental Learning in the industrial use case. In that regard, interpreting the results using the top-1 accuracy and average incremental accuracy of classification would be insufficient. These metrics are sufficient when assessing standardised Class-IL scenarios such as the ones benchmarked at [15]. However, with a variable amount of data and classes, the bigger increments (where a greater amount of data is introduced) become more important than the smaller ones (with fewer new classes). In that regard, a modified metric must be used to study the results from the Class-IL research.

The developments presented in Chapter 6 and the resulting inferences (Chapter 7) give insights into how Incremental Learning implementations work under different scenarios. These are then used to develop an implementation that makes use of all the possible tools with no need to adhere to the standards of Incremental Learning research. Such a scenario is presented in Chapter 8 on the EIBA Photostudio data 4.3.

5.3 Data Analysis and Pruning

With dataset management, the aim is to develop an implementation to get quick, accurate and interpretable results from the input data with minimum computational investment. This means being able to prune the dataset based on user preference, as well as provide the user with a clear way to visually understand the distribution of the image data. For the former, a simple random image sampling can be taken as a baseline. For carefully curated image datasets, random sampling can be effective. However, it does not help with the latter.

A simple test was designed to determine the preliminary efficacy of a sampling strategy. The Industrial-100 and ImageNet-Subset were used. The dataset was to be reduced to only 20 images per class (in line with the general exemplar limit in Class-IL literature). The best sampling strategy ought to distil the dataset down to 20 relevant images with the most important and varied data distribution. This reduced dataset was compared against a random exemplar

selection (baseline) by training a ResNet18 model on it. A good exemplar selection strategy must give at least as good accuracy as the baseline, preferably better.

In the context of Machine Learning, an image is a tensor, which the trained model converts into a meaningful output for inference. The thesis work started with exploring the simplest possible options using image hashing [28]. It is the process of assigning a unique hash value to an image. The image values of different images can then be compared; similar images will tend to have similar hash values. However, preliminary testing carried out by the author showed that the hash values can change drastically for an image upon performing simple transforms (such as rotation, padding, and stretching). This makes it unfeasible to implement it in a project environment where the image size and contextual data can vary.

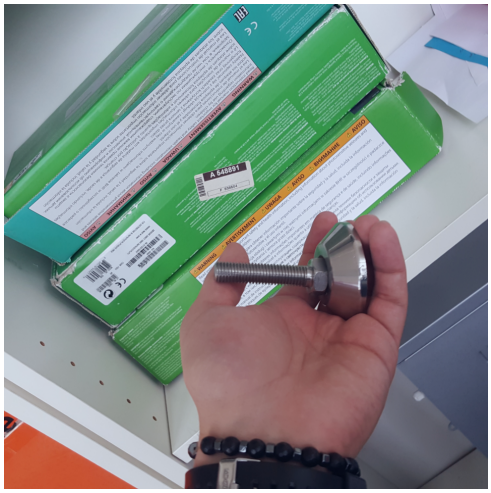
The next option was to use pre-trained Machine Learning models [12] and extract image features for analysis. While this approach offers more nuance than image hashing, datasets such as Industrial-100 contain object data in different contexts and out-of-the-box features obtained by passing the images through the model did not provide satisfactory results. It failed to outperform the baseline.

As previously discussed, it is appealing to use the incrementally trained ML model which has already been trained on the available classes to derive image features. However, there are two issues: Firstly, if more classes are added to the dataset, a further inference cannot be made unless a new task is added to the increments. Secondly, this trained model would be ideal for the current dataset only, using it for any other dataset or a set of images will require more computational overhead and training time.

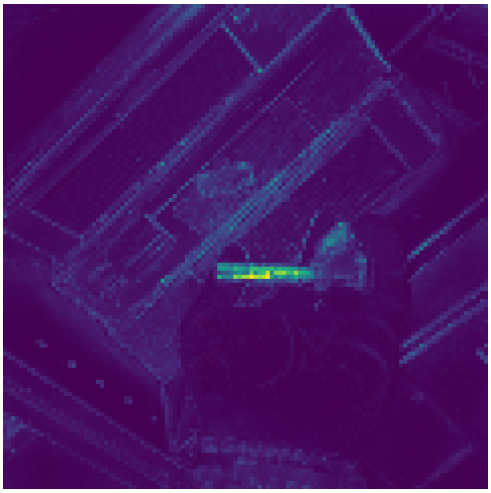
The use of Vision Transformers was explored in this context since several recent studies show that these powerful transformer models can outperform ResNet models without pretraining or strong data augmentations [66]. Furthermore, Dino [60] can visualize attention and understand the context of image and video data in a variety of contexts. An example of this is shown in Figure 5.1 for two objects from the Industrial-100 dataset. Initial assessment of various objects under different contexts yielded accurate results. Dino ViT-Small was used for the thesis, which has 21 Million parameters and claims 78.3% k-nn accuracy on the ImageNet dataset. The preliminary investigation shows that large pre-trained Vision models have proven to be excellent for feature encoding and interpretation.

A simple k-NN evaluation using the pre-trained ViT-Small model yielded the results shown in Table 5.2. The results obtained by training the transformer model from scratch on the Industrial-100 dataset were much less accurate since sufficient data and computational resources were not available.

From these results, it is evident that Dino is an ideal tool for extracting meaningful feature

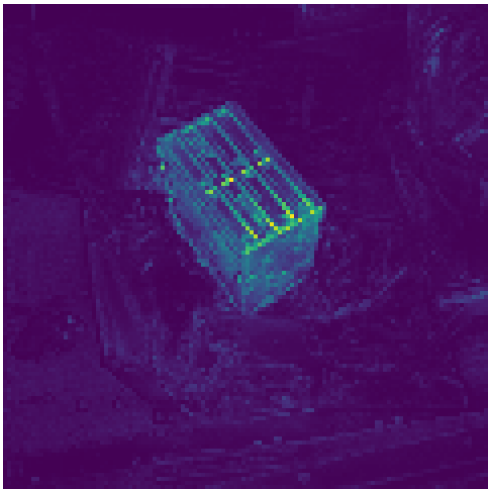


(a) Object: Kochtopfdeckelgriff



(b) Attention Map

(c) Object: Dieselfilter



(d) Attention Map

Figure 5.1: Attention maps for the Industrial100 Dataset objects generated by DINO ViT-Small [60]

Table 5.2: k-NN classification into 100 clusters using ViT-Small pre-trained model

Dataset	Top-1 kNN accuracy	Top-5 kNN accuracy
Industrial-100 White background	97.18%	99.63%
Industrial-100 Clean background	91.24%	98.87%
Industrial-100 Handheld	81.27%	95.09%
Industrial-100 Messy background	74.45%	91.45%
ImageNet Subset	98.37%	99.77%

vectors from the image dataset. The ViT-Small model outputs a feature vector of the size [1, 384] for each image [60]. These dimensions make it still computationally expensive to be used on large datasets. This encoded data (N images corresponds to (N, 384)) must be downscaled to a lower dimension (n < N) for better computational efficiency without losing the general structure of the data. Additionally, vectors encoded in higher dimensions cannot be visualized by the end user. Hence, these vectors need to be downscaled to either 2D or 3D to be projected into an interactive plot.

5.4 Green AI

From a broader perspective, the carbon footprint of an AI system heavily depends on the location of the operation. This dictates the energy infrastructure and resource availability. Good approximations exist that parameterise the carbon footprint of ML training based on the location, the hardware setup and the running time, such as [30] and [31]. Since the location of the operation is fixed, this factor is neutralised. Furthermore, a dedicated computer setup is used for the experimental work, and no other operations were run on it. This also nullifies the second variable of the hardware setup. The specific details of the setup are given in Section 6.4.

As proposed by Schwartz et al. (2020) [35], measuring the computational load of an AI algorithm using MACs/FLOPs is the most impartial method. These can be measured using simple operations in the Python environment. The number of MAC operations of an ML training is roughly approximated as:

$$Computationalcost \propto E * D * H$$

Where,

E = Processing cost of one sample/image

D = Size of the dataset

H = Hyperparemeter tuning experiments

For this thesis, the hyperparameter settings from the original work are used. The processing time for each image can be readily computed for the backbone architecture. However, it is difficult to parameterise and measure the number of operations for the changing network architecture, as previously discussed. Additionally, the base architecture is the same for all Class-IL implementations used, so another method of measuring the energy consumption of the algorithms is necessary.

6 Design and Implementation

The methods proposed in this chapter follow the analysis conducted on the state-of-the-art and expand them to suit the current project requirements and the operating environment. The proposed methodology follows, with an account of the developments incorporated during the thesis work. They are, once again, divided into the three core themes of this thesis for the sake of linearity. However, these developments are all an integral part of the EIBA project and the results are intended to be implemented in tandem. Finally, details of the experiments conducted during the thesis are presented.

6.1 Incremental Learning

Multiple Class Incremental Learning approaches can be introduced into the ML framework in a manner similar to that presented by the FACIL [18] or PyCIL [55] toolboxes. With such a setup, there is no restriction on the number of implementations that can be integrated into the project environment. However, since the model architecture progressively expands (as elaborated in the previous chapter), the subsequent class increments would use the same approach until the performance is no longer acceptable and full joint training is necessary.

In order to implement a computationally efficient solution to the problem, it is essential to investigate the computational cost of the Class-IL implementations. One other objective of this work is to qualitatively and quantitatively understand the behaviour of the chosen implementations based on their performance on data of varying visual complexity, and under varying increment sizes.

The general algorithm for implementing class incremental learning in real-world scenarios is presented in the form of pseudocode in Algorithm 2. After the initial joint training, the increase in the number of classes can be tracked and logged, based on which incremental learning training can be activated during a suitable time. This may also be voluntarily triggered by the end user. Based on the experience with industrial projects, it can be said that it might be prudent to allow for the accumulation of a few new classes and initiate the incremental training loop after a certain minimum threshold is met. Another part can be added to the code for ensuring that the accuracy does not fall below a certain limit. This scenario is unpacked further in Chapter 7.

Another practical scenario that needs to be addressed is the implementation of Class-IL with other continual learning approaches. The most frequent use case was estimated to be the one where the lab operator digitalises a given set of parts on a weekly (or a similar periodic schedule) basis. Some parts are reprocessed, and more image data becomes available on them. In this case, the CIL model must be supplemented with additional online learning increments. Another additional consideration is whether the newly added images provide a sufficient feature gain

Algorithm 2: Class Incremental Learning approach with variable increments and preferred exemplar selection method

```
Data: Dataset, Exemplar Selection Method, Hyperparameters
Result: Accuracy for each task, Average incremental accuracy
Accuracy List = [ ]
From Dataset get number of classes = N_0;
while task T ← 0 do
    Initial Joint Training with N_0 classes
    Model M ← M_0
        Knowledge Distillation // train student from teacher
    // Exemplar Selection:
    for i in range(0, S) do
        Get θ // Model M or ViT
        Select exemplars // Use preferred method
    end for
    Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
end while
N ←N_0
Get N_1 // From Dataset get the number of classes N_1
if N_1 > N_0 then
    Get Model M_T
    Training Set = New Classes + Exemplar set E_T
    Incremental training loop for T
    Model M ← M_T
        Knowledge Distillation // train student
    // Exemplar Selection:
        Reduce exemplars from old class
    for i in range(0, S + T*N) do
        Get θ // Model M or ViT
        Select new exemplars // Use preferred method
    end for
    // Update Model Representation and Architecture:
        Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
    N_0 ←N_1
else
end if
Result ←Accuracy List
```

that the model can use to train on and improve its prediction. This comes under the purview of data sampling and pruning, which is discussed in the next section. This can also be controlled and monitored in the incremental learning algorithm, as shown in Algorithm 4. The addition of new data within a particular class also introduces an additional problem of balancing the class mean and selecting appropriate exemplars based on the Herding algorithm. An argument can be made that the new exemplars ought to be discarded if the additional online training does not improve (or even slightly diminish) the performance on the validation/test set. In the context of an expanding dataset, however, the test set can also be appended and changed at a later stage. It is necessary to expand the continual learning approach further to adjust for new input of data regardless of the online training performance in the given increment. The accompanying online learning step is elaborated in Algorithm 3.

Algorithm 3: Online Learning Increment

```
Data: Dataset, Exemplar Selection Method, Hyperparameters
Result: Updated Model, Model accuracy
if Data > N_1 * K then
    // New data is available for older classes
    Online Learning Increment
        Get Model M_T
        Training Set = New Images E_o
        Model Training with N_1 classes
    if Acc_o > Accuracy then
        // Exemplar Selection:
            Reduce exemplars from old class
        for i in range(0, S + T*N) do
            Get θ // Model M or ViT
            Select new exemplars // Use preferred method
        end for
        // Update Model Representation and Architecture:
            Model M ← M_o
        Result ←Acc_o
    else
        Model M ← M_T // Discard Training
    end if
else
end if
Result ←Accuracy
```

Lastly, in order to interpret the results from the variable class incremental learning sequences, we present a metric: **cumulative accuracy ratio**. It is expressed as follows.

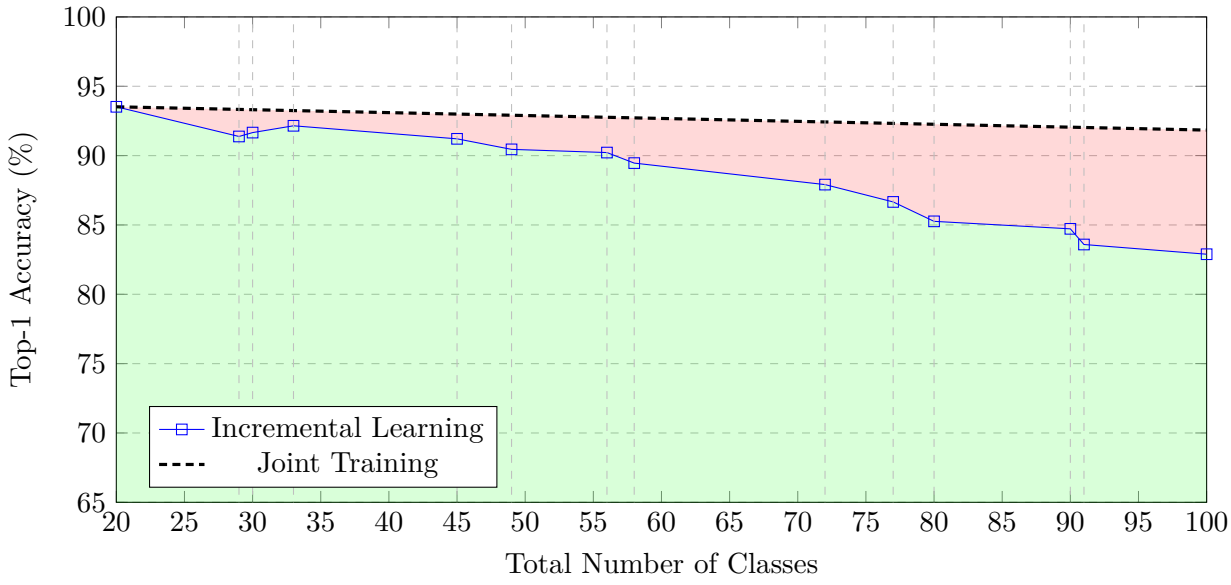
$$\eta_{acc} = \frac{\sum_{I=0}^{T} acc_i}{\sum_{i=0}^{T} acc_i^{joint}} = \frac{A_{green}}{(A_{green} + A_{red})}$$

Algorithm 4: Class Incremental Learning in tandem with Online Learning Increments

```
Data: Dataset, Exemplar Selection Method, Hyperparameters
Result: Accuracy for each task, Average incremental accuracy
Accuracy List = [ ]
From Dataset get number of classes = N_0
while task T ← 0 do
    Initial Joint Training with N_0 classes
    Model M ← M_0
        Knowledge Distillation // train student from teacher
    // Exemplar Selection:
    for i in range(0, S) do
        Get θ // Model M or ViT
        Select exemplars // Use preferred method
    end for
    Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
end while
N ←N_0
Get N_1 // From Dataset get the current number of classes N_1
if N_1 > N_0 then
    Get Model M_T
    Training Set = New Classes + Exemplar set E_T
    Incremental training loop for T
    Model M ← M_T
        Knowledge Distillation // train student
    // Exemplar Selection:
        Reduce exemplars from old class
    for i in range(0, S + T*N) do
        Get θ // Model M or ViT
        Select new exemplars // Use preferred method
    end for
    // Update Model Representation and Architecture:
        Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
    N_0 ←N_1
    if Data > N_1 * K then
        // New data is available for older classes
        Online Learning Increment
    else
    end if
else
end if
Result ←Accuracy List
```

The closer the ratio to 1, the better. A low value indicates that a lot of performance potential was lost during the life of the incremental learning project. In a commercial project, the aim of the incremental learning training is to provide inferences on incoming objects. In that regard, the drop in accuracy with the increasing number of classes is more significant when there are more objects to be classified. This also provides a proportional weighing factor to the accuracy of the given class increment based on its size. As previously discussed, the increments with a larger increment are more important. The areas under the curves are represented in the Chart 3. For setups with a constant increment size, the cumulative accuracy ratio reduces to the ratio of the average incremental accuracy and average joint training accuracy. It is the goal of Incremental learning to minimise the difference between the two accuracy curves, i.e., the cumulative accuracy ratio needs to be maximised.

Chart 3: Cumulative Accuracy curves



6.2 Dataset Pruning and Preliminary Analysis

The issues are elaborated below The table shows that the data representation must be representative of the test data.

Table 6.1: k-NN Classification accuracy on Industrial-100 data subtypes using pretrained Dino model

Training Data	Test Data	Top-1 Accuracy	Top-5 Accuracy
White Background	White Background	97.18%	99.63%
White Background	Messy background	30.0%	48.1%
Messy background	White Background	48.0%	74.45%
Messy background	Messy background	74.90%	91.54%

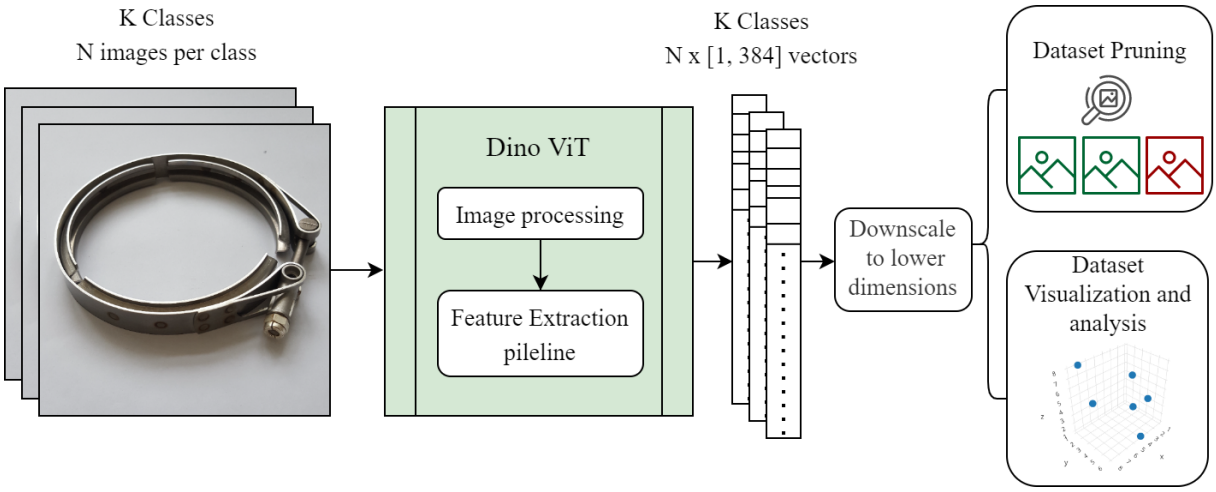


Figure 6.1: Concept of dataset pruning, visualization and analysis using Dino.

An experimental strategy was developed using a scale-invariant loss function with a simple MLP architecture. The features from the ImageNet dataset were derived and stored. The MLP architecture has an input of (batch size, 384) and an output of (batch size, 3). The MLP model was trained on the feature dataset to maximise the intra-class distance between the individual vectors and project them into a 3D space between 0 and 1. Multiple training configurations and setups were experimented with. It was inferred that the model tended to project the vectors into a 3D space along the diagonal of the cube of unit size. Testing this approach on the Industrial-100 dataset showed that some image-based context was still maintained. The projected vectors were converted into a 3D point cloud and then further downsampled using Open3D [67]. Comparing the resulting pruned dataset showed that the approach was inferior to pure random sampling. A further use for visualisation is possible, but the results remained unsatisfactory.

The authors of the original work present a class representation example on the ImageNet dataset. They reduced the dimensions of the features down to 30 using PCA, and then further down to 2D using t-SNE. They represent the class average feature vectors in a 2D space with the feature embeddings placed in a semantically consistent manner w.r.t. the other classes.

For the work of this thesis, using PCA was sufficient. Using t-SNE in addition did not provide any advantages and led to a loss of context in some cases. This may be because t-SNE preserves only local similarities between the features, whereas PCA preserves global data structure and large pairwise distance to maximise variance. In the case of dataset pruning and visualisation, the general intra-class and inter-class structure of the data is important, which is handled by PCA. Testing this approach on the Industrial-100 dataset showed that the intra-class and inter-class structure was very well maintained.

The plot 6.2 shows an example of the PCA downscaling for one object from the Industrial-100 dataset. A broad separation between the subcategories can be seen. Further investigation

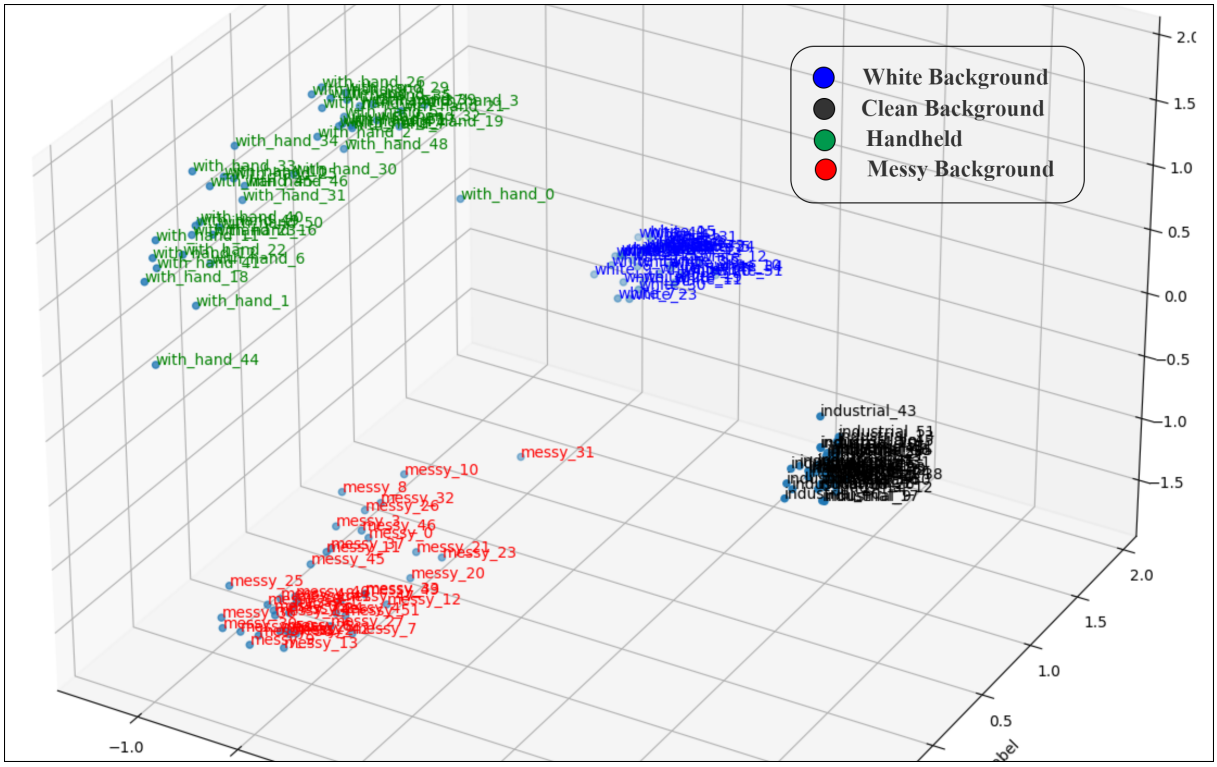


Figure 6.2: Intra-Class distribution of feature vectors from a class in the Industrial-100 dataset.

into the intra-class subcategories shows that the data distribution in lower dimensions still contains global structure and meaning. Figure 6.3 shows an example of an object against a white background. The object has a circular profile, the object is captured from all orientations. The images with the object captured in a more flat/oval profile are located towards the right corner of the 3D plot, whereas images with a more round profile are distributed away from that corner towards the left. Similar visual encoding can be observed from other objects in varied contexts. Another example is shown in Figure 6.4, with the object against a messy background. The 3D plot shows that the images with the wires in the background are clumped together, similarly, images with the working desk in the background are grouped close to each other. It can also be seen that the variance in the feature vectors with clean and white background is much lesser than the data features with handheld objects and messy background.

The next step in the development is adding an optimal way to prune the dataset. The previous approach of converting the feature vectors into a 3D point cloud is a viable option. Using the K-means clustering algorithm proved to be quite successful. This also has an advantage over using Open3D, since the K-means algorithm can work with feature vectors in any dimension. Experimenting with various feature dimensions showed that reducing the dimensions down to 32 had approximately the same results as with the original dimensions (384), which means that the loss of information was minimal. The sequence of the dataset pruning process is depicted in

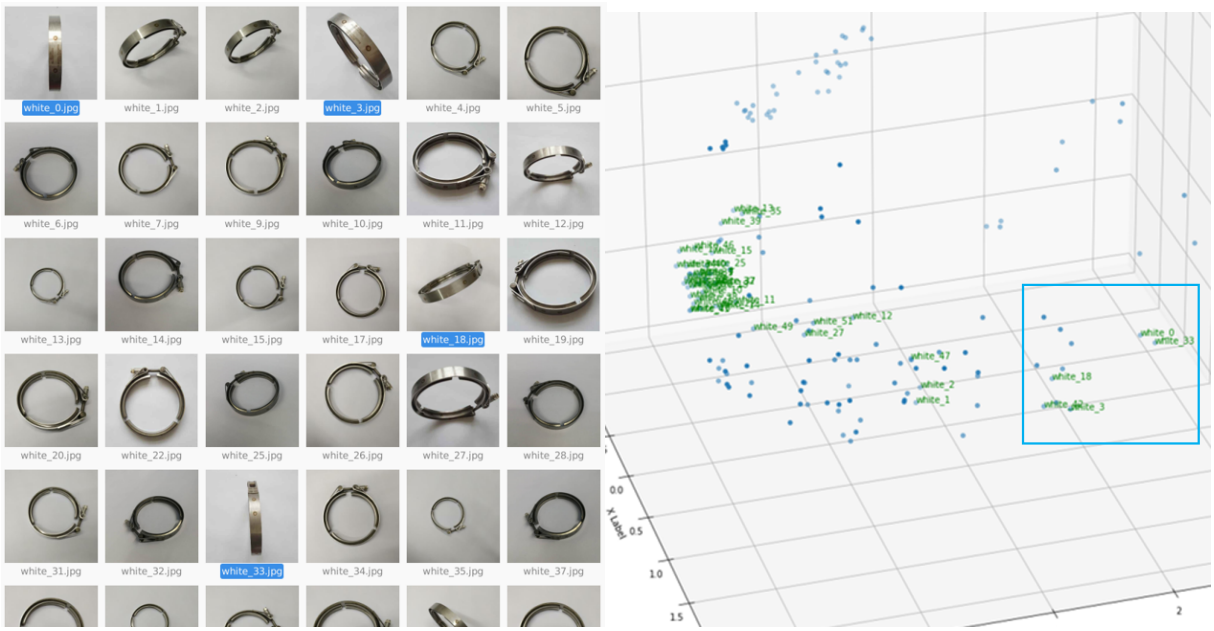


Figure 6.3: Intra-class data structure for an object against a white background from the Industrial-100 dataset.

