Fine-Grained and Continual Visual Recognition for Assisting Visually Impaired People

MARCUS KLASSON

Doctoral Thesis
Stockholm, Sweden, 2022

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
TRITA-EECS-AVL-2020:4 Division of Robotics, Perception, and Learning
ISBN 100- SE-10044 Stockholm, Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av Teknologie doktorexamen i elektroteknik fredagen den 18 januari 2022 klockan 14.00 i Sal F3, Lindstedtsvägen 26, Kungliga Tekniska Högskolan, Stockholm.

Tryck: Universitetsservice US AB

Abstract

In recent years, computer vision-based assistive systems have enabled visually impaired people to use automatic object recognition on their mobile phones. These systems should be capable of recognizing objects that are important for the user on a fine-grained level. To this end, we have focused on the particular application of classifying food items which can be challenging for blind/low-vision people since visual information is often required for distinguishing between similar items. In Paper A, we present a challenging image dataset of groceries taken in grocery stores where each item is hierarchically labeled to capture the fine-grained structure of the various items. Furthermore, we demonstrate in Paper B how more easily accessible information about the items, such as web-scraped images and text descriptions, can be utilized for enhancing the classification performance of groceries compared to only using the real-world images for training.

A valuable feature of assistive vision systems is the capability of adapting to new object classes. The main challenge here is to avoid catastrophically forgetting previously learned knowledge when the classifier is updated with new classes. In Paper C, we propose a new continual learning setting for replay-based methods that aligns well with real-world needs where constraints are placed on processing time rather than the storage capacity of old samples. We then study the timing of replaying certain tasks and show that learning replay schedules over which tasks to replay can be critical for the final classification performance in our proposed setting. Finally, in Paper D, we present a method based on reinforcement learning for learning a policy for selecting which tasks to replay at different times. The benefit of our learned replay scheduling policy is that it can be applied to any new continual learning scenario for mitigating catastrophic forgetting in a classifier without additional computational cost.

To conclude, I will discuss some potential future directions for the development of the next generation of computer vision-based assistive technologies.

Keywords: Visual Recognition, Fine-grained Classification, Continual Learning, Visually Impaired People, Assistive Technologies

Sammanfattning

hej

List of Papers

A ***A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels***
Marcus Klasson, Cheng Zhang, Hedvig Kjellström
In *IEEE Winter Conference on Applications of Computer Vision (2019)*

B ***Using Variational Multi-View Learning for Classification of Grocery Items***
Marcus Klasson, Cheng Zhang, Hedvig Kjellström
In *Patterns, Volume 1(8) (2020)*

C ***Learn the Time to Learn: Replay Scheduling for Continual Learning***
Marcus Klasson, Hedvig Kjellström, Cheng Zhang
Under submission

D ***Meta Policy Learning for Replay Scheduling in Continual Learning***
Marcus Klasson, Hedvig Kjellström, Cheng Zhang
Under preparation

Acknowledgements

hej

Acronyms

List of commonly used acronyms:

DQN	Deep Q-Network
CL	Continual Learning
CNN	Convolutional Neural Network
FGIR	Fine-Grained Image Recognition
LSTM	Long Short-Term Memory
MCTS	Monte Carlo Tree Search
MLP	Multilayer Perceptron
PCA	Principal Component Analysis
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
VAE	Variational Autoencoder
VCCA	Variational Canonical Correlation Analysis
VI	Vision Impairment

Contents

Part I

Overview

Chapter 1

Introduction

Vision is probably the most important of all senses that humans possess. Our society is built on having this ability. For example, if we would like to cross a street, there are thick colored stripes on the road or signs above head height that indicate where the cross walk is located such that we can cross the street in an appropriate way. Another example is how we use text to communicate with each other, where words and sentences are composed by structured sequences of symbols that constitute a specific language. Furthermore, it has been shown that learning from both images and text can improve comprehension over learning from text only [1,2]. Possessing normal vision capabilities basically make everyday tasks easier when it comes to reaching destinations in the world, communicating with other people, and learning new concepts.

In 2020, it was estimated to be 43.3 million people who are blind and 295 million people with moderate to severe visual impairment in the world [3]. To enhance the mobility of visually impaired (VI) people, there exist various kinds of assistive devices and tools, such as screen readers and Braille typewriter machines, for supporting them with receiving information and communicating through text. More recently, several computer vision-based assistive vision tools have emerged in the form of wearable devices and mobile applications for helping VIs with tasks where visual information is a must, for example, object recognition [4–6] and wayfinding in natural environments [7–9] and .

Despite the recent successes in computer vision [10–12], these methods can face several challenges when deployed in the real-world which makes their recognition performance suffer. For example, it can be difficult for the methods to distinguish between similar items on a fine-grained level, such as different brands of apples and pears, as well as performing robustly in environments with noisy backgrounds and poor lighting. Part of the reason for such challenges is that specifying a model of the visual world that has been injected with knowledge about the rich complexity that can exist in images is very difficult [13]. Therefore, there is a necessity for developing computer vision methods that can recognize different appearances of objects, adapt to changes of known objects, and learn what new objects look like. At the same time, these tasks should be possible to execute in a time-efficient and robust manner.

In this thesis, we address the challenges on robustness in fine-grained classification as well as how the method can learn to recognize new object classes. We will begin this introduction by briefly describing vision impairments in Section 1.1, followed by a summary of assistive vision technologires in Section 1.2. Then we describe the scope of the thesis in Section 1.3 and summarize the contributions of the included papers in Section 1.4. Finally, in Section 1.5, we give the outline to the rest of the contents in this thesis.

1.1 Vision Impairments

Vision impairment (VI) is defined as the decrease of one's ability to see from various distances [14]. There are different types of VIs ranging from various degrees of blindness to having issues with seeing from far or near distances. The visual capabilities are in general assessed by measuring the *visual acuity* (sharpness) of seeing, for example, a letter or symbol, from some fixed distance. The visual acuity measured differently based on whether near- or far-sighted VI is assessed. For far-sighted VI, the visual acuity is calculated by the ratio between the distance that the subject can see the item and the distance a normal-sighted person could recognize the item. When assessing near-sighted VI, one checks the font size of letters that the subject can see using a standardized point system for measuring the symbol size [15].

In 2020, it was estimated that 338 million people possess moderate to severe VI globally, including 43 million people that are blind [3]. Furthermore, the World Health Organization (WHO) have estimated that at least 2.2 billion people live with a near or distance VI, where at least 1 billion cases could have been prevented or yet has to be addressed [15]. The untreated cases are projected to grow to 1.7 billion people by 2050 mainly due to population growth in the world as well as increased aging among the populations [3]. The leading causes for vision loss are uncorrected refractive errors, untreated cataracts, age-related macular degenerationm, glaucoma, diabetic retinopathy, where 90% of such cases are preventable and treatable [16]. The causes for vision loss also differs between countries and areas with different incomes.

There exists several tools for assisting VI people with everyday tasks. The *white cane* is probably the most common tool among VI people which is used for wayfinding to help the user anticipate what is present in their near surroundings. Also, guiding dogs are used for enhancing mobility by helping VI people to maintain a direct route, avoid obstacles, and prepares owner by stopping at curbs and stairways until they are told to proceed [17]. There also exist several tools for recognition tasks. For example, currency markers are used for keeping track of different bills in wallets, color indicators can be used to tell the user of the color of clothes, and labeling apparatus are used for distinguishing between similar items. Means for communication also exists in the form of Braille keybords and screen readers that are used in both computers and mobile phones to provide nearly equal opportunities for VI people when it comes to office-related tasks. There has been a recent emergence of various devices that are aimed to assist VI people with object recognition tasks which we will discuss next.

1.2 Assistive Vision Technologies

Cameras are used by people with VIs, including blindness, to record events and memories similarly as normal-sighted people [18]. This has opened up for opportunities where VI people can use their cameras for more than recording events, for example, object recognition, document text recognition, and color identification. Object recognition has been shown to be considered an everyday challenge, where VI people would like to ask questions about objects where visual information is necessary for identification [19]. For example, it can be very difficult to distinguish between different containers and packages that have similar shapes but different content without being able to see. These findings have encouraged development of technical aids that use computer vision for assisting VI people.

In the last decade, we have seen several variants of assistive vision technologies emerging on the market. There exist many applications for mobile phones where various visual tasks have been cramped in into the app, such as object and face recognition, barcode scanning, color and currency identification, and text recognition [20–23]. Moreover, there exists wearable devices with similar capabilities as the mobile phone apps [24,25] that also use computer vision for assistance. An alternative to the computer vision-based apps there are other mobile applications where VI users can have a video call with sighted volunteers that help them with any kind of task requiring visual capabilities [26,27]. Despite that these assistive vision technologies has opened up for VI people being more independent, there remains several challenges to tackle regarding system requirements [28–30] and privacy concerns [31–33].

Current assistive vision technologies face several challenges that needs to be tackled to enhance their utility for VI people. In the past decade, machine learning techniques have been applied successfully to various computer vision tasks such as object recognition [10, 34, 35], generating scene descriptions [12, 36, 37], and visual question answering [38–40]. In addition to better computer hardware, the main reason for these successes is the immense data collection and annotation that is required for obtaining large-scale computer vision datasets. However, the annotation becomes even more costly if the object classes should be separated based on fine-grained details about the objects, which makes it challenging for assistive vision systems to provide users with further information about objects than the general object class. Another challenge is how to update the assistive vision devices with information about new objects to recognize and ensuring that the system is still able to recognize the previous known items correctly. Furthermore, assistive vision devices should have the ability to answer questions about the surroundings of the user, should perform in real-time and be robust when applied in different environments, as well as ensuring privacy for the user.

1.3 Scope of Thesis

This thesis is focused on two applications for machine learning and computer vision-based assistive technologies, namely *fine-grained classification* [41] and *continual learning* [42, 43]. Fine-grained classification involves identifying subcategories and details of general object classes, which can be important when distinguishing between visually similar items. An example is when one has to distinguish between two juice packages from the same brand where the main ingredients are apples and oranges in the packages. The general setting in fine-grained classification is that all data and classes to learn are given all at once to the classifier to learn, but can be extended to the continual learning setting where the classes to learn are divided into tasks that are learned at different points in time. Continual learning methods are used for updating the classifier's current knowledge with information about the new classes and making sure that the classifier remembers the previously learned classes. The common denominator of these fields is classification, but both have challenges of their own that has to be addressed before adding such features into assistive vision devices. Next, we describe the challenges that we have focused on in this thesis.

Fine-grained Classification

One main challenge for fine-grained classification is the data collection procedure and there are several reasons for this. Firstly, the annotation of the collected data becomes more time-consuming as the annotators must know specific details about the objects to label the data as accurately as possible. Secondly, as fine-grained classes might be rare, there might be few examples per class that the classifier can learn from to discriminate between the objects. An application where an assistive vision device would need to learn fine-grained classes from sparse datasets is grocery shopping for helping VI people [5, 44]. Grocery items usually require visual information to distinguish between them, for example, when one needs to know how the ingredients differ in two juice packages. This also goes for raw grocery items where it might be difficult for a VI customer to tell the difference between two different brands of green apples unless the customer knows how the apples smell or how the texture of their peel differs when touching them. Furthermore, situations in the grocery store environment can disturb the recognition performance of the assistive vision device, for example, when multiple and misplaced items appear in the camera view and also when there are poor lighting settings in some areas of the store. Collecting training data that covers all possible scenarios that can occur in the store would be a cumbersome procedure. Our goal is to reduce the need for training data in the grocery stores by collecting web-scraped information about the items and using this for easing the learning of the classifier.

Continual Learning

The main challenge in continual learning is called *catastrophic forgetting* [45] which means that the classifier will overwrite previously learned knowledge with information about the new objects of interest during learning. Therefore, we must use additional training techniques that prevents this forgetting effect to maintain the recognition performance on all classes during the lifespan of the classifier. A simple yet efficient approach in continual learning for mitigating catastrophic forgetting is replay-based methods [46, 47]. The main assumption is that we are allowed to keep a low number of examples from every seen class in a small memory buffer. The idea is then to mix the old examples with the training data from new classes, such that we learn the new classes and aim to retain the performance on the old classes by replaying the memory examples for the classifier.

Most previous works on replay-based continual learning ignores the time to replay certain tasks. However, the timing of rehearsal has been shown to be very important for humans to retain memory on various tasks [48–52]. Moreover, in contrast to the constraint on the small memory size, machine learning systems used in real-world applications may be limited by processing times rather than data storage capacity [Add REFs]. In such settings, there is a need for methods that select what data from the huge storage to replay as the problem of catastrophic forgetting still remains. Our goal is to demonstrate that scheduling over which tasks to replay can be crucial for continual learning performance in this setting. Hence, we will need to develop efficient methods that can automatically propose replay schedules that mitigate catastophic forgetting in classifiers to enable this strategy in real-world settings.

1.4 Thesis Contributions

In this section, we provide summaries of the included papers as well as stating the contributions of each author to the manuscripts.

Paper A: A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels

Marcus Klasson, Cheng Zhang, Hedvig Kjellström. In *IEEE Winter Conference on Applications of Computer Vision (WACV) 2019.*

Summary We collect a dataset with natural images of raw and refrigerated grocery items taken in grocery stores in Stockholm, Sweden, for evaluating image

Table 1.1: Examples of grocery item classes in the Grocery Store dataset. We display four different items (coarse-grained class in parenthesis), followed by two natural images taken with a mobile phone inside grocery stores. Next comes the web-scraped information of the items consisting of an iconic image and a text description. We have highlighted ingredients and flavors in the text description that are characteristic for the specific item.

Class Labels	**Natural Images**	**Iconic Images**	**Text Descriptions**
Granny Smith (Apple)			*“...**green** apple with **white, firm** pulp and a **clear acidity** in the flavor.”*
Tropicana Mandarin (Juice)			*“. . . is a **ready to drink** juice **without pulp** pressed on **orange**, **mandarin** and **grapes**. Not from concentrate. Mildly **pasteurized**.”*

classification models on a challenging real-world scenario. The data collection was performed by taking photos of groceries with a mobile phone to simulate a scenario of grocery shopping using an assistive vision app. Furthermore, we downloaded iconic images and text descriptions of each grocery item by web-scraping a grocery store website to enhance the dataset with information describing the semantics of each individual item. The items are grouped based on their type, e.g., apple, juice, etc., to provide the dataset with a hierarchical labeling structure. We show two examples of grocery item classes and their corresponding web-scraped information in Table 1.1.

We provide benchmark results evaluated using pre-trained and fine-tuned CNNs for image classification. Moreover, we take an initial step towards utilizing the rich product information in the dataset by training the classifiers with representations where both natural and iconic images have been combined through a multi-view VAE.

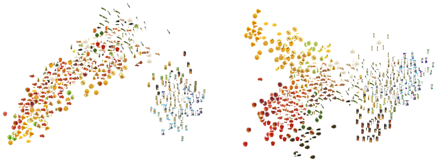
Author Contributions CZ and HK presented the idea and the data collection procedure for the natural images and web-scraped information. MK performed the data collection including visiting the grocery stores for taking the natural images and the web-scraping of the grocery store website for iconic images and text descriptions. MK performed all the experiments. All authors contributed to discussing the results and contributed to writing the manuscript.

Paper B: Using Variational Multi-View Learning for Classification of Grocery Items

Marcus Klasson, Cheng Zhang, Hedvig Kjellström. In *Patterns, Volume 1(8) (2020).*

Summary We investigate whether training image classifiers can benefit from learning joint representations of grocery items using multi-view learning over the natural images and web-scraped information of the grocery items in the Grocery Store dataset (see Paper 1.4). We employ a deep multi-view model based on VAEs called Variational Canonical Correlation Analysis (VCCA) [53] for learning joint representations of the different data types, i.e., natural images, iconic images, and text descriptions. We performed a thorough ablation study over all data types to demonstrate how they contribute individually to enhancing the classification performance. Furthermore, we apply two classification approaches where we (i) train the classifier on the joint latent representations, and (ii) using a generative classifier by incorporating a class decoder to the VCCA model that can be used for classifying images.

We performed a thorough ablation study over all data types to demonstrate how they contribute individually to enhancing the classification performance. To gain further insights into our results, we visualized the learned representations of the grocery items from VCCA and discussed how the iconic images and text descriptions help the model to better distinguish between the groceries. Our results show that the iconic images help to group the items based on their color and shape while text descriptions separate the items based on differences in ingredients and flavor. Figure 1.1 shows visualizations of the latent representations projected in 2-dimensional space using Principal Component Analysis (PCA), where we illustrate how the latents change when adding the iconic image and text description into the VCCA model. Finally, we concluded that utilizing the iconic images and text descriptions yielded better classification results than only using natural images.



(a) VAE$_{\boldsymbol{x}}$ (b) VCCA$_{\boldsymbol{xiwy}}$

Figure 1.1: Visualizations of the latent representations projected in 2D space with PCA from models VAE$_{\boldsymbol{x}}$ and VCCA$_{\boldsymbol{xiwy}}$, where we plot the corresponding iconic image for each latent representation. We observe that VCCA$_{\boldsymbol{xiwy}}$ structures the items based on visual similarities by incorporating the web-scraped information into the learning.