Figure 6.5. After downscaling the features (32 for strict dataset pruning, 2/3 for visualisation), k-means is used to divide the features into individual clusters. The number of clusters can be set to the desired value based on the requirement or user preference. The cluster centre can be considered as a generalisation or an exemplar of the entire cluster. The image closest to the centre is taken as the exemplar image. Figure 6.5c shows the 1300 images from the ImageNet class reduced to 20 representative images. Tests on this approach show that it outperforms the baseline, as shown in Table 6.2 (the tests were repeated multiple times, averaged results are given).

For the PCA downscaling, the cost for covariance matrix computation is $O(p^3)$; the cost for its eigenvalue decomposition is n^3), where n is the number of data points and p is the original feature dimension (384 in this case). The computational complexity of the PCA operation is thus $O(min(p^3n + p^3))$.

The k-means algorithm aims to minimise the following argument:

$$argmin \sum_{i=1}^{K} \sum_{X_j \subseteq s_i} ||X_j - \mu_i||^2$$

The algorithm is NP-hard, with K being the total number of clusters, however, it is usually run for a fixed set of iterations (100 was found to be satisfactory for exemplar selection). Taking the

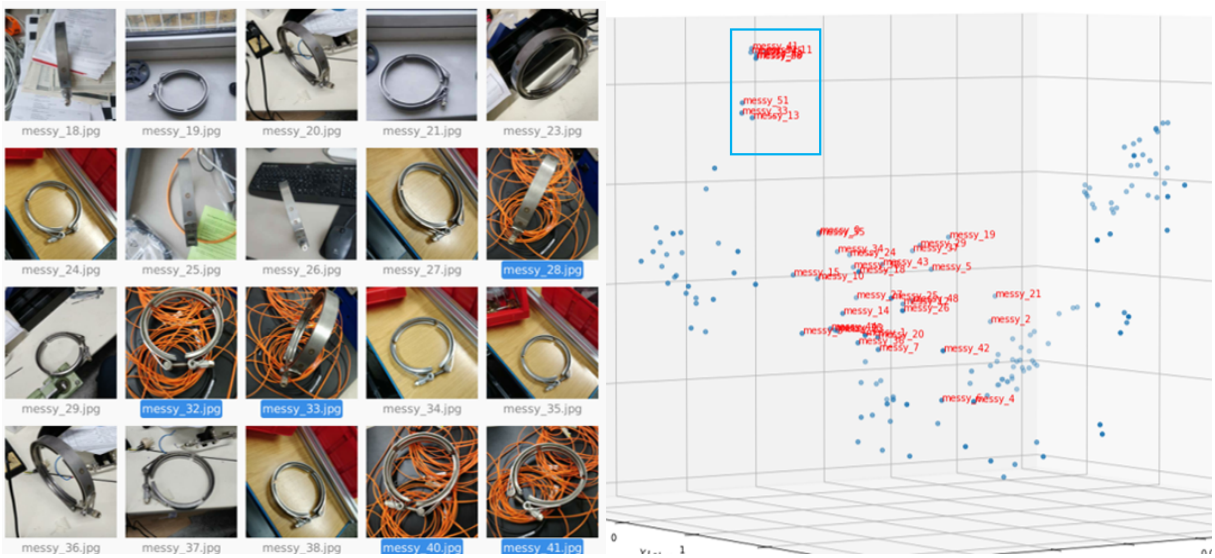


Figure 6.4: Intra-class data structure for an object against a messy background from the Industrial-100 dataset.

downscaled d dimensions from PCA, the computational complexity of the k-means algorithm is $O(t * k * n * d)$.

Based on the experiments conducted during the thesis, the total computational cost is comparable to the herding algorithm, unless the k-means is run on high dimensional data or without downsampling the original dimensions of the vectors. The herding approach also allows reducing the exemplars and balance the class mean to store new class data, which is infeasible with k-means. However, given the other tertiary applications of this approach, the tradeoff is relatively low. Compared to the incremental step training time, the computational cost of the approach is much lower and does not affect the total computational cost of the complete operation significantly.

A summary of the developed algorithm is available in Algorithm 5. The usefulness of the developed approach also translates directly to exemplar selection for incremental learning tasks. The pseudocode for incorporating this in any class incremental learning implementation is given in Algorithm 6.

The result of implementing this approach for pruning the dataset can be seen in the form of an example in Figure 6.6. The class images are reduced to 20 unique images in terms of object orientation, context, and background. In agreement with the earlier observation about the variance within different subcategories of the Industrial-100 dataset, it can be seen that fewer images are selected from the *white* and *clean* backgrounds and a greater amount of *handheld* and *messy* background images are selected. This helps the approach outperform the baseline,

Table 6.2: Comparing Random Exemplar selection against the approach developed during this thesis

Dataset	Random Selection	Exemplar Selection with Dino + PCA + k-means
Industrial-100 (20 images per class)	74.15%	78.32%
ImageNet-Subset (20 images per class)	81.24%	83.06%

since the images with noisier backgrounds are harder to train on.

This approach can also be further extended to detect similar images and poor-quality images. Images with feature vector values within a certain tolerance threshold can be flagged as similar for the operator to inspect and make a decision. Similarly, basic statistics can be implemented to detect outliers in the data and flag them as poor-quality images. Tests conducted on both these show the validity of this approach; though slight fine-tuning of the threshold parameters on a given dataset may be needed.

The algorithm 6 effectively preserves the original feature distribution and balances the intra-class variance for each class. However, this approach would be less effective in cases where not all classes have the same visual complexity. In practical applications, some objects might be digitalised better than others. The objects containing cleaner images would require comparatively lesser data to be stored as exemplars. This allows additional memory to be directed towards classes with higher visual complexity and noise. The algorithm 7 shows the implementation of this approach with pseudocode. Since PCA normalises the feature vectors, the relative feature distribution of each class is equalised. Hence, the inter-class feature comparison is to be carried out prior to the downsampling. The average feature variance is computed for all the features within each class, which then leads to the class weight as follows:

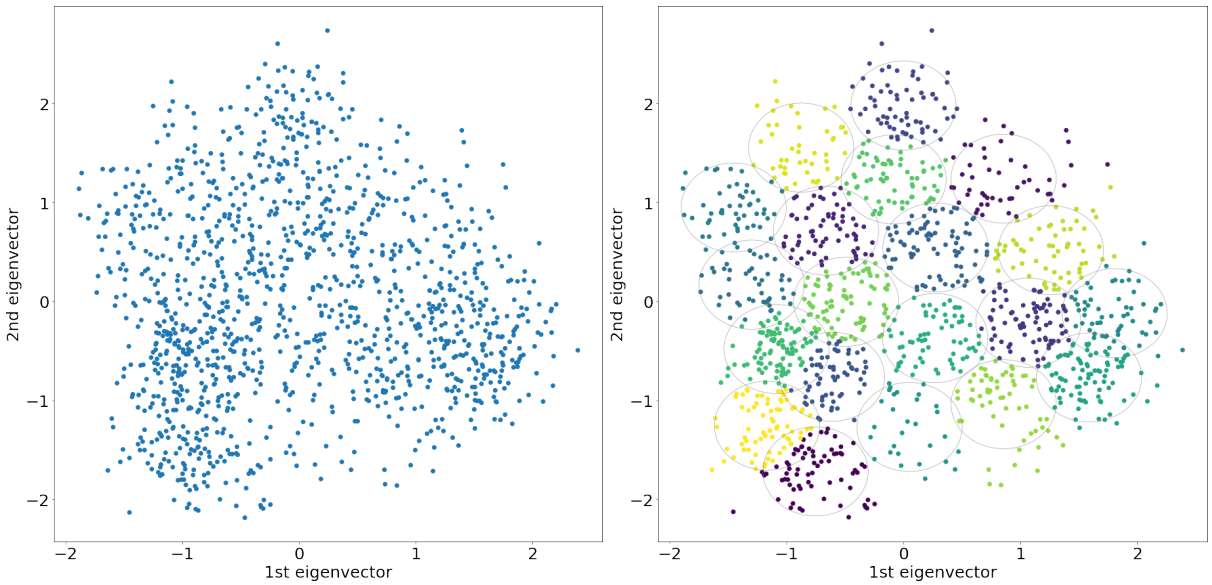
For class i, which contains j total images:

$$n_i = \frac{\sum_j \Theta_j}{\sum_i \sum_j \Theta_{ij}}$$

The operation can be computed on a subset of the classes (for instance the newly appended objects) or on the full dataset. This class weight governs the number of exemplars that shall be stored from the given class. During the K-means clustering, the k value is given as:

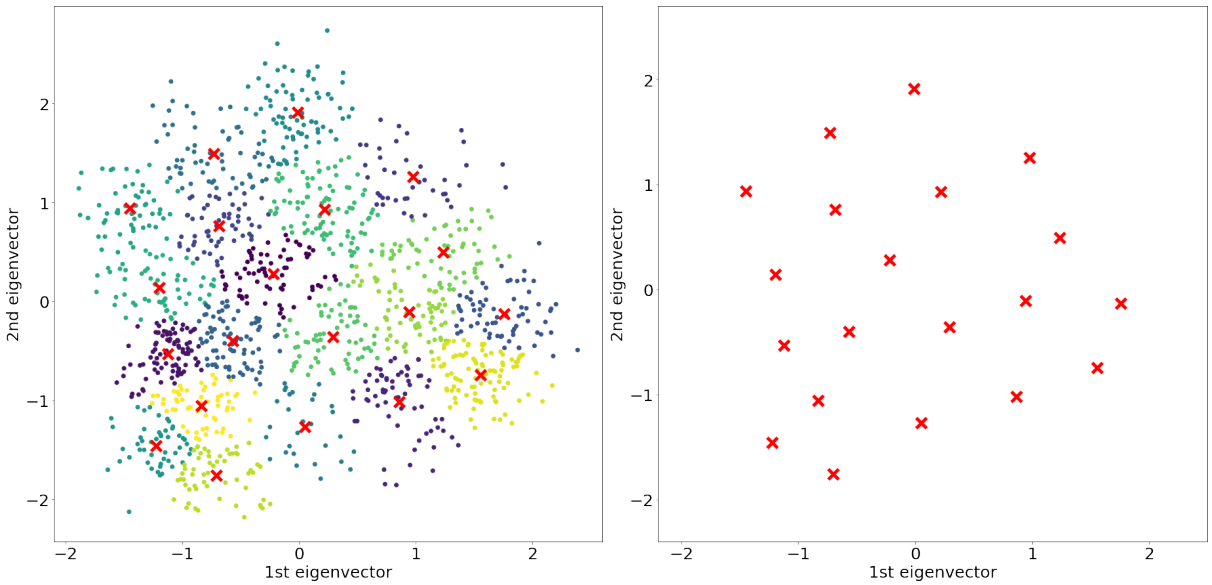
$$K_i = n_i * \frac{memory}{num_{classes}}$$

This is illustrated in Figure 6.7, with two classes that have very different complexity and re-



(a) Feature Vectors

(b) K-means clustering



(c) Clusters and their centres

(d) Exemplars

Figure 6.5: (a) Encoded feature vectors for the ImageNet class *n15075141* in 2 dimensions. For the actual analysis, PCA is used to downsample to 32 dimensions, but 2D plots are shown here for visualisation. (b)Using K-means, optimal intra-class clusters are identified. Please note that the highlighted circles are meant for visualisation and have no bearing on the underlying sampling process. (c)For each cluster, the sample point closest to the cluster centre is taken as the representative image. Thus, the 1300 images in the class are reduced to 20 representative exemplars. (d) 20 representative exemplars

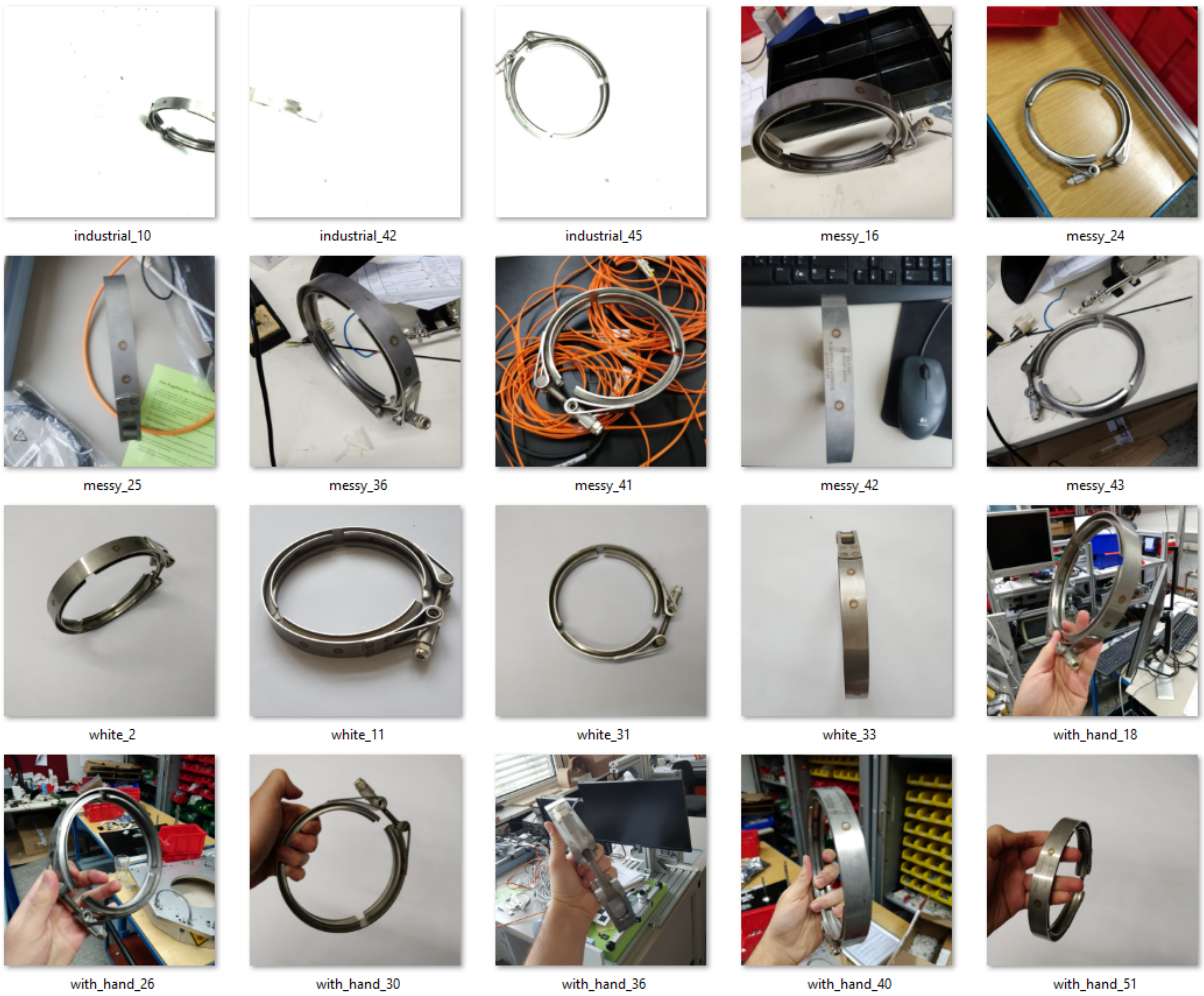
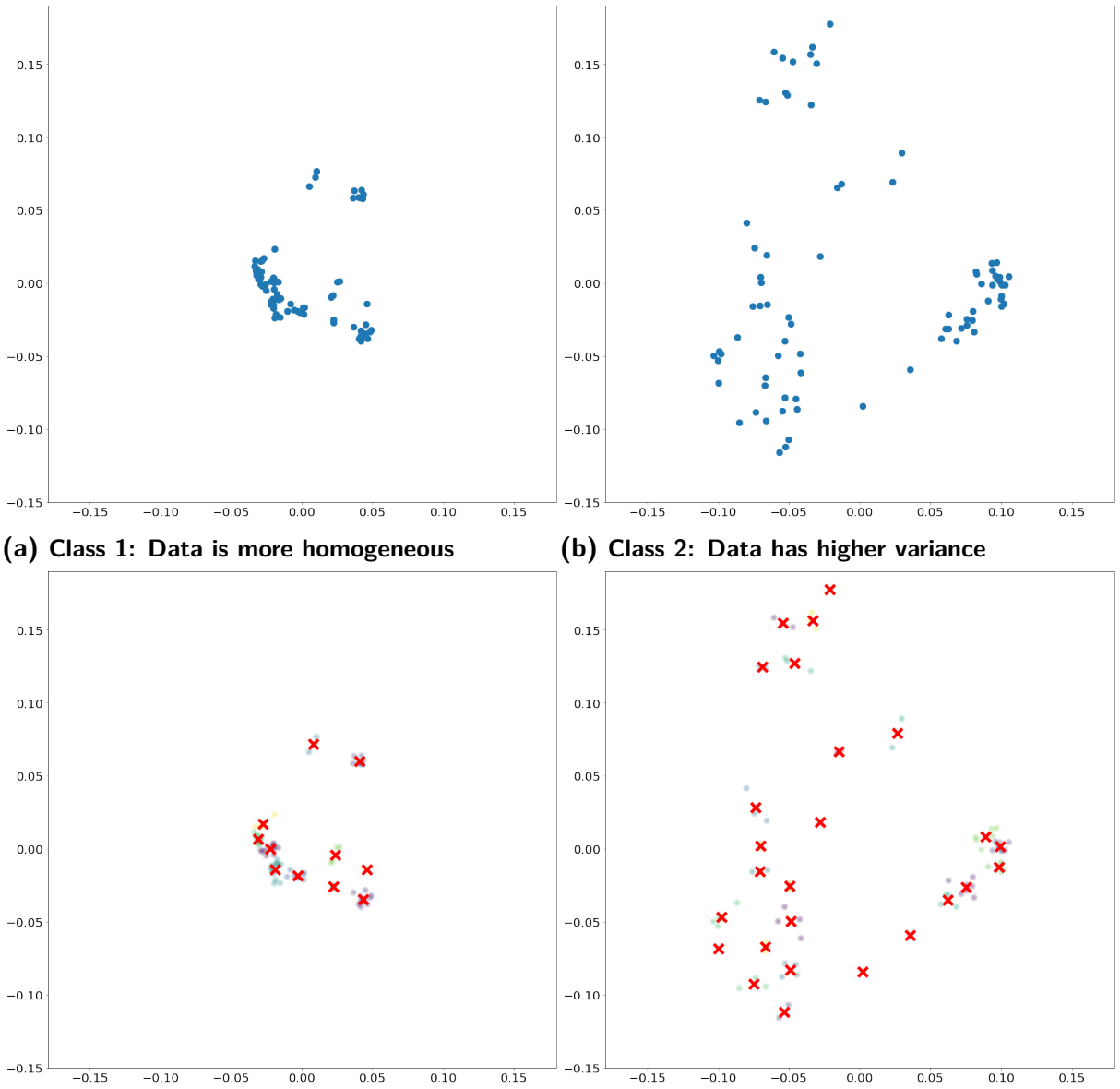


Figure 6.6: 20 selected exemplars for a class from the Industrial-100 dataset.

sulting feature distribution. Please note that the figure only illustrates the concept, the actual computation is performed in the original 384-dimensional space.



(a) Class 1: Data is more homogeneous

(b) Class 2: Data has higher variance

(c) Fewer exemplars are necessary to represent the class distribution

(d) More exemplars are chosen to represent the class distribution

Figure 6.7: A visual representation for the variable exemplar selection strategy as elaborated in Algorithm 7. Please note that the 2D plot is meant for illustration purposes. The original computation is done on the high dimensional data prior to PCA downscaling and normalisation.

Algorithm 5: Dataset Sampling and Pruning with Dino

```
Input  : Original dataset, K classes, N images per class, n
Output: Pruned Dataset, K classes, n < N images per class.
Pruned List = [ ]
import ViT-Small // Pre-trained Vision Transformer Model
for i ← 0 to K do
    for j ← 0 to N do
        θ = Feature Vector [1, 384] // process image via ViT
    end for
    Class Feature Vectors = [N, 384]
    if No Visualization then
        Downsample:
            Principal Component analysis
            Modified Feature Vectors = [N, 32]
        Prune Data:
        K-means Clustering
        Sampled Feature Vectors = [n, 32]
        Pruned List ←Sampled Feature Vectors
    end if
    if Visualize Data then
        Downsample:
            Principal Component analysis
            Modified Feature Vectors = [N, 3]
        Prune Data:
        K-means Clustering
        Sampled Feature Vectors = [n, 3]
        Plot Feature Distribution in a 3D interactive plot
    end if
end for
return Pruned List
```

Algorithm 6: Exemplar selection for Class Incremental Learning with Dino

```
Input  : Old Class data, K classes, N images per class, n
Output: Exemplars, K classes, n < N images per class, n*K <=2000 (User
         defined limit)
Exemplar List = [ ]
import ViT-Small // Pre-trained Vision Transformer Model
for i ← 0 to K do
    for j ← 0 to N do
        θ = Feature Vector [1, 384] // process image via ViT
    end for
    Class Feature Vectors = [N, 384]
    Downsample:
       Principal Component analysis
       Modified Feature Vectors = [N, 32]
    Prune Data:
    K-means Clustering
       Sampled Feature Vectors = [n, 32]
       Exemplar List ←Sampled Feature Vectors
end for
return Exemplar List
Repeat for the next incremental Task
```

Algorithm 7: Variable exemplar selection for Class Incremental Learning with Dino

```
Input  : Old Class data, K classes, N images per class
Output: Exemplars, K classes, n < N sampled images (varies according to
         class feature distribution)
Exemplar List = [ ]
import ViT-Small // Pre-trained Vision Transformer Model
for i ← 0 to K do
    for j ← 0 to N do
        θ = Feature Vector [1, 384] // process image via ViT
    end for
    Class Feature Vectors = [N, 384]
    Feature distribtion analysis
       θ_a = Average feature variance for all vectors within the class
       n = Weighted class exemplar count // proportional to θ_a
    Downsample:
       Principal Component analysis
       Modified Feature Vectors = [N, 32]
    Prune Data:
    K-means Clustering
       Sampled Feature Vectors = [n, 32]
       Exemplar List ←Sampled Feature Vectors
end for
return Exemplar List
Repeat for the next incremental Task
```

6.3 Green AI Analysis

The most accurate solution for tracking energy consumption of the ML training is to measure the power usage during training using an external smart power plug. Shelly Plug S [58] was used for this thesis. Using the plug, it is possible to measure the Wattage of operation, sampled every second. For larger operations (such as incremental training), the statistical sampling of the power consumption can be added up to the total energy consumed by the system. The impact of GPU and CPU operations on the energy consumption can thus be accurately logged over longer periods of time with statistical precision. The plug, however, cannot distinguish between the energy consumed by the ML training and that consumed by other background operations on the workstation. Hence, the idle power consumption of the system was periodically monitored and measured. Based on the calculations, it averages to approx. 54W. This baseline power consumption is subtracted from the power consumption logged by the shelly plug for each experiment.

Since the operating conditions for all sets of training are kept the same, the energy used during each training will tend to be proportional to the total training time. However, other factors such as System memory utilisation and GPU usage are also essential. A cross-referential system of monitoring the different system metrics is also necessary. The chosen metrics are listed in Table 6.3. Different metrics are continuously monitored and logged. The implementation for measuring and logging the continuous real-time power consumption is given in algorithm 8.

Table 6.3: Energy Consumption Metrics for the ML Experiments

Metric	**Detail**
Training Time (h)	Logged
Energy Consumption (kWh)	Tracked using the Shelly plug and subtracting the idle system energy consumption rate
Maximum CPU Utilization (%)	Tracked and logged
Maximum System Memory Utilization (%)	Tracked and logged
GPU Utilization (%)	Tracked and logged

Cumulative Joint Training: Considering a scenario where class-incremental learning is not implemented, and the model is trained anew (full joint training) at each step, the equivalent cost of this approach must be considered. Although this approach would offer the maximum possible accuracy of classification, the training time and computational cost would be higher.

The cumulative joint training cost is computed as:

$$E_{cumulative} = \sum_{i=0}^{T} E_i$$

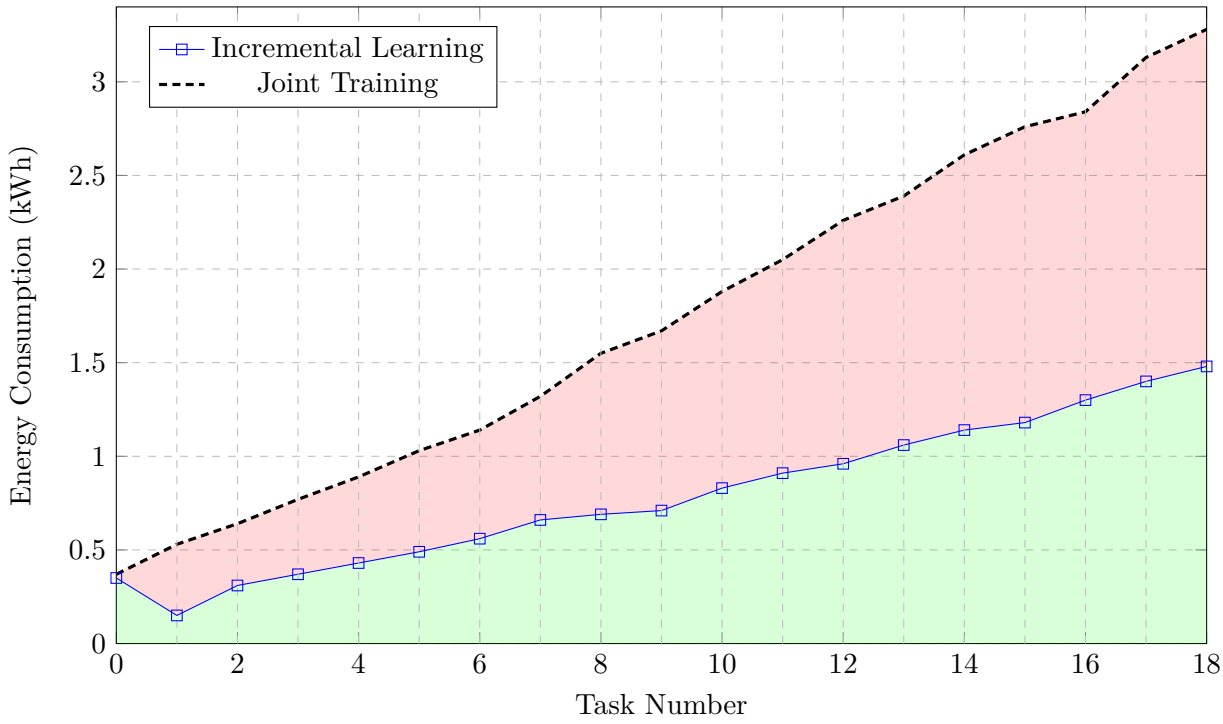
where T is the total number of increments.

Based on this, the cumulative energy consumption ratio for the given Class-IL implementation is computed as:

$$\eta_{energy} = \frac{E_{CIL}}{E_{cumulative}} = \frac{A_{green}}{(A_{green} + A_{red})}$$

This ratio depends on the number of class increments and the size of the dataset. When those parameters are controlled, the various implementations can be readily compared with each other. Contrary to the accuracy curves, the two energy curves, as shown in Chart 4 should ideally be as far apart as possible, which corresponds to higher cost saving. The goal of Green Incremental Learning is to minimise the cumulative energy consumption ratio.

Chart 4: Energy Consumption curves



Algorithm 8: Energy Consumption tracking for Incremental Learning Experiments

```
Data: Class Incremental Learning framework, System Utilisation metrics
Result: Power consumption log, Training time log, other metrics
Power Consumption Log = [ ]
Prompt to close all other applications
Start logging shelly plug power output
Prompt to close all other applications
for t in range(0, 1800) do
    log.append(power,t)
end for
Baseline Power Consumption = sum(log)/1800 Watt
From Dataset get number of classes = N_0
while task T ← 0 do
    Initial Joint Training with N_0 classes
    // Perform model update operations
    for t in range(0, log size) do
        Updated Log = [Log - Baseline Power Consumption] for all i
    end for
    Compute average power consumption
    Energy Consumption = Power Avg * 144/1000 kWh
    EnergyLog ←EnergyConsumption
    TimeLog ←TimeConsumption
end while
N ←N_0
Get N_1 // From Dataset get the current number of classes N_1
if N_1 > N_0 then
    Incremental training loop for T
    EnergyLog ←EnergyConsumption
    TimeLog ←TimeConsumption
end if
Result ←All metrics
Back up logs
```

6.4 Experimentation and Testing

This section gives an overview of the overall setup for the Green Incremental Learning experiments within the purview of the three key areas of the project. The new developments presented in this chapter were sequentially implemented and assessed using the designed experiments.

6.4.1 Hardware, Software, and Operation Setup

The specifics of the system setup are given in Table 6.4. The Shelly smart meter plug was attached to the system and active at all times. The results obtained w.r.t. the time consumption and energy consumption would be specific to this operating system. Hence, the absolute values are not transferable to other systems and setups. However, since all the other parameters are controlled, the experiments will give a good look into the relative computational load for each Class-IL implementation. Most hyperparameters were kept consistent across the implementations where possible.

Table 6.4: System details for the workstation used for the ML experiments

OS	Linux-5.15.0-43-generic-x86_64-with-glib2.17
System Memory	16GB
CPU Count	8
GPU Count	1
GPU type	NVIDIA GeForce GTX 1070
Python version	3.8.12

Resource and project time limitations do not allow all possible permutations and combinations of real-world scenarios to be tested. Secondly, it is also not prudent for a project investigating the energy efficiency of ML implementations to itself have a high energy footprint. This is the reason the ImageNet-Subset was used for most of the thesis instead of the full ImageNet dataset. It may seem like a good idea to train the models a few times and then average the performance metrics to get rid of randomness in the results, but this isn't feasible. Precautions were taken to minimise the randomness and make sure that the results were as accurate and reproducible as possible.

6.4.2 Design of Experiments

The experiments stated below test the developments laid out in this chapter under varying scenarios. They were designed to provide a thorough insight into each of the three key areas of the thesis. The individual experiments, along with their objectives, are discussed below, and the results are discussed in the next chapter. 404543 (author's Matriculation number) was used as the seed for the experiments, including the shuffling of the classes. Most hyperparameters were maintained from the original published work.

EXPERIMENT 1: Class-IL with fixed increments on the ImageNet-Subset.

Objective: *To test the performance of different Class-IL implementations on the ImageNet subset under the condition of a fixed number of total exemplars during each new task (2000).*

Research Questions Addressed: *1, 5, 9, 10, 11*

This experiment will be used as the baseline for the other CIL-based experiments. Based on previous experience and discussions with machine lab engineers, 10 classes were used for initial joint training and 5 classes were introduced during each new task (roughly the number of classes that an operator would digitalise in 45-60min.). The constant introduction of tasks is also shown in the plot. The total number of tasks is 19 (base task 0 + 18 incremental tasks).

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Energy consumption for the different approaches.*
- *Energy consumption for each new task.*
- *A comparison of the energy consumption of the approaches with an equivalent joint training.*

EXPERIMENT 2: Class-IL with fixed increments on the Industrial-100 dataset.

Objective: *To test the performance of different Class-IL implementations on the Industrial-100 dataset under the condition of a fixed number of exemplars for each class (20).*

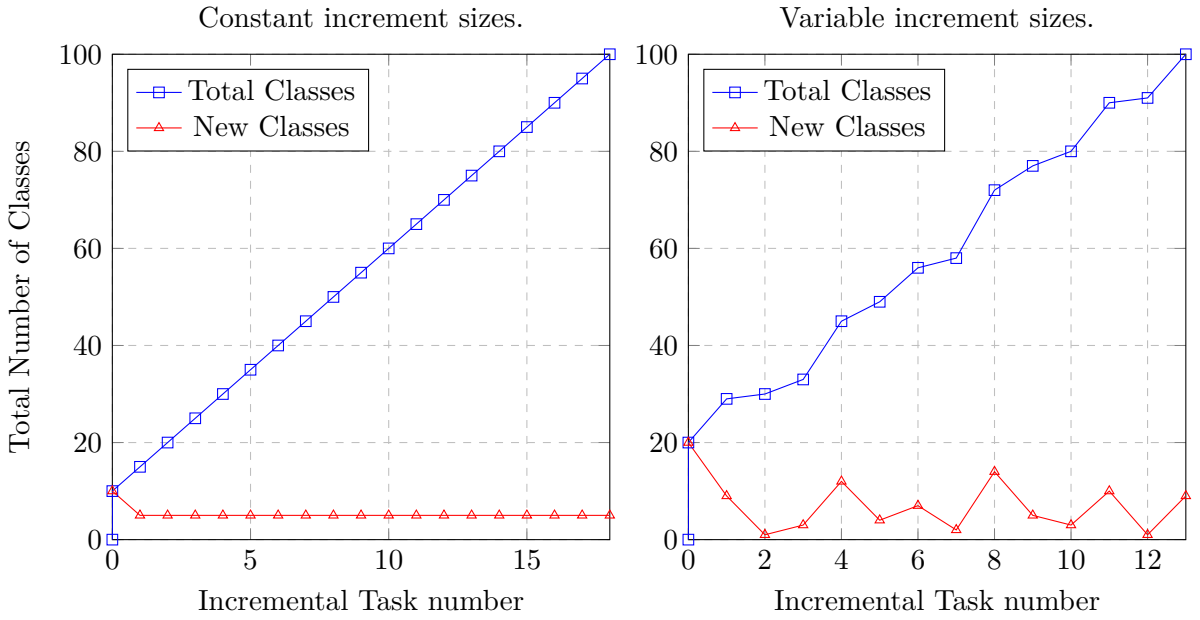
Research Questions Addressed: *1, 5*

This experiment will be compared with the baseline for comparing the performance of different implementations. The exemplar count per class is kept constant for all tasks. This simulates the scenario where a very limited amount of data can be maintained from the previous classes. Secondly, the impact of data sampling strategies will also be equal during each task, such that the results from this experiment can be used as a baseline in future experiments.

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Performance on new classes and old classes.*

Chart 5: Incremental Step Sequences for CIL Experiments



EXPERIMENT 3: Class-IL with variable increments on the Industrial-100 dataset.

Objective: *To test the performance of different Class-IL implementations on the Industrial-100 dataset under the condition of a variable number of exemplars for each class.*

Research Questions Addressed: *1, 2, 3, 4, 9, 10, 11*

A randomised sequence was generated that was tested on each Class-IL implementation. 20 classes were taken during the initial joint training and new classes were introduced during each new task increment as shown in the plot. This sequence was chosen to replicate a scenario where a sporadic jump in the size of the increments happens. The total number of tasks is 14 (base task 0 + 13 incremental tasks).

The Industrial-100 dataset was split into four subcategories (White background, clean background, handheld and messy background) and run on these random increments to study the effect of data complexity and variable task size for the class incremental learning scenario. Since such an investigation has not been reported on before, this will be a unique contribution of this thesis work that will help other researchers in understanding the field of incremental learning better.

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Accuracy for the four subcategories of the dataset.*

- *Performance on new classes and old classes.*
- *Energy consumption for the different approaches.*
- *Energy consumption for each new task.*
- *A comparison of the energy consumption of the approaches with an equivalent joint training.*

EXPERIMENT 4: Class-IL with second variable increment sequance on the Industrial-100 Subset.

Objective: *To test the performance of different Class-IL implementations on the Industrial-100 dataset under a different set of variable increments to study the difference against Experiment 3.*

Research Questions Addressed: *2, 3, 5*

A second randomised sequence was generated that was tested on each Class-IL implementation. 20 classes were taken during the initial joint training and new classes were introduced during each new task increment at a varying rate.

The Industrial-100 *clean background* dataset was used for this experiment.

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Comparison with results from Experiment 3*
- *Energy consumption comparison against Experiment 3*

EXPERIMENT 5: Exemplar Selection for Class Incremental Learning using Dino.

Objective: *To test the performance of different Class-IL implementations on the Industrial-100 dataset with exemplar selection using Dino.*

Research Questions Addressed: *6, 8*

The constant task increment and variable task increment scenarios were tested for a few different Class-IL implementations, with an aim to verify that Dino yields better results irrespective of the underlying method. The Industrial-100 dataset was used for this experiment. Since the

number of exemplars stored remains constant (20 images per class), the effect of the sampling strategy remains equal across all increments.

The dynamic exemplar selection approach was also tested on the Industrial-100 imbalanced dataset (containing varying background images for different classes). The total exemplar limit is controlled, such that the sum of all stored exemplars remains below 2000 for the 100 classes.

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Comparison with results from Experiment 3 (baseline)*
- *Energy consumption comparison against Experiment 3*
- *Comparison of the variable exemplar selection strategy with the other approaches*

EXPERIMENT 6: Dataset Analysis using Dino

Objective: *To test the use of the data analysis strategy developed for Dino on an EIBA component digitalisation prototype.*

Research Questions Addressed: *6, 7, 8*

The EIBA prototype described in Chapter 1 was modified at IPK to incorporate 10 RGB-D cameras, as shown in Figure 6.8. The cameras capture the component from multiple and varied perspectives.

Evaluation:

- *Feature embedding of the images from the different cameras in a 3D space for visualisation.*
- *A comparison of the intra-class feature variation for the different cameras.*
- *Analysis on the selection of the best camera angles and combinations to capture the target object.*

EXPERIMENT 7: Incremental Learning in tandem with Online Learning

Objective: *To test the use of class incremental learning in tandem with online learning epochs to improve efficiency and accuracy.*

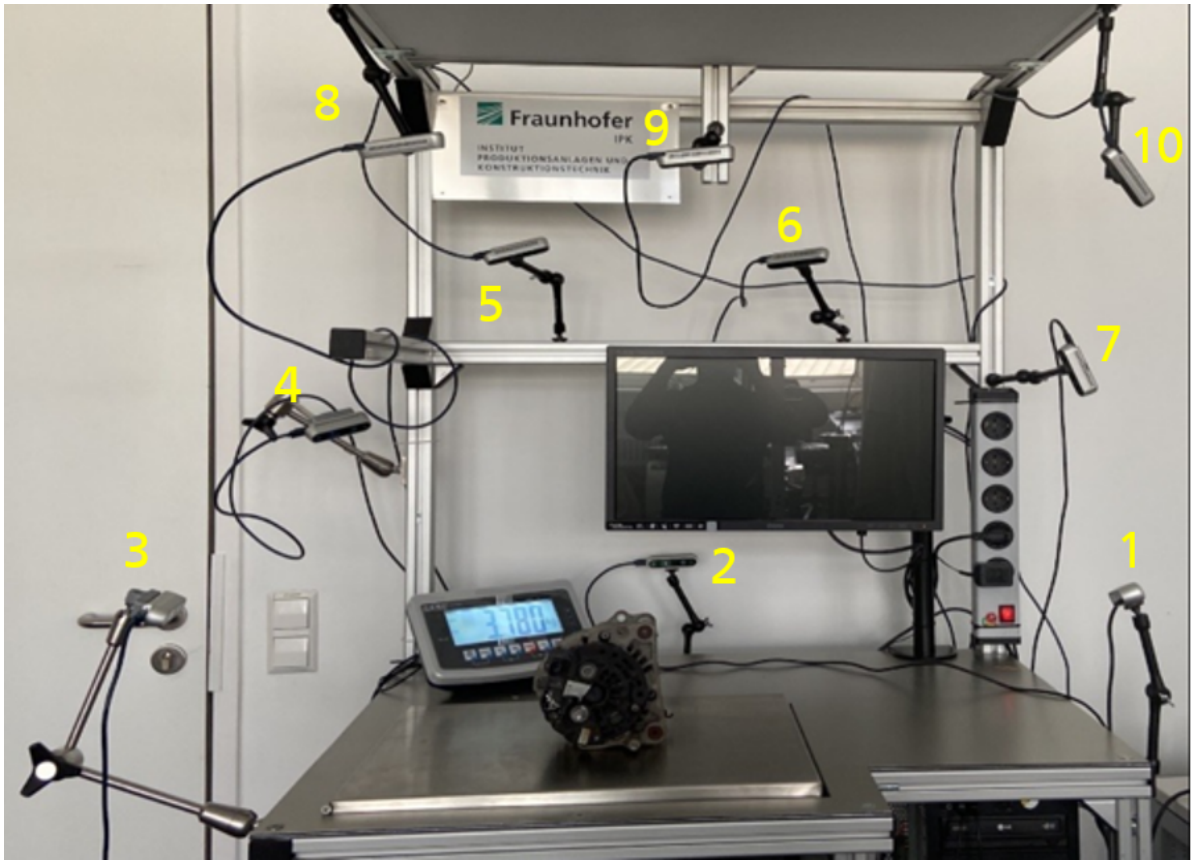


Figure 6.8: Prototype Setup of the EIBA 10 Camera System

Research Questions Addressed: *3, 5*

Consider the scenario when new objects are weekly digitalised, similar to the Class-IL case. However, each week, new images of the objects become available after the incremental training has been carried out, so the model needs to be trained intermittently on online learning and Class-IL steps. This scenario is tested on the Industrial-100 dataset. Two such trainings were performed. The first is where the data is homogeneous and the feature domain of the data is relatively similar during incremental learning and online learning. The second had the Industrial-100 subcategories split among the training sets for incremental learning and online learning.

Evaluation:

- *Top-1 accuracy on all the tasks.*
- *Energy consumption for each incremental learning and online learning task.*
- *A comparison of the energy consumption of the approach with an equivalent joint training*

6.4.3 Data logging and monitoring

The Class-IL experiments were logged locally and were also backed up on Wandb.ai.[1]
They will be publicly available as a supplement to the thesis.

The code implementation of the developments and experiments in this report will also be made available on the author's GitHub account.[2]

[1] See Appendix VI
[2] https://github.com/Vivek9Chavan/Green_Incremental_Learning

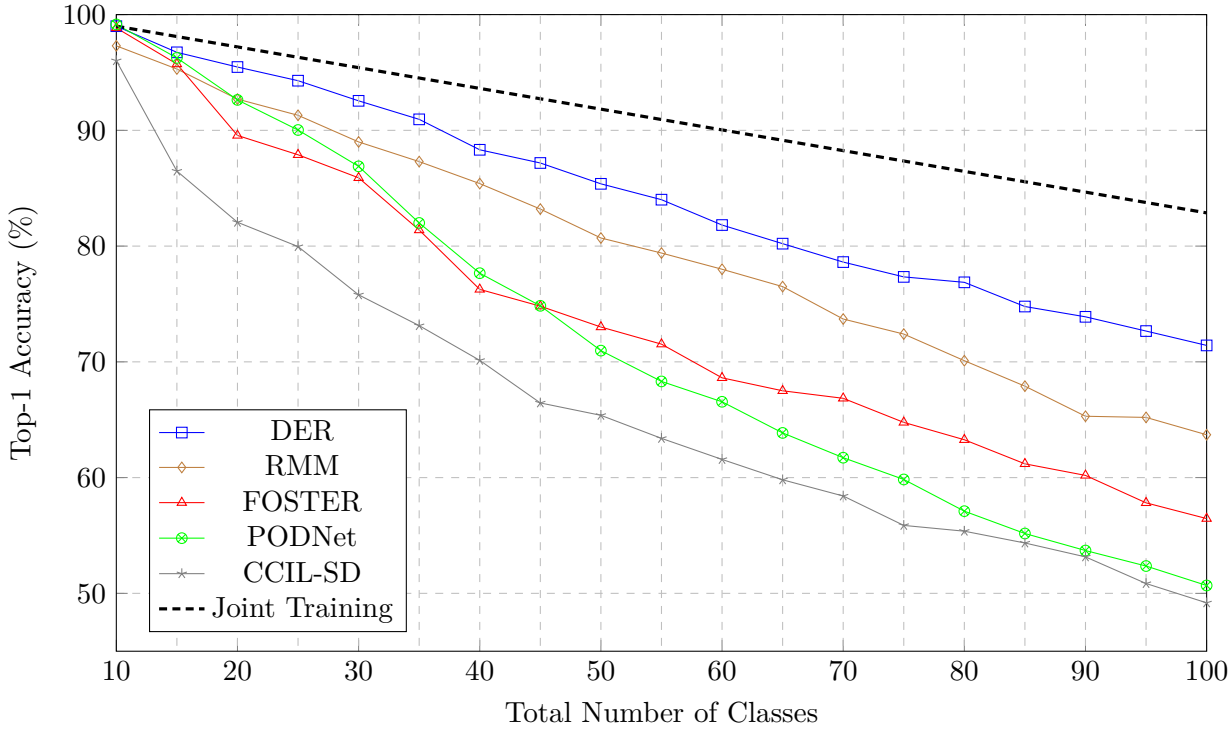
7 Results and Discussion

The results of the experiments are presented along with general inferences in their individual sections.

7.1 Experiment 1: Class-IL with fixed increments on the ImageNet-Subset

The Top-1 accuracy for the incremental learning approaches is given in Chart 6 below. Each method exhibits a significant drop in accuracy by the end of the 18 increments. The model performance was also assessed on the test dataset, and no statistical difference was observed between the test set and the validation set accuracies. Based on the Top-1 accuracy metric, the DER implementation outperforms other implementations, whereas the results from the CCIL implementation were markedly worse.

Chart 6: Accuracy for ImageNet-Subset with Constant Increment Sizes



The performance of the implementations on newer and older classes for this particular experiment has already been presented in the preliminary analysis (5). Taking those results with the added context of the results below shows that the DER implementation maintains high accuracy on old classes due to all the learned weights from older tasks being preserved in separate feature extractors. Whereas, the FOSTER implementation suffers from the opposite issue due to the compression of all the old knowledge into a single feature extractor which needs updating

Table 7.1: Results for Experiment 1

Task Number	PODNet	CCIL-SD	DER	FOSTER	RMM	Cumulative Joint Training
Training Time (h)	54.9	41.6	119.5	73.5	179.5	261.3
Power Consumption (kWh)	6.86	5.19	15.03	6.12	21.8	33.29
Cumulative Energy Consumption Ratio	**0.21**	**0.16**	**0.45**	**0.18**	**0.66**	–
Cumulative Accuracy Ratio	**0.78**	**0.72**	**0.92**	**0.81**	**0.87**	**1**
Top-1 accuracy at last stage	50.68%	49.18%	71.42%	56.46%	63.7%	84.25%
Average Incremental accuracy	71.57%	66.18%	84.29%	73.73%	79.70%	91.37%

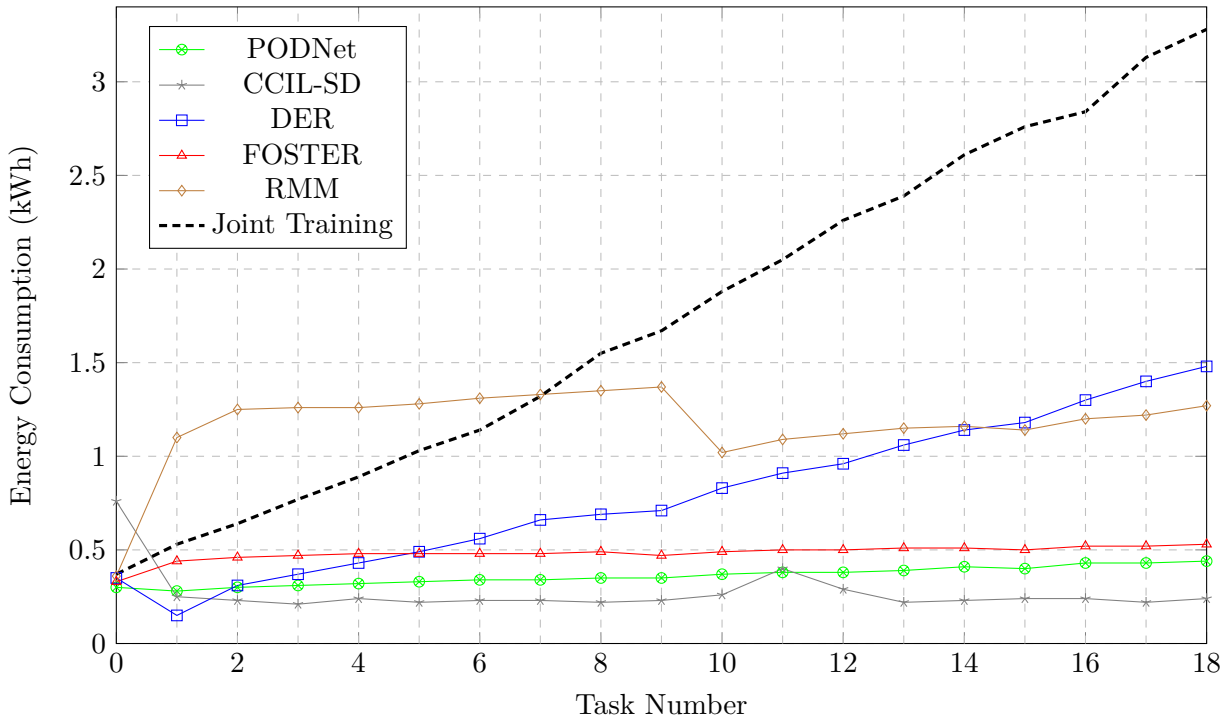
after every increment. The PODNet and the PODNet+AANet showed a greater homogeneity between the performance on older and newer classes.

A summary of all the key metrics from the first experiment is given in Table 7.1. Contrasting model accuracy with energy consumption metrics gives a more complete picture of the actual performance gain obtained from the incremental learning approach. The results show that the training time and the energy consumption tend to be approximately proportional to each other, provided that all the operating conditions are made as similar as possible for all the trainings. The **energy ratio** indicates the relative saving in computational cost that was achieved using a given Class-IL implementation. As discussed in 6, it is the ratio of the cumulative joint training energy consumption to the total energy consumption for the incremental learning training.

The energy consumption of each incremental learning step is shown in Chart 7. The total energy consumption of the entire training (as summarised in Table 7.1) is the area under the curve. As expected, the computational cost of carrying out joint training at each step is generally much higher than an equivalent incremental learning update. The number of parameters for the DER implementation increases linearly with the number of learning tasks; as a result, the computational cost also increases linearly. The RMM implementation uses PODNet + AANet and introduces model architecture modification in each residual block; based on the results, this directly corresponded to an increase in the training times and energy consumption during incremental training.

It can be inferred that the energy ratio would increase with an increase in the number of

Chart 7: Energy Consumption of individual tasks during Incremental Learning for Experiment 1



incremental tasks. This can also be interpreted from the cumulative training times given in Table 7.2. As discussed, the training times have a proportional relation to the energy consumption (and thus, also the energy ratio).