Author Contributions CZ and HK presented the idea and all authors contributed to formalizing the methodology. MK performed all the experiments and created the visualizations. All authors took part in discussing the results. All authors contributed to writing the manuscript.

Paper C: Learn the Time to Learn: Replay Scheduling for Continual Learning

Marcus Klasson, Hedvig Kjellström, Cheng Zhang. Submitted to *International Conference on Machine Learning (ICML) 2022.*

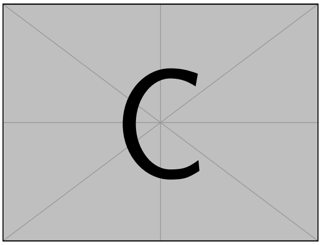


Figure 1.2: **MK: Paper C could illustrate the new CL setup with a figure that shows a network and a scheduler that needs to fetch a small replay memory from a huge pool of historical data.**

Summary In this paper, we show that learning the time to replay different tasks can be critical for continual learning (CL) performance in replay-based methods. As the main assumption in replay-based CL is that only a small set of historical data can be re-visited for mitigating catastrophic forgetting, most works have focused on improving the sample quality of the replay memory. However, in many real-world applications, historical data is accessible at all times, e.g., by storing it on the cloud. But although all historical data could be stored, retraining machine learning systems on a daily basis is prohibitive due to processing times and operational costs. Therefore, small replay memories are still needed in CL to mitigate catastrophic forgetting when learning new tasks. To this end, we propose to learn the time to learn for a CL system, in which we learn schedules over which tasks to replay at different times. Inspired by human learning, we demonstrate that scheduling over the time to replay is critical to the final CL performance with finite memory resources. We then illustrate our idea with scheduling over which tasks to replay by learning such policy with Monte Carlo tree search. We perform extensive evaluation showing that learning replay schedules can significantly improve the performance compared to baselines without learned scheduling. We also show that our method can be combined with any replay-based method and memory selection technique. Finally, our results indicate that the learned schedules are also consistent with human learning insights.

Author Contributions CZ presented the idea and MK and CZ contributed to formalizing the methodology. MK performed all the experiments. All authors took part in discussing the results and contributed to writing the manuscript.

Paper D: Meta Policy Learning for Replay Scheduling in Continual Learning

Marcus Klasson, Hedvig Kjellström, Cheng Zhang. Under preparation for submission to *Conference on Neural Information Processing Systems (NeurIPS)*

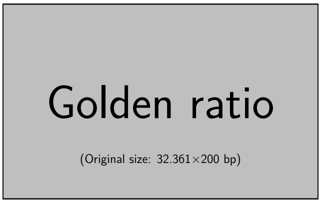


Figure 1.3: **MK: Paper D could show illustration of the RL agent scheduler that gets performance measures as input and outputs an action proportion of how to select the replay memory. RL agent could also have a replay buffer where data is collected from several environments. Maybe it can also show the test case, so that it would be two separated "at training/test phase".**

2022.

Summary In this paper, we propose a reinforcement learning-based method for learning policies for replay scheduling that can be applied in new continual learning scenarios. We demonstrated in Paper C that learning the time to replay different tasks is important in continual learning. However, a replay scheduling policy that can be applied in any continual learning scenario is currently absent, which makes replay scheduling infeasible in real-world scenarios. To this end, we propose using reinforcement learning to enable learning general policies that can generalize across different data domains. The learned policy can then propose replay schedule that efficiently mitigate catastrophic forgetting to improve the continual learning performance without any additional computational cost in the new domain. We compare the learned policies to several replay scheduling baselines and show that the learned policies can improve the continual learning performance given task orders and datasets unseen during training.

Author Contributions CZ presented the idea and MK and CZ contributed to formalizing the methodology. MK performed all the experiments. All authors took part in discussing the results and contributed to writing the manuscript.

1.5 Thesis Outline

The rest of the thesis is organized as follows. We provide some background and preliminaries on the machine learning methodology used in this thesis in Chapter 2. Chapter 3 is focused on our contributions in fine-grained classification where we first describe the related work in this field followed by explaining the frameworks used in our work. Similarly, in Chapter 4, we focus on our contributions in continual learning by first describing the related work to place our contributions in context and then describing the framework we used for enabling replay scheduling in continual learning. Finally, in Chapter **??**, we provide our conclusions of the presented works and present some research directions that we believe would be interesting to look deeper into in the future.

Chapter 2

Background

The goal with this chapter is to provide the reader with preliminaries that are useful for comprehending the included papers. We assume that the reader has some knowledge in calculus, linear algebra, and probability theory, but we intend to keep it on a basic level. First, we give the notation that will be used throughout the thesis. Then, we will introduce a selection of related works to place the thesis into context.

2.1 Notation and Terminology

We will begin by providing some algebraic notation that will be used for representing various types of data in the thesis. Scalars (both integer and real) are denoted by italic letters such as a. Vectors are denoted by lowercase bold italic letters such as $\boldsymbol{x}$, where all vectors are assumed to be column vectors. A superscript T denotes the transpose of a vector or matrix, such that $\boldsymbol{x}^T$ becomes a row vector. Matrices are denoted as uppercase bold italic letters such as $\boldsymbol{W}$. The notation $(w_1, \dots, w_m)$ denotes a row vector with m elements, where the corresponding column vector is denoted as $(w_1, \dots, w_m)^T$.

A dataset is denoted by the set $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}$, where $\boldsymbol{x}^{(i)}$ is the i-th example among the N data points. Each data point is assumed to belong in a space of vectors denoted by $\mathcal{X}$, such that $\boldsymbol{x} \in \mathcal{X}$. The data generating distribution is denoted by $p_{data}(\mathcal{X})$ which is usually unknown. To provide an example, we let the vector $\boldsymbol{x} = (x_1, \dots, x_m)$ represent a flattened image of m pixels. In this case, all possible images that can exist belong to the space $\mathcal{X}$ and the data generating distribution $p_{data}(\mathcal{X})$ gives the probability of how likely each image is to occur in the world. In supervised learning, there is also a target, either denoted as $y^{(i)}$ or $\boldsymbol{y}^{(i)}$, associated with $\boldsymbol{x}^{(i)}$. The target belongs to the target space $\mathcal{Y}$, where the space is discrete $\mathcal{Y} = \{1, \dots, C\}$ for classification tasks over C number of classes, or continuous $\mathcal{Y} = (-\infty, \infty)$ over an interval of real values for regression tasks.

Throughout this thesis, we take a machine learning approach to solve tasks by tuning an adaptive model using a dataset called the *training set*. In our case, we will represent the model with a function $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ that allows us to predict outcomes of events/data $\boldsymbol{x}$ from the task of interest. The parameters $\boldsymbol{\theta}$ expresses the function and we use machine learning algorithms for tuning parameters with the given dataset during the training phase. Once the model is trained, we often enter the *test phase* where we want to evaluate the model by predicting outcomes on an unseen dataset called the *test set*. The ability to predict outcomes of new data that is different from the examples seen during training is called *generalization*, which is a central goal for most applications in machine learning and pattern recognition.

2.2 Problem Settings in Machine Learning

Machine learning problems can be divided into three main fields, namely, *supervised learning*, *unsupervised learning*, and *reinforcement learning* (RL). Since this thesis includes work from each of these problem settings, we will briefly introduce these topics to provide the reader with context on the tasks that we are trying to solve.

MK: TO-DO: Add references to papers and sections!

Supervised Learning. In this setting, we are given a dataset $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$ where each data example $\boldsymbol{x}^{(i)}$ is accompanied with a target $\boldsymbol{y}^{(i)}$. The goal is to estimate a function $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ that assigns the correct target to each example in the training set as accurately as possible. In classification problems, each target belongs to one of K discrete categories, such that $\boldsymbol{y} = \{1, \ldots, K\}$, and we want to predict which of the categories that new data belongs to. Classification tasks will be involved in all included papers of this thesis wherein Paper **??** and B we focus on assigning the correct product category to images of grocery items. Another problem type in supervised learning is regression where the targets are continuous and real-valued. An example of a regression task is to predict the outdoors temperature tomorrow given the observed temperature today.

Unsupervised Learning. Here, we are given a dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ without access to any corresponding targets. The goal in this unsupervised setting may then be to find hidden structures in the given dataset. For example, we might be interested in discovering groups of similar examples with *clustering* techniques, or we may want to use *density estimation* where we approximate the true data distribution p_{data} with a parametric distribution $p_{\boldsymbol{\theta}}$ using the collected dataset, or we may want to project high-dimensional data into two or three dimensions for *visualization* purposes. We will get back to these goals when we introduce representation learning in Section X.

Reinforcement Learning. For these problems, we have a *learning agent* that wants reach a goal in an environment by performing a given set of actions. After performing an action, the agent observes the state of the environment and receives a reward from the environment saying how good or bad the taken action was to reach the goal. The objective for the agent is to maximize the reward signal within the time the agent reaches the goal. The agent then has to learn a policy for deciding which actions to perform in certain situations in the environment. The policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ is a mapping from perceived states $\boldsymbol{s}$ in the environment to actions $\boldsymbol{a}$ that should maximize the reward signal. An example of a task that can be framed as a RL problem is the so called Mountain Car problem, where the agent is a car that is trying to drive up to the top of a hill. The state represents the position and velocity of the car and the agent must take actions that will move the car forward or backwards. The objective is to reach the goal with as little time as possible, and the agent is encouraged to do so by the environment by sending the agent a negative reward for every time step that passes without reaching the top of the hill. We will return to the RL framework later when we describe the prerequisites for Paper D.

There exist many different methods for solving problems within these three fields. In this thesis, we employ deep learning methods which has been successfully applied in each field by representing the models with deep neural networks [10, 54, 55].

2.3 Deep Learning

In this section, we give a brief overview of deep learning [56] which is the main building block for the models we use in this thesis. Deep learning contains a family of machine learning models based on neural networks that are parametric function approximators used for representing some function of interest. Much of the successes of deep learning methods have been in supervised learning settings, especially in applications where there are large amounts of labelled data and sufficiently large model in terms of number of parameters in the network. In the following sections, we will introduce the deep learning frameworks and models that we have used in this thesis.

Deep Neural Networks

Neural networks is a class of machine learning models which popularity have grown immensely due to their ability to learn from large and high-dimensional datasets. Moreover, neural networks have been successfully applied in various number of fields in computer vision [10, 34], natural language processing [57], and reinforcement learning [55, 58]. These models are constructed by stacking layers of parameters that extract intermediate representations of the input data. The last layer outputs the target answer from the queried input and is specific for the task. For example, in image classification, the last layer outputs class scores representing which class the image is most likely to belong to.

Next, we will describe three popular types of neural networks, namely, multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). **MK: TO-DO: It would be nice to have some simple illustrations of how the networks process the data, I can take inspiration from how other people have done it.**

Multilayer Perceptrons. The simplest form of feedforward neural networks is the MLP. Let the vector $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ be some form of data where x_i is the i-th feature for $i = 1, \ldots, d$. Every layer in the MLP constitutes of weights $\boldsymbol{W}$ that are used for transforming the input such that the output reveals some hidden structure useful for solving the task of interest. The transformation is performed with a matrix multiplication, i.e., $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x}$, to receive the intermediate representation $\boldsymbol{h}$. An essential part for enabling neural networks to learn non-linear functions is to add an activation function right after the matrix multiplication of each layer. Otherwise, the neural network would only be capable of learning linear functions since the matrix multiplication is a linear mapping. A common activation function is the $a(\boldsymbol{x}) = \max(0, \boldsymbol{x})$, or the so called Rectified Linear Unit (ReLU) activation, which outputs $\boldsymbol{x}$ when $\boldsymbol{x} > 0$ or otherwise zero. By stacking two layers together in a neural network with a ReLU activation, we then obtain the representation $\boldsymbol{h} = \boldsymbol{W}_2 \max(0, \boldsymbol{W}_1\boldsymbol{x})$. Note here that this representation could be predicted class scores, such that $\boldsymbol{h} = \hat{\boldsymbol{y}}$, if we would use two-layer MLP for a classication task.

Convolutional Neural Networks. For data where the spatial order of each feature can be salient for prediction tasks such as image classification, we need a network that can capture relationships between features. CNNs are special kinds of neural networks that can process data with grid-like structures. Convolutional layers constitutes of a set of filters with adaptable weight parameters. To produce the output, we slide each filter across the input across the width and height of the input and compute dot products between the filter weights and the input at any position. Each filter will then produce a 2-dimensional feature map that gives the responses of that filter at every spatial position. The 2-dimensional feature maps from all filters are then stacked depth-wise to obtain the output volume. The obtained feature maps are often downsampled along their spatial dimensions using a pooling operation after the activation function. The parameter sharing in convolutional layers where each weight in a filter is applied to every position of the input comes from the idea that if some visual features are important in one part of the image, it should intuitively be useful at some other location as well. Furthermore, this design choice also makes the model require fewer parameters and a lower number of operations to compute the outputs.

Recurrent Neural Networks. RNNs are a family of neural network models specialized for processing sequences of data. Similar to CNNs, these models use parameter sharing by applying the same weights across several time steps. The parameter sharing is important in RNNs as it enables the model to handle different sequence lengths as well as being capable of recognizing relevant information that can appear at different locations in the sequence. Many RNNs follow the same procedure through the equation $\boldsymbol{h}^{(t)} = f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$, where the RNN produces the current state $\boldsymbol{h}^{(t)}$ by incorporating the input data $\boldsymbol{x}^{(t)}$ at time t into the previous hidden state $\boldsymbol{h}^{(t-1)}$. Hence, the hidden state $\boldsymbol{h}^{(t)}$ will now contain information about the whole past sequence up to time t. In most applications, there will be an extra output layer that reads the information from state $\boldsymbol{h}^{(t)}$ to make predictions. An example application for RNNs is predicting the next word in a sentence given previous words, where the RNN should store the necessary information about previous words to predict the rest of the sentence. A common choice of RNN model is the Long Short-Term Memory [59] (LSTM) which mitigates problems with vanishing gradients during the training phase.

For training deep networks, *loss* functions are used for measuring how well the network performs to solve the task of interest. In classification tasks, the cross-entropy loss is commonly used where the predicted class scores $\hat{\boldsymbol{y}}$ are compared against the true target class $\boldsymbol{y}$,

$$\mathcal{L}_{CE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{k=1}^{K} \boldsymbol{y}[k] \log \hat{\boldsymbol{y}}[k], \tag{2.1}$$

where the true target vector $\boldsymbol{y}$ uses a one-hot representation where the true class i is denoted in the vector by setting the i-th element in $\boldsymbol{y}$ to one, as in $\boldsymbol{y}[i] = 1$, and zero elsewhere.

Probably the most common optimization algorithm for deep learning is stochastic gradient descent (SGD). The model parameters are updated by first computing the gradient of the loss function with respect to the weights of the network, as in $\nabla_{\boldsymbol{\theta}} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})$, for a single input-output pair with backpropagation [60]. We can then update the parameters with the equation

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}), \tag{2.2}$$

where η is the learning rate which is an important parameter for SGD determining how much the weights should be updated with the computed gradient.

Deep Autoencoders

A common architecture type for deep learning in unsupervised learning are autoencoders for learning hidden representations of unlabeled data. Autoencoders are commonly used for dimensionality reduction of high-dimensional data, where the lower-dimensional representation can be used for classification tasks, or to visualize hidden structures in the data that are hard to reveal from the original input data. The objective of the model is to reconstruct the original input data. The network architecture is built using two networks called *encoder* and *decoder* with a bottleneck layer between the networks for extracting the hidden representation $\boldsymbol{h}$. The encoder and decoder architectures can be of any neural network type, such as MLPs, CNNs, or RNNs, that fits the given dataset. The encoder is used for obtaining the hidden representation of the input data, while the decoder tries to reconstruct the original input from the obtained representation. Therefore, the idea is that the learned representation should contain the relevant information for reconstructing the data.

Mathematically, we denote the decoder as $f_{\boldsymbol{\theta}}$ and the encoder as $g_{\boldsymbol{\phi}}$. The encoder extracts the hidden representation $\boldsymbol{h} = g_{\boldsymbol{\phi}}(\boldsymbol{x})$ from the input $\boldsymbol{x}$, then the decoder produces a reconstruction $\hat{\boldsymbol{x}} = f_{\boldsymbol{\theta}}(\boldsymbol{h})$ from $\boldsymbol{h}$. We train the encoder and decoder simultaneously by minimizing a reconstruction loss $\mathcal{L}(f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{x})), \boldsymbol{x})$, for instance mean-squared error loss, using SGD similarly as for the feedforward networks described above. There exist various kinds of methods for improving the quality of the learned representations in autoencoders. For example, we can adjust target task by adding noise to the inputs and let the decoder reconstruct the original input from noise variants [61], or we can induce different constraints in the bottleneck layer to, for example, obtain a sparse lower-dimensional representation of the data. Next, we will introduce the variational autoencoder which originates from latent variable models.