Table 7.2: Cumulative training times for different implementations (h)

Task Number	PODNet	CCIL-SD	DER	FOSTER	RMM	Joint Training
0	2.37	6.07	2.76	2.62	2.92	3.1
1	4.63	8.08	3.95	6.12	11.77	7.3
2	7.03	9.94	6.38	9.83	21.83	12.4
3	9.55	11.66	9.33	13.62	31.97	18.5
4	12.13	13.55	12.75	17.45	42.23	25.7
5	14.75	15.28	16.67	21.30	52.73	33.9
6	17.45	17.11	21.12	25.17	63.43	43.1
7	20.18	18.92	26.40	29.05	74.25	53.2
8	23.00	20.65	31.88	32.97	85.25	65.4
9	25.80	22.73	37.55	36.73	96.23	78.6
10	28.78	24.82	44.18	40.70	104.45	93.3
11	31.83	28.12	51.46	44.73	113.17	109.8
12	34.90	30.42	59.14	48.72	122.18	127.7
13	38.05	32.17	67.63	52.77	131.40	146.8
14	41.30	34.01	76.71	56.85	140.70	167.2
15	44.52	35.97	86.12	60.85	149.83	188.7
16	47.92	37.93	96.47	65.02	159.47	210.8
17	51.38	39.73	107.64	69.22	169.30	235.1
18	54.92	41.64	119.46	73.47	179.47	261.3

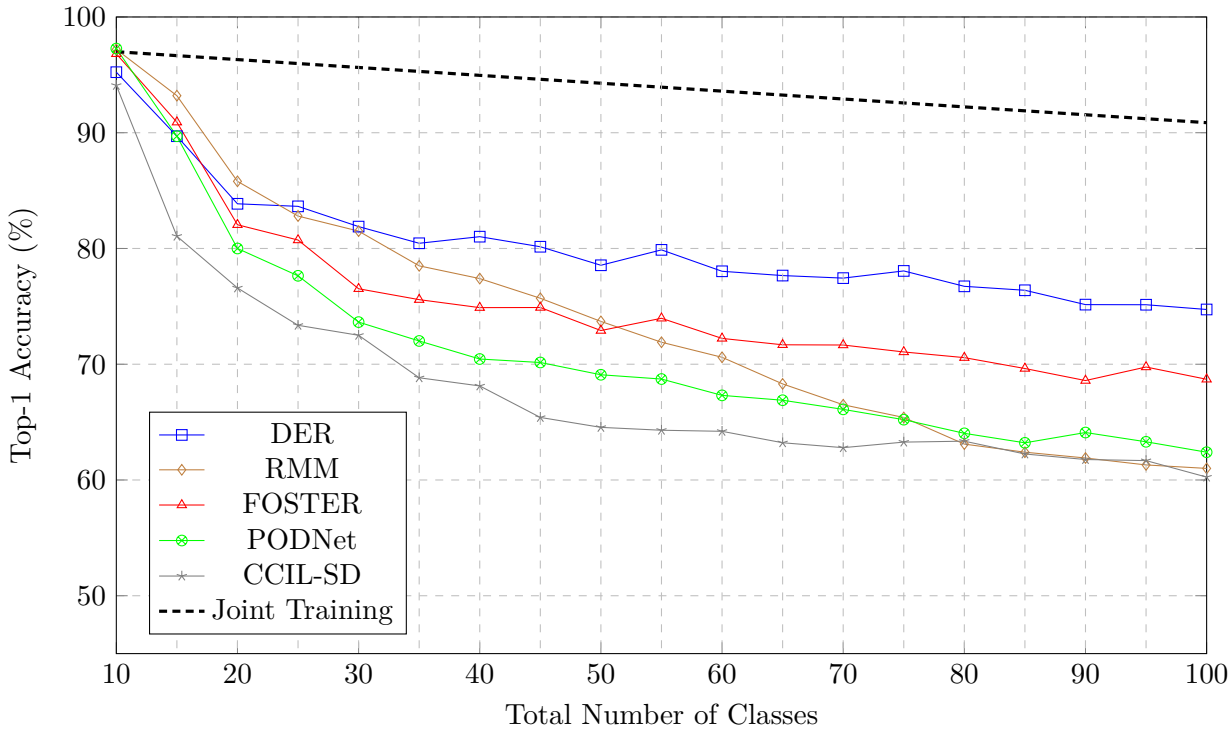
7.2 Experiment 2: Class-IL with fixed increments on the Industrial-100 dataset

The second experiment[1] was conducted on the Industrial-100 dataset with the same task increment conditions as Experiment 1. The results are compiled in the Table 7.3. The DER implementation, once again, outperformed other implementations. However, the margin between the accuracies at each step (and the resulting average incremental accuracy) for the different methods is less narrow.

Table 7.3: Results for Experiment 2

Task Number	PODNet	CCIL-SD	DER	FOSTER	RMM	Cumulative Joint Training
Top-1 accuracy at last stage	62.41%	60.25%	74.73%	68.7%	61.05%	90.87%
Average Incremental accuracy	71.12%	67.98%	80.04%	71.66%	73.59%	93.94%
Accuracy Ratio	**0.76**	**0.72**	**0.85**	**0.76**	**0.78**	**1**

Chart 8: Accuracy on Industrial-100 with Constant Increment Sizes

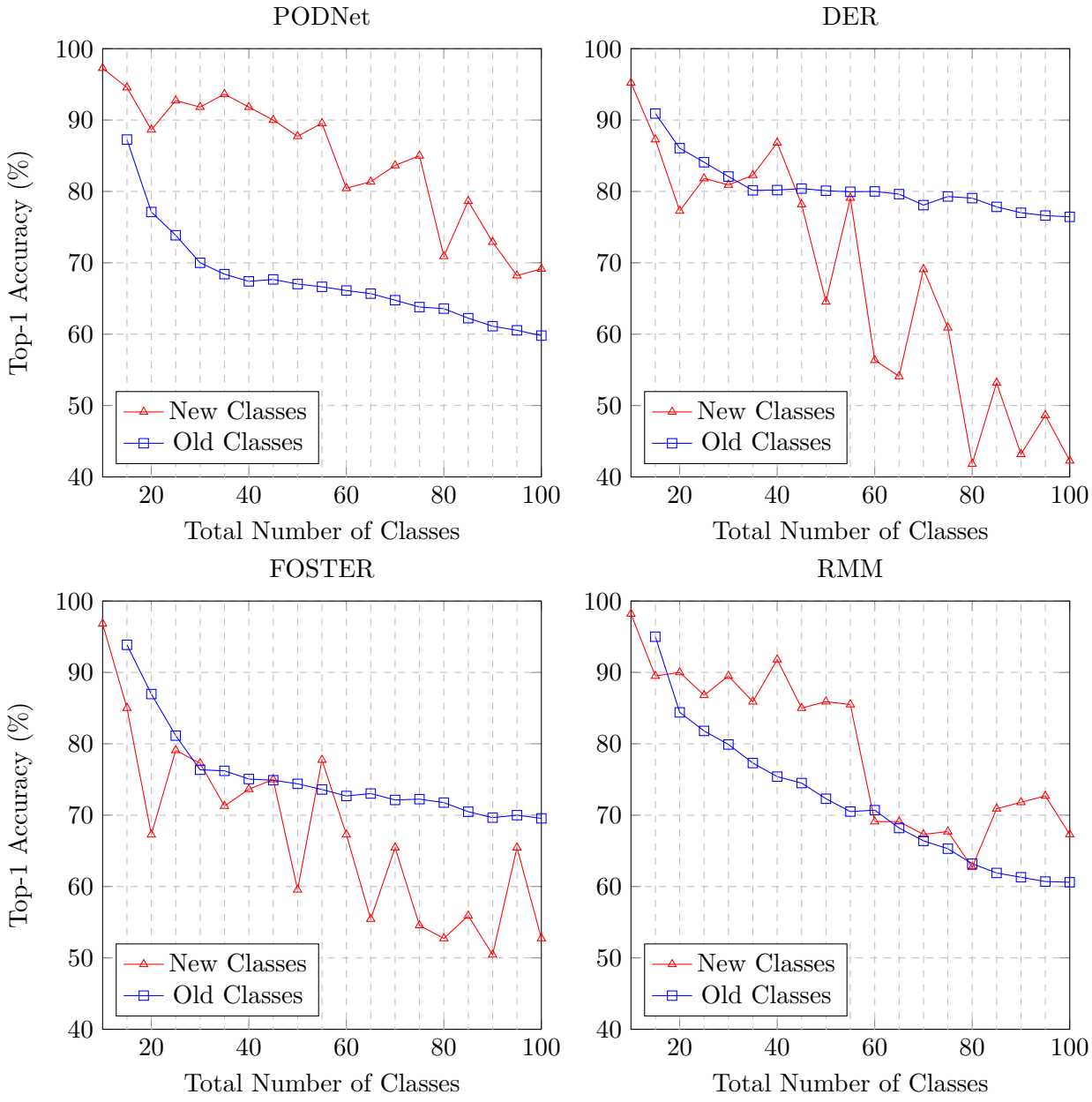


The performance comparison between the old and newer classes is shown in the plots for the

[1]This experiment was done on another workstation where the energy consumption was not tracked and logged.

individual implementations. [2] The results show a similar pattern as Experiment 1. However, the FOSTER implementation was seen to struggle with the new classes, which contrasts with the previous results. This can be attributed to the difference in the intra-class and inter-class feature distribution for the Industrial-100 dataset. The number of stored exemplars per class was strictly kept constant for each increment. As a result, the performance difference between the RMM and the PODNet (with herding) implementation vanishes.

Chart 9: Performance on old and new classes for Experiment 2 with Industrial-100



The energy consumption metrics were not tracked for this experiment. However, the training time pattern was closely similar to that from Experiment 1 for the different implementations.

[2] This comparison was not measured and tracked for the CCIL implementation.

Table 7.4: Results for Experiment 3

Task Number	PODNet	CCIL-SD	DER	FOSTER	RMM	Cumulative Joint Training
Training Time (h)	96.9	87.3	110.1	111.4	188.6	138.19
Power Consumption (kWh)	11.7	10.5	15.0	13.4	23.2	17.5
Cumulative Energy Consumption Ratio	**0.67**	**0.59**	**0.85**	**0.76**	**1.33**	–
Cumulative Accuracy Ratio	**0.956**	**0.81**	**0.921**	**0.972**	**0.902**	**1**
Top-1 accuracy at last stage	82.89%	72.10%	80.29%	84.04%	81.37%	90.87%
Average Incremental accuracy	88.65%	74.01%	86.65%	90.45%	87.06%	93.94%

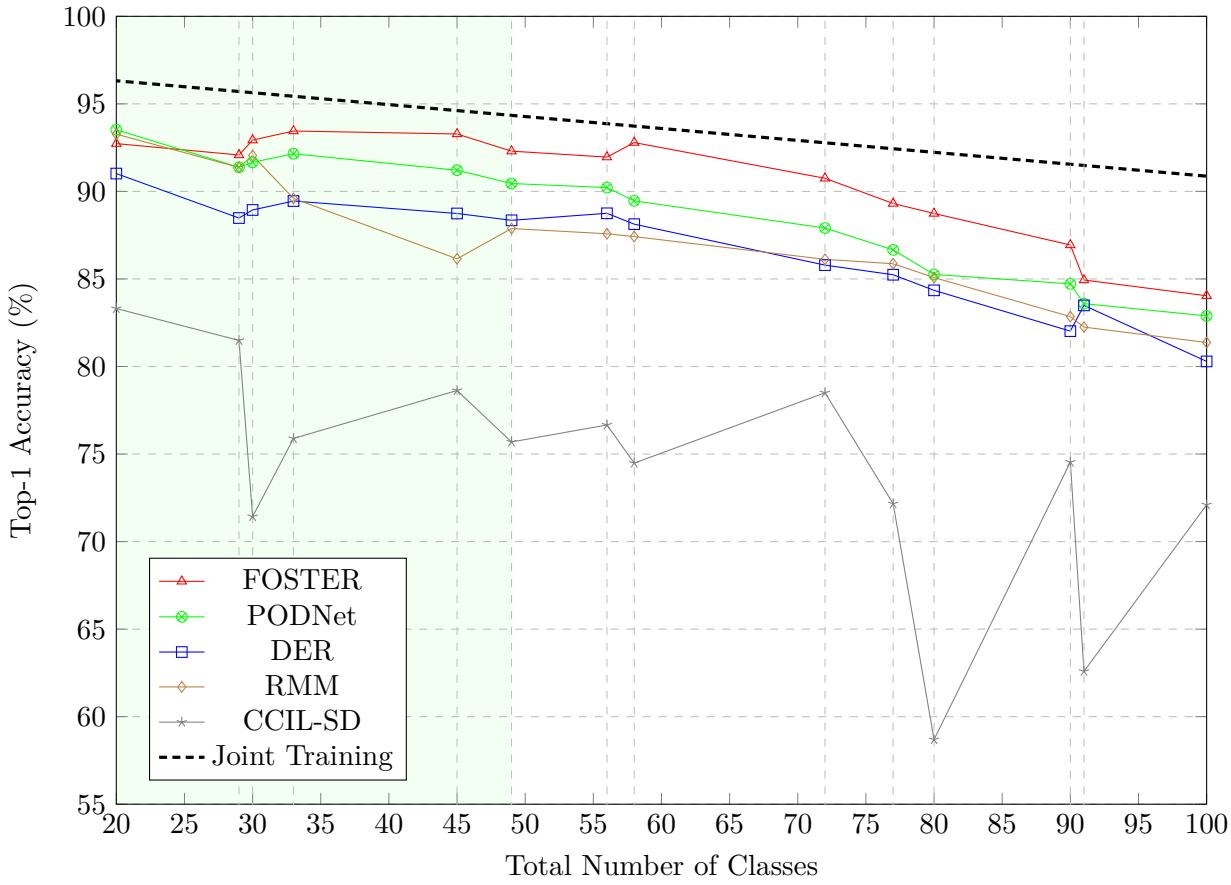
7.3 Experiment 3: Class-IL with variable increments on the Industrial-100 dataset

This experiment was conducted on the Industrial-100 dataset with a variable task increment sequence. The number of exemplars stored per class was also increased fourfold, such that until the 5th increment, all the data from the older classes is retained. This part of the sequence is highlighted in all the plots below in the shaded (light green) region. As a result, it can be seen that the general accuracy values were much higher for most implementations compared to the previous experiment. However, despite retaining all the data from the previous classes, joint training performance was not matched for most implementations. The performance on the four subcategories of the dataset is shown for the individual implementations.

The table 7.4 gives a summary of the key metrics for this experiment. Due to an increase in the stored exemplar data for each increment, the training times and energy consumption of the class-IL implementations have increased. In fact, the RMM implementation has a cumulative energy ratio of 1.33, which means that, for the given conditions, conducting joint training at each increment at each step would be cheaper. The ratio values for other implementations are significantly lower as well. A positive effect of the higher exemplar count is that the average incremental accuracy values match the joint training performances more closely. However, based on this performance-to-cost comparison, the strategy of having more exemplars does not offer a distinctive boost.

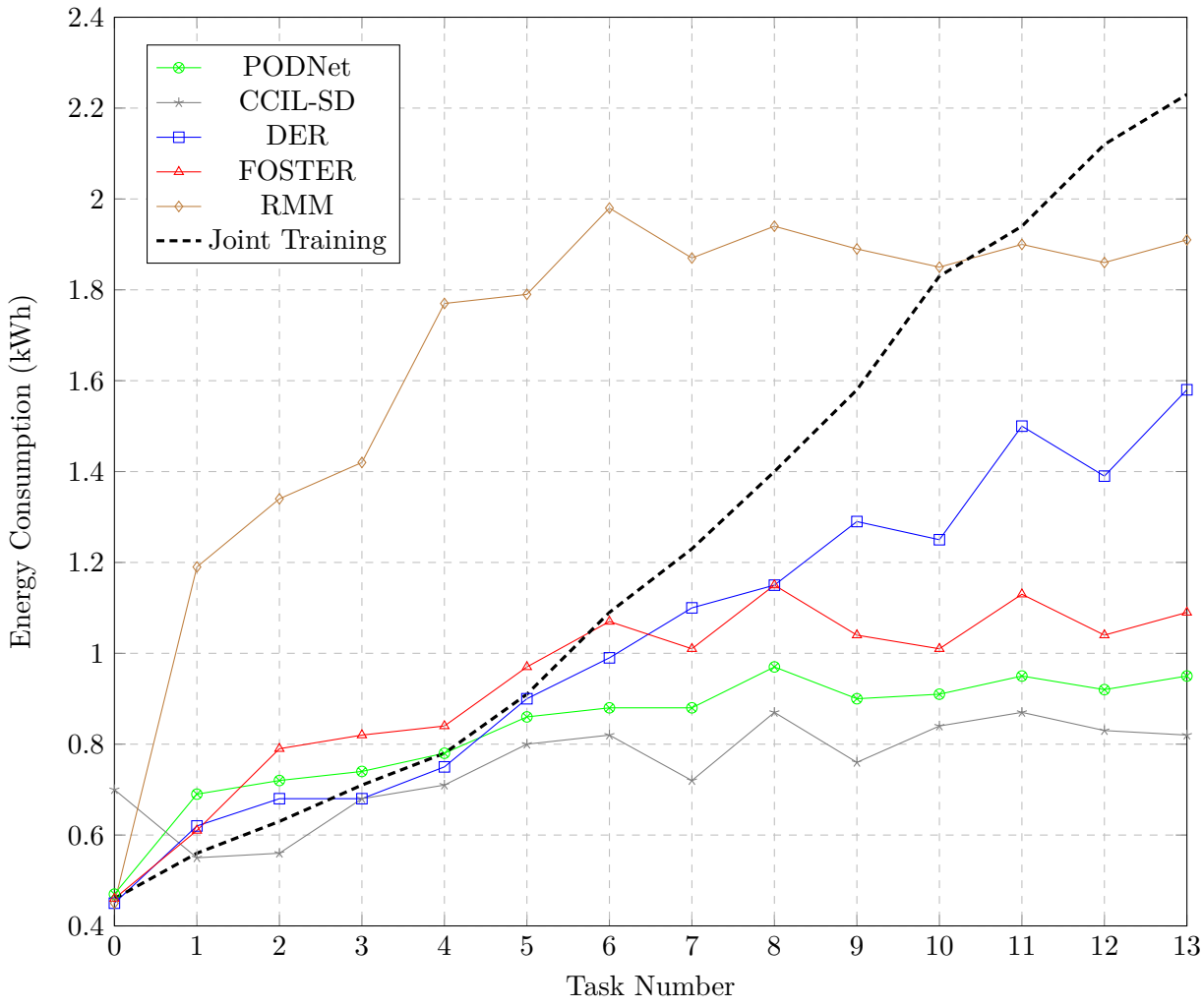
The energy consumption values for each step in the learning sequence are shown in Chart 11

Chart 10: Accuracy results for Experiment 3: Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.



below. Similar to Experiment 1, the area under the curve would yield the cumulative energy consumption for the entire training. The relative patterns in the energy consumption between the different implementations remain the same as in Experiment 1. However, there is a less noticeable separation between the curves due to the higher computational load at each increment. Secondly, since the total number of increments is lower (13, as opposed to 18 for Experiment 1), the difference between the cumulative joint training and other Class-IL implementations is less apparent. This yields an important conclusion that the energy saving from incremental learning would be higher if the model can be incrementally trained on a higher number of increments. Additionally, the change in the number of new classes or training data does not significantly impact energy consumption; rather, the model architecture and the increment count have a much greater impact.

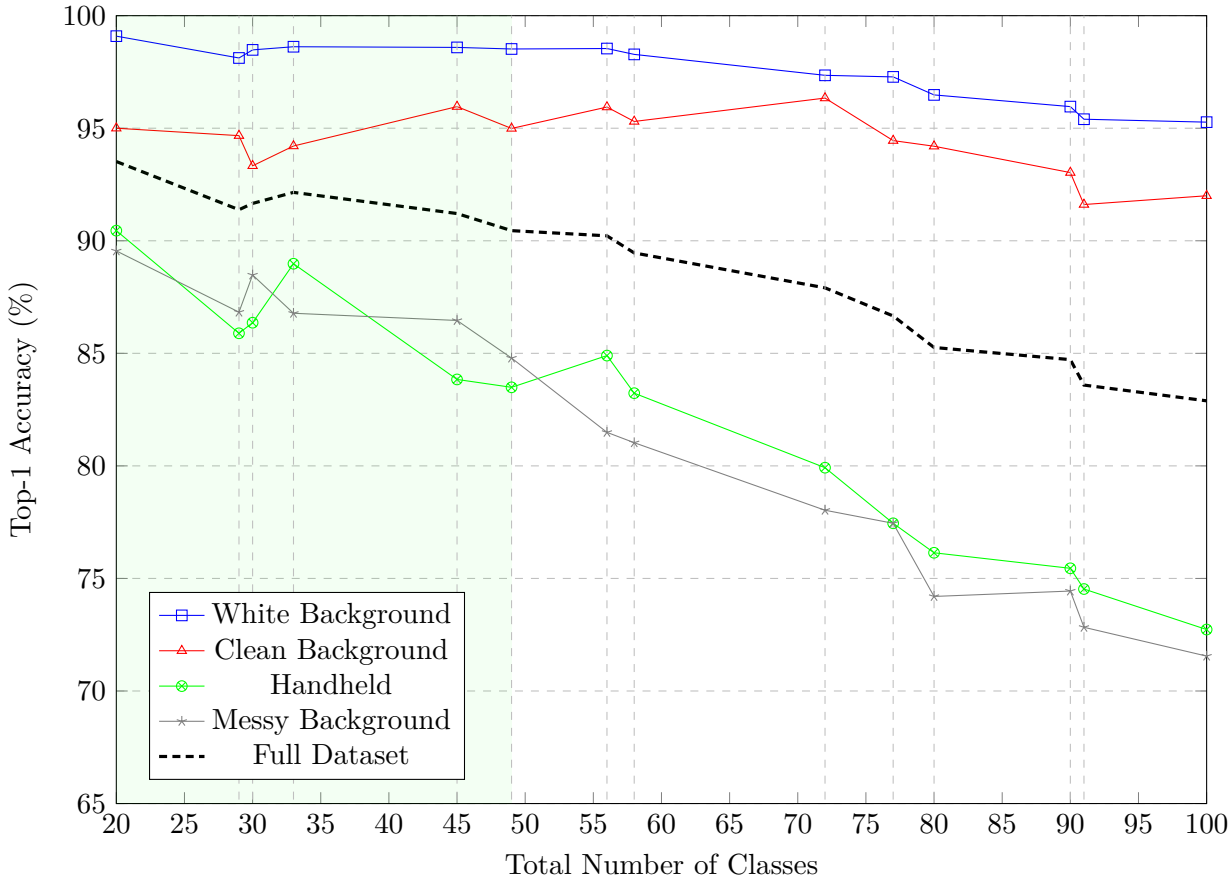
Chart 11: Energy Consumption of individual tasks during Experiment 3



PODNet: The model performance on the Industrial-100 dataset subcategories is elaborated in the following plot. The accuracy on the data with white and cleaner backgrounds is significantly higher and closely resembles joint training performance. The data with handheld and messy

backgrounds exhibits a much lower performance. Secondly, greater catastrophic forgetting is seen on the data with higher visual complexity.

Chart 12: PODNet: Accuracy for Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.



The performance on the older and newer classes has also been broken down for the individual subcategories. For PODNet, the performance on both old and new classes remains comparable, which confirms the original claim by the authors of the implementation regarding its suitability for bigger increment sequences.

CCIL-SD: This implementation showed markedly worse performance on the data with variable incremental sequences. [3] This indicates a mismatch between model behaviour under ideal conditions and practical real-world scenarios. Especially, a significant drop in the Top-1 accuracy can be seen in cases where a large increment is followed directly by a small increment or vice versa.

DER: The performance on the different subcategories of the dataset shows a similar pattern to that of the PODNet implementation.

[3] The trainings failed to finish on the original workstation. The experiment was repeated on another computer and the equivalent time and energy consumption values were measured.

Chart 13: PODNet: Performance on old and new classes for Experiment 3. Earlier increments (shaded region) have access to all old data in the form of exemplars.

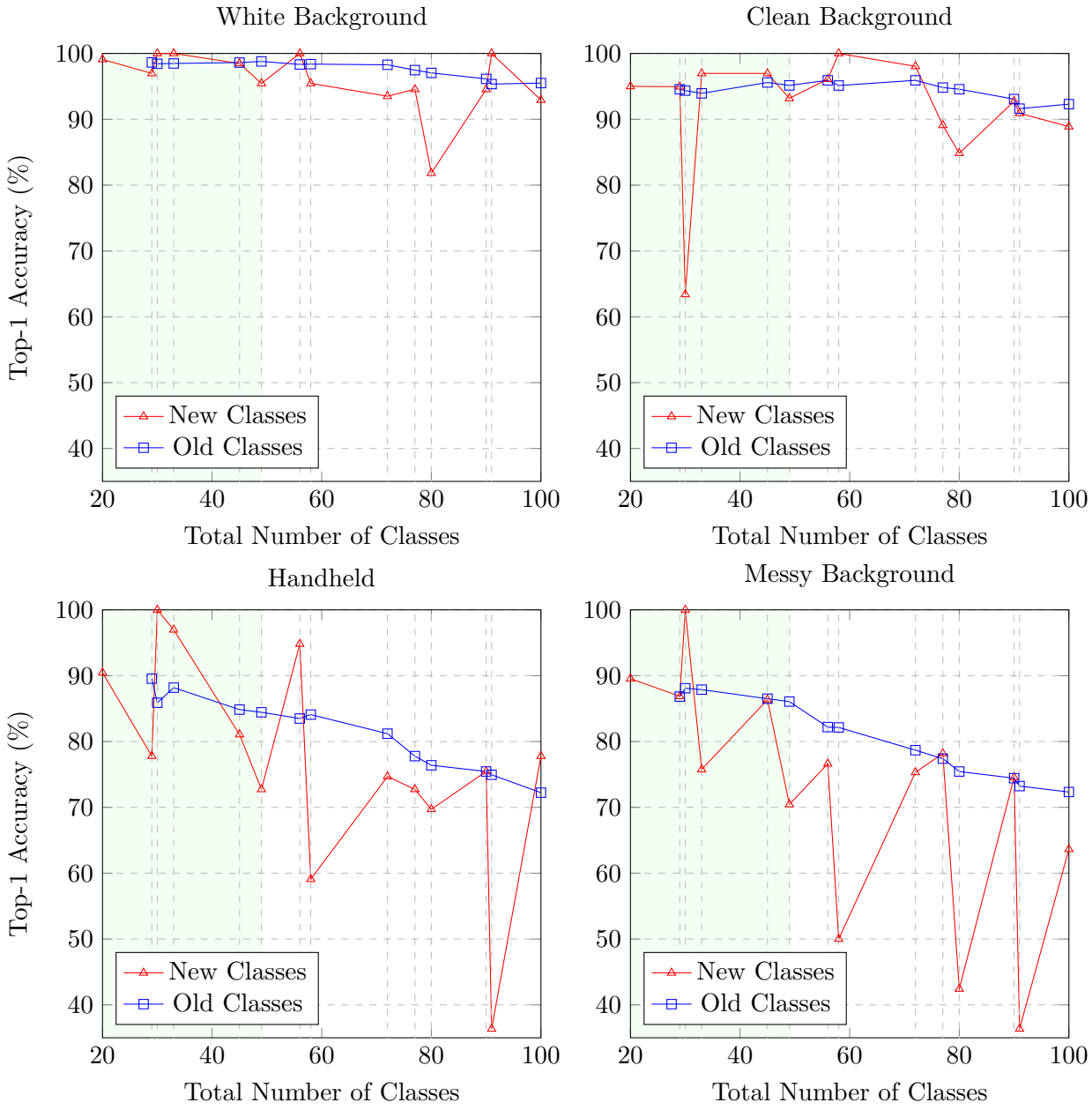


Chart 14: CCIL-SD: Accuracy on Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.