Variational Autoencoders

The variational autoencoder [62] (VAE) is a variant of autoencoders where learning is viewed from the perspective of probabilistic modeling. These models come from the family of deep generative models, where the goal is to approximate some underlying data distribution p_{data} with a parametric distribution $p_{\boldsymbol{\theta}}$ learned from a dataset $\mathcal{D} \sim p_{data}$. A common approach for estimating $p_{\boldsymbol{\theta}}$ is to use a latent variable model that infers hidden structures in the data to facilitate learning the distribution. VAEs is a deep latent variable model that uses neural networks for learning $p_{\boldsymbol{\theta}}$ making the training scalable to large high-dimensional datasets.

The main idea with introducing latent variables is that they should encode some semantically meaningful information about the observed data. Latent variable models are usually expressed by the joint distribution

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}), \tag{2.3}$$

where $\boldsymbol{z}$ denotes the latent variables and $\boldsymbol{x}$ the observed variables that represents the observed data points. The distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood of the data and $p(\boldsymbol{z})$ is the prior distribution for the latents. This model describes the generative process of the data $\boldsymbol{x}$ by following the steps 1) sample the latent vector $\boldsymbol{z} \sim p(\boldsymbol{z})$ from the prior, and 2) generate data point $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$ from the sampled latent $\boldsymbol{z}$. We are now interested in learning the model $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ that best fits a given dataset $\mathcal{D}$, as well as inferring the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ over the latent variables $\boldsymbol{z}$ given the data $\boldsymbol{x}$.

The overall goal with latent variable models is to maximize the marginal log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ given a dataset $\mathcal{D} \sim p_{data}$. However, computing $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by with marginalizing out the $\boldsymbol{z}$ from the model $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z}$ is in general intractable due to the many settings of $\boldsymbol{z}$ we would need to evaluate. Consequently, the posterior distribution also becomes intractable since $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})/p_{\boldsymbol{\theta}}(\boldsymbol{x})$ from Bayes' rule. Variational inference [63, 64] is a technique for enabling learning of latent variable models. The idea of variational inference is to provide means for calculating the marginal log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by selecting a parameterized distribution $q_{\boldsymbol{\phi}}$ for approximating the true posterior distribution. In VAEs, the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is represented as a neural network with parameters $\boldsymbol{\phi}$ that outputs the latents $\boldsymbol{z}$ given data points $\boldsymbol{x}$. With this approach, we can now form a lower bound on the marginal log-likelihood given by

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathbb{E}_{z \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})]. \tag{2.4}$$

The right-hand side is called the evidence lower bound (ELBO) and comprises of two quantities that we can evaluate to train the model. The expectation over the log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ can be estimated with Monte Carlo sampling. The KL divergence between $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ can be computed analytically depending on how we choose these distributions. The standard choice for the prior is to use a zero-mean unit-variance Gaussian distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ where $\mathbf{I}$ is the identity matrix. The approximate posterior is also selected to be a Gaussian distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})))$, where the encoder network parameterized by $\boldsymbol{\phi}$ outputs the the means $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and standard deviations $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$ for the latent dimensions. The latent vector is sampled using the "reparametrization trick" [62, 65] by computing $\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ denotes element-wise multiplication, which enables backpropagating gradients through the sampling operation. The likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is the decoder network that tries to reconstruct the original input to the encoder. The likelihood distribution depends on the type of data $\boldsymbol{x}$ we wish to generate. If $\boldsymbol{x}$ is a continuous variable, then we can let the decoder output Gaussian parameters for the likelihood similar as for the encoder.

VAEs have been used in various applications for modeling images, text, and audio data, as well as when combining data from different modalities. Next, we will briefly introduce how autoencoders can be used when learning representations from multiple data types from different modalities.

Multimodal Learning using Autoencoders

Learning representations from different types of data is a highly active research field in deep learning [66]. Combining visual data with other modalities such as natural language and audio signals for learning more rich representations of data has been studied frequently the last decade [67–70] [Add REFs]. A common framework in such applications is to use autoencoders for incorporating information from the different data types into a single joint representation. Learning from multiple sources then opens up for capturing correspondences between the data types and obtaining better representations that can be used for downstream tasks such as classification.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ originate from two different data sources but share some high-level information. For example, $\boldsymbol{x}$ could be an image of a living room and $\boldsymbol{y}$ is text describing the appearance of the room, where objects are located etc. Constructing a joint representation is then done by projecting both $\boldsymbol{x}$ and $\boldsymbol{y}$ with separate encoder networks into the same latent space. The joint multimodal representation is then passed through two separate decoder networks used for predicting the original input data individually. The advantage of multimodal autoencoders is that they can be trained end-to-end for both learning representations as well as making predictions of the used modalities. However, a major challenge is how to handle scenarios where modalities might be missing. One option is to only encode the data modality that we know will be available at both training and test phases and then establish a joint representation by decoding two both modalities [68].

Multimodal autoencoders have frequently been extended to deep generative models, mainly VAEs [53, 71–74]. These models are capable of generating new data through sampling from the latent space in addition to learning joint representations. Furthermore, they can handle missing modalities for the encoders which enables cross-modal data generation between the modalities. In Paper B [Add REF], we employ Variational Canonical Correlation Analysis (VCCA) for learning joint representations of natural images and web-scraped information of grocery items to facilitate learning image classifiers.

Deep Reinforcement Learning

In this section, we give a brief overview to Reinforcement Learning [75] (RL) and introduce some approaches for how to incorporate deep neural networks in RL. We begin by outlining the notation for the problem setup in RL that will be used in Paper D. Next, we introduce two popular methods for learning RL policies, namely, Deep Q-Networks [55, 76] (DQNs) and policy-based methods [77, 78].

The RL setup considers an agent interacting with an environment $\mathcal{E}$ over a number of discrete time steps. The environment is modeled with a Markov Decision Process [79] represented as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ an action space, $P(s'|s, a)$ is the state-to-state transition probability distribution with $s, s' \in \mathcal{S}$, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each time step t, the agent receives a state s_t from the environment, selects an action $a_t \in \mathcal{A}$ using a policy $\pi(a|s)$, and enters the next state s_{t+1} with transition probability $P(s_{t+1}|s_t, a_t)$ and receives a numerical reward following r_t from the environment. This procedure is repeated until the agent reaches a terminal state in which the procedure can be restarted. The return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted accumulated reward from time step t. The goal of the agent is to maximize the expected return from every state s_t.

The action value $Q^\pi(s,a) = \mathbb{E}[G_t | s_t = s, a]$ is the expected return for selecting action a in state s and following policy π. The optimal action value $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ is defined as the maximum action value for state s and action a for any given policy π. Similarly, the value function $V^\pi(s) = \mathbb{E}[G_t | s_t = s]$ defines the expected return following policy π from state s. In value-based RL methods, the action value function $Q_{\boldsymbol{\theta}}(s,a)$ is represented using function approximators, such as neural networks, parameterized by $\boldsymbol{\theta}$. A popular algorithm for updating the parameters $\boldsymbol{\theta}$ is Q-learning [80] where the goal is to directly approximate the optimal action value function as $Q^*(s,a) \approx Q_{\boldsymbol{\theta}}(s,a)$. The parameters $\boldsymbol{\theta}$ are learned by iteratively minimizing the loss

$$\mathcal{L}(\boldsymbol{\theta}_i) = \Big(r + \max_{a'} Q_{\boldsymbol{\theta}_{i-1}}(s', a') - Q_{\boldsymbol{\theta}_i}(s,a)\Big)^2 . \tag{2.5}$$

Next, we briefly introduce some popular algorithms in deep RL.

Deep Q-Networks. The Deep Q-Network [55,76] (DQN) is common algorithm for RL problems with discrete action spaces and high-dimensional state spaces where the Q-function is parameterized by a neural network. This method requires some additional steps for stabilizing the learning process, including the usage of target networks [76], applying L1-smoothing to the loss, and using experience replay [81] to sample training data stored in a replay buffer. Furthermore, the DQN can incorporate extensions that are also helpful for stable learning such as correcting the action value estimates [82] and imrpoving generalization across actions [83]. The DQN uses a greedy action selection strategy where the policy is given by $\pi(a|s) = \arg\max_{a'} Q_{\boldsymbol{\theta}}(s,a')$.

Policy-based Methods. The optimal policy can be estimated directly by using a parameterized form of the policy as $\pi_{\boldsymbol{\theta}}(a|s)$, where the parameters $\boldsymbol{\theta}$ represent a neural network in deep RL. The network takes a state s_t as input and outputs a distribution over the possible actions a_t. The agent collects experiences with the current policy to use for learning by interacting with the environment. The policy parameters are updated by minimizing the loss

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}[\log \pi_{\boldsymbol{\theta}}(a|s) \hat{A}(s_t, a_t)], \tag{2.6}$$

where $\hat{A}_{\boldsymbol{\theta}_v}(s_t, a_t)$ is an estimate of the advantage function given by $\sum_{i=0}^{k-1} = \gamma^i r_i + \gamma^k V_{\boldsymbol{\theta}_v}(s_{t+k}) - V_{\boldsymbol{\theta}_v}(s_t)$. The value function here is estimated with a neural network parameterized by $\boldsymbol{\theta}_v$, which usually share the parameters $\boldsymbol{\theta}$ with the policy. The policy and value function are updated after t_{max} actions or when a terminal state is reached [77]. A popular variant of policy-based methods is Proximal Policy Optimization [78] that incorporates constraints on the policy updates, as well as mini-batches and epochs of learning between each interaction with the environment.

Chapter 3

Fine-Grained Recognition

This chapter presents an approach for enhancing fine-grained classification performance of grocery items by using web-scraped information. We focus on classification of grocery items due to applicability in assistive vision and its potential to enhance the independence of visually impaired (VI) people [ADD groceries/shopping/object recognition for VI REFs]. Initially, we were interested in learning classifiers with natural images taken in the grocery stores combined with web-scraped information about the grocery items, such as iconic images and text descriptions from supermarket websites. Using iconic images have been used in grocery image classification earlier [ADD grocery paper REFs], however, utilizing text descriptions was as far we know absent for this application even if it has been successfully applied in other image classification problems [84–86]. Thus, we collected our own dataset of grocery items images using a mobile phone camera as well as web-scraped images and text descriptions to study whether this multi-view approaches would benefit training the classifiers (Section 3.2). We then select a multi-view learning framework based on the Variational Autoencoder (VAE) for investigating how the different data views affect the fine-grained classification performance (Section 3.3).

3.1 Related Work

In this section, we will briefly discuss the related work on fine-grained image recognition [41], particularly when learning from external information, and multi-view learning.

Fine-Grained Image Recognition

The goal with fine-grained image recognition (FGIR) is to distinguish between images with multiple visually similar sub-categories that belong to a super-category. For example, various attempts have been made to discriminate between sub-categories of different animals [87], cars [88], fruits [89], retail products [90], etc. The challenge is to recognize differences that are sufficient for discriminating between objects that are generally similar but differ in fine-grained visual details. In recent years, the successes with deep learning in computer vision have encouraged researchers to explore various approaches for FGIR that can broadly be divided into three directions for recognition by utilizing (i) localization-classification subnetworks, (ii) end-to-end feature encoding, and (iii) external information. In (i), the goal is to find object parts that are shared across the sub-categories for discovering details that make the part representations different. This can be achieved by utilizing feature maps from the activations of convolutional layers as local descriptors [91–93], employing detection and segmentation techniques for localizing object parts[REFs], or leveraging attention mechanisms when common object parts are difficult to represent or even define [REFs]. With (ii), the goal has been to learn features that are better at capturing subtle and local differences by, for instance, performing high-order features interactions [94] as well as designing novel loss functions [Add REFs]. In the third approach (iii), the goal is to leverage external information, for example, web data and multimodal data, in FGIR as additional supervision to the images. We will put more focus on the approach on FGIR with external information next, as we use this approach in Paper A and B.

Recognition with External Information

Learning fine-grained details about objects often requires large amounts of labeled data. To ease the need for large amounts of accurately labeled images, there have been several attempts to let either web-scraped or multimodal data influence learning the fine-grained features of the sub-categories to boost the FGIR performance. Web-scraped images may be noisy in the sense that retrieved images may have high-variations of the objects. For example, the objects of interest can look different in appearance, and there could also be other irrelevant objects in the images that potentially occlude the category to recognize. Hence, incorporating web-scraped data into the training set may establish a domain gap between the easily acquired web data and the original training set which we need to overcome by reducing the domain gap or reducing the negative effects of the noisy web data that can disturb the learning. Another direction than using web-scraped data is to utilize multimodal data, for example, images, text and knowledge bases, for boosting the classification performance. In FGIR, the goal is to establish a joint representation between the images and additional data sources, where the additional data should act like extra guidance for learning useful representations that capture the fine-grained details of objects. Text descriptions have been a popular data type to combine with images, which can be both easy and cheap to collect as they can be accurately generated by non-experts. High-level knowledge graphs of objects have also been used and can contain rich knowledge useful for fine-grained recognition. In addition to FGIR, both web-scraped and multimodal external information has been used for zero-shot learning to transfer knowledge from annotated categories to new fine-grained categories. In Paper A, we collect web-scraped images and text descriptions of grocery items to accompany real-world images of groceries for FGIR. Then, in Paper B, we perform a study using multi-view learning to investigate how the external information can enhance the classification performance. Next, we will cover the related work for the multi-view learning approach that we used.

Multi-view Learning

Learning from combinations of different data sources has been studied frequently in machine learning [66, 95]. Especially, there has been many research efforts made in the intersection of computer vision and natural language processing for applications in image captioning [12,96,97], visual question answering [38,40,98], and learning joint representations with visual and text data [71–73]. In this thesis, we focus on the latter where we apply ideas from multi-view learning [95] for learning joint representations across data views. A view can be defined as any signal or data measured by some appropriate sensor [99]. A common approach for learning joint representations from multiple views is to assume that each view have been generated from the same latent space shared between all views. An example method is Canonical Correlation Analysis [100] (CCA) where the goal is to linearly project pairs of different views into a lower-dimensional space where the correlation between the projections is maximized. There exist various extensions of CCA which uses deep neural networks for learning nonlinear mappings to extract more rich features of the views, such as Deep CCA [101] and Deep Canonically Correlated Autoencoders [102]. Inspired from the Deep CCA variants, Variational CCA [53] (VCCA) is a multi-view deep generative model that can both learn joint representations and generate new data by sampling from the shared latent space. In the case of noisy views, an advantage of VCCA is that it can be extended to modeling both a shared latent space and private latent spaces for each view which capture view-specific variations to ease the learning of shared variations [53, 99, 103–106]. In Paper B, we investigate whether the classification performance of grocery items can be improved by extracting the view-specific variations in the web-scraped views by comparing the classification performance of the joint latent representations from standard VCCA and the VCCA-private.

Datasets for Fine-Grained Image Recognition

There exist many image datasets for benchmarking computer vision models [84, 85, 87, 107–111]. These datasets ranges from large-scale datasets annotated by groups of classes, such as [107,112], to smaller but densely-annotated datasets [109, 113–115]. There also exist domain-specific datasets for FGIR tasks of animals and species [87, 116–120], fashion [121, 122], airplanes and cars [123–127], faces [128–131], and foods [89, 132]. Several of the mentioned datasets have also been extended to benchmarking in zero/few-shot image classification [86, 133–135]. In computer vision for visually impaired people, we have recently seen an emergence of datasets collected by people who are blind/low-vision [6,32,110,136,137]. Datasets for FGIR of grocery items in their natural environments, such as grocery stores, shelves, and kitchens, have been addressed in plenty of previous works [90,138–142]. Similarly, there exist other kinds of food datasets with images of various food dishes [132, 143–145], cooking videos [146, 147], recipes [148–150], and also restaurant-oriented information [151, 152]. Our grocery item dataset (see Section 3.2) shares many similarities with the mentioned works above, for instance, all images of raw and packaged groceries are taken in their natural environment, images are taken with a mobile phone camera from an egocentric view-point, grocery classes are hierarchically labeled, and each class have an additional web-scraped iconic image and text description.

3.2 Dataset Collection

In this section, we describe our procedure for collecting the image dataset of grocery items in Paper A. As the target use case is grocery shopping with an assistive vision device, we visited several supermarkets and collected natural images of the groceries with a mobile phone camera to imitate such scenarios. Hence, the collected images will capture situations that can be challenging for the assistive device, such as, various lighting conditions, multiple instances and classes present, hand occlusions, and misplaced items. All images were taken with a single targeted item in mind, such that each image is paired with a single label. For items which belong to a clear super-class, for example, various kinds of apples and milk packages, we also provided the general class of the items to establish a hierarchical labeling structure of the data. Collecting natural images of the grocery items is unfortunately a time-consuming process. Furthermore, as the surroundings in every grocery store varies, it may be difficult to build accurate classifiers that can recognize fine-grained details solely from natural images. Hence, we need some cheaper procedure that can complement the collection of real-world images for boosting the classification performance of the groceries.

We have complemented the image dataset with external information from the web of each grocery item that can be used for training classifiers. In the past years, most supermarket chains have the option for consumers to purchase groceries online from their websites. The website usually provides each grocery item with an iconic image of the item on a white background, a text description that describes the flavor and ingredients of the item, as well as nutrition values if applicable. We downloaded these information types of all grocery item classes by web-scraping the online shopping website of a supermarket chain. We show four examples of grocery items and their web-scraped information in Table 3.1. Since these data types are on a class-based level, we can use the web-scraped information as weak supervision to guide the classifier to learn fine-grained details

Table 3.1: Examples of grocery item classes in the Grocery Store dataset. We display four different items (coarse-grained class in parenthesis), followed by two natural images taken with a mobile phone inside grocery stores. Next comes the web-scraped information of the items consisting of an iconic image and a text description. We have highlighted ingredients and flavors in the text description that are characteristic for the specific item.

Class Labels	**Natural Images**	**Iconic Images**	**Text Descriptions**
Granny Smith (Apple)			*“...**green** apple with **white, firm** pulp and a **clear acidity** in the flavor.”*
Royal Gala (Apple)			*“...**crispy** and **very juicy** apple, with **yellow-white** pulp. The peel is **thin** with a **red yellow** speckled color.”*
Tropicana Mandarin (Juice)			*“. . . is a **ready to drink** juice **without pulp** pressed on **orange**, **mandarin** and **grapes**. Not from concentrate. Mildly **pasteurized**.”*
Yoggi Vanilla (Yoghurt)			*“...**creamy vanilla** yoghurt original... added **sugar** than regular flavored yoghurt. Great for both **breakfast and snacks**.”*

that helps discriminating between visually similar items.

3.3 Fine-Grained Classification of Grocery Items

This section describes the approaches we used for classification of grocery items from the available data types in the collected dataset. We begin by introducing the problem setting, followed by describing the methods for learning representations from the available data views.

Problem Setting

The application we are focusing on is grocery shopping with an assistive vision device. The device could for instance be a mobile phone app where the groceries are recognized by an in-built image classifier from natural images taken with the camera. Training such image classifier to be robust in grocery store environments would typically require an immense amount of labeled training examples of all available groceries. To reduce the need for labeled training data, we aim to combine the collected natural images with web-scraped information about the groceries when training the classifier. The goal is that incorporating the web-scraped information should help the classifier to learn fine-grained details about the items to enhance the classification performance and robustness.

The available data views that are available for training image classifiers is denoted as follows:

- $\boldsymbol{x}$: Natural images of the grocery items in the grocery stores.
- $\boldsymbol{y}$: Class labels of grocery items from corresponding natural images.
- $\boldsymbol{i}$: Iconic images of the grocery items scraped from a supermarket website.
- $\boldsymbol{w}$: Text description of the grocery items scraped from the same supermarket website as $\boldsymbol{i}$.

The simplest approach is to take a standard supervised approach and train a CNN from the natural image and class label pairs. An alternative is leverage from CNNs pre-trained on a large dataset, such as Imagenet [107], and fine-tune the final classification layer to the grocery item recognition task [153]. We use ideas from multi-view learning [95] and VAEs [62] for learning joint representations from the available data views that can be used for training the image classifiers, which we present in the next section.

Multi-view Representation Learning of Grocery Items

This section describes the approach we took for learning representations of grocery items that are shared across the available data types. We employ a deep latent variable model called Variational Canonical Correlation Analysis [53] (VCCA) for learning the shared representation. The main assumption in VCCA is that each data view have been generated from the same latent space. The goal then is to learn this latent space that captures the correspondences between all views into representations shared across the views for the grocery items. This representation can then be utilized for enhance the learning more accurate classifiers as well as for performing tasks such as synthesis and prediction of novel images. Next, we describe how to enable learning the shared latent space.

Capturing variations from each view in the learned representation is performed by predicting the original views from the latent space. To obtain the latent representation, we extract the representation by encoding the natural images with neural network. The extracted representation is then used for predicting each view individually by inputting the representation through separate neural networks. Note that we only use the natural images for extracting the latent representation here since it is the only view that is available at test time when we want to use the learned classifier in the grocery store. We have two options for exploiting the new representation to train classifiers. The first option is to train the classifier with the latent representations after we have learned the latent space as described above. The second option is to train the classifier and learning the latent space simultaneously by adding an additional classifier network predicting the class label with the latent representation as input.

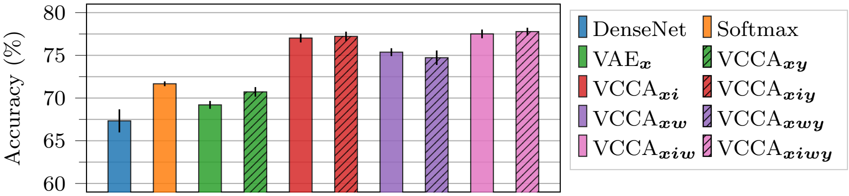


Figure 3.1: Fine-grained accuracies on the Grocery Store dataset for all classification methods. We show the means and standard deviations averaged over 5 seeds. Adding the iconic image $\boldsymbol{i}$ and text description $\boldsymbol{w}$ for learning joint representations with VCCA improves the classification performance over approaches that only utilize the natural images and class labels.

MK: TO-DO: I should add a figure of the architecture for extra clarity on the method.

3.4 Experiments

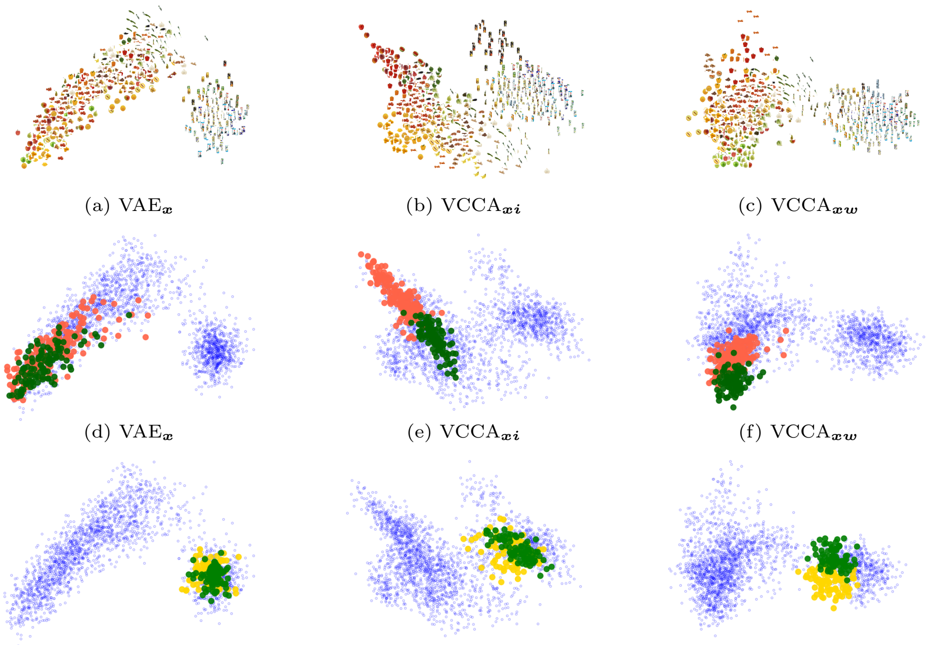
In this section, we summarize the main results from the experimental study in Paper B. We performed an ablation study with VCCA over the combinations of available data views to investigate how each data view contributes to the classification performance. First, we present results on fine-grained classification performance between the compared methods. Then, we provide insights in how the web-scraped icnoc images and text descriptions contribute to the boosting the classification performance by visualizing their the joint latent spaces. Finally, we demonstrate how the iconic image decoder in VCCA can be used for explaining the misclassifications by generating iconic images from natural images at test time.

We compare the VCCA models against two CNN baselines that uses the DenseNet [154] architecture. The first baseline is a DenseNet trained from scratch on the dataset, and the second baseline is a Softmax classifier trained from image features extracted from a DenseNet pre-trained on ImageNet. We denote which data views that are used by VCCA using subscripts. For instance, VCCA$_{\boldsymbol{xiw}}$ means that the natural images $\boldsymbol{x}$, iconic images $\boldsymbol{i}$, and text descriptions $\boldsymbol{w}$ are utilized for learning the joint latent representation. These VCCA models uses the two-stage classifier setup with steps 1) train VCCA on the data views, and 2) train 1-layer MLP classifier using the extracted latent representations from VCCA and the corresponding class labels. We also compare against VAE$_{\boldsymbol{x}}$ only using the natural images $\boldsymbol{x}$ trained in this setting. The VCCA models with class label decoders are denoted by using $\boldsymbol{y}$ in the subscript, such as VCCA$_{\boldsymbol{xiwy}}$.

Fine-Grained Classification Results. Adding the web-scraped views in VCCA improves the classification performance over approaches that only utilize the natural images and class labels. Figure 3.1 shows a bar plot over the fine-grained accuracies achieved by all classification methods. Among the methods only using natural images and class labels, we see that the Softmax baseline performs best, which could be due to some information loss when compressing the images into as low-dimensional latent representations with VAE$_{\boldsymbol{x}}$ and VCCA$_{\boldsymbol{xy}}$. The performance of VCCA significantly improves over Softmax when the iconic image $\boldsymbol{i}$ and text description $\boldsymbol{w}$ are used, which shows that both data views are useful for enhancing the fine-grained classification performance. Comparing VCCA$_{\boldsymbol{xi}}$ and VCCA$_{\boldsymbol{xw}}$, we see that utilizing the iconic image has an advantage over using the text description for improving the performance. This could be due to the fact that the text descriptions are providing information on ingredients and flavors rather than visual appearance. Combining both $\boldsymbol{i}$ and $\boldsymbol{w}$ achieves on par performance as only utilizing the iconic image, which could potentially be improved by filtering the text descriptions for obtaining words relevant for describing the fine-grained details of the items. Finally, we observe that both classification options for VCCA performs similar, which could potentially be since $\boldsymbol{i}$ and $\boldsymbol{w}$ acts as labels since there is only a single instance of these views for all classes.

Visualization of Latent Space. The web-scraped iconic images and text descriptions structures the grocery items based on view-specific similarities in the latent space that are beneficial for fine-grained classification. In Figure 3.2, we illustrate how adding either the iconic image $\boldsymbol{i}$ or the text description $\boldsymbol{w}$ changes the structure of the latent space, where we have used PCA to project the latent representations into a 2-dimensional space. In Figure 3.2(a-c), we have plotted the corresponding iconic image for all latent representations of models VAE$_{\boldsymbol{x}}$, VCCA$_{\boldsymbol{xi}}$, VCCA$_{\boldsymbol{xw}}$ for visualization purposes. The latent space for VAE$_{\boldsymbol{x}}$ separates raw and packaged grocery items into two separate clusters. When adding the iconic image in VCCA$_{\boldsymbol{xi}}$, we observe that the latent space becomes structured according to the color of the items. For VCCA$_{\boldsymbol{xw}}$, the latent space becomes structured according to the ingredients of the items, where we see in the raw food cluster that bell peppers are placed in the upper region while apples are in the lower region.

Next, we focus on a certain set of classes to inspect to gain more insights in how the additional data views affect the latent space. First, we focus on the green and red apples classes to inspect how the views handle visually different items in color. In Figure 3.2(d-f), we plot the latent representations for the three models but highlight the green and red apple classes by plotting them as green and red dots respectively. All other classes are plotted as smaller blue dots. We see that both VCCA$_{\boldsymbol{xi}}$ and VCCA$_{\boldsymbol{xw}}$ manages to separate the apple classes better than VAE$_{\boldsymbol{x}}$, where adding the iconic images yields the most clear separation. Next, we want to study the benefits of the text descriptions. We focus on some yoghurt and juice package classes that are visually similar but have very different ingredients and flavors. In Figure 3.2(g-i), we plot the latent representations for the three models again where the yoghurt and juice classes are plotted in green and yellow colored dots respectively. Here, we see that VCCA$_{\boldsymbol{xw}}$ manages to separate these items better than VCCA$_{\boldsymbol{xw}}$ due to the differences in the text descriptions between



(a) VAE$_{\boldsymbol{x}}$ (b) VCCA$_{\boldsymbol{xi}}$ (c) VCCA$_{\boldsymbol{xw}}$

(d) VAE$_{\boldsymbol{x}}$ (e) VCCA$_{\boldsymbol{xi}}$ (f) VCCA$_{\boldsymbol{xw}}$

(g) VAE$_{\boldsymbol{x}}$ (h) VCCA$_{\boldsymbol{xi}}$ (i) VCCA$_{\boldsymbol{xw}}$

Figure 3.2: Visualizations of the latent representations from VAE$_{\boldsymbol{x}}$, VCCA$_{\boldsymbol{xi}}$, and VCCA$_{\boldsymbol{xw}}$ projected in 2-dimensional space with PCA. In (a-c), we show the latent representations plotted using the iconic images of the corresponding object class. In (d-f), we illustrate how the iconic images structures the items based on visual similarities by focusing on the green and red apple classes in the dataset plotted in their corresponding colors. Similarly, in (g-i), we show how the text descriptions structure items based on ingredients and flavor by focusing on visually similar yoghurt (green) and juice (yellow) packages. The blue dots correspond to all other grocery item classes.

the selected package classes.

Iconic Image Generation. The iconic image decoder can provide explanations for the predicted classes. Figure 3.3 shows two examples of decoded iconic images from two natural images where the class labels are *Orange Bell Pepper* and *Anjou Pear*. On the first row, we see that VCCA$_{\boldsymbol{xiwy}}$ has recognized the green bell peppers in the natural image and generated a mixed orange and green bell pepper in the iconic image. On the second row, we see that the decoded iconic image is a *Granny Smith* apple instead of a pear

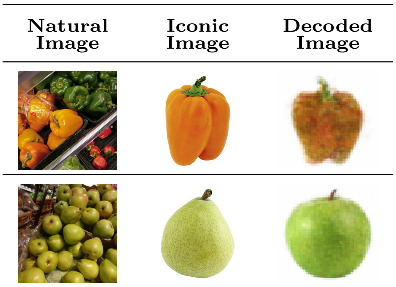


Figure 3.3: Examples of decoded iconic images from VCCA$_{\boldsymbol{xiwy}}$ with their corresponding natural image and true iconic image.

which was the true class. The classifier consequently predicts the natural image to be a *Granny Smith* apple. Hence, the iconic image decoder can be used as a tool for providing an intuition of why the classifier made an error.

3.5 Discussion

In the experiments, we showed that utilizing the web-scraped information with VCCA can enhance the performance of grocery item classifiers. This shows that the cheaper web-scraped views can serve as a good alternative for improving classification performance over collecting more natural images in the grocery stores. Furthermore, we illustrated how the iconic images and text descriptions affects the structure of the latent space based on view-specific information. More specifically, the iconic images structures the latent space after visual similarities such as colors and shapes, while the text descriptions pushes items with similar ingredients and flavors closer to each other in the latent space. Finally, we demonstrated how the iconic images can be used for providing potential explanations for misclassifications, which could help us detect hard classes and give us indications of how robust the latent representations are to classifying images with different classes.

We observed that utilizing the iconic images in VCCA affects the classification performance significantly better than the text descriptions. Potentially, this is due to the text description view to be more noisy than the iconic images as there are few words that are relevant to the recognition task. However, the text descriptions could be utilized more efficiently, for example by pre-processing the text to keep words describing fine-grained details about the items as well as removing stop words ('the', 'it', 'and', etc.) and other irrelevant words. A second option would be to use attention mechanisms [155, 156] that helps the model to learn which words to emphasize on when learning the joint latent representations. We could also encode the text into a single-vector embedding with various methods [157–159] and replace the RNN with an MLP predicting in the embedding space which the text embedding the natural image is closes to.

Regarding the dataset collection, we have suggested extensions on how to provide more useful information about the items to improve the recognition task. Firstly, it would be valuable to download instances of iconic images and text descriptions from more supermarket websites to potentially allow the VCCA model to capture more view-specific variations into the learned representations. Secondly, the model should be extended to handle video data rather than still images. This would require record videos with the mobile phone camera in the grocery stores to properly evaluate the classifiers. Nevertheless, the extension to video would be beneficial for user experience of the recognition app since the classifier would receive more chances to classify the items correctly by utilizing multiple frames.

Finally, the Grocery store dataset could be extended to zero/few-shot learning [160, 161] and continual learning [42, 43] settings. Such applications are important for assistive vision devices to build data-efficient and adaptable systems that improves their usability in real-world scenarios.

Chapter 4

Continual Learning

This chapter introduces the idea of replay scheduling for mitigating catastrophic forgetting in continual learning (CL). The problem setting of CL is on learning tasks of recognizing a new set of classes with a dataset given at the current time step. In the standard setting, one main assumption is that the data from past tasks can never be fully revisited by the model. However, in the real-world, many organizations record data from incoming streams for storage rather than deleting it [162, 163] [Add at least 1 more REF]. In contrast to the assumption on data storage in standard CL, we suggest a new setting where we assume that all seen data is accessible at any time for the model to revisit. The challenge then becomes how to select which tasks that needs to be remembered via replay as the data is still incoming from a stream. We propose to learn the time when replaying a certain task is necessary when the model is updating its knowledge with new incoming tasks. In Paper C, we propose the new CL setting where historical data is accessible and introduce the idea of replay scheduling and how it can be used in CL. In Paper A, we propose a framework based on reinforcement learning [75] (RL) for learning replay scheduling policies that can be applied in new CL scenarios.