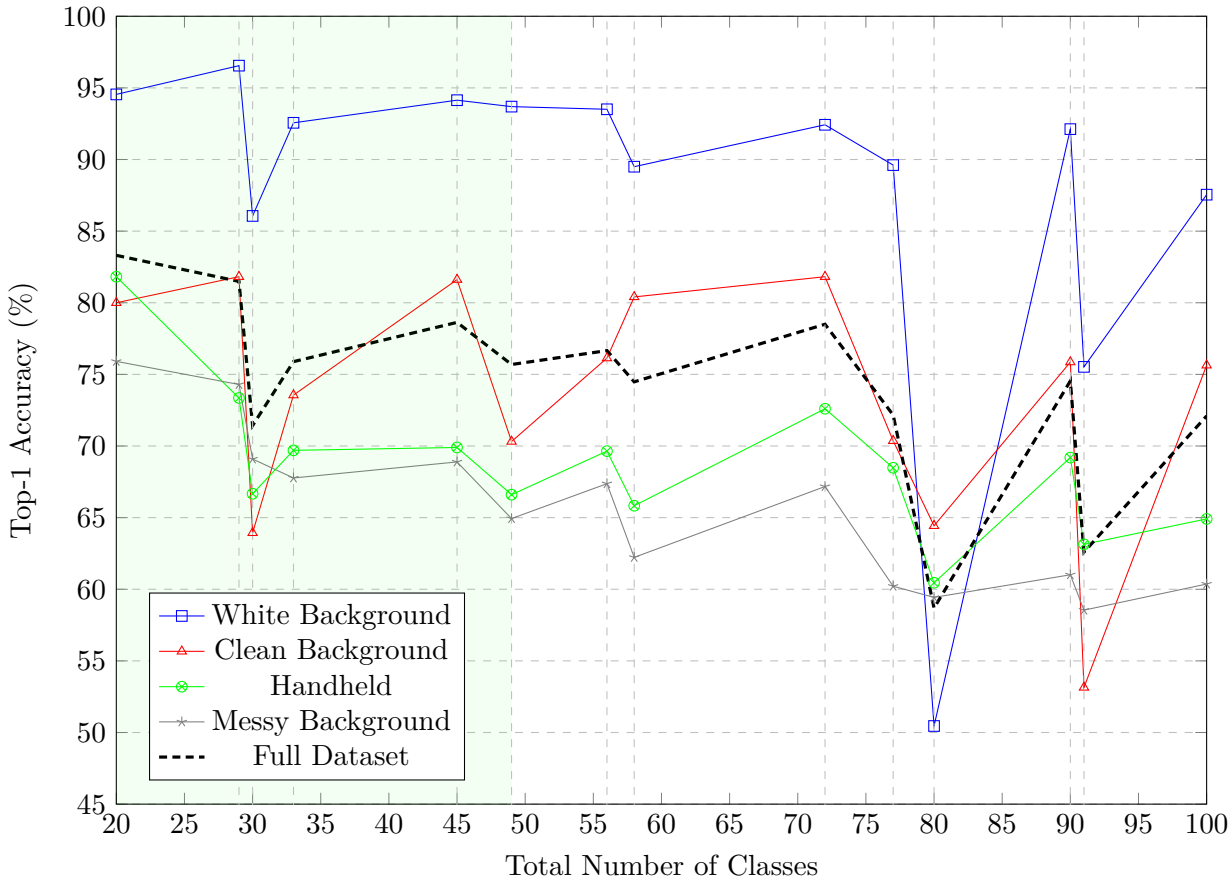
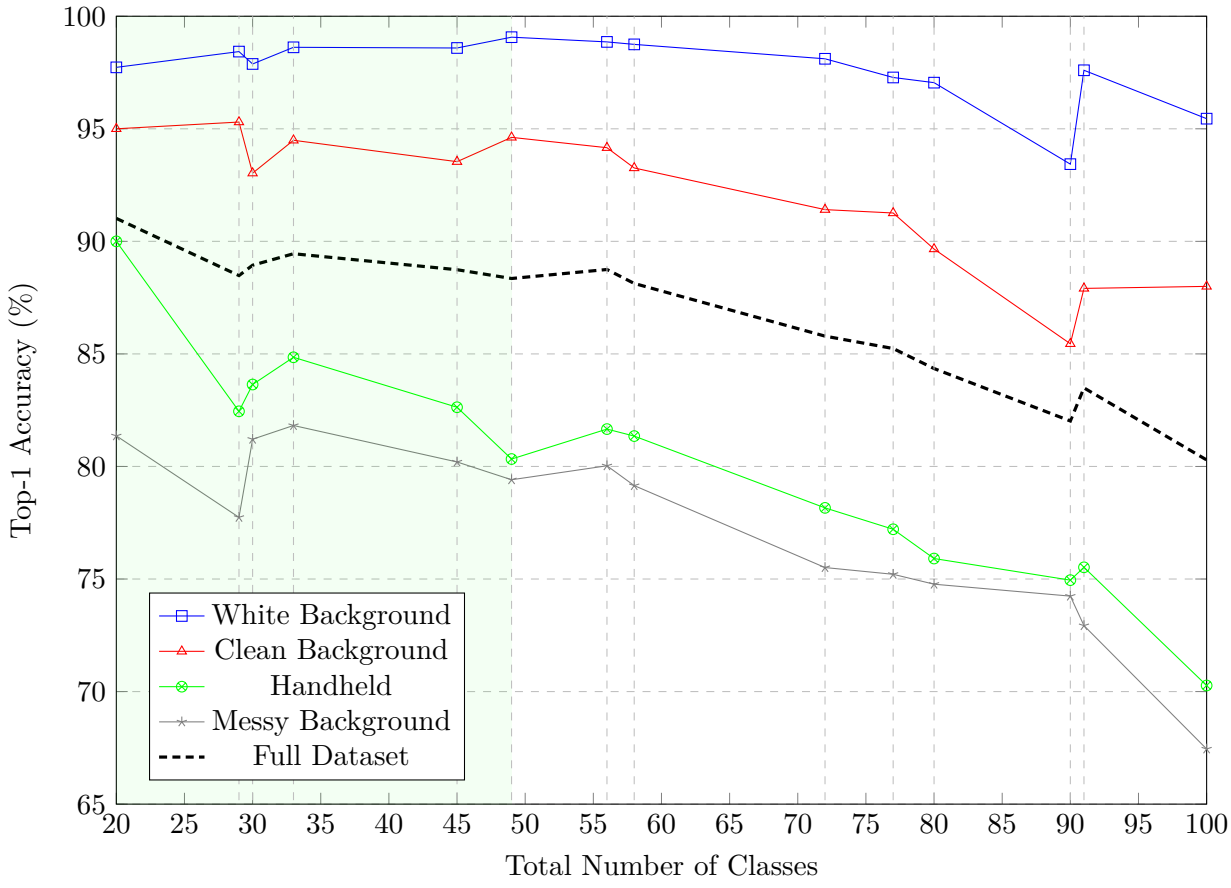
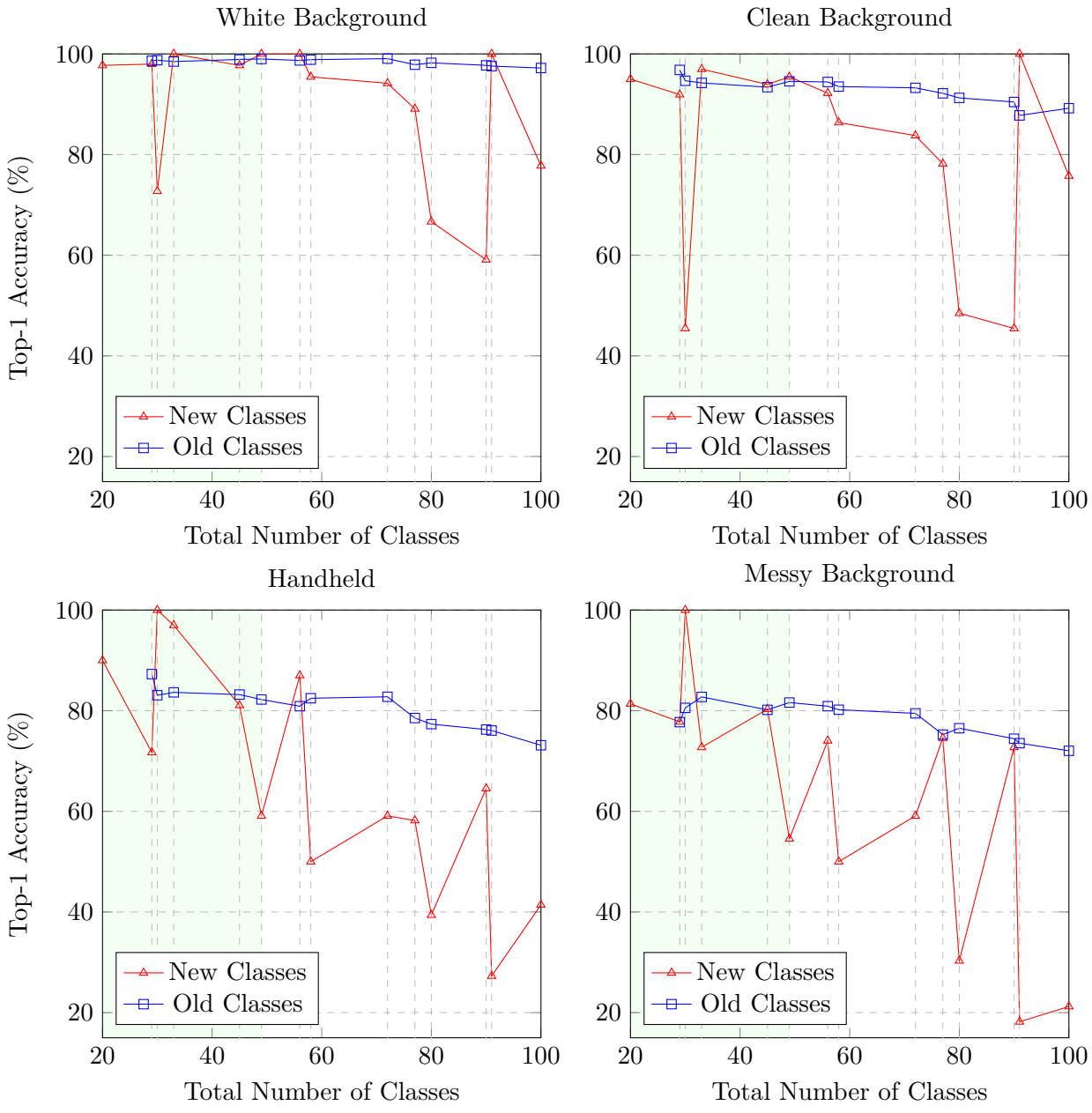


Chart 15: DER: Accuracy for Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.



Inspecting the accuracies for old and new class data shows that even though model performance on new classes for DERNet traditionally drops with an increasing number of feature extractors, the drop can be much lower if the data is clean and has lesser visual complexity. On the other side, the performance on the handheld and messy background data drop so precipitously, that it can be inferred that adding new increments beyond 90 classes was futile. In fact, taking the increasing number of parameters into context, and adding more increments would also be uneconomical.

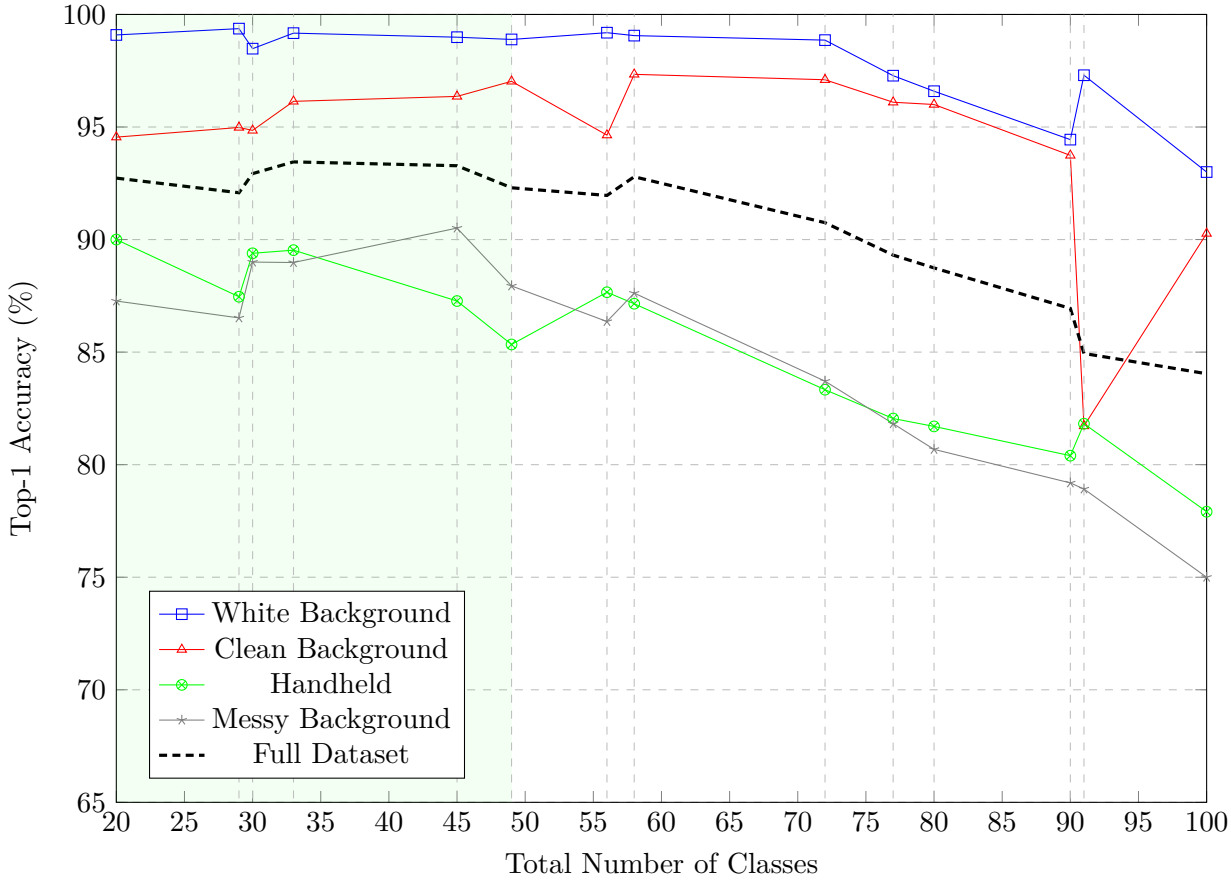
Chart 16: DER: Performance on old and new classes on Experiment 3. Earlier increments (shaded region) have access to all old data in the form of exemplars.



FOSTER: The performance on the cleaner dataset matches the joint training performance,

especially during the initial increments. This implementation incorporates feature compression and distillation of the extractor after each step. This introduces an additional loss parameter that governs the ability of the model to learn newer classes and remember the old classes.

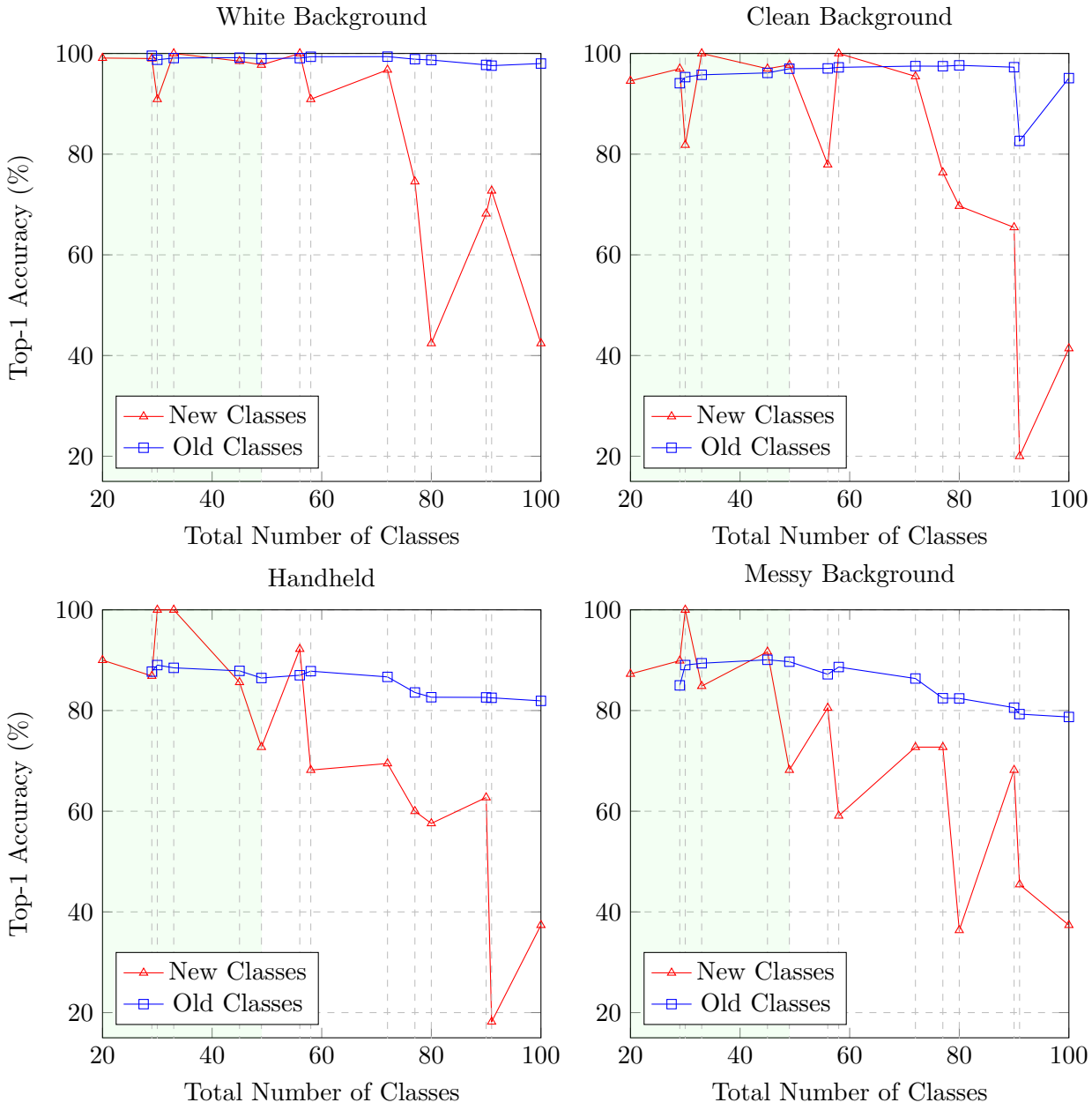
Chart 17: FOSTER: Accuracy for Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.



This is patently clear in the plots showing the old and new class performance. The model has nearly perfect performance for data with low complexity. However, the ability of the model to learn newer classes was observed to sharply decreases for all subcategories. This can be attributed to the variable sequence, especially smaller ones. Contrary to traditional models, the FOSTER implementation needs to learn the new classes and then compress the learned weights with the old weights. Hence, with smaller class increments, the training can be detrimental to the overall model performance. This implementation would not be suitable for a long sequence of increments, similar to the DER implementation.

RMM (PODNet + AANet): The Top-1 accuracies closely resemble those from the PODNet results discussed earlier in this section. The conditions of Experiment 3 serve to diminish the effect of the RMM exemplar selection strategy since a large portion of the data is maintained throughout. As a result, the performance was no better than that of the base PODNet im-

Chart 18: FOSTER: Performance on old and new classes for Experiment 3. Earlier increments (shaded region) have access to all old data in the form of exemplars.



plementation. Additionally, given the higher computational cost of the architecture, the overall performance was comparatively inferior. The plots of old and new class accuracy follow a similar pattern to other implementations and do not yield any additional insights.

Chart 19: RMM: Accuracy for Variable Class Incremental Learning Scenario. Earlier increments (shaded region) have access to all old data in the form of exemplars.

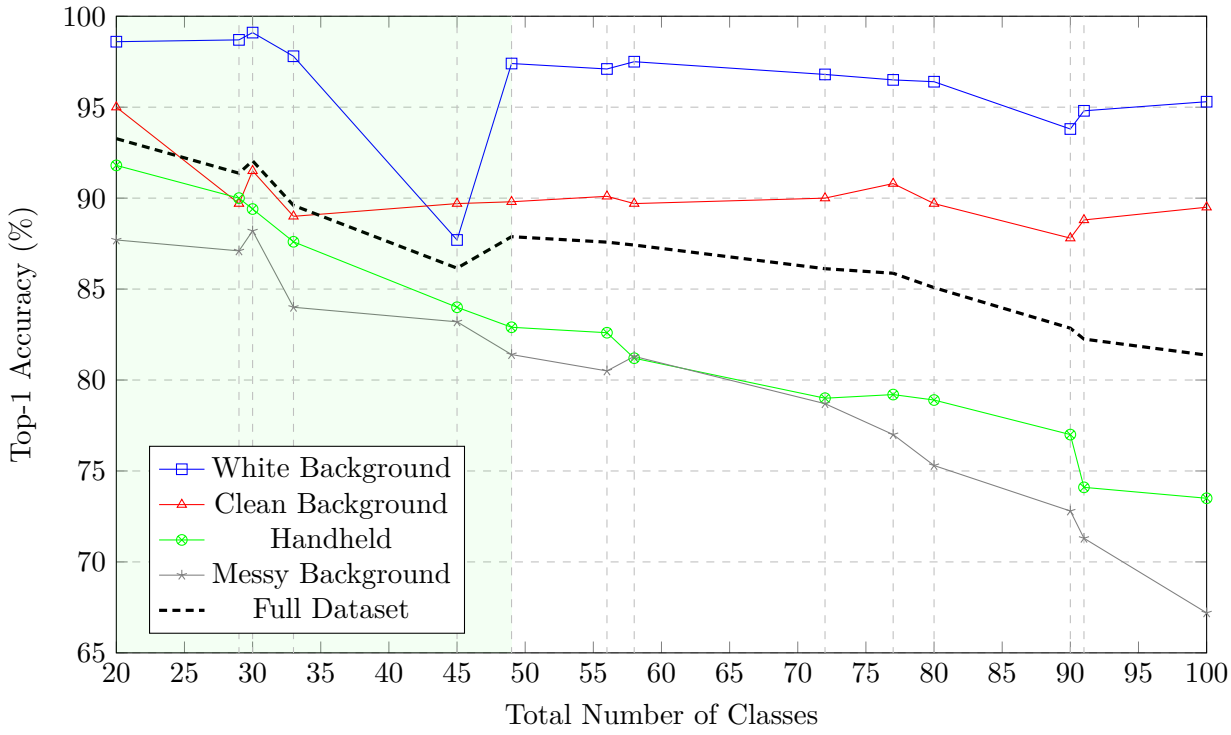
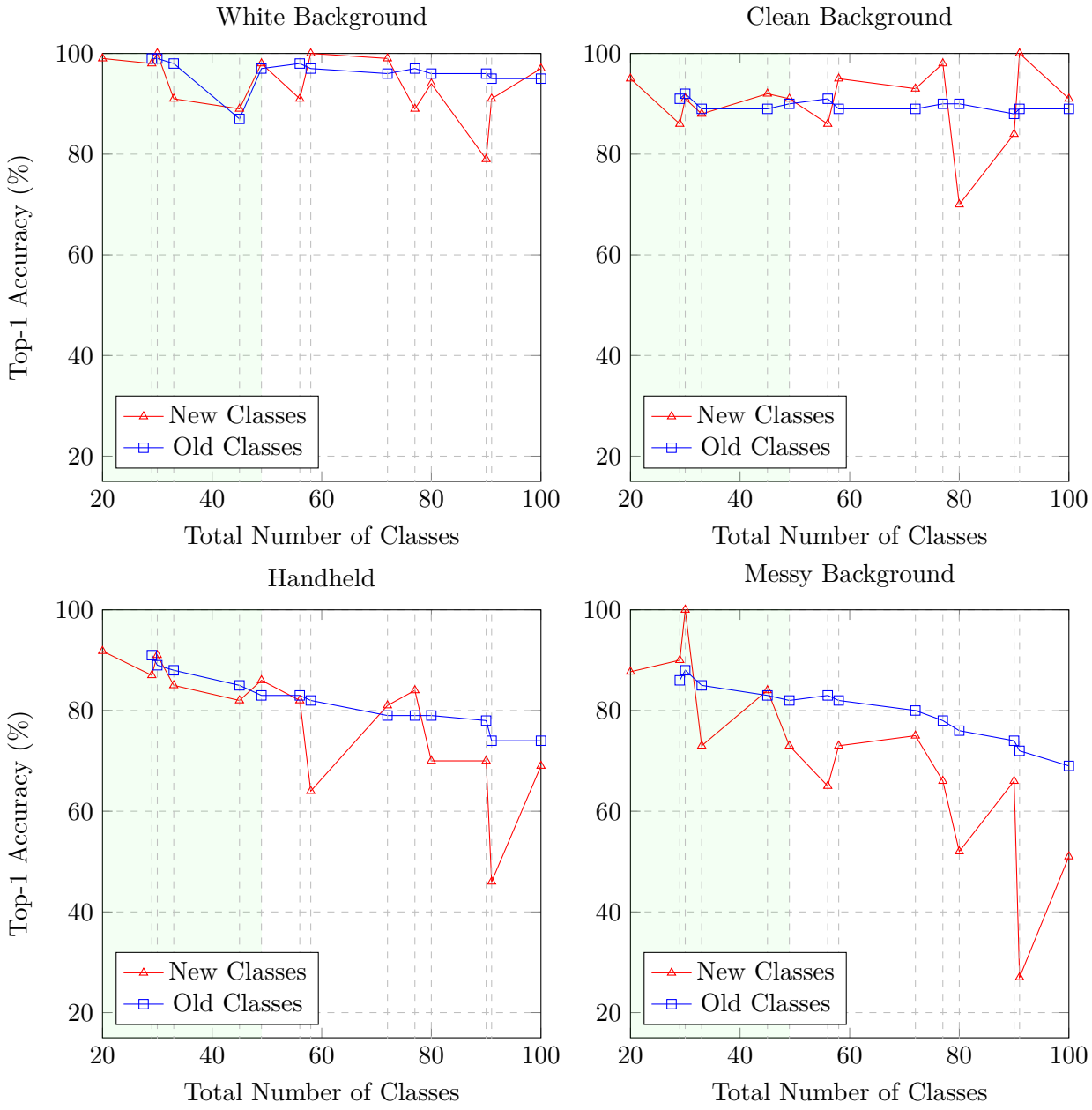


Chart 20: RMM: Performance on old and new classes for Experiment 3. Earlier increments (shaded region) have access to all old data in the form of exemplars.



7.4 Experiment 4: Class-IL with second variable increment sequence on the Industrial-100 Subset (Clean background data)

This experiment was performed as a supplement to the larger Experiment 3. With a variable increment sequence, it can be questioned whether a specific sequence can be deliberately or accidentally engineered to favour one implementation over another. This experiment was performed under similar conditions with a different sequence of class increment sizes. Table 7.5 clearly shows that the average incremental accuracy from both experiments closely matches one another, except for the CCIL implementation, whose issues have already been discussed. Thus, no evidence can be observed in favour of incremental size bias and the inferences drawn from the experiments can be verified to hold true under varying scenarios. The energy consumption from Experiments 3 and 4 was also nearly identical and has not been presented for the sake of pithiness.