4.1 Related Work

In this section, we give an overview of previous works related to Paper C and D. We begin by describing different approaches in CL, especially replay-based approaches, and then discuss some approaches for fostering generalization in RL.

Continual Learning. There exist many different approaches in CL for mitigating catastrophic forgetting in neural networks. In general, these approaches can be divided into three main cores, namely, *regularization-based*, *architecture-based*, and *replay-based* methods. Regularization-based methods are mainly focused on applying regularization techniques on parameters important for recognizing old tasks and fit the remaining parameters to new tasks [164–166]. Knowledge distillation methods [167] also belong to these approaches where classification logits are used for regularizing the output units for previous tasks in the network [168, 169]. More recently, there are some works that uses projection-based approaches for constraining the parameter updates to subspaces which avoid interference with previous tasks [170, 171]. Architecture-based approaches focuses on adding task-specific network modules for every seen task [172–175], or isolating parameters for predicting specific task in fixed-size networks [176–178]. Replay-based methods re-trains on samples of old tasks when learning new tasks. The old samples are either stored in an external memory [46,179,180], or synthesized with a generative model [72,181–183]. Both regularization- and architecture-based methods can be combined with replay for improving the models capability of remembering tasks [Add REFs]. Our replay scheduling idea is originated from replay-based methods which we will cover more in detail next.

Replay-based Continual Learning. In this thesis, we focus on replay-based methods with external memories for storing historical data. The most common selection strategy for filling the memory is random sampling from the used datasets. There exist several works focusing on selecting high quality samples for storing in the memory [Add REFs]. However, in image classification problems, random sampling has been shown to often perform on par with more elaborate selection strategies [179,184]. In contrast to using various memory selection methods, there has been proposals of retrieval policies over which samples to select for replay from the memory, for instance, selecting the samples that will mostly interfere with the parameter update with batches of new data [185]. Our replay scheduling approach differs from this method as we focus on selecting which tasks to select for replay rather than the individual samples to retrieve from the memory. More recent works have focused on evolving the memory samples through data augmentation to avoid overfitting to the memory [Add REFs], and also by using contrastive learning to improve discriminating between tasks. Another direction has been to increasing the storage capacity to store more samples by compressing raw data into features that are more memory-cheap [Add REFs]. The above mentioned methods assume that the memory is small and allocates equal storage amount for all tasks. Our new problem setting for memory-based CL is different from this assumption as we argue that data storage is cheap in many real-world applications. Hence, we compose a replay memory with data from historical tasks before learning new tasks because the amount of compute is limited. However, replay scheduling can be combined with of the mentioned methods as it only differs with the standard memory-based CL setting in that the replay memory has to be selected at every new task.

Generalization in Reinforcement Learning. Generalization in RL is an active research field where much focus has been put on developing proper benchmark datasets for evaluating generalization capabilities of RL agents [186–189]. Regularization techniques from supervised learning have been used to investigate whether these can enhance generalization in RL [186, 190, 191], and also how variations in the environments help to obtain agents that generalize better [192,193]. Algorithms for enabling fast adaptation to new environments, such as new tasks [194, 195] and new action spaces [196, 197], has also been studied. The survey by Kirk et al. [198] focuses on zero-shot policy transfer where the policy must generalize to unseen dynamics in the test environments as additional queries could be disallowed in real-world scenarios [199, 200]. In Paper A, we consider using RL for learning policies for scheduling which tasks to replay in CL settings to mitigate catastrophic forgetting in the classifier. The policy is trained using experiences from multiple CL environments and then tested in new CL environments with unseen dynamics such as new task orders and datasets.

4.2 Replay Scheduling in Continual Learning

In this section, we introduce a slightly new CL setting considering the real-world needs where all historical data can be available since data storage is cheap. However, the amount of compute is limited when the model is updated on new data due to operational costs. Hence, it is impossible for the model to leverage from all the available historical data to mitigate catastrophic forgetting. The goal then becomes to learn how we can select subsets of historical data for replay to efficiently reduce forgetting of the old tasks. We will refer to these subsets of historical data as the *replay memory* throughout this chapter. The size of the replay memory affects the processing time when learning new tasks as well as the allowed time for the training phase. When composing the replay memory, we focus on determining the number of samples to draw from the seen tasks in the historical data rather than selecting single stored instances. Next, we introduce the problem setting in more detail as well as the notation of the new CL setting.

Problem Setting

We introduce the notation of our problem setting which resembles the traditional CL setting for image classification. We let a neural network $f_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta}$, learn T tasks from the datasets $\mathcal{D}_1, \dots, \mathcal{D}_T$ arriving sequentially one at a time. The t-th dataset $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, \boldsymbol{y}_t^{(i)})\}_{i=1}^{N_t}$ consists of N_t samples where $\boldsymbol{x}_t^{(i)}$ and $\boldsymbol{y}_t^{(i)}$ are the i-th data point and class label respectively. The training objective at task t is given by

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N_t} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(i)}), \boldsymbol{y}_t^{(i)}), \tag{4.1}$$

where $\mathcal{L}(\cdot)$ is the loss function, which in our case is the cross-entropy loss. When learning task t, the network $f_{\boldsymbol{\theta}}$ is at risk of catastrophically forgetting the previous $t-1$ tasks. The forgetting effect shows as the decrease in task accuracy between time steps, for example, $A_{t,i} < A_{t,i-1}$ where $A_{t,i}$ is the accuracy for task t at time step i. Replay-based methods mitigate catastrophic forgetting by storing a few number of examples from historical tasks in an external memory. The network $f_{\boldsymbol{\theta}}$ is then allowed to fetch old examples from the memory and mix these with the current task dataset to remind itself about the previous tasks during training.

In the new problem setting, we assume that historical data from old tasks are accessible at any time step. However, we can only fill a small replay memory $\mathcal{M}$ with M historical samples for replaying old tasks due to processing time constraints prohibiting re-using all historical data at the same time. The challenge then becomes how to know which tasks to include in the replay memory that efficiently retain the previous knowledge when learning new tasks. We decide to fill the replay memory with M historical samples using sequence of task proportions $(p_1, \dots, p_{t-1})$ where $\sum_{i=1}^{t-1} p_i = 1$ and $p_i \geq 0$. The number of samples from task i to place in $\mathcal{M}$ is given by $p_i \cdot M$. In the next section, we introduce a method for selecting the task proportions of which old tasks to replay.

Comparison to Traditional CL. The new setting has several similarities to the traditional CL setting. Both settings share the fundamental setting that the data arrive in streams and re-training on all historical data is prohibited. Also, the goal that the model should perform well both historical tasks and tasks associated with new data remains the same. In replay-based CL, we also share the same constraints that the memory size is limited. However, we argue that this limitation is mainly associated with compute rather than of storage. Our assumption aligns with the real-world where data storage is cheap and easy to maintain, but retraining large machine learning models is computationally expensive. The only difference is that we allow filling the limited replay memory from historical data or some other external memory. Here, we argue that historical data is stored rather than deleted in many real-world settings [162]. Thus, we should keep the limited memory assumption for training but allow access to historical data to fill the replay memory to make CL align with real-world needs.

Replay Scheduling for Mitigating Catastrophic Forgetting

In this section, we describe our replay scheduling method for selecting the replay memory at different time steps. A replay schedule is defined as a sequence $S = (\boldsymbol{p}_1, \dots, \boldsymbol{p}_{T-1})$, where $\boldsymbol{p}_i = (p_1, \dots, p_{T-1})$ for $1 \leq i \leq T-1$ is the sequence of task proportions for determining how many samples per task to fill the replay memory with at time step i. To make the selection of task proportions tractable, we construct an action space with a discrete number of choices for the task proportions from historical tasks.

We show the procudre for creating the discrete action space in Algorithm 1. At task i, we have $i-1$ historical tasks that we can choose from. We then generate all possible bin vectors $\boldsymbol{b}_i = [b_1, \dots, b_i] \in \mathcal{B}_i$ of size i where each element are a task index $1, ..., i$. We sort all bin vectors by the order of task indices and only keep the unique bin vectors. For example, at $i = 2$, the unique choices of vectors are $[1, 1], [1, 2], [2, 2]$, where $[1, 1]$ indicates that all samples in the replay

Algorithm 1 Discretization of action space with task proportions

```
Require: Number of tasks $T$
$\mathcal{T} = ()$                                  ▷ Initialize sequence for storing actions
for $i = 1, \dots, T-1$ do
    $\mathcal{P}_i = \{\}$                          ▷ Set for storing task proportions at $i$
    $\mathcal{B} = \texttt{combinations}([1:i], i)$  ▷ Get bin vectors of size $i$ with bins $1, ..., i$
    $\bar{\mathcal{B}} = \texttt{unique}(\texttt{sort}(\mathcal{B}))$  ▷ Only keep unique bin vectors
    for $\boldsymbol{b}_i \in \hat{\mathcal{B}}$ do
        $\boldsymbol{p}_i = \texttt{bincount}(\boldsymbol{b}_i)/i$   ▷ Calculate task proportion
        $\mathcal{P}_i = \mathcal{P}_i \cup \{\boldsymbol{p}_i\}$   ▷ Add task proportion to set
    end for
    $\mathcal{T}[i] = \mathcal{P}_i$                ▷ Add set of task proportions to action sequence
end for
return $\mathcal{T}$                                ▷ Return action sequence as discrete action space
```

memory should be from task 1, $[1, 2]$ indicates that half memory is from task 1 and the other half are from task etc. The task proportions are then computed by counting the number of occurrences of each task index in $\boldsymbol{b}_i$ and dividing by i, such that $\boldsymbol{p}_i = \texttt{bincount}(\boldsymbol{b}_i)/(i)$. From this specification, we have built a tree $\mathcal{T}$ with different task proportions that can be selected at different time steps. We construct a replay schedule S by traversing through $\mathcal{T}$ and select a task proportion on every level to append to S. We can then evaluate the replay schedule S by training a network on the CL task sequence and use S to compose the replay memory to use for mitigating catastrophic forgetting at every task.

We are interested in studying whether the time to replay different tasks is important in the new CL setting. One option is to use a brute-force approach, for example, breadth-first search, and evaluate every possible replay schedule in the tree. However, as the tree grows fast with the number of tasks, we need a scalable method that can perform searches in large action spaces. We suggest using Monte Carlo tree search [201] (MCTS) due to its previous successes in applications with similar conditions as ours [58, 202, 203]. The use of MCTS enables performing search for datasets with longer task horizons. Furthermore, MCTS encourages searches in promising paths based on the selected reward function, which we set as the average accuracy of all tasks after learning the final task achieved by the network. We provide the full details on how MCTS is used to search for replay schedules in Paper C.

4.3 Meta Policy Learning for Replay Scheduling

In this section, we present an RL-based framework to learn policies for selecting which tasks to replay in CL scenarios. We are interested in learning such policy that can be transferred to new CL scenarios, such as new task orders and new datasets, without any additional computational cost for updating on the new domain. We take a meta-learning approach where the policy learns from episodes of experience collected from training a classifier in CL settings. The experience from the environment is represented as the classification performance on each seen task in the dataset. The policy receives the task performances for basing its action on which task that needs to be replayed at the next time step. Our goal is to obtain a policy that can generalize to be used for replay scheduling in new CL scenarios to mitigate catastrophic forgetting. Next, we describe in more detail how the framework for learning this policy works.

Problem Setting

We consider the setting where an agent selects replay schedules to mitigate catastrophic forgetting in a classifier trained in the CL setting. The environment that the agent interacts with contains a network $f_{\boldsymbol{\phi}}$ and a dataset $\mathcal{D}_{1:T}$ of T tasks that $f_{\boldsymbol{\phi}}$ should learn in sequential order. The dataset is split into training, validation, and test sets as $\mathcal{D}_{1:T} = \{\mathcal{D}_{1:T}^{\text{train}}, \mathcal{D}_{1:T}^{\text{val}}, \mathcal{D}_{1:T}^{\text{test}}\}$ respectively. The training sets $\mathcal{D}_{1:T}^{\text{train}}$ are for the network to learn all T tasks sequentially, while the $\mathcal{D}_{1:T}^{\text{val}}$ are for evaluating how well the network performs on each task during training. The task performances on the validation sets can be used for dense rewards to the RL agent. The test sets are for final evaluation and are unseen during training as standard practice to avoid overfitting.

We consider having a distribution $p(E)$ of Markov Decision Process [79] (MDP) where the MDP is represented as a tuple $E_i = (\mathcal{S}_i, \mathcal{A}, P_i, R_i, \mu_i, \gamma)$ consisting of the state space $\mathcal{S}_i$, action space $\mathcal{A}$, state transition probability $P_i(s'|s,a)$, reward function $R_i(s,a)$, initial state distribution $\mu_i(s_0)$, and discount factor γ. For training, we assume that we have access to a fixed set of training environments $\mathcal{E}_{\text{train}} = \{E_1, \dots, E_K\}$, where $E_i \sim p(E)$ for $i = 1, ..., K$. Each environment E_i contains of a network $f_{\boldsymbol{\phi}}$ and T datasets $\mathcal{D}_{1:T}$ where the t-th dataset is learned at time step t. We let the dynamics of the environments depend on the network initialization and the task order of the datasets. The states s are given by the validation performance on each task from $f_{\boldsymbol{\phi}}$ which can be accuracies. Hence, the state space $\mathcal{S}_i$ depends on the parameter initialization for $\boldsymbol{\phi}$. Regarding the datasets, we allow the task orders in the datasets $\mathcal{D}_{1:T}$ to be shuffled for each sampled environment $E_i \sim p(E)$. Therefore, the state space is also affected by the task order as this can yield different CL performances. We assume that the action space is the same for every environment as the agent will interact with each training environment in $\mathcal{E}_{\text{train}}$.

The goal for the agent is to maximize the accumulated returns in each training environment. The reward is given by the CL performance. We assume that we have a dense reward that we can compute from the validation sets at each time step t. We are therefore after a policy that works well on all environments. The intention for learning such policy is that it should generalize across environments of new CL settings, such as new task orders and datasets. As we are not allowed to go back in time in CL, we are not allowed to query the testing environments during the transfer. Hence we must apply the policy in a zero-shot setting where tuning the policy on the testing environment is prohibited.

Deep Q-Networks for Learning Replay Scheduling Policy

We employ DQNs for learning the policy for replay scheduling. The architecture takes states as input and outputs action values for the valid actions at the current time step t. We define the states s as a vector with the validation performance of every seen task, where we use the validation accuracy to represent the task performance. We use zero-padding on the vector elements in the states that represent the performance for future tasks. The states can also be represented by other performance metrics, for example, the accuracy difference between time steps or forgetting measures such as backward transfer [204] (BWT). For the actions, we use the discrete action space built using Algorithm 1 such that the action space $\mathcal{A}_t$ is time-dependent and grows per seen task. To handle the growing action space, we assume that we know the total number of actions such that the DQN can use a fixed output layer. Therefore, we use action masking on the predicted action values to prevent the DQN from selecting invalid actions.