Chart 21: Accuracy with the modified variable Class Incremental Learning sequence in Experiment 4. Earlier increments (shaded region) have access to all old data in the form of exemplars.

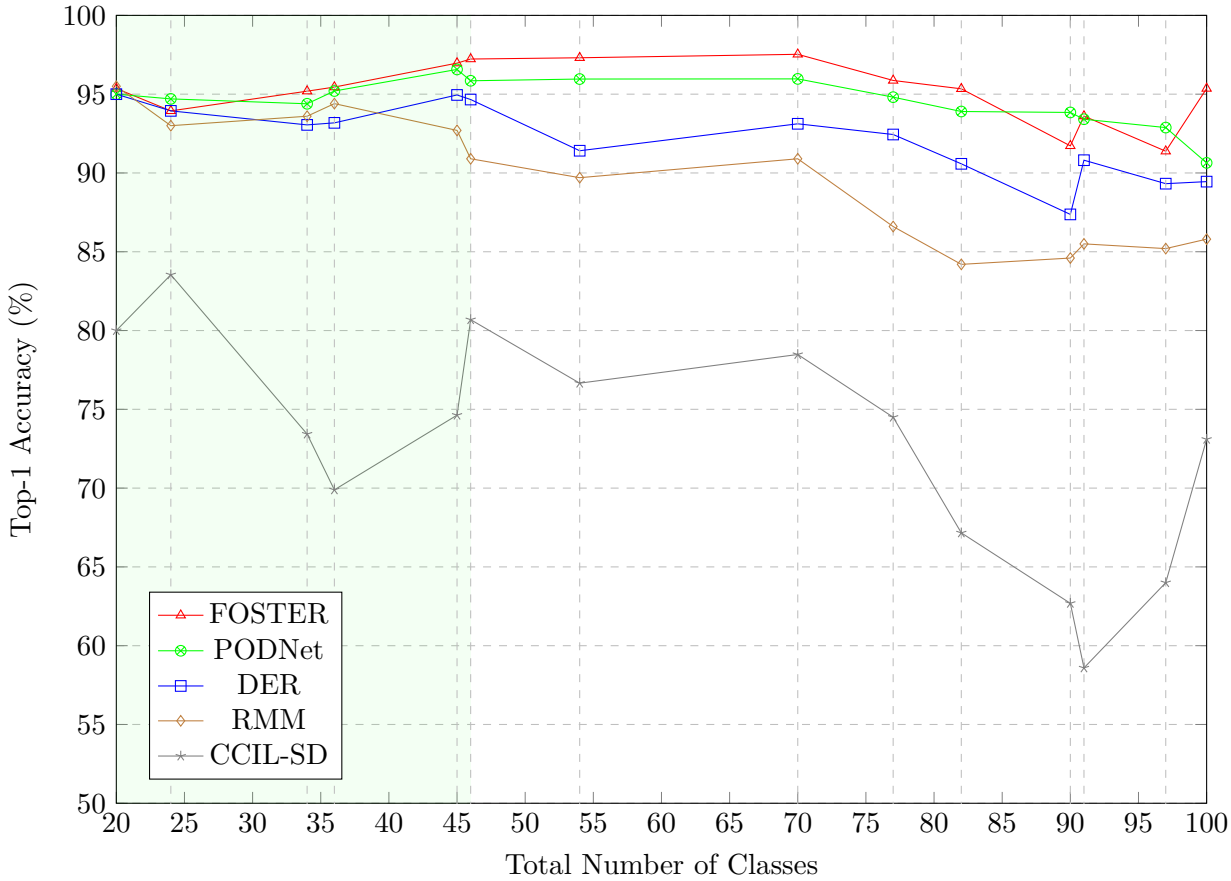


Table 7.5: Results for Experiment 4

Task Number	PODNet	CCIL-SD	DER	FOSTER	RMM
Experiment 3: Average Incremental accuracy	94.4%	73.51%	91.95%	94.35%	90.0%
Experiment 4: Average Incremental accuracy	94.61%	77.13%	92.09%	95.16%	89.48%
Exp 3 Cumulative Accuracy Ratio	**0.964**	**0.853**	**0.924**	**0.971**	**0.912**
Exp 4 Cumulative Accuracy Ratio	**0.973**	**0.884**	**0.932**	**0.982**	**0.904**

7.5 Experiment 5: Exemplar Selection for Class Incremental Learning using Dino

Chart 22 below shows the performance of the Dino exemplar selection approach against the state-of-the-art herding approach for the PODNet implementation. The size of the exemplar set was set to 20 per class for all increments. It can be seen that the strategy using Dino outperforms herding. Additionally, the PODNet implementation employs fine-tuning epochs to allow the model to resolve the class imbalance. By default, this is also carried out on the exemplar set, hence, if the exemplar set maintains the original data distribution, the model is able to maximize its learning.

The plots in Chart 23 below show a comparison of the two strategies for DER implementation. It can be observed that the Dino exemplar selection approach outperforms the baseline. It can be conjectured the size of the increment should have little bearing on the performance gain obtained by an exemplar selection strategy. This is verified by the results shown. The newly developed approach remains effective regardless of the scenario.

Thirdly, the strategy comparison is given for the FOSTER implementation (Chart 24). Thus, the Dino exemplar selection approach developed during this project proves to be a superior strategy in varied scenarios and CIL implementations.

Variable exemplar selection: The scenario of imbalance in the class feature distribution was studied on the Industrial-100 imbalanced dataset, which is a derivative of the original dataset. The strategy aims to maintain the same number of exemplars while distributing the

Chart 22: Comparison of Dino exemplar selection approach against Herding using PODNet implementation

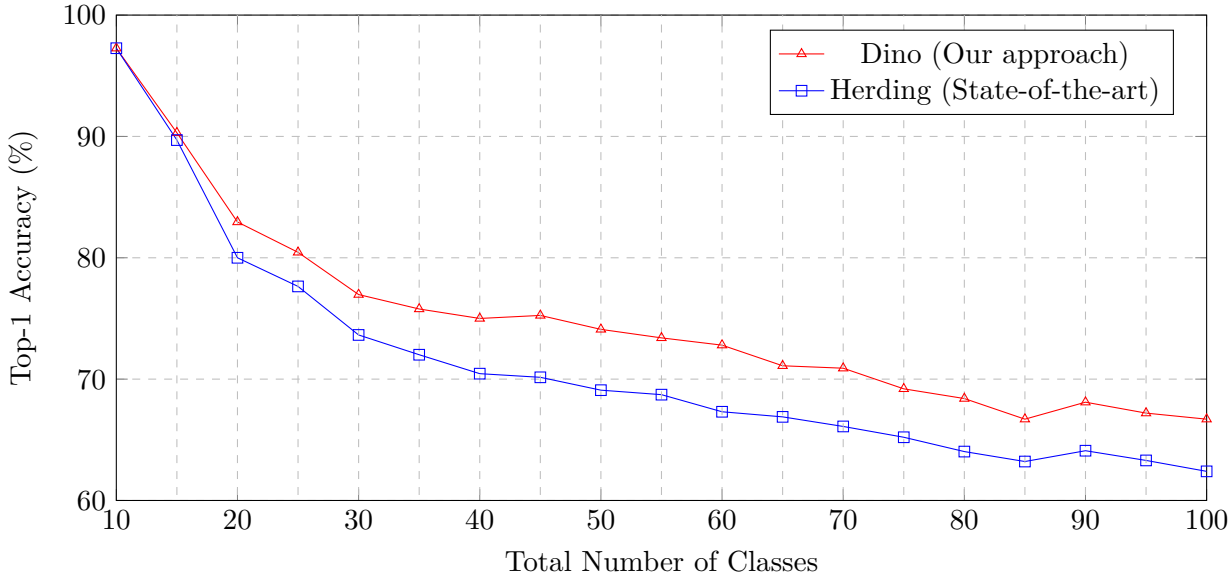


Chart 23: Comparison of Dino exemplar selection approach against Herding using DER implementation

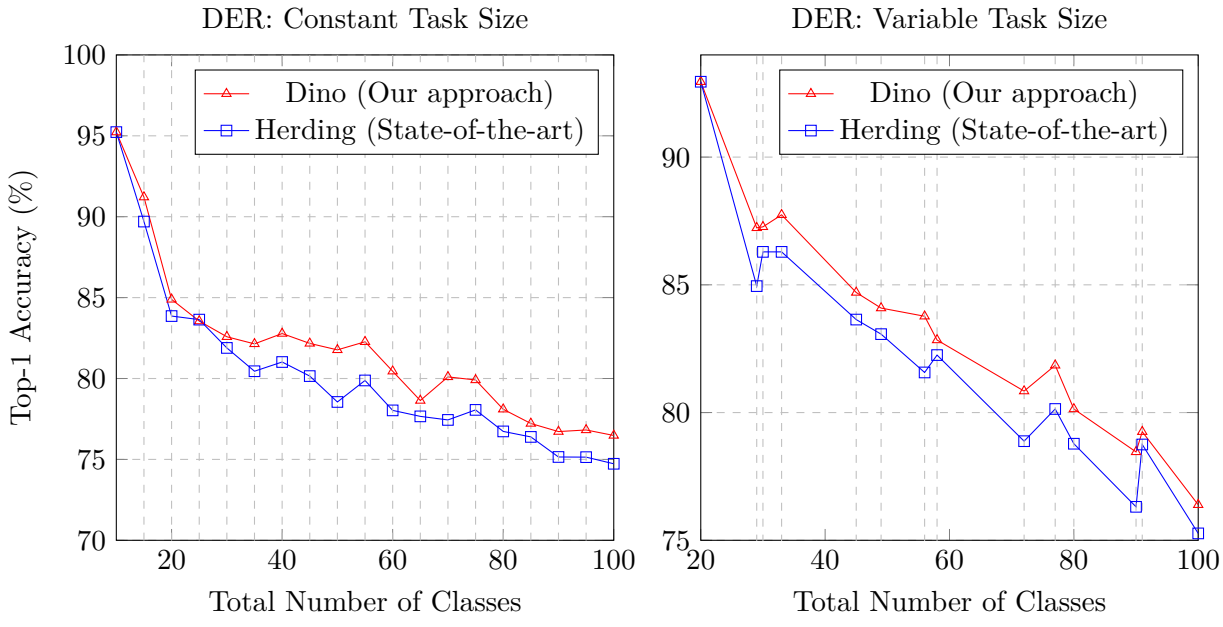
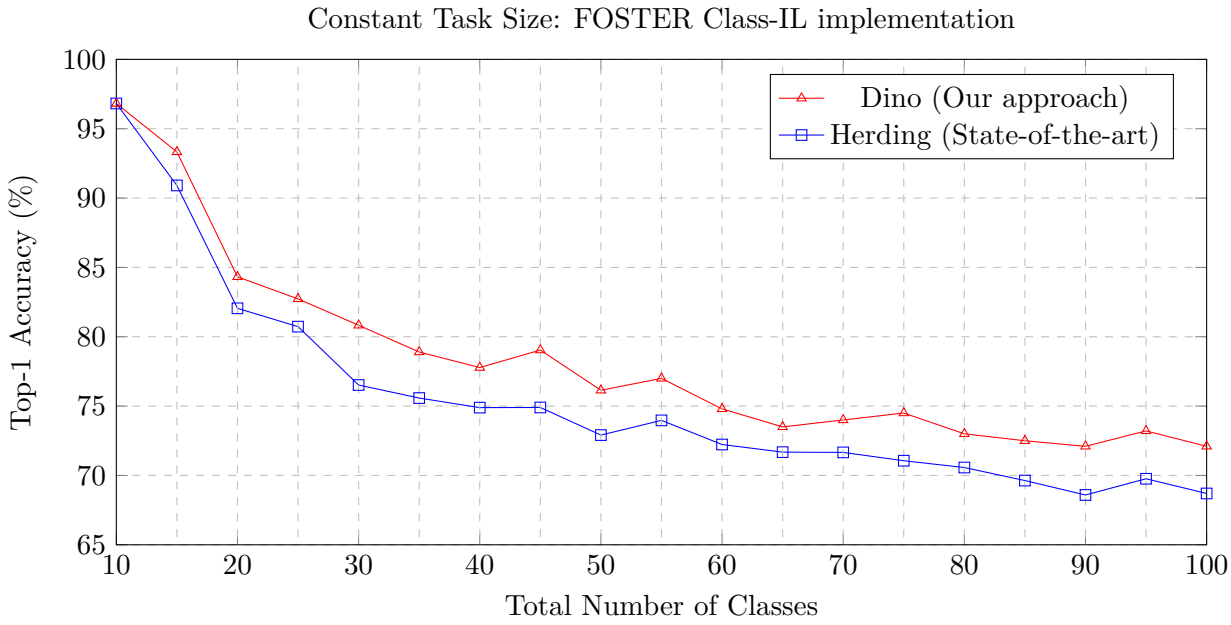
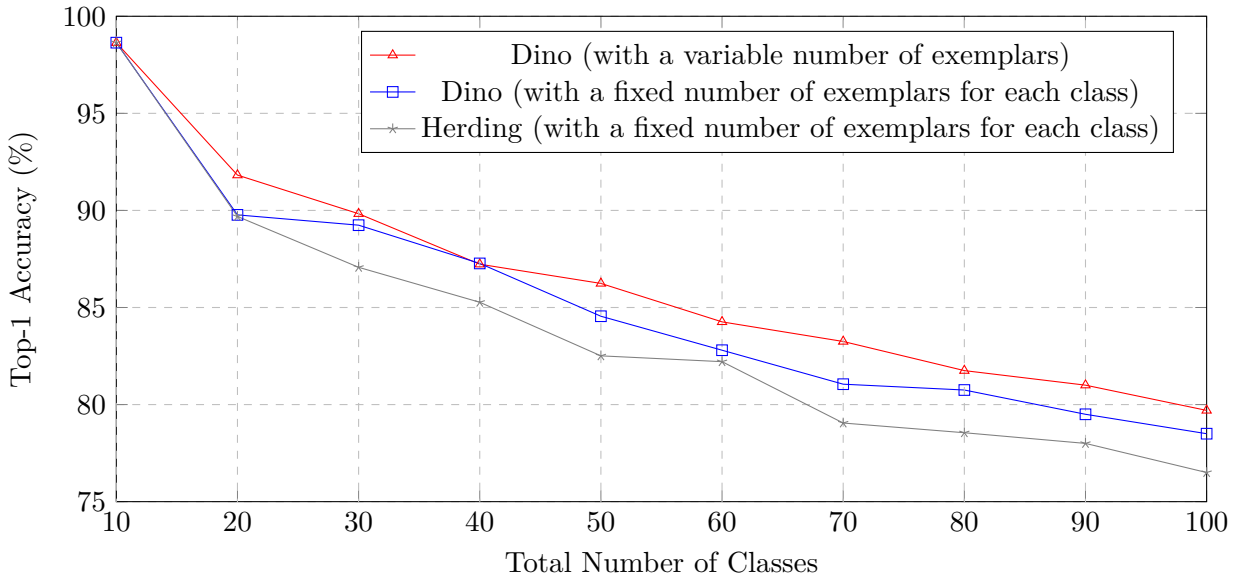


Chart 24: Comparison of Dino exemplar selection approach against Herding using FOSTER implementation



exemplar space based on class distribution. The Chart 25 shows that this approach offers a slight gain in accuracy compared to the traditional strategies with a constant memory allocation. This indicated that there is scope to further investigate and improve this approach. With an imbalance in the memory allocation, the model fine-tuning can become a secondary issue and explain why the performance gain remained low.

Chart 25: Comparison of exemplar selection strategies: PODNet Class-IL implementation



7.6 Experiment 6: Dataset Analysis using Dino

The features from the 10 cameras (Figure 6.8) are embedded in the 3D space as shown in Figure 7.1. Each object is rotated in 12 different orientations, giving (12 images * 10 cameras) 120 images per class. A clustering of the images from different cameras is observed. Each cluster has 12 images. The camera labels are placed at the location of the centroid of all the features from that camera. Furthermore, investigation of the clusters shows that the variance of the cluster corresponds to the variance of the image features that are captured with the varied object orientation. This investigation employs a similar approach to the variable exemplar selection strategy. Moreover, it can clearly be seen that the camera clusters that are closer to each other contain similar image features. However, this visual similarity goes beyond the simple interpretation that two close clusters must come from cameras that are placed close by. Rather, the clusters show the overall distribution of the features captured from the given camera. This can be verified with cameras 1 and 3. These are located at the opposite ends of the prototype setup and are located far apart. However, as the object is rotated with different orientations, the resultant feature average for the two cameras is embedded close to each other.

Such effective feature encoding can be used to study the data distribution and add more understanding at a relatively low computational cost. The total cost of studying the embedding was no higher than that of running an evaluation on the Dino pretrained model and performing a PCA downscaling.

It can be argued that the images captured from the setup contain other background objects, which are dependent on the camera location and orientation. As a result of this, the features would overfit the background data and would explain why the images are automatically clustered

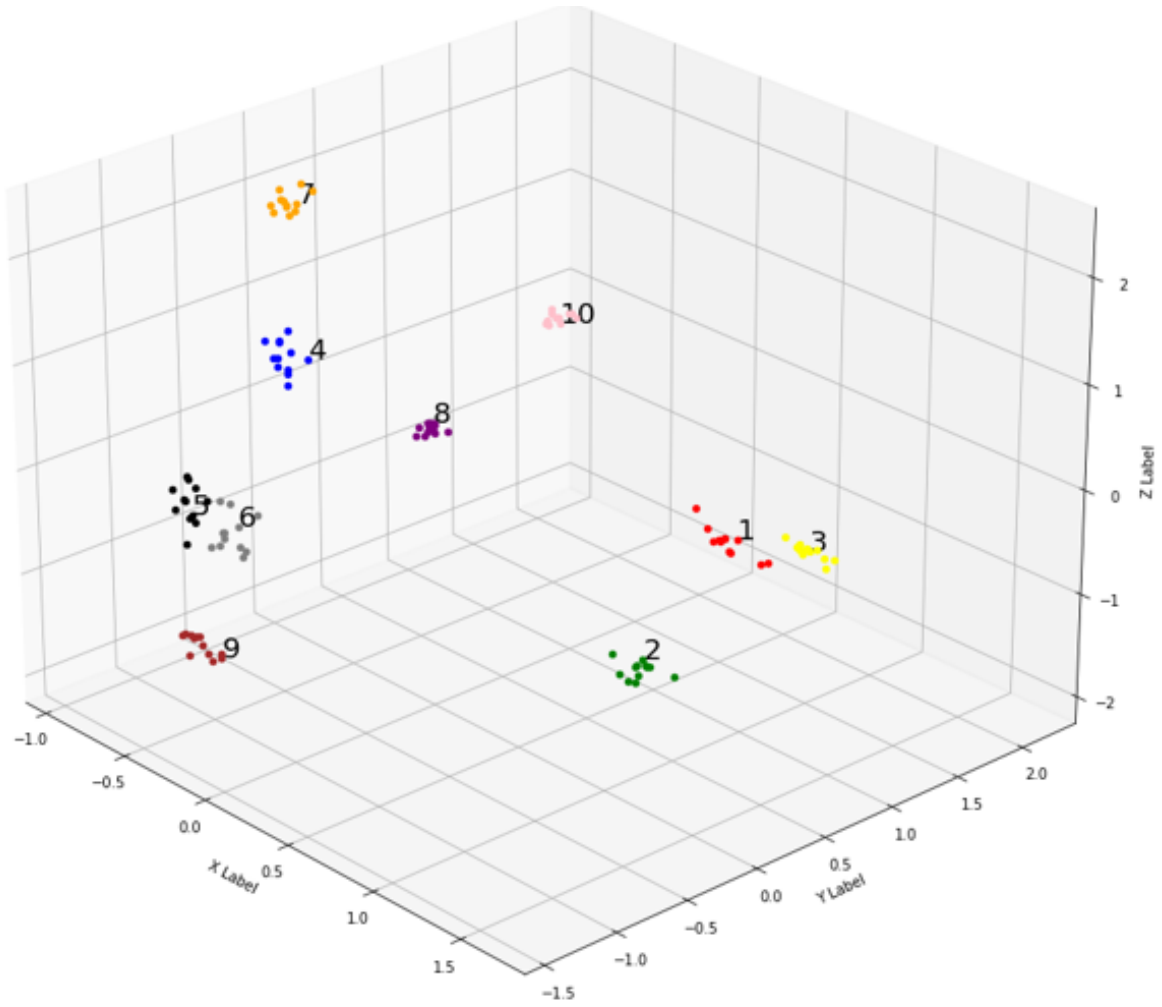


Figure 7.1: Encoded features from the images captured from various cameras in a 3D scatter plot. The location of the camera number label represents the centroid of all the features from the respective camera

based on the camera. Hence, a second investigation was conducted after segmenting the images and removing the background. The embedding of these images is shown in Figure 7.2. It is clear that the feature embeddings are more spread out. This verifies the hypothesis about the clusters overfitting to the background data. However, the distribution of the cluster centroids is still maintained. For example, cameras 1, 2 and 3, which are placed on the same plane, produce clusters whose centroids are in proximity to each other. Incorporating the analysis of the cluster variance, it is inferred that the cluster with a higher variance once again correlates with the object being captured from the varied perspective (for example the images in Figure 7.3a) and a lower variance corresponds to a cluster with more identical images (for example images in Figure 7.4a).

Similar to the K-means clustering from exemplar selection, an approach can be developed to

maximise the object features captured while reducing the number of cameras used.

This experiment illustrates the multifunctionality of the dataset analysis and pruning approach developed in this thesis. The approach can be utilised in various scenarios and deployed with low computational overhead.

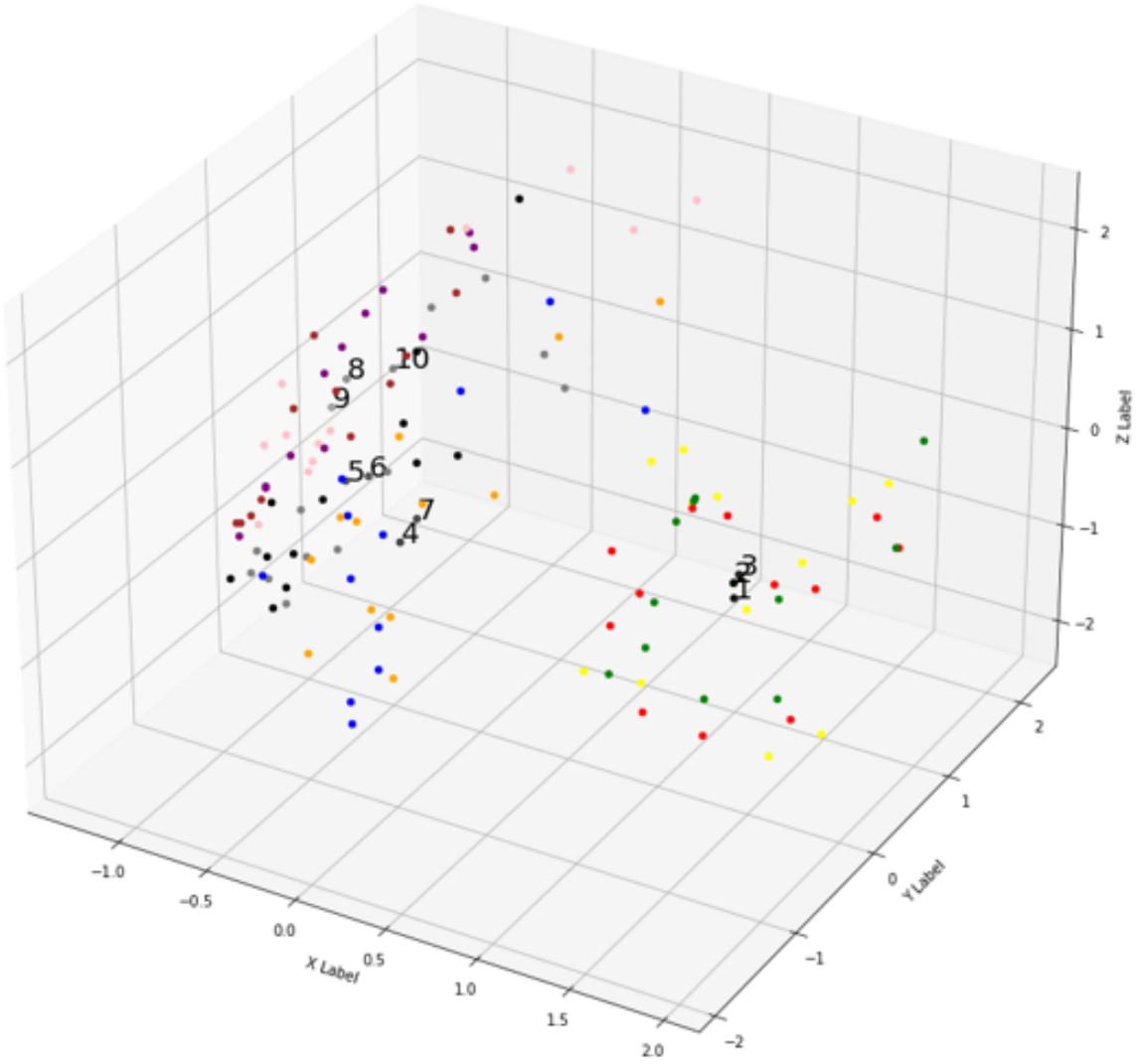
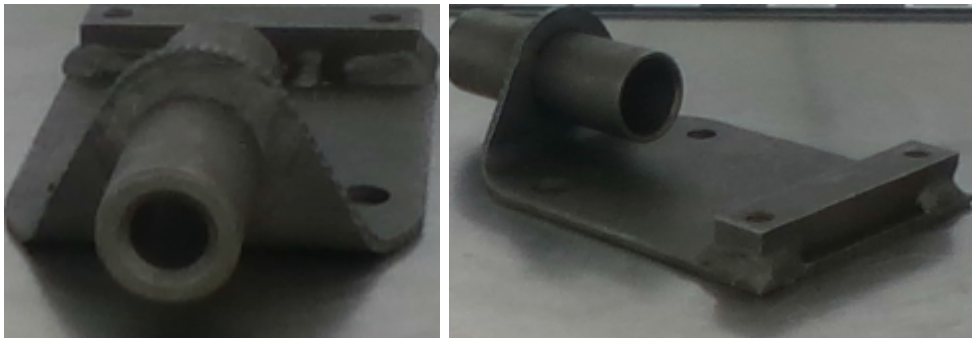
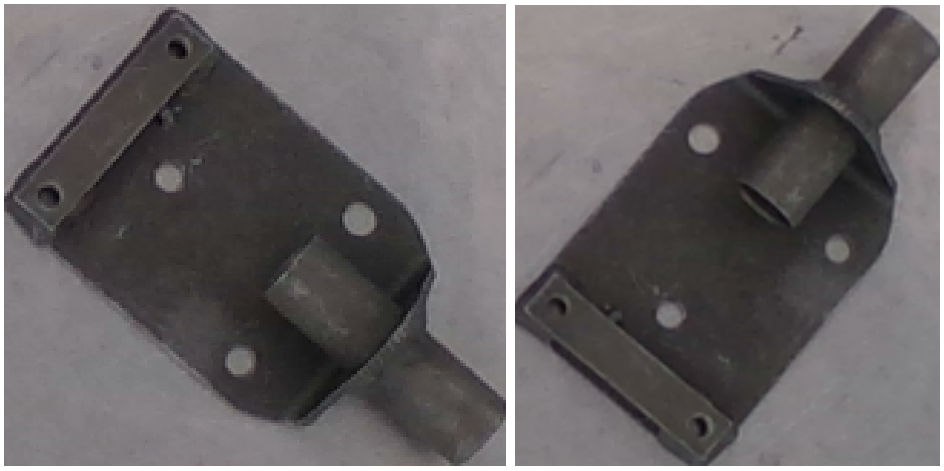


Figure 7.2: Encoded features from the segmented images captured from various cameras in a 3D scatter plot. The location of the camera number label represents the centroid of all the features from the respective camera



(a) orientation 1 (b) orientation 2

Figure 7.3: Images from the camera cluster with the highest variance



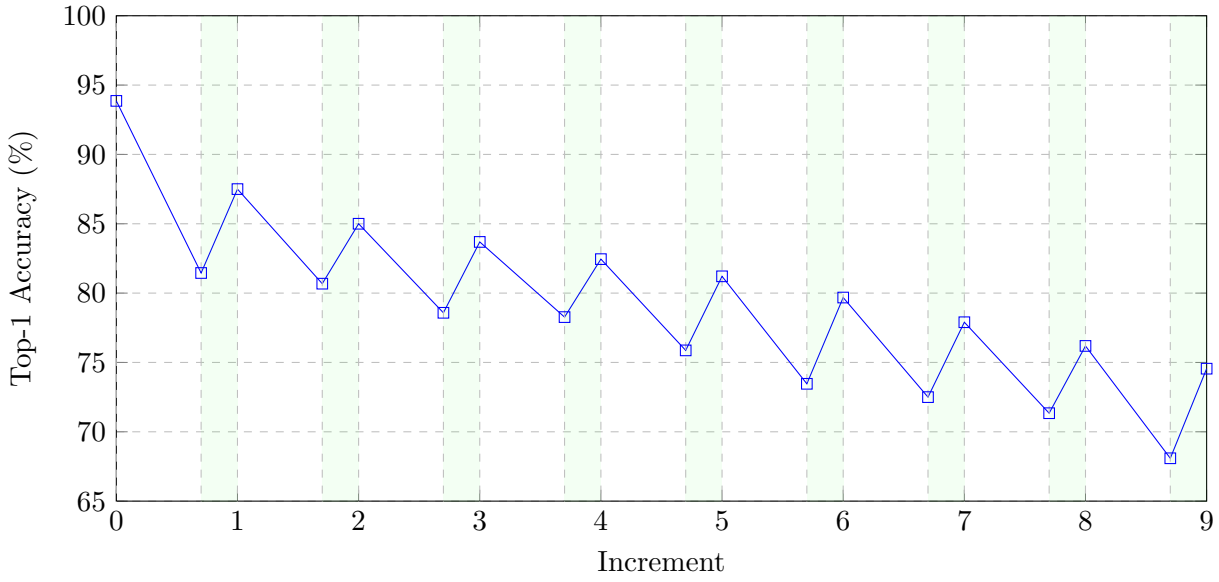
(a) orientation 1 (b) orientation 2

Figure 7.4: Images from the camera cluster with the lowest variance.