We outline the procedure for training the DQN on multiple CL environments in Algorithm 2. The main idea to obtain a policy that generalize is to train it on several environments with different dynamics for learning from more diverse training data [193]. We let the DQN store transitions $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ from all environments in the same experience replay buffer $\mathcal{B}$ to use for training the network. The first step is to receive the intial state s_1 for all environments by training the network f_ϕ on the first dataset $\mathcal{D}_1^{\text{train}}$ in each environment, such that $s_1^i = [A_{1,1}^{\text{val}}, 0, ..., 0]$. The DQN then selects the action a_t for the current environment from the input state or selects a random action with probability ϵ. The agent retrieves the reward and next state by training the environment-specific network f_ϕ^i on the current task t in `CLStep`. Before training starts, the action a_t^i is used for sampling the replay memory $\mathcal{M}_t$ using the task proportions $(\boldsymbol{p}_1, \dots, \boldsymbol{p}_{T-1})$ translated from a_t^i. The replay memory is then used for mitigating catastrophic forgetting when training on the current task. After training, the agent obtains the reward from the reward function

$$R(s, a) = \frac{1}{t} \sum_{i=1}^{t} A_{t,i}^{\text{val}}, \tag{4.2}$$

where $A_{t,i}^{\text{val}}$ is the validation accuracy on task i after f_ϕ has learned task t. The next state is also obtained by utilizing the validation accuracies up to task t, such as $s_{t+1}^i = [A_{t,1}^{\text{val}}, ..., A_{t,t}^{\text{val}}, 0, ..., 0] \in [0, 1]^{T-1}$. The DQN is then updated by sampling a mini-batch of B samples from the shared buffer $\mathcal{B}$ and taking a

Algorithm 2 Learning replay scheduling policy with DQN

```
Require: $\mathcal{E}_{\text{train}}$: Training environments
Require: $\boldsymbol{\theta}$: DQN parameters
$\mathcal{B} = \{\}$                                                    ▷ Initialize replay buffer
for $i = 1, \dots, n_{\text{episodes}}$ do
    $s_1^i \sim \mu_i \, \forall E_i \in \mathcal{E}_{\text{train}}$                                ▷ Get initial states
    for $t = 1, \dots, T-1$ do
        for $E_i \in \mathcal{E}_{\text{train}}$ do
            $a_t^i = \arg\max_{a' \in \mathcal{A}_t} Q_{\boldsymbol{\theta}}(s_t^i, a')$                ▷ Get actions from states
            $r_t^i, s_{t+1}^i = \texttt{CLStep}(t, a_t^i, E_i)$
            $\mathcal{B} = \mathcal{B} \cup \{(s_t^i, a_t^i, r_t^i, s_{t+1}^i)\}$               ▷ Store transition in buffer
            $(s_j, a_j, r_j, s_{j+1}) \overset{B}{\sim} \mathcal{B}$        ▷ Get mini-batch of $B$ samples from buffer
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} (r_t + \max_{a'} Q_{\boldsymbol{\theta}^-}(s_{j+1}, a') - Q_{\boldsymbol{\theta}}(s_j, a_j))$ ▷ Update DQN
        end for
    end for
end for
return $\boldsymbol{\theta}$                                                  ▷ Return DQN

function CLSTEP($t$, $a_t$, $E$)
    $\mathcal{M}_t \leftarrow \texttt{SampleMemory}(a_t)$                 ▷ Sample historical data from action
    $f_{\boldsymbol{\phi}}^E \leftarrow \texttt{Train}(f_{\boldsymbol{\phi}}^E, \mathcal{D}_{t+1}^{\text{train}}, \mathcal{M}_t)$   ▷ Train network on current task with replay
    $r_t \leftarrow \frac{1}{t+1} \sum_{i=1}^{t+1} A_i^{\text{val}}$                ▷ Compute reward from validation datasets
    $s_{t+1} \leftarrow \texttt{GetState}(f_{\boldsymbol{\phi}}^E, \mathcal{D}_{1:t+1}^{\text{val}})$ ▷ Get next state from validation performance
    return $r_t$, $s_{t+1}$                                 ▷ Return reward and next state
end function
```

gradient step to minimize the loss

$$\mathcal{L}_{\text{DQN}}(\boldsymbol{\theta}) = r_j + \max_{a'} Q_{\boldsymbol{\theta}^-}(s_{j+1}, a') - Q_{\boldsymbol{\theta}}(s_j, a_j), \tag{4.3}$$

where $\boldsymbol{\theta}^-$ is target network which is copy of the parameters $\boldsymbol{\theta}$ from some previous time step. By using the shared replay buffer, the DQN is trained on diverse set of environments that can generalize well to new environments with unseen dynamics from an unseen task order or a new dataset. Next, we will describe the experiments we perform to evaluate the generalization capability of the policy to new CL scenarios.

4.4 Experiments

In this section, we give an overview of the experimental results in Paper C and A. In Paper C, we perform experiments to show that the importance of replay scheduling in our proposed problem setting for CL described in Section 4.2. We demonstrate that learning when to replay different tasks by using MCTS as an exemplar method for finding good replay schedules that are efficient in mitigating catastrophic forgetting in the classifier. In Paper A, we demonstrate on benchmark datasets for CL how our RL-based framework can be used for learning replay scheduling policies. We show that our proposed method can learn policies that generalize to CL scenarios with new task orders and datasets unseen during training.

Datasets and Architectures. In Paper C and A, we perform experiments on common CL benchmark datasets with 5 tasks such as Split MNIST [165], Split Fashion-MNIST [205], Split notMNIST [206], and Split CIFAR10 [108], 10-task Permuted MNIST [207], and 20-task Split CIFAR-100 [108, 204, 208] and Split miniImagenet [209]. We use a 2-layer MLP with 256 hidden units and ReLU activation for Split MNIST, Split FashionMNIST, Split notMNIST, and Permuted MNIST. We use a multi-head output layer for each dataset except Permuted MNIST where the network uses single-head output layer. For Split CIFAR-10 and CIFAR-100, we use a multi-head CNN architecture built according to the CNN in [169, 209, 210]. For Split mniImagenet, we use the reduced ResNet-18 from [204] with multi-head output layer. See each paper for training details and hyperparameter settings.

Baselines. We compare our methods with several scheduling baselines to verify the importance of learning replay schedules. The baselines we compare against are

- **Random Schedule**: Randomly select which tasks to replay.
- **Equal Task Schedule (ETS)**: Replay all seen tasks equally.
- **Heuristic Schedule**: Replay tasks which validation performance is below a tuned threshold found through hyperparameter searches.

Evaluation. The evaluation metric we use for measuring the CL performance is defined as

$$\text{ACC} = \frac{1}{T} \sum_{j=1}^{T} A_{T,j}^{\text{test}}, \tag{4.4}$$

where $A_{T,j}^{\text{test}}$ is the accuracy of task j after learning the final task T. In Paper A, we use a ranking method for comparing the ACC performance of each scheduling method within the testing environments, since the performance between environments can differ significantly when the task order changes. We then average all obtained ranking lists over the number of testing environments to obtain the ranking metric.

Replay Scheduling Using Monte Carlo Tree Search

In this section, we provide an overview of the experimental results from Paper C. We have named our approach as Replay Scheduling MCTS (RS-MCTS) below.

Firstly, we evaluate the CL performance of RS-MCTS and the baselines on experiments with varying memory sizes. Secondly, we add replay scheduling to three recent CL methods to show that scheduling is applicable to any memory-based method. Finally, we demonstrate the efficiency benefits of replay scheduling in settings when the memory size is smaller than the number of classes. See Paper C for the full experimental results and settings.

Varying Memory Size. We show that the replay schedules found by MCTS improves the CL performance across different memory sizes. In Figure 5, we observe that RS-MCTS obtains better task accuracies than ETS, especially for small memory sizes. Both RS-MCTS and ETS perform better than Heuristic as M increases showing that Heuristic requires careful tuning of the validation accuracy threshold. Finally, these results show that replay scheduling can outperform simpler scheduling methods on both small and large datasets.

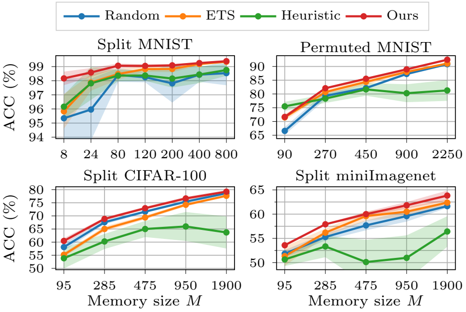


Figure 4.1: Performance comparisons with ACC over different replay memory sizes M for RS-MCTS (Ours) and the baselines. All results have been averaged over 5 seeds.

Applying Scheduling to Recent Replay Methods. We show that replay scheduling can be combined with any memory-based CL method to enhance the performance. We combine replay scheduling with three recent memory-based CL methods called Hindsight Anchor Learning (HAL) [211], Meta-Experience Replay (MER) [212], Dark Experience Replay (DER) [213]. Table 2 shows the performance comparison between our RS-MCTS against using Random, ETS, and Heuristic schedules for each method. Especially for HAL and DER, the schedule found by RS-MCTS mostly outperforms the other schedules across the different datasets. For MER, the Heuristic baseline performs better than RS-MCTS and Permuted-MNIST and Split miniImagenet which could potentially be adjusted with more hyperparameter searches. Furthermore, the Heuristic baseline occasionally did not converge for some seeds which could indicate that this scheduling method is more sensitive to the settings than the other scheduling methods. Finally, the results in this experiment confirm that replay scheduling is important for the final performance given the same memory constraints and it can benefit any existing CL framework.

Efficiency of Replay Scheduling. We illustrate the efficiency of replay scheduling with comparisons to several common replay-based CL baselines in an even more extreme memory setting. Our goal is to investigate if scheduling over which tasks to replay can be more efficient in situations where the memory size is even smaller than the number of classes. To this end, we set the replay memory size

Table 4.1: Performance comparison with ACC between scheduling methods RS-MCTS (Ours), Random, ETS, and Heuristic combined with replay-based methods HAL, MER, and DER. We use 'S' and 'P' as short for 'Split' and 'Permuted' for the datasets. Replay memory sizes are $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation averaged over 5 seeds. Results on Heuristic where some seed did not converge is denoted by *. Applying RS-MCTS to each method can enhance the performance compared to using the baseline schedules.

		5-task Datasets			**10- and 20-task Datasets**		
Method	**Schedule**	S-MNIST	S-FashionMNIST	S-notMNIST	P-MNIST	S-CIFAR-100	S-miniImagenet
HAL	Random	97.24 (± 0.70)	86.74 (± 6.05)	93.61 (± 1.31)	88.49 (± 0.99)	36.09 (± 1.77)	38.51 (± 2.22)
	ETS	94.02 (± 4.25)	95.81 (± 3.53)	91.01 (± 1.39)	88.46 (± 0.86)	34.90 (± 2.02)	38.13 (± 1.18)
	Heuristic	97.69 (± 0.19)	*74.16 (± 11.19)	93.64 (± 0.93)	*66.63 (± 28.50)	35.07 (± 1.29)	39.51 (± 1.49)
	Ours	**97.93** (± 0.56)	**98.27** (± 0.17)	**94.64** (± 0.39)	**89.14** (± 0.74)	**40.22** (± 1.57)	**41.39** (± 1.15)
MER	Random	93.07 (± 0.81)	85.53 (± 3.30)	91.13 (± 0.86)	75.90 (± 1.34)	42.96 (± 1.70)	31.48 (± 1.65)
	ETS	92.89 (± 3.53)	96.47 (± 0.85)	**93.80** (± 0.82)	73.01 (± 0.96)	43.38 (± 1.81)	33.58 (± 1.53)
	Heuristic	94.30 (± 2.79)	96.91 (± 0.62)	90.90 (± 1.30)	**83.86** (± 3.19)	40.90 (± 1.70)	**34.22** (± 1.93)
	Ours	**98.20** (± 0.16)	**98.48** (± 0.26)	93.61 (± 0.71)	79.72 (± 0.71)	**44.29** (± 0.69)	32.74 (± 1.29)
DER	Random	98.23 (± 0.53)	96.56 (± 1.79)	92.89 (± 0.86)	87.51 (± 1.10)	56.83 (± 0.76)	42.19 (± 0.67)
	ETS	98.17 (± 0.35)	97.69 (± 0.58)	94.74 (± 1.05)	85.71 (± 0.75)	52.58 (± 1.49)	35.50 (± 2.84)
	Heuristic	94.57 (± 1.71)	*72.49 (± 19.32)	*77.88 (± 12.58)	81.56 (± 2.28)	55.75 (± 1.08)	**43.62** (± 0.88)
	Ours	**99.02** (± 0.10)	**98.33** (± 0.51)	**95.02** (± 0.33)	**90.11** (± 0.18)	**58.99** (± 0.98)	43.46 (± 0.95)

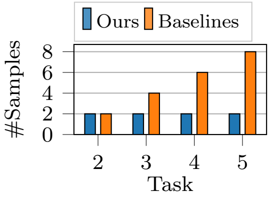


Figure 4.2: Number of replayed samples per task for the 5-task datasets for RS-MCTS (Ours) and the baselines in the tiny memory setting.

	5-task Datasets		
Method	S-MNIST	S-FashionMNIST	S-notMNIST
Random	92.56 (± 2.90)	92.70 (± 3.78)	89.53 (± 3.96)
A-GEM	94.97 (± 1.50)	94.81 (± 0.86)	92.27 (± 1.16)
ER-Ring	94.94 (± 1.56)	95.83 (± 2.15)	91.10 (± 1.89)
Uniform	95.77 (± 1.12)	97.12 (± 1.57)	92.14 (± 1.45)
Ours	96.07 (± 1.60)	97.17 (± 0.78)	93.41 (± 1.11)

Table 4.2: Performance comparison with ACC averaged over 5 seeds in the 1 ex/class memory setting. RS-MCTS (Ours) has replay memory size $M = 2$, while the baselines replay all available memory samples.

for our method to $M = 2$ for the 5-task datasets. We then compare against the most memory efficient CL baselines, namely A-GEM [214], ER-Ring [46] which show promising results with 1 sample per class for replay, as well as with uniform memory selection as reference. Additionally, we compare to using random replay schedules (Random) with the same memory setting as for RS-MCTS. We visualize the memory usage for our method and the baselines when training on a 5-task dataset in Figure 6. In Paper C, we provide results for the same experiments with the larger datasets Permuted MNIST, Split CIFAR-100, and Split miniImagenet.

Table 4.2 shows the ACC for each method across the 5-task datasets. Despite using significantly fewer samples for replay, RS-MCTS performs better or on par with the best baselines on each dataset. These results show that replay scheduling can be even more efficient than the common memory efficient replay-based methods, which indicate that learning the time to learn is an important research direction in CL.

Policy Generalization to New Continual Learning Scenarios

In Paper A, we perform two experiments to evaluate how well the learned replay scheduling policy can generalize to new CL scenarios. First, we investigate whether the policy can be applied to environments with new dataset task splits which are different from the splits that the policy was trained on. Secondly, we study if the policy can be applied to environments with new datasets than it was originally trained on. Finally, we visualize the learned policy and compare it against one of the heuristic baselines to bring further insights into the advantages of learning the replay scheduling policy.

We run experiments on CL datasets of $T = 5$ tasks to have a discrete action space of reasonable size. The replay memory size is set to $M = 10$ in all experiments. We use 15% of the training set for validation for the DQN and the Heuristic baselines. The Random and ETS baselines use the validation set for training since these policies are fixed. We use three different heuristic rules for scheduling, namely

- **Heuristic Global Drop (Heur-GD)**: Replay task when the validation accuracy has dropped below threshold of the highest achieved validation accuracy across all previous time steps.
- **Heuristic Local Drop (Heur-LD)**: Replay task when the validation accuracy has dropped below threshold of the achieved validation accuracy at the previous time step.
- **Heuristic Accuracy Threshold (Heur-AT)**: Replay task when the validation accuracy has dropped below threshold of the minimum allowed validation task accuracy.

The thresholds are found from grid searches and we provide the used values in Paper A.

Generalization to New Task Orders. In this experiment, we evaluate the performance of each scheduling method when generalizing to environments of Split MNIST with new task orders unseen during training. The DQN uses 30 environments for training and we average its results over 3 seeds for parameter initialization. Table 4.3 shows the average ranking over 10 test environments for all methods. We observe that the DQN achieves the best average ranking compared to the baselines. These results indicate that learning the replay scheduling policy is more beneficial than creating heuristic scheduling rules and fixed policies such as ETS.

Table 4.3: Average rankings for generalizing to Split MNIST environments with new task orders.

Method	Avg. Rank
Random	3.77
ETS	3.27
Heur-GD	4.47
Heur-LD	3.27
Heur-AT	3.37
DQN (Ours)	**2.87**

Table 4.4: Average rankings for generalizing to Split notMNIST environments from training on Split MNIST and FashionMNIST.

Method	**Avg. Rank**
Random	3.27
ETS	3.60
Heur-GD	4.53
Heur-LD	4.10
Heur-AT	2.80
DQN (Ours)	**2.70**

Generalization to New Datasets. Here, we evaluate the performance of each scheduling method when generalizing to environments of Split notMNIST datasets while the methods have only been trained on environments of Split MNIST and Split FashionMNIST datasets. The DQN uses 30 environments in total for training (15 Split MNIST + 15 Split FashionMNIST) and we average its results over 3 seeds for parameter initialization. Table 4.4 shows the average ranking over 10 Split notMNIST environments for all methods. We observe again that the DQN achieves the best average ranking compared to the baselines, where Heur-AT performs almost on par with the DQN. These results further shows that the it is indeed possible to learn general scheduling policies that are capable of performing in new environments without additional computational time and training.

Visualization of Learned Policy. We visualize the learned policies for the DQN to bring further insights into the advantages of learning the replay scheduling policy. We analyze the task accuracy progress and the replay schedule used for mitigating catastrophic forgetting by visualizing the schedule as a bubble plot. In the bubble plots shown in Figure 4.3, the x- and y-axis shows the current and replay task respectively, and the bubbles represent the proportion of the replayed task in the replay memory at the current task. Each color of the circles corresponds to a seen task that is being replayed at the current task. The sum of all points in the circles at a current task column is fixed since the replay memory size is constant.

Figure 4.3 shows the task accuracy progression and replay schedules from the DQN and Heur-LD in one test environment with Split MNIST. Using the replay schedule from the DQN gives higher ACC since the DQN manages to mitigate forgetting tasks 1 and 2 better than Heur-LD. The DQN decides to replay task 1 and task 2 as soon as these tasks have been learned. At task 4, the DQN decides to replay both tasks 1 and 2 which helps the classifier to retain its knowledge of task 2. However, as the classifier forgets task 1, the DQN selects to only replay task 1 at task 5 which makes the classifier improve its performance on task 1. The heuristic baseline Heur-LD enables replay on tasks 3 and 5 according to its rule since the performance of task 1 and 2 has decreased below its threshold at those time steps, but the classifier forgets these tasks more than when using the learned replay schedule from the DQN. This example shows creating a good heuristic for replay scheduling can be difficult which is why learning such a policy is preferable.

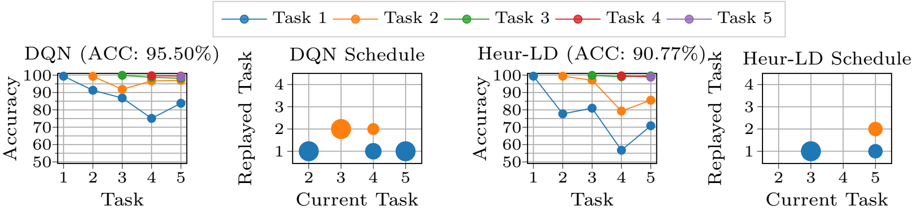


Figure 4.3: Comparison of test classification accuracies for tasks 1-5 on Split MNIST from using the replay schedules from the DQN and the Heuristic Local Drop (Heur-LD) scheduling baseline from 1 seed. We visualize the corresponding replay schedule used by both methods as a bubble plots. The learned policy (DQN) remembers tasks 1 and 2 better than when replaying based on this heuristic.

4.5 Discussion

In this chapter, we have focused on a new problem setting in CL where historical data is cheap to store but processing time is still limited. In Paper C, we present this new setting and demonstrate the importance of scheduling which tasks to replay at different times with MCTS. We show that using a good replay schedule can mitigate catastrophic forgetting significantly across several small and large datasets as well as different backbone choices which indicates that learning the time to learn old tasks again will be important in settings where the historical data storage is huge. As MCTS requires multiple trials to find a good scheduling policy, we are disallowed to use MCTS in a real CL setting as this would violate the premise that tasks can never be fully revisited in CL. Therefore, in Paper A, we propose a framework based on RL for learning a general replay scheduling policy that can be trained on several CL environments and then transferred to new CL scenarios. We used DQN as the RL agent and showed that the learned policies managed to generalize to new task orders and a new dataset without additional computational cost. The next step here is to apply RL-methods that scale to datasets with longer task horizons and large action spaces.

These results bring the applicability of replay scheduling closer to applying this idea to real-world CL scenarios. For example, the policy could be used for mitigating catastrophic forgetting in assistive vision devices that are personalized to the specific environments of users where new objects need to recognized regularly. Assume that we have access to data collected by several users that have applied their assistive device to learn how to recognize different objects. The stored data could then be used to learn a replay scheduling policy that can be deployed in assistive devices of new users for learning personalized classifiers without additional computational cost in potentially unseen environments. Another example when replay scheduling would be useful is in scenarios where the replay memory size is much smaller than the total number of seen classes. For instance, consider a machine learning system operated by a large company, such as Facebook or Amazon, that regularly receives examples of new object classes to be classified in addition to the old ones, where the incoming data is stored rather than deleted. However, as the system cannot reprocess all stored data when it has to learn new concepts, CL is needed to circumvent bottlenecks on processing and transmission times since the number of samples to use for replay is much smaller than the number of stored classes. Then, it becomes crucial for the system to have a policy for scheduling which classes to replay to mitigate catastrophic forgetting efficiently.

There exist limitations with our proposed framework for learning policies for replay scheduling. Firstly, using RL potentially requires lots of training data for learning a policy that generalizes. Acquiring state transitions in the CL environment is more expensive compared to common benchmark environments in RL [215–217] since a step involves learning a task with the classifier given some training protocol. Moreover, RL is well-known for needing many learning episodes before obtaining successful policies as well as time for hyperparameter tuning. There are also other algorithms that should be studied for replay scheduling, such as policy gradient methods [77,78], that can be applied to continuous action spaces for making the framework applicable to longer task horizons. In the future, it will be important to focus on how to enhance the sample-efficiency as well as scaling up replay scheduling policies to datasets with larger number of tasks to make our framework more useful in real-world CL scenarios.

From a CL perspective, there are indeed privacy concerns that needs to be fulfilled as we are currently training the classifiers on raw image data. A benefit with our framework here is that the CL environment utilizes the data in an isolated way and transmit the achieved performance to the policy learning module rather than raw data, as the policy only uses CL metrics to represent the states. Furthermore, the classifier can also be trained using compressed features as has been done in previous works on replay-based CL [30, 179]. Another important direction is to study how the replay scheduling approach behaves in so called *class incremental learning* scenarios where task labels are unknown [218]. This setting is considered as the most challenging in CL as the classifier must both infer the task and the class of the input data making the classifier more prone to catastrophic forgetting. Replay scheduling can indeed be applied in this setting to mitigate the forgetting effect by always replaying at least one example of all seen classes and then letting the scheduler decide how to compose the rest of the memory. However, this setting requires that the replay memory size can at least store one example from all classes, which may not be the case in real-world scenarios. This is why further investigations of replay scheduling approaches needs to be made to reduce the gap between CL research and real-world applications.

Chapter 5

Conclusions and Future Directions

In this chapter, we summarize and provide conclusions to each included paper in the thesis as well as discuss future work (Section 5.1). Finally, we discuss potential long-term future directions for this project that could be interesting for future researchers to dive into for developing computer vision methods for assisting the visually impaired (Section 5.2).

5.1 Conclusions

We divide the conclusions into two parts where Paper A and B are discussed in the first part on fine-grained classification followed by the the second part on continual learning where the conclusions for Paper C and A are discussed.

Fine-Grained Image Classification of Groceries

In Paper A, we presented a challenging fine-grained and hierarchically labeled image classification dataset of grocery items[1]. All images are taken with a mobile phone camera to simulate the scenario of shopping with an assistive vision app for recognizing groceries. Therefore, the images includes natural variations common in grocery stores such as misplaced items, varying lighting conditions and backgrounds, as well as images with multiple instances of the target class and items from different classes. In addition to the natural images, we also collect external information by web-scraping a supermarket website for an iconic image and a text description of each grocery class. There are several lessons learned from collecting this dataset. First of all, recording videos would have enhanced the number of images in the dataset and potentially eased the collection process. The dataset could then be used for benchmarking on both image and video recognition tasks, where the latter would also be more user-friendly as this increases the chance of capturing frames containing the item. Furthermore, we could have provided

[1] Publicly available at `https://github.com/marcusklasson/GroceryStoreDataset`.

more annotations for the items in the natural images. For example, we could have placed bounding boxes over the location of items of the target class as well as provided labels for if the whole item is captured in the image, and whether there are multiple instances of the target class or if there are several different classes present. Moreover, collecting multiple instances of the web-scraped information could be beneficial to increase the chances of extracting the fine-grained details of the grocery items. Finally, it would have been valuable to have blind/low-vision collectors of the images to reduce the gap between the collected data and how the data is later encountered in application use [219].

In Paper B, we showed that the web-scraped iconic images and text descriptions in the dataset from Paper A are useful for enhancing the fine-grained classification performance. We employ the multi-view autoencoder VCCA [53] for learning joint representations between the data views to capture the fine-grained details of the groceries items better by utilizing the web-scraped views. We perform an ablation study over the available data views and find that VCCA structures the latent space differently depending on which views that are used in the model. More specifically, the iconic images structures the items based on visual similarities, such as color and shape, in the latent space, while the text descriptions structures the items based on ingredients and flavors. Using the more web-scraped views yields better fine-grained classification performance compared to approaches that only use the natural images for training the classifier. Hence, the more easily available views can potentially reduce the need for large amounts of natural images to obtain reasonable classification performance of groceries in assistive vision devices. One limitation of VCCA is its inability of learning the view-specific variations when learning from single instances of the web-scraped views, which would help the model to separate the shared information across the utilized views from the private information for each view. A different direction than using autoencoder-based approaches could be to study attention-based methods [155, 220, 221] in combination with the natural images and the web-scraped images and text. Furthermore, it would also be interesting to test VCCA on images of the items in different environments than the gropcery stores to check its robustness. Finally, we believe that the dataset would be suitable for zero/few-shot learning and continual learning settings where the model should adapt fast to new classes from potentially few available samples, which would be interesting to investigate for testing the VCCA model in other realistic scenarios.

Replay Scheduling in Continual Learning

In Paper C and A, we focused on a slightly new setting for continual learning (CL) with the aim to bring CL research closer to real-world needs. The main contrast to the traditional CL setting is that historical data is stored rather than deleted which stems well with how major companies handle their incoming data [222,223]. However, the model has a budget on how many historical samples that can be used for mitigating catastrophic forgetting when learning from new data due to constraints on the processing and data transmission times. We present this setting in Paper C and propose scheduling over which tasks to replay at different times to be capable of selecting subsets of historical data from the huge storage. To demonstrate the replay scheduling procedure, we construct a discrete action space of options on how to compose a replay memory with different proportions of the seen tasks. We then select MCTS [201, 202] to enable searching for replay schedules that efficiently reduce catastrophic forgetting for a CL model in large action spaces. The experiments showed that selecting a good replay schedule over the tasks to remember yields better CL performance compared to replaying all tasks equally and could also outperform a heuristic scheduling rule. Furthermore, we showed that recent replay methods [211–213] also benefits from the replay schedules selected by MCTS. Finally, we illustrated by letting MCTS select fewer samples for replay than the number of seen classes that replay scheduling can be as efficient as always replaying 1 sample per class. We emphasize that MCTS was used as an exemplar method for demonstrating the importance of learning the memory scheduling in our new CL setting, since MCTS requires multiple rollouts in the action space which is prohibited in CL. Hence, we need a method that learns a general policy for replay scheduling that can be applied across CL domains, which we address in Paper A.

To the best of our knowledge, there is a lack of efficient methods for applying replay scheduling policies in real-world scenarios. To this end, in Paper A, we propose a framework based on reinforcement learning [75] (RL) for learning policies that can mitigate catastrophic forgetting in new CL scenarios. Our framework learns a single policy for selecting which tasks to replay for a CL classifier based on how well the classifier has performed on previous tasks. To foster generalization to new CL scenarios, such as new task orders and datasets, we train the policy with experiences from several CL environments where the policy schedules which tasks to be replayed at different times. We show through experiments on common CL benchmark datasets that the learned policies with our framework can be applied to new CL scenarios and successfully mitigate catastrophic forgetting in the classifiers without any additional computational cost. As we employed Deep Q-Networks [76], we suggest for future work to test more RL algorithms, such as policy-gradient methods [77, 78], for learning the replay scheduling policy. This also opens up for learning policies with continuous action spaces that could ease the scalability issues from using discrete action spaces in CL datasets with long task horizons. Scaling up the policy to large number of tasks opens up for utilizing replay scheduling approaches in real-world scenarios where a classifier must adapt fast to recognize new objects of interest. As an example, assistive vision devices that are user-personalized could make use of replay scheduling policies learned from other users for mitigating catastrophic forgetting when learning new tasks. However, training the policy to generalize to unseen domains currently requires lots of training from CL environments where each agent-environment query is expensive. Furthermore, RL approaches typically need highly diverse training data for the policy to generalize [193, 198] as well as hyperparameter tuning that can be costly. Hence, it will be important to enhance the sample-efficiency and adaptation ability of the policy in the future.

5.2 Future Directions

In this section, we discuss two future directions that we believe are important to advance assistive vision technologies. First, we suggest future researchers in machine learning and computer vision to work with federated learning [224] applied to assistive vision devices to enhance the robustness in their image recognition performance. Secondly, we highlight the importance of involving visually impaired (VI) people throughout research processes to enhance the usefulness of the next generation of assistive vision technologies.

Federated Learning

In recent years, there has been a growing interest for federated learning [224–226] in the machine learning community. Federated learning involves leveraging the storage and computational capabilities of remote devices, such as mobile phones, for training local models while keeping the data on the device to preserve privacy.

We give an example scenario in the context of assistive vision where Alice and Bob are grocery shopping in a supermarket using an assistive vision app installed on their personal smartphones. Alice needs to buy milk and wants to recognize a milk package held in her hand. She capture several frames of the milk package by recording a video with the camera that is processed in the app. However, the app responds that this milk package is an unknown item since this type of milk brand is new in the supermarket. Alice asks Charles the clerk what item she is holding, gets help with updating her app with the new milk class from Charles, and purchases the milk. The next day, Bob does his grocery shopping in the same supermarket as Alice. Bob is curious about a new milk package he has never seen before, which is of the same class as the milk Alice bought the day before. Charles app can successfully recognize the correct class of the milk package although this is the first time Bob takes photos of this milk with the app. This is enabled through a central server communicating with all mobile phones with the assistive app installed for learning a global model from local updates from the connected devices in the network. Thus, after the local update of the new milk package in Alice's app has been incorporated, the server transmits the new global model to Bob's app. This is the standard procedure for federated learning which has the potential to utilize the computational resources for predictions on smartphones while maintaining the user experience and preserving privacy.

There are many interesting challenges that remains to tackle in federated learning. Li et al. [224] mentions challenges with statistical heterogeneity in the data from different users [224, 227, 228], variability of the systems on each device [229], privacy concerns [230, 231], and expensive communication rounds between the server and the devices [232]. From the perspective of this thesis, it would be valuable to study the statistical heterogeneity as the generated data can vary immensely across different users in terms of sample frequency, data quality, and context of where the data has been recorded. Here, it may be important to consider issues regarding fairness in addition to accuracy, since the learned model may become biased towards devices with large amounts of data or to commonly occurring groups of devices. Another relevant extension to this thesis would be on systems heterogeneity among different devices, since it will be important to push research and development towards image recognition methods that are accurate, robust, and performs in real-time regardless of the hardware, network connectivity, and power of the mobile devices. This could enable assistive vision apps to be less reliant on the mobile phone type, which in return may yield assistive vision apps that work equally well across different geographical locations and income levels.

Disability-First Approach in Development of Assistive Vision Technologies

This section is for encouraging future researchers in computer vision and machine learning to have their target user in mind for projects motivated by real-world applications such as assistive technologies. There exist many studies on computer vision-based assistive technologies describing the benefits and challenges to ease everyday life for VI people [18, 233–237]. The past years, there has been an increasing interest of deploying computer vision and machine learning models on mobile devices [238, 239]. Additionally, it is well-known that VI people use mobile phones, including the camera to record events, and want help with object recognition and navigation [18, 136, 240, 241]. Even if the combination is evidently a good fit, many of the mentioned works stress that researchers should keep VI people involved throughout the process to ensure that the developed technology is requested and, in fact, useful for blind/low-vision people. An example of close collaboration between the two communities is the collection of the recent ORBIT dataset [110, 219] where the data collection was performed by blind/low-vision participants. They take on a *disability-first approach* in the process to produce good quality dataset that should primarily serve the VI community by continuously engaging with and supporting the participants to balance the usefulness versus the effort of data to be collected [219]. Involving VI users in the research process should provide benefits on for both the VI users as well as the vision community by bringing insights on how to build more robust image recognizers.

Aspects on usefulness, fairness, and privacy needs to be considered when developing assistive vision technologies. Regarding usefulness, it will be important to engage with VI people and people from technical disciplines, such as human-computer interaction and user experience designers, to understand what VI people want their app to assist them with and how to build functionality to achieve those goals. For computer vision researchers, we have seen that the models suffer from biases that can disadvantage users based on their gender [242–244], geographical location [245], and co-occurrences between objects and surrounding contexts [246]. Such challenges on fairness [247–250] and various types of distribution shifts [251, 252] will be crucial to tackle for establishing trust in image recognition systems for practitioners. Furthermore, we should strive for developing privacy-preserving systems to avoid unintended leakages of user-sensitive information [31,32,253,254]. However, we also need to take into account the point-of-view of bystanders and whether they are comfortable with being photographed by mobile and wearable devices. This aspect has been studied recently [255–257], however, it is still non-trivial how to approach users and ask them to stop using their device. This matter of privacy and social acceptance between assistive vision users and surrounding people needs to be further studied in realistic environments for the purpose of understanding the needs of those who will be involved in the usage of the technology.

Bibliography

[1] Alexander Eitel, Katharina Scheiter, Anne Schüler, Marcus Nyström, and Kenneth Holmqvist. How a picture facilitates the process of learning from text: Evidence for scaffolding. *Learning and Instruction*, 28:48–63, 2013.

[2] Anne Nielsen Hibbing and Joan L. Rankin-Erickson. A picture is worth a thousand words: Using visual images to improve comprehension for middle school struggling readers. *The Reading Teacher*, 56, 2003.

[3] Rupert Bourne, Jaimie D Steinmetz, Seth Flaxman, Paul Svitil Briant, Hugh R Taylor, Serge Resnikoff, Robert James Casson, Amir Abdoli, Eman Abu-Gharbieh, Ashkan Afshin, et al. Trends in prevalence of blindness and distance and near vision impairment over 30 years: an analysis for the global burden of disease study. *The Lancet global health*, 9(2):e130–e143, 2021.

[4] Dragan Ahmetovic, Daisuke Sato, Uran Oh, Tatsuya Ishihara, Kris Kitani, and Chieko Asakawa. Recog: Supporting blind people in recognizing personal objects. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[5] Rabia Jafri, Syed Abid Ali, Hamid R Arabnia, and Shameem Fatima. Computer vision-based object recognition for the visually impaired in an indoors environment: a survey. *The Visual Computer*, 30(11):1197–1222, 2014.