7.7 Experiment 7: Incremental Learning in tandem with Online Learning

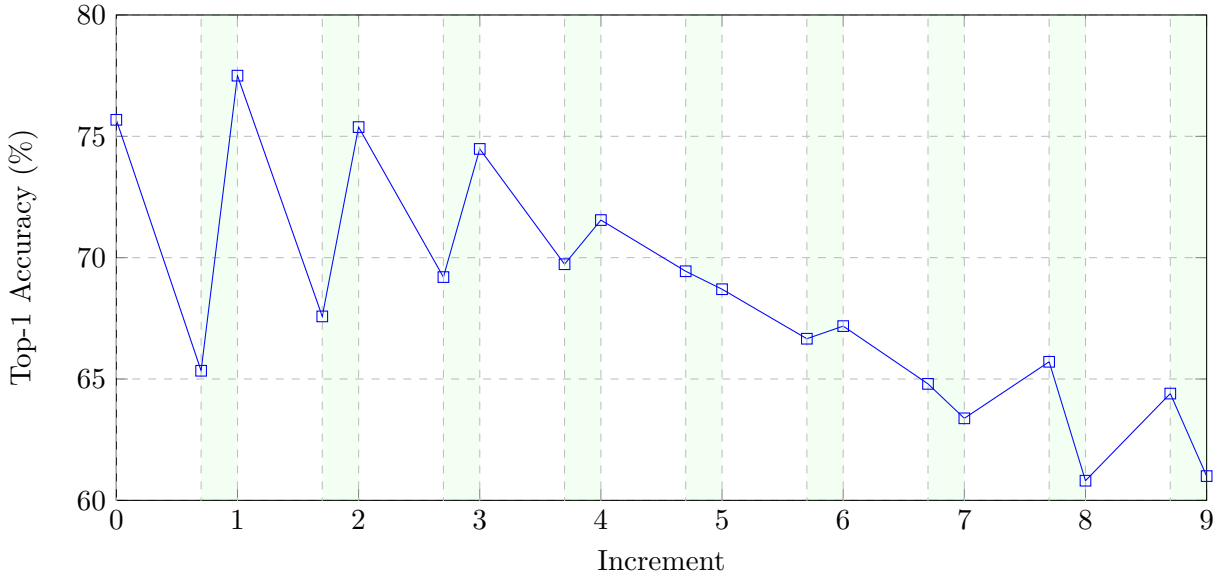
Based on the results shown in Experiment 3, it was concluded that retaining a large portion of the dataset as exemplars is an ineffective and expensive strategy for improving the model performance during incremental learning. Instead, a small unseen portion of the data was introduced to the model after the class increment. This approach utilises the fine-tuning epochs of the PODNet implementation to introduce new data instead of the exemplar. This shows a boost in the model accuracy at a relatively low computational cost, as seen in Chart 26. This strategy assumes that the incremental learning and online learning data (and by extension the test data) belong to the same distribution. For example, if all the images in the dataset have the same clean background or have an equal ratio of clean to messy data.

Chart 26: Homogeneous Data Distribution: PODNet Class-IL implementation with online learning steps (shaded)



However, this developed approach proves to be rather ineffective when there is a domain imbalance in the incremental learning and online learning data. In such cases, the model struggles to balance the class imbalance and, as a result, the online learning epochs result in a greater performance loss (Chart 27). This peculiar case involves a combination of class incremental learning with domain incremental learning and can be investigated further. It can be formulated as a more specialised and challenging case of incremental learning.

Chart 27: Domain Imbalance: PODNet Class-IL implementation with online learning steps (shaded)



8 Conclusion

A preliminary conclusion that can be drawn from the thesis work is that Class Incremental Learning is an effective solution to reducing the overall training time and energy consumption of the Machine Learning project over its lifetime. The implementations discussed in this thesis offer varied solutions to the problem of balancing model plasticity and rigidity.

The results presented in Chapter 7 show that Class-IL performance remains satisfactory for a larger number of increments when the data is clean and has less visual complexity and background noise. For data containing higher visual complexity, a sharper decline in model performance was observed. Hence, it is recommended to assess the dataset in order to assess how long the incremental learning algorithm can be implemented. However, in a framework of lifelong continual learning, it will eventually be necessary to train the model anew using a joint training update to upgrade the model accuracy. In the case of relatively clean data, approx. 15-20 class increments may be conducted before joint training is introduced. For data with more visual complexity and noise, 5-10 increments are recommended. Additionally, the continual learning framework can track the current model accuracy (or drop in average incremental accuracy) to decide when incremental learning is no longer effectual.

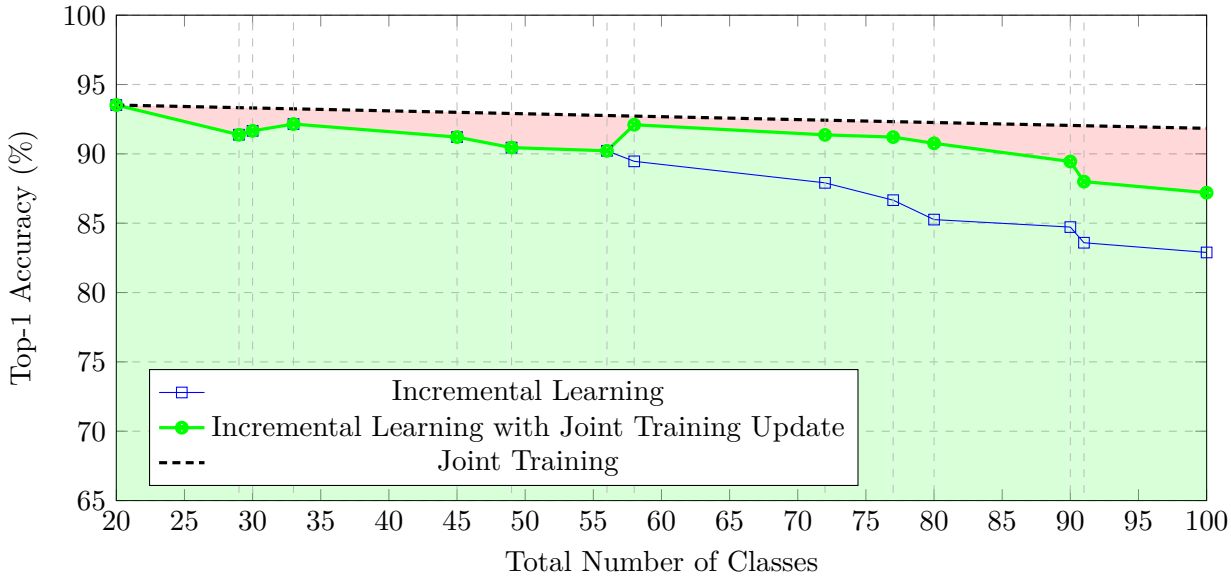
The Algorithm 9 shows such an implementation in pseudocode. The desired Class-IL implementation, exemplar selection strategy and online learning updates can be incorporated alongside a conditional resetting to periodically initiate joint training. This approach strikes a good balance between model accuracy and computational load. Since the training time and energy consumption of the Class-IL approaches also increases linearly with increasing increments (especially for implementations such as DER), it is not desirable to keep adding more increments. As shown in Charts 28 and 29, adding a joint training increment in such cases can boost the performance of the ML framework over its lifetime in terms of accuracy and energy consumption (given that the exemplar size and incremental training data is managed accordingly). The two performance metrics established in this thesis: **cumulative accuracy ratio**, and **cumulative energy consumption ratio** can be used by other researchers to test their incremental learning setups under diverse scenarios.

Considering the ImageNet-100 or the Industrial-100 dataset, the proposed hybrid approach provides a cumulative energy consumption ratio of 0.26 to 0.55 and can save up to approx. 25kWh of energy. This approach is meant to be deployed on a larger scale on multiple applications, which will result in a scaling up of energy saving. Projecting this implementation to the larger datasets such as ImageNet (with 1000 classes or higher) and assuming an increment after 10 increments shows that approx. 1400kWh energy could potentially be saved, which directly translates to a saving of EUR 840 [71] on a single project. [1]

[1] Considering the energy cost in Germany at the time of this project (EUR 0.55-0.6/kWh) and assuming the same computational setup as the one used in Experiments 1 and 3

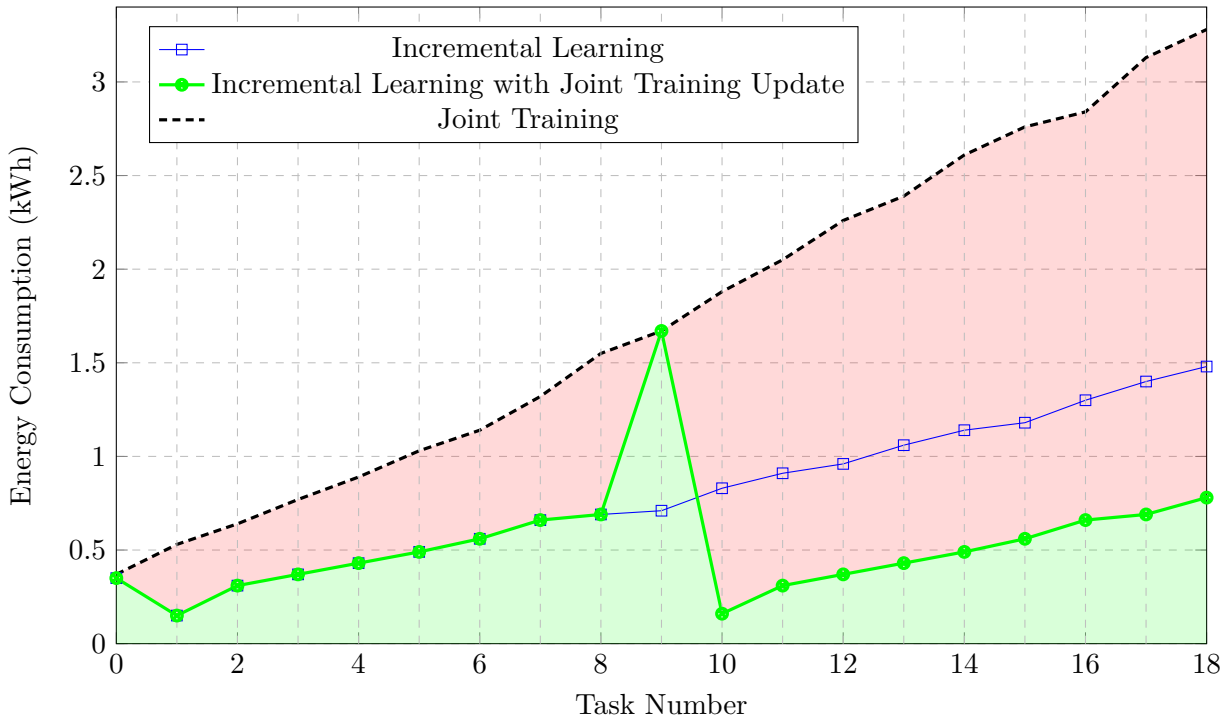
Case Study: The industrial application of the developments presented in this thesis was implemented on the EIBA photo studio data. ResNet50 was used as the backbone architecture with ImageNet pertained weights. Based on the various experiments conducted in this thesis, the PODNet [47] implementation was chosen, since it provides good model performance at low overall energy consumption. Additionally, the method was expanded to incorporate online learning steps in tandem with incremental learning. This selection is based on the industry-specific project requirements of EIBA and is not meant to be a generalised verdict on the Class Incremental Learning as a whole. As previously discussed, multiple approaches can be incorporated into the larger comprehensive continual learning framework; this allows the user to select the preferred implementation. The larger pretrained model aids in faster learning and reduces the number of epochs necessary for initial joint training and the joint training updates. This makes it computationally cheaper to update the framework when the growing size of the CIL model slows down the training. A year-long project scenario was simulated, where an initial joint training was conducted in the first week and an incremental update was carried out once a week followed by an online learning update for the rest of the year. It was gauged that a joint training update becomes necessary only after 25 increments (corresponds to 6 months) to ensure higher accuracy with minimal increase in energy consumption. The developments and insights from this thesis were satisfactorily implemented onto the industrial project environment.

Chart 28: Accuracy curves for practical incremental learning projects (DER Implementation, Experiment 3 setup)



The second area of focus for the thesis was data pruning and analysis. It has been shown that the Dino-based multifaceted approach can be employed in diverse contexts, including exemplar selection for incremental learning, visualising data structure and deriving preliminary conclusions. The features encoded using the pretrained ViT model were downsampled using PCA for further analysis. This approach follows the same Green AI principles and provides meaningful results

Chart 29: Energy Curves for practical incremental learning projects (DER Implementation, Experiment 1 setup). Joint training update resets the model architecture, and the energy consumption per task decreases as a result.



with low computational overhead. It is recommended to study the intra-class and the inter-class distribution, identify outliers and redundant data prior to commencing training. Similar to the aforementioned year-long case study, the data pruning approach can be used to evaluate the incoming data every week prior to commencing the incremental and online trainings.

The third area of focus, Green AI, was pivotal in driving the research questions and the resulting developments in this thesis. It was established (through Experiments 1 and 3) that the energy consumption of the Class-IL approaches corresponds with the time consumption metric, provided that all the other parameters are kept the same. Researchers working on incremental learning are encouraged to present the training times and (if possible) the energy consumption of their experiments to provide the readers with a more complete picture of their research. Moreover, researchers are also encouraged to use a metric like the cumulative energy consumption ratio that parameterises the relative computational costs from incremental learning and joint training.

Algorithm 9: Class Incremental Learning with online learning steps and periodic joint training updates

```
Data: Dataset, Exemplar Selection Method, Accuracy Threshold,
      Hyperparameters
Result: Accuracy for each task, Average incremental accuracy
Accuracy List = [ ]
From Dataset get number of classes = N_0
while task T ← 0 do
    Initial Joint Training with N_0 classes
    Model M ← M_0
        Knowledge Distillation // train student from teacher
    // Exemplar Selection:
    Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
end while
N ←N_0
Get N_1 // From Dataset get the current number of classes N_1
if N_1 > N_0 && Accuracy >= Threshold then
    Get Model M_T
    Training Set = New Classes + Exemplar set E_T
    Incremental training loop for T
    Model M ← M_T
        Knowledge Distillation // train student
    // Exemplar Selection:
    // Update Model Representation and Architecture:
        Model = M_T
    Exemplar set = E_T
    Accuracy List ←Accuracy
    if Data > N_1 * K then
        // New data is available for older classes
        Online Learning Increment
    end if
    N_0 ←N_1 // Update task number T_i and information
end if
if Accuracy < Threshold or T==10 then
    Set T ←0
        Reinitiate Joint Training
end if
Result ←Accuracy List
```

9 Further Work

A further deeper investigation into the various Class-IL implementations is desirable. Some recent implementations use Vision Transformers as the base architecture and add separate classifiers on top of the large pretrained models. As previously discussed, improving feature encoding and dark knowledge of the model directly correlates to superior performance with continual learning. Hence, the use of pretrained ViT models is a promising avenue.

This thesis successfully dealt with the case of class incremental learning in tandem with online learning for homogeneous data distribution (Experiment 7, Chart 26). The more challenging case of a domain shift introduction via online learning increments can be further studied and investigated. A feature encoder such as Dino may also be incorporated here to balance the mean of the online learning data with the original data from the class increment.

The exemplar selection strategy using Dino has proven to be quite expedient. However, it operates with the assumption that the training data and the validation/test data are drawn from the same distribution. In cases where this does not hold, the optimisation conducted by the approach might not provide any considerable advantages. A further investigation into understanding how this approach can be adapted to other use cases is necessary.

As an overarching recommendation, it is necessary for the AI community to develop new methods and explore new avenues while balancing the energy consumption added by this research [72]. Reporting the training times and energy consumption of the developed methods is an important first step and would serve to make AI implementations greener. Additionally, researchers working in the field of Class Incremental Learning are encouraged to test their developments on varied incremental sequences to provide better context on the working of their implementations in practical scenarios. The **cumulative energy consumption ratio** and **cumulative accuracy ratio** metrics are ideal for these case studies. The Industrial-100 dataset will be available to other researchers for investigations similar to those conducted in this thesis. [1]

[1] See Appendix VI

Acknowledgement

A big thanks to Simon Storms (WZL, RWTH) for his continued support and supervision of this thesis project. The author would also like to thank Univ.-Prof. Dr.-Ing. Christian Brecher for the opportunity to work on the thesis in tandem with IPK.

The author is deeply grateful for the support provided by the Machinelles Sehen and Automatisierungstechnik Team at Fraunhofer IPK. The author would like to thank Paul Koch and Clemens Briese for providing the work opportunity. The work presented in this report would not have been possible without the continued support, guidance, and encouragement provided by Paul Koch. The work presented in this thesis would not have been possible without the efforts of the AI community to make their work open source and available to everyone. The author would also like to thank Marian Schlüter, Dipl.-Ing. Johannes Hügle and Prof. Dr.-Ing. Jörg Krüger for their leadership and support.

Bibliography

[1] **Über Uns - Fraunhofer IPK.** Fraunhofer, Accessed June, 2022. https://www.ipk.fraunhofer.de/de/ueber-uns.html

[2] **Eiba: Automatisierte Identifikation Gebrauchter Produkte - Fraunhofer IPK.** Fraunhofer, Accessed June, 2022. https://www.ipk.fraunhofer.de/de/referenzen/eiba.html

[3] **EIBA: Circular Economy Initiative Deutschland** Accessed June 2022, https://www.circular-economy-initiative.de/de-eiba

[4] Schlüter, Marian, Hannah Lickert, Katharina Schweitzer, Pinar Bilge, Clemens Briese, Franz Dietrich, and Jörg Krüger. **AI-Enhanced Identification, Inspection and Sorting for Reverse Logistics in Remanufacturing.** Procedia CIRP 98 (2021): 300–305. https://doi.org/10.1016/j.procir.2021.01.107

[5] **EIBA - Sensorische Erfassung, automatisierte Identifikation und Bewertung von Altteilen anhand von Produktdaten sowie Informationen über bisherige Lieferungen. (n.d.).** Acatech. Accessed June 2022, https://www.acatech.de/projekt/

[6] **Vier Augen für die Kreislaufwirtschaft** Accessed June 2022, https://www.ipk.fraunhofer.de/de/medien/futur/futur-2021-2/vier-augen-fuer-die-kreislaufwirtschaft.html

[7] **Energy demands limit our brains' information processing capacity** Accessed July 2022, https://www.ucl.ac.uk/news/2020/aug/energy-demands-limit-our-brains-information-processing-capacity

[8] **CIFAR-10 and CIFAR-100 datasets** Accessed June 2022, https://www.cs.toronto.edu/~kriz/cifar.html

[9] **ImageNet Dataset**, Accessed July 2022. https://www.image-net.org/

[10] **t-SNE visualization of CNN codes**, Accessed June 2022, https://cs.stanford.edu/people/karpathy/cnnembed/

[11] **What Is a Convolutional Neural Network?** MATLAB and Simulink. Accessed June 2022, https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

[12] **MODELS AND PRE-TRAINED WEIGHTS** , Accessed Aug., 2022 https://pytorch.org/vision/stable/models.html

[13] French, Robert M. **Catastrophic interference in connectionist networks: can it be predicted, can it be prevented?,** In Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS 1993, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1176–1177. https://dl.acm.org/doi/10.5555/2987189.2987341

[14] James Kirkpatricka, Razvan Pascanua, Neil Rabinowitza, Joel Venessa, Guillaume Desjardinsa, Andrei A. Rusua, Kieran Milana, John Quana, Tiago Ramalhoa, Agnieszka Grabska-Barwinska, Demis Hassabisa, Claudia Clopathb, Dharshan Kumarana and Raia Hadsell, **Overcoming catastrophic forgetting in neural networks**, Proceedings of the National Academy of Sciences, Vol 114, No 13, 2017, https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[15] **Incremental Learning.** Papers With Code. Accessed June 2022, https://paperswithcode.com/task/incremental-learning

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, **Deep Residual Learning for Image Recognition** , CoRR, abs/1512.03385, 2015, https://arxiv.org/abs/1512.03385

[17] Belouadah, Eden, Popescu, Adrian, and Kanellos, Ioannis. **A Comprehensive Study of Class Incremental Learning Algorithms for Visual Tasks.** Neural Networks 135 (2021): 38-54. doi:10.1016/j.neunet.2020.12.003. https://arxiv.org/abs/2011.01844

[18] Masana, Marc, Liu, Xialei, Twardowski, Bartłomiej, Menta, Mikel, Bagdanov, Andrew D., van de Weijer, Joost, **Class-incremental learning: survey and performance evaluation on image classification** CoRR, abs/2010.15277, 2020 https://arxiv.org/abs/2010.15277

[19] Rebuffi, Sylvestre-Alvise, Kolesnikov, Alexander, Sperl, Georg and Lampert, Christoph. **ICaRL: Incremental Classifier and Representation Learning.** 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. https://doi.org/10.1109/cvpr.2017.587

[20] Belouadah, Eden, Popescu, Adrian, Aggarwal, Umang and Saci, Léo. **Active Class Incremental Learning For Imbalanced Datasets.** Computer Vision – ECCV 2020 Workshops Lecture Notes in Computer Science, 2020, 146-62. doi:10.1007/978-3-030-65414-6-12. https://arxiv.org/abs/2008.10968

[21] Mittal, Sudhanshu, Silvio Galesso, and Thomas Brox, **Essentials for Class Incremental Learning.** 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021. https://doi.org/10.1109/cvprw53098.2021.00390

[22] Liu, Yaoyao, Schiele, Bernt and Sun, Qianru. **Adaptive Aggregation Networks for**

Class-Incremental Learning. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. doi:10.1109/cvpr46437.2021.00257. https://arxiv.org/abs/2010.05063

[23] Yan, Shipeng, ie, Jiangwei, He, Xuming, **DER: Dynamically Expandable Representation for Class Incremental Learning,** 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. doi:10.1109/cvpr46437.2021.00303. https://arxiv.org/abs/2103.16788

[24] Hinton, Geoffrey, Vinyals, Oriol and Dean, Jeffrey. **Distilling the Knowledge in a Neural Network**, NIPS Deep Learning and Representation Learning Workshop, 2015, arXiv:1503.02531 https://arxiv.org/abs/1503.02531

[25] van de Ven, Gido M., Tolias, Andreas S., **Three scenarios for continual learning**, CoRR, abs/1904.07734, Apr 2019, http://arxiv.org/abs/1904.07734

[26] **OpenAI's GPT-3 Language Model: A Technical Overview**, Lambda, Accessed June 2022, https://lambdalabs.com/blog

[27] Lannelongue, Loïc, Grealey, Jason and Inouye, Michael. **Green Algorithms: Quantifying the Carbon Footprint of Computation.** Advanced Science 8, no. 12 (2021): 2100707. https://doi.org/10.1002/advs.202100707

[28] **Imagehash**, Accessed July 2022, https://pypi.org/project/ImageHash/

[29] Lacoste, Alexandre, Luccioni, Alexandra, Schmidt, Vicor, Dandres, Thomas, **Quantifying the Carbon Emissions of Machine Learning**, CoRR, abs/1910.09700, Oct 2019, http://arxiv.org/abs/1910.09700

[30] **Machine Learning CO2 Impact Calculator.** Machine Learning CO2 Impact Calculator. Accessed July 2022. https://mlco2.github.io/impact

[31] **Green Algorithms.** Green Algorithms. Accessed July 2022. https://www.green-algorithms.org/

[32] Wu, Yue, Chen, Yinpeng, Wang, Lijuan, Ye, Yuancheng, Liu, Zicheng, Guo, Yandong and Fu, Yun. **Large Scale Incremental Learning**, CoRR, abs/1905.13260, Jan 2021, http://arxiv.org/abs/1905.13260

[33] Wu, Bichen, **Efficient Deep Neural Networks**, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Aug 2019, http://arxiv.org/abs/1908.08926

[34] Sze, Vivienne, Chen. Yu-Hsin, Yang, Tien-Ju , Emer, Joel S. **Efficient Processing of Deep Neural Networks: A Tutorial and Survey**, Proceedings of the IEEE, Volume: 105, Issue: 12, Dec. 2017, 10.1109/JPROC.2017.2761740, https://arxiv.org/abs/1703.09039

[35] Schwartz, Roy, Dodge, Jesse, Smith, Noah A. and Etzioni, Oren. **Green AI**, Communications of the ACM, December 2020, Vol. 63 No. 12, Pages 54-63, 10.1145/3381831 https://dl.acm.org/doi/10.1145/3381831

[36] Castro, Francisco M., Marín-Jiménez, Manuel J., Guil, Nicolás, Schmid, Cordelia, Alahari, Karteek. **End-to-End Incremental Learning**, CoRR, Jul 2018, https://arxiv.org/abs/1807.09536

[37] Liu, Yaoyao, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. **Mnemonics Training: Multi-Class Incremental Learning without Forgetting.** 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. https://doi.org/10.1109/cvpr42600.2020.01226

[38] García-Martín, Eva, Rodrigues, Crefeda Faviola, Riley, Graham and Grahn, Håkan. **Estimation of Energy Consumption in Machine Learning.** Journal of Parallel and Distributed Computing 134 (2019): 75–88. https://doi.org/10.1016/j.jpdc.2019.07.007

[39] Hayes, Tyler L., Nathan D. Cahill, and Christopher Kanan. **Memory Efficient Experience Replay for Streaming Learning.** 2019 International Conference on Robotics and Automation (ICRA), 2019. https://doi.org/10.1109/icra.2019.8793982

[40] Dodge, Jesse, Gururangan, Suchin, Card, Dallas, Schwartz, Roy and Smith, Noah A. **Show Your Work: Improved Reporting of Experimental Results.** Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019. https://doi.org/10.18653/v1/d19-1224

[41] Douillard, Arthur, Valle, Eduardo, Ollion, Charles, Robert, Thomas and Cord, Matthieu. **Insights from the Future for Continual Learning.** 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021. https://doi.org/10.1109/cvprw53098.2021.00387

[42] Wu, Yue, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. **Large Scale Incremental Learning.** 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. https://doi.org/10.1109/cvpr.2019.00046

[43] Krizhevsky, Alex, **Learning multiple layers of features from tiny images.** 2009, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[44] Hou, Saihui, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. **Learning a Unified Classifier Incrementally via Rebalancing.** 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. https://doi.org/10.1109/cvpr.2019.00092

[45] Kimin Lee and Kibok Lee and Honglak Lee and Jinwoo Shin **A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks.** NIPS 2018, 1807.03888, https://arxiv.org/abs/1807.03888

[46] Fei Zhu, Zhen Cheng, Xu-yao Zhang, Cheng-lin Liu. **Class-Incremental Learning via Dual Augmentation.** Advances in Neural Information Processing Systems 34 pre-proceedings, NeurIPS 2021, https://proceedings.neurips.cc/paper/2021

[47] Douillard, Arthur, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. **PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning.** Computer Vision – ECCV 2020, 2020, 86–102. https://doi.org/10.1007/978-3-030-58565-5_6

[48] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, Yun Fu. **Incremental Classifier Learning with Generative Adversarial Networks.** CoRR, Feb 2018, https://arxiv.org/abs/1802.00853

[49] Javed, Khurram, and Faisal Shafait. **Revisiting Distillation and Incremental Classifier Learning.** Computer Vision – ACCV 2018, 2019, 3–17. https://doi.org/10.1007/978-3-030-20876-9_1

[50] Jong-Yeong Kim, Dong-Wan Choi **Split-and-Bridge: Adaptable Class Incremental Learning within a Single Neural Network.** CoRR, July 2021, https://arxiv.org/abs/2107.01349

[51] van de Ven, Gido M., Zhe Li, and Andreas S. Tolias. **Class-Incremental Learning with Generative Classifiers.** 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021. https://doi.org/10.1109/cvprw53098.2021.00400

[52] Zhao, Bowen, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. **Maintaining Discrimination and Fairness in Class Incremental Learning.** 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. https://doi.org/10.1109/cvpr42600.2020.01322

[53] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, Matthieu Cord, **DyTox: Trans-**

formers for Continual Learning with DYnamic TOken eXpansion, CVPR, 2022, https://arxiv.org/pdf/2111.11326v3.pdf

[54] Jieren Deng, Haojian Zhang, Jianhua Hu and Yunkuan Wang **Incremental Prototype Tuning for Class Incremental Learning**, CoRR, arXiv:2204.03410v6, 2022, https://arxiv.org/pdf/2204.03410.pdf

[55] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, De-Chuan Zhan. **PyCIL: A Python Toolbox for Class-Incremental Learning** CoRR, Dec 2021, https://arxiv.org/abs/2112.12533

[56] Arun Mallya, Svetlana Lazebnik, **PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning**, CoRR, abs/1711.05769, 2017 https://arxiv.org/abs/1711.05769

[57] Zhizhong Li, Derek Hoiem, **Learning without Forgetting**, CoRR, abs/1606.09282, 2016, https://arxiv.org/abs/1606.09282

[58] **Shelly Technical Documentation.** Accessed July, 2022. https://shelly-api-docs.shelly.cloud/

[59] Enrico Fini, Victor G. Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, Julien Mairal **Self-Supervised Models are Continual Learners** https://arxiv.org/pdf/2112.04215.pdf

[60] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, Armand Joulin, **Emerging Properties in Self-Supervised Vision Transformers** DBLP:journals/corr/abs-2104-14294, https://arxiv.org/abs/2104.14294

[61] Laurens van der Maaten, Geoffrey Hinton, **Visualizing Data using t-SNE,** Journal of Machine Learning Research 9 (2008) 2579-2605, https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[62] **PCA Explained - Papers With Code**, Accessed July 2022, https://paperswithcode.com/method/pca

[63] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, **Learning Transferable Visual Models From Natural Language Supervision,** DBLP:journals/corr/abs-2103-00020, https://arxiv.org/abs/2103.00020

[64] Yaoyao Liu, Bernt Schiele, Qianru Sun, **RMM: Reinforced Memory Management for Class-Incremental Learning** Advances in Neural Information Process-

ing Systems 34 (NeurIPS 2021) https://proceedings.neurips.cc/paper/2021/file/1cbcaa5abbb6b70f378a3a03d0c26386-Paper.pdf

[65] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, **FOSTER: Feature Boosting and Compression for Class-Incremental Learning**, Accepted to ECCV 2022, https://arxiv.org/abs/2204.04662

[66] Xiangning Chen, Cho-Jui Hsieh, Boqing Gong, **When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations**, Submitted to ICLR 2022, https://openreview.net/forum?id=LtKcMgGOeLt

[67] **Open3D**, Accessed July 2022, http://www.open3d.org/

[68] Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, **Co-Transport for Class-Incremental Learning**, DBLP:journals/corr/abs-2107-12654, https://arxiv.org/abs/2107.12654

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, **Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification**, ICCV 2015, https://arxiv.org/pdf/1502.01852v1.pdf

[70] **OOD Detection: Papers with code** Accessed August 2022, https://paperswithcode.com/task/ood-detection

[71] Germany Electricity Price: Trading Economics, Accessed September 2022, https://tradingeconomics.com/germany/electricity-price

[72] Wenninger, Simon; Kaymakci, Can; Wiethe, Christian; Römmelt, Jörg; Baur, Lukas; Häckel, Björn; and Sauer, Alexander, **How Sustainable is Machine Learning in Energy Applications? – The Sustainable Machine Learning Balance Sheet" (2022).** Wirtschaftsinformatik 2022 Proceedings. 1. https://aisel.aisnet.org/wi2022/sustainable_it/sustainable_it/1

VI Appendix

A.1 Supplementary Logged Data

The experiments conducted during this thesis are logged and publicly available on the author's Weights and Biases account.

They can be accessed using the links below. The aggregate logged metrics can be viewed, and each training run can also be inspected separately. Moreover, the energy consumption metrics discussed are also logged for each training.

Class Incremental Learning: https://wandb.ai/vivek9chavan/01_Class_Incremental_Learning

Dataset Sampling: https://wandb.ai/vivek9chavan/02_Dataset_Sampling

A.2 Code Implementations and Data

The code implementation related to the developments in this thesis shall be shared on author's GitHub account. The Industrial-100 dataset shall be made publicly available and linked to the project repository.

https://github.com/Vivek9Chavan/Green_Incremental_Learning

A.3 Class Incremental Learning Implementations: Architectures

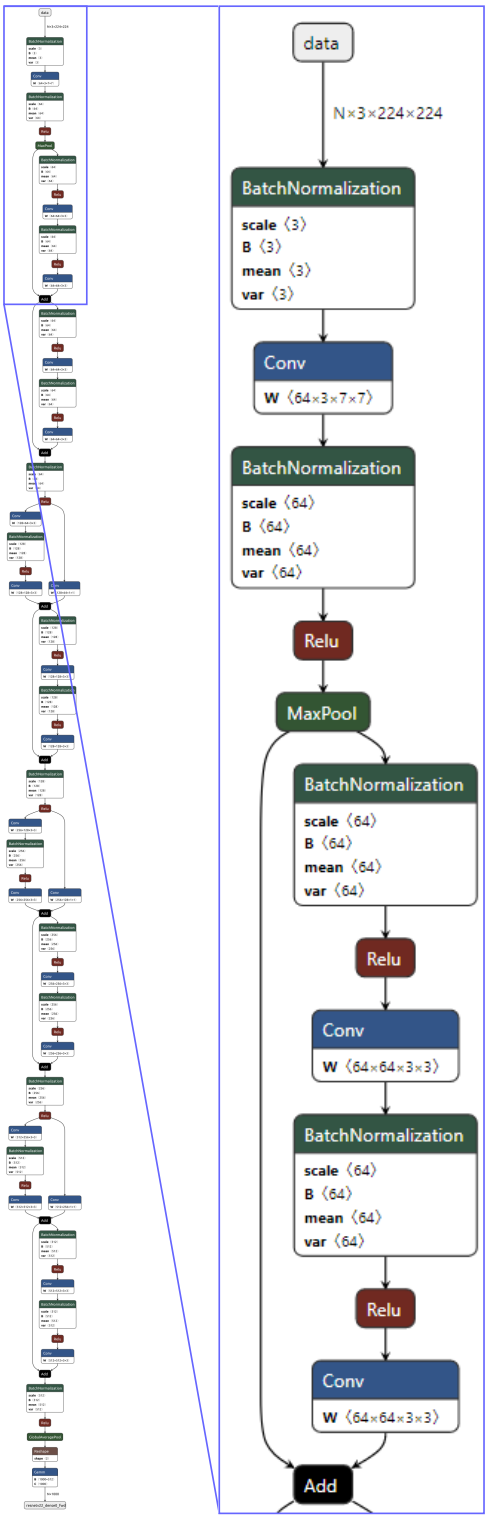


Figure A.1: Model architecture of the ResNet18, with the recurring block highlighted

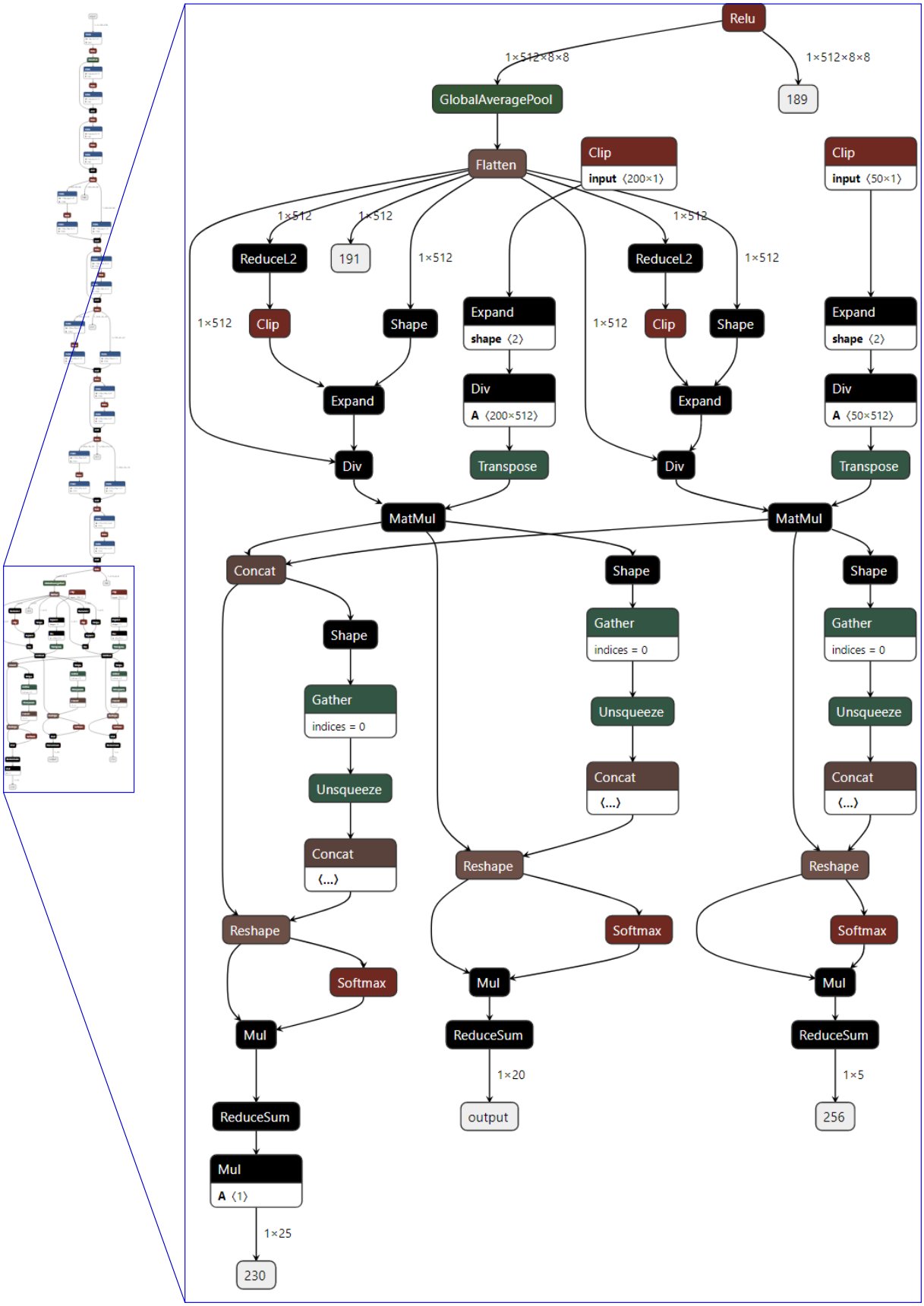


Figure A.2: Model architecture of the PODNet implementation with KD during incremental learning at Task 3. The amendment to the backbone ResNet-18 architecture is highlighted.

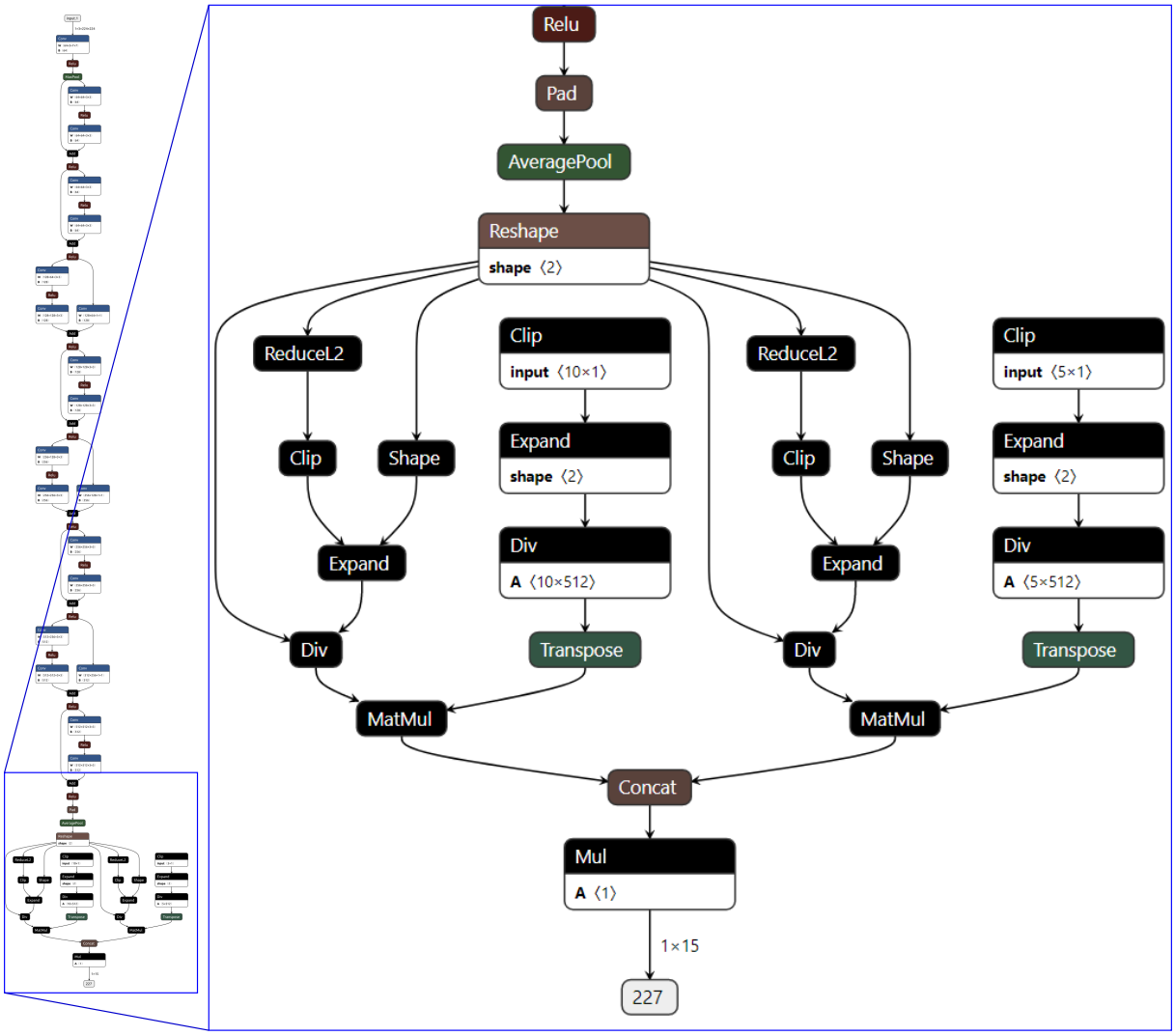


Figure A.3: Model architecture of the CCIL-SD implementation with KD during incremental learning at Task 2. The amendment to the backbone ResNet-18 architecture is highlighted

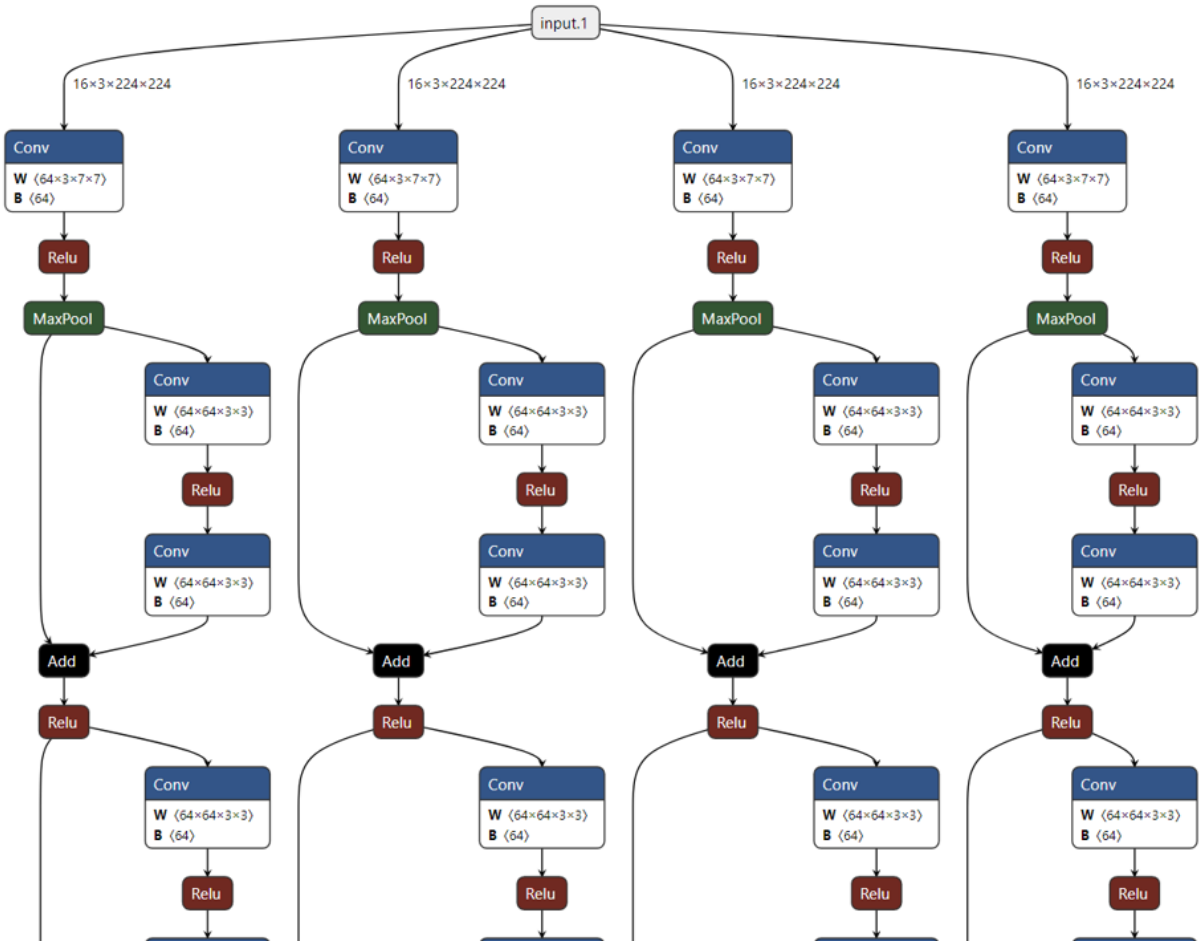


Figure A.4: Model architecture of the DER implementation during incremental learning at Task 3. The first feature extractor (left) consists of the joint training weights; a new extractor is added for each new increment and their weights are frozen.

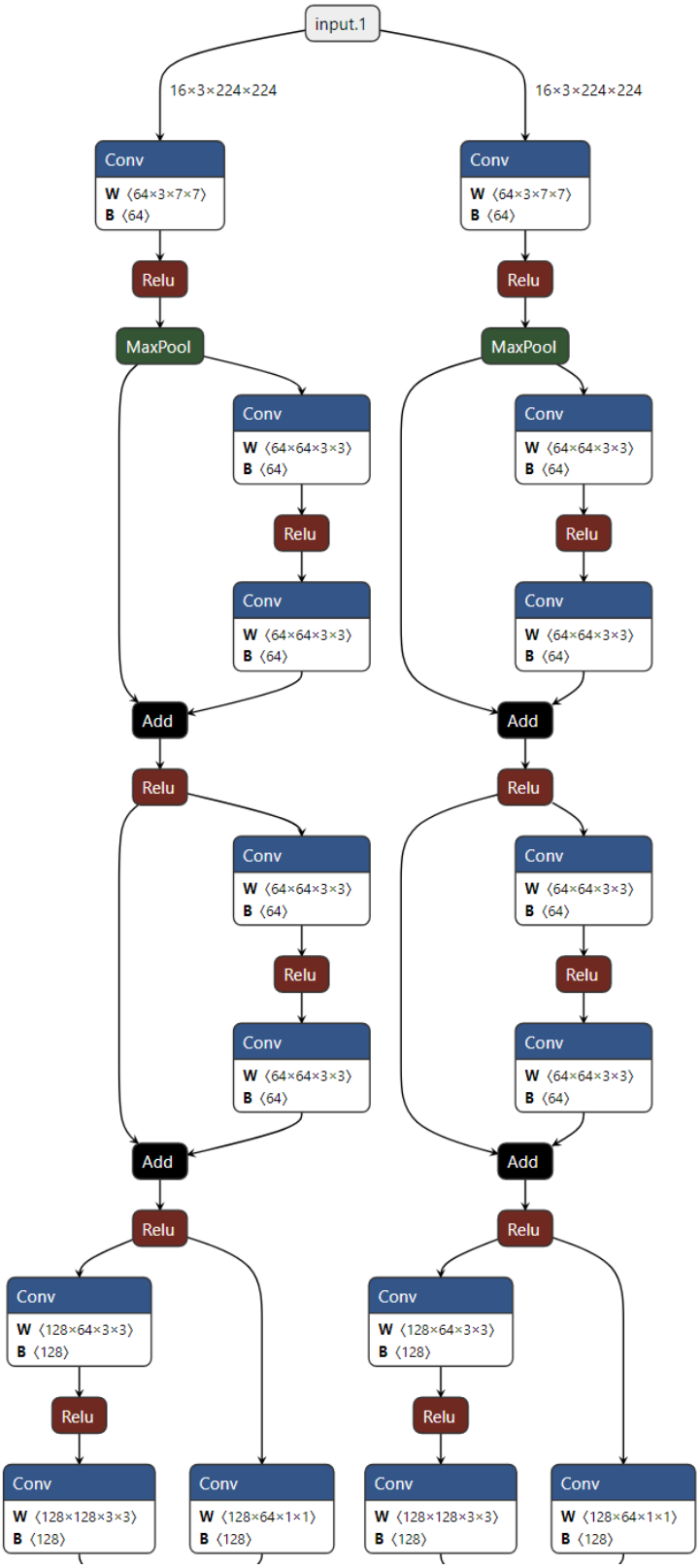


Figure A.5: Model architecture of the FOSTER implementation during incremental learning at Task 3. The feature extractor to the left contains the distilled weights for all the old classes; the second extractor is trained on the new classes.