[6] Hernisa Kacorri. Teachable machines for accessibility. *ACM SIGACCESS Accessibility and Computing*, (119):10–18, 2017.

[7] James Coughlan and Roberto Manduchi. Functional assessment of a camera phone-based wayfinding system operated by blind and visually impaired users. *International Journal on Artificial Intelligence Tools*, 18(03):379–397, 2009.

[8] Hernisa Kacorri, Eshed Ohn-Bar, Kris M Kitani, and Chieko Asakawa. Environmental factors in indoor navigation based on real-world trajectories of blind users. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[9] Jack M Loomis, Reginald G Golledge, Roberta L Klatzky, and James R Marston. Assisting wayfinding in visually impaired travelers. In *Applied Spatial Cognition*, pages 179–202. Psychology Press, 2020.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[12] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[13] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[14] World Health Organization. International Classification of Diseases 11th Revision (ICD-11), 2022. URL https://icd.who.int/en. Accessed 2022-03-22.

[15] World Health Organization. *World report on vision*. World Health Organization, 2019.

[16] Jaimie D Steinmetz, Rupert RA Bourne, Paul Svitil Briant, Seth R Flaxman, Hugh RB Taylor, Jost B Jonas, Amir Aberhe Abdoli, Woldu Aberhe Abrha, Ahmed Abualhasan, Eman Girum Abu-Gharbieh, et al. Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to vision 2020: the right to sight: an analysis for the global burden of disease study. *The Lancet Global Health*, 9(2):e144–e160, 2021.

[17] Roberto Manduchi and James Coughlan. (computer) vision without sight. *Communications of the ACM*, 55(1):96–104, 2012.

[18] Chandrika Jayant, Hanjie Ji, Samuel White, and Jeffrey P Bigham. Supporting blind photography. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, pages 203–210, 2011.

[19] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White, and Jeffrey P Bigham. Visual challenges in the everyday lives of blind people. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2117–2126, 2013.

[20] Microsoft Corporation. Seeing AI, 2017. URL https://www.microsoft.com/en-us/ai/seeing-ai. Accessed 2022-03-14.

[21] Patrick Clary. Lookout: an app to help blind and visually impaired people learn about their surroundings, 2018. URL https://blog.google/outreach-initiatives/accessibility/lookout-app-help-blind-and-visually-impaired-people-learn-about-their-surrou Accessed 2022-04-05.

[22] Inc Cloudsight. TapTapSee, 2013. URL https://taptapseeapp.com/. Accessed 2022-03-22.

[23] Envision. Envision App, 2018. URL https://www.letsenvision.com/envision-app. Accessed 2022-03-22.

[24] OrCam. OrCam MyEye 2, 2019. URL https://www.orcam.com/sv/myeye2/. Accessed 2022-03-22.

[25] Envision. Envision Glasses, 2020. URL https://www.letsenvision.com/envision-glasses. Accessed 2022-03-22.

[26] Be My Eyes. Be My Eyes, 2017. URL https://www.bemyeyes.com/. Accessed 2022-03-22.

[27] Aira Tech Corp. Aira, 2017. URL https://aira.io/. Accessed 2022-03-14.

[28] Tai-Yin Chiu, Yinan Zhao, and Danna Gurari. Assessing image quality issues for real-world problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3646–3656, 2020.

[29] Hernisa Kacorri, Kris M Kitani, Jeffrey P Bigham, and Chieko Asakawa. People with visual impairment training personal object recognizers: Feasibility and challenges. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5839–5849, 2017.

[30] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. *arXiv preprint arXiv:1912.01100*, 2019.

[31] Tousif Ahmed, Roberto Hoyle, Kay Connelly, David Crandall, and Apu Kapadia. Privacy concerns and behaviors of people with visual impairments. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3523–3532, 2015.

[32] Danna Gurari, Qing Li, Chi Lin, Yinan Zhao, Anhong Guo, Abigale Stangl, and Jeffrey P Bigham. Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 939–948, 2019.

[33] Roberto Hoyle, Robert Templeman, Steven Armes, Denise Anthony, David Crandall, and Apu Kapadia. Privacy behaviors of lifeloggers using wearable cameras. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 571–582, 2014.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[36] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4565–4574, 2016.

[37] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[38] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[39] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[40] Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. Language-conditioned graph networks for relational reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10294–10303, 2019.

[41] Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge Belongie. Fine-grained image analysis with deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[42] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[43] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[44] Patrick E Lanigan, Aaron M Paulos, Andrew W Williams, Dan Rossi, and Priya Narasimhan. Trinetra: Assistive technologies for grocery shopping for the blind. In *2006 10th IEEE International Symposium on Wearable Computers*, pages 147–148. IEEE, 2006.

[45] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[46] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[47] Tyler L Hayes, Giri P Krishnan, Maxim Bazhenov, Hava T Siegelmann, Terrence J Sejnowski, and Christopher Kanan. Replay in deep learning: Current approaches and missing biological elements. *Neural Computation*, 33(11):2908–2950, 2021.

[48] Frank N Dempster. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330, 1989.

[49] Hermann Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

[50] Karri S Hawley, Katie E Cherry, Emily O Boudreaux, and Erin M Jackson. A comparison of adjusted spaced retrieval versus a uniform expanded retrieval schedule for learning a name–face association in older adults with probable alzheimer's disease. *Journal of Clinical and Experimental Neuropsychology*, 30(6):639–649, 2008.

[51] T. Landauer and Robert Bjork. Optimum rehearsal patterns and name learning. *Practical aspects of memory*, 1, 11 1977.

[52] Paul Smolen, Yili Zhang, and John H Byrne. The right time to learn: mechanisms and optimization of spaced learning. *Nature Reviews Neuroscience*, 17(2):77, 2016.

[53] Weiran Wang, Xinchen Yan, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis. *arXiv preprint arXiv:1610.03454*, 2016.

[54] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[56] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[61] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[62] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[63] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[64] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[65] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[66] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[67] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[68] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[69] Carina Silberer and Mirella Lapata. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, 2014.

[70] Michelle A Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *IEEE Transactions on Robotics*, 36(3):582–596, 2020.

[71] Mike Wu and Noah Goodman. Multimodal generative models for scalable weakly-supervised learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[72] Yuge Shi, Brooks Paige, Philip Torr, et al. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

[73] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. *arXiv preprint arXiv:1705.10762*, 2017.

[74] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2016.

[75] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[76] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[77] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[78] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[79] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[80] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[81] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[82] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[83] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

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

[85] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[86] Sebastian Bujwid and Josephine Sullivan. Large-scale zero-shot image classification from rich and diverse textual descriptions. *arXiv preprint arXiv:2103.09669*, 2021.

[87] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[88] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[89] Yushan Feng Saihui Hou and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *IEEE International Conference on Computer Vision*, 2017.

[90] Xiu-Shen Wei, Quan Cui, Lei Yang, Peng Wang, and Lingqiao Liu. Rpc: A large-scale retail product checkout dataset. *arXiv preprint arXiv:1901.07249*, 2019.

[91] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1134–1142, 2016.

[92] Yaming Wang, Vlad I Morariu, and Larry S Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4148–4157, 2018.

[93] Yao Ding, Yanzhao Zhou, Yi Zhu, Qixiang Ye, and Jianbin Jiao. Selective sparse sampling for fine-grained image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6599–6608, 2019.

[94] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[95] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[96] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Neural baby talk. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7219–7228, 2018.

[97] Jyoti Aneja, Harsh Agrawal, Dhruv Batra, and Alexander Schwing. Sequential latent spaces for modeling the intention during diverse image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4261–4270, 2019.

[98] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[99] Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. Factorized orthogonal latent spaces. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 701–708. JMLR Workshop and Conference Proceedings, 2010.

[100] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[101] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255, 2013.

[102] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International conference on machine learning*, pages 1083–1092, 2015.

[103] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[104] Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. Learning factorized multimodal representations. In *International Conference on Learning Representations*, 2019.

[105] Cheng Zhang, Hedvig Kjellström, and Carl Henrik Ek. Inter-battery topic representation learning. In *European Conference on Computer Vision*, pages 210–226. Springer, 2016.

[106] Andreas Damianou, Neil D Lawrence, and Carl Henrik Ek. Multi-view learning as a nonparametric nonlinear inter-battery factor analysis. *Journal of Machine Learning Research*, 22(86):1–51, 2021.

[107] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[108] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[109] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[110] Daniela Massiceti, Luisa Zintgraf, John Bronskill, Lida Theodorou, Matthew Tobias Harris, Edward Cutrell, Cecily Morrison, Katja Hofmann, and Simone Stumpf. Orbit: A real-world few-shot dataset for teachable object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10818–10828, 2021.

[111] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[112] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2017.

[113] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[114] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[115] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[116] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12884–12893, 2021.

[117] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[118] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[119] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[120] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE conference on computer vision and pattern recognition*, pages 951–958. IEEE, 2009.

[121] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[122] Menglin Jia, Mengyun Shi, Mikhail Sirotenko, Yin Cui, Claire Cardie, Bharath Hariharan, Hartwig Adam, and Serge Belongie. Fashionpedia: Ontology, segmentation, and an attribute localization dataset. In *European conference on computer vision*, pages 316–332. Springer, 2020.

[123] Andrea Vedaldi, Siddharth Mahendran, Stavros Tsogkas, Subhransu Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, David Weiss, et al. Understanding objects in detail with fine-grained attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3622–3629, 2014.

[124] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[125] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[126] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3973–3981, 2015.

[127] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[128] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[129] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[130] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[131] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016.

[132] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014.

[133] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot learning of object categories. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (01):1–1, 2013.

[134] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 49–58, 2016.

[135] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[136] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018.

[137] Joan Sosa-García and Francesca Odone. “hands on” visual recognition for visually impaired users. *ACM Transactions on Accessible Computing (TACCESS)*, 10(3):1–30, 2017.

[138] Philipp Jund, Nichola Abdo, Andreas Eitel, and Wolfram Burgard. The freiburg groceries dataset. *arXiv preprint arXiv:1611.05799*, 2016.

[139] Georg Waltner, Michael Schwarz, Stefan Ladstätter, Anna Weber, Patrick Luley, Horst Bischof, Meinrad Lindschinger, Irene Schmid, and Lucas Paletta. Mango - mobile augmented reality with functional eating guidance and food awareness. In *International Workshop on Multimedia Assisted Dietary Management*, 2015.

[140] Marian George and Christian Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. In *European Conference on Computer Vision*, pages 440–455. Springer, 2014.

[141] Michele Merler, Carolina Galleguillos, and Serge Belongie. Recognizing groceries in situ using in vitro training data. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[142] Weidong Geng, Feilin Han, Jiangke Lin, Liuyi Zhu, Jieming Bai, Suzhen Wang, Lin He, Qiang Xiao, and Zhangjiong Lai. Fine-grained grocery product recognition by one-shot learning. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1706–1714, 2018.

[143] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[144] Weiqing Min, Linhu Liu, Zhengdong Luo, and Shuqiang Jiang. Ingredient-guided cascaded multi-attention network for food recognition. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1331–1339, 2019.

[145] Jaclyn Rich, Hamed Haddadi, and Timothy M Hospedales. Towards bottom-up analysis of social food. In *Proceedings of the 6th International Conference on Digital Health Conference*, pages 111–120, 2016.

[146] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[147] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, , Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 2021.

[148] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[149] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[150] Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. *arXiv preprint arXiv:1809.00812*, 2018.

[151] Oscar Beijbom, Neel Joshi, Dan Morris, Scott Saponas, and Siddharth Khullar. Menu-match: Restaurant-specific food logging from images. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 844–851. IEEE, 2015.

[152] Ruihan Xu, Luis Herranz, Shuqiang Jiang, Shuang Wang, Xinhang Song, and Ramesh Jain. Geolocalized modeling for dish recognition. *IEEE transactions on multimedia*, 17(8):1187–1199, 2015.

[153] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[154] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[155] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[157] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[158] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[159] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[160] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[161] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[162] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 541–556, 2017.

[163] Tom M. Mitchell. Machine learning and data mining. *Communications of the ACM*, 42:30–36, 1999.

[164] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[165] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[166] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[167] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[168] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[169] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[170] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

[171] Ta-Chu Kao, Kristopher Jensen, Gido van de Ven, Alberto Bernacchia, and Guillaume Hennequin. Natural continual learning: success is a journey, not (just) a destination. *Advances in Neural Information Processing Systems*, 34, 2021.

[172] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[173] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[174] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019.

[175] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *arXiv preprint arXiv:2003.09553*, 2020.

[176] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[177] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[178] Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in Neural Information Processing Systems*, 34:28889–28903, 2021.

[179] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020.

[180] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

[181] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[182] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020.

[183] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

[184] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[185] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019.

[186] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[187] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[188] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

[189] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[190] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

[191] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

[192] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[193] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[194] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[195] Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. *arXiv preprint arXiv:2106.02940*, 2021.

[196] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pages 941–950. PMLR, 2019.

[197] Ayush Jain, Andrew Szot, and Joseph J Lim. Generalization to new actions in reinforcement learning. *arXiv preprint arXiv:2011.01928*, 2020.

[198] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

[199] Jiachen Yang, Brenden Petersen, Hongyuan Zha, and Daniel Faissol. Single episode policy transfer in reinforcement learning. *arXiv preprint arXiv:1910.07719*, 2019.

[200] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pages 619–629. PMLR, 2021.

[201] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[202] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[203] Muhammad Umar Chaudhry and Jee-Hyong Lee. Feature selection for high dimensional data using monte carlo tree search. *IEEE Access*, 6:76036–76048, 2018.

[204] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

[205] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[206] Yaroslav Bulatov. The notMNIST dataset. http://yaroslavvb.com/upload/notMNIST/, 2011.

[207] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[208] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[209] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[210] Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). *arXiv preprint arXiv:1911.09514*, 2019.

[211] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6993–7001, 2021.

[212] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[213] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.

[214] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[215] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[216] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[217] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[218] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[219] Lida Theodorou, Daniela Massiceti, Luisa Zintgraf, Simone Stumpf, Cecily Morrison, Edward Cutrell, Matthew Tobias Harris, and Katja Hofmann. Disability-first dataset creation: Lessons from constructing a dataset for teachable object recognition with blind and low vision data collectors. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–12, 2021.

[220] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4438–4446, 2017.

[221] Heliang Zheng, Jianlong Fu, Zheng-Jun Zha, and Jiebo Luo. Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5012–5021, 2019.

[222] Nathan Bronson, Thomas Lento, and Janet L Wiener. Open data challenges at facebook. In *2015 IEEE 31st international conference on data engineering*, pages 1516–1519. IEEE, 2015.

[223] Anthony Asta. Observability at twitter: technical overview, part i, 2016. URL `https://blog.twitter.com/engineering/en_us/a/2016/observability-at-twitter-technical-overview-part-i`. Accessed 2022-05-04.

[224] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[225] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[226] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[227] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[228] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[229] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[230] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[231] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[232] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[233] YingLi Tian, Xiaodong Yang, Chucai Yi, and Aries Arditi. Toward a computer vision-based wayfinding aid for blind persons to access unfamiliar indoor environments. *Machine vision and applications*, 24(3):521–535, 2013.

[234] Juan R Terven, Joaquín Salas, and Bogdan Raducanu. New opportunities for computer vision-based assistive technology systems for the visually impaired. *Computer*, 47(4):52–58, 2013.

[235] Marco Leo, G Medioni, M Trivedi, Takeo Kanade, and Giovanni Maria Farinella. Computer vision for assistive technologies. *Computer Vision and Image Understanding*, 154:1–15, 2017.

[236] Ruxandra Tapu, Bogdan Mocanu, and Titus Zaharia. Wearable assistive devices for visually impaired: A state of the art survey. *Pattern Recognition Letters*, 137:37–52, 2020.

[237] Linda Wang and Alexander Wong. Implications of computer vision driven assistive technologies towards individuals with visual impairment. *arXiv preprint arXiv:1905.07844*, 2019.

[238] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287, 2019.

[239] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[240] Marynel Vázquez and Aaron Steinfeld. Helping visually impaired users properly aim a camera. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*, pages 95–102, 2012.

[241] Sarit Szpiro, Yuhang Zhao, and Shiri Azenkot. Finding a store, searching for a product: a study of daily challenges of low vision people. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 61–72, 2016.

[242] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

[243] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[244] Kaylee Burns, Lisa Anne Hendricks, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. *arXiv preprint arXiv:1803.09797*, 2018.

[245] Terrance De Vries, Ishan Misra, Changhan Wang, and Laurens Van der Maaten. Does object recognition work for everyone? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 52–59, 2019.

[246] Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. Don't judge an object by its context: learning to overcome contextual bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11070–11078, 2020.

[247] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[248] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudik, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–16, 2019.

[249] Kaiyu Yang, Klint Qinami, Li Fei-Fei, Jia Deng, and Olga Russakovsky. Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the imagenet hierarchy. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 547–558, 2020.

[250] Priya Goyal, Adriana Romero Soriano, Caner Hazirbas, Levent Sagun, and Nicolas Usunier. Fairness indicators for systematic assessments of visual feature extractors. *arXiv preprint arXiv:2202.07603*, 2022.

[251] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. Dataset shift in machine learning, 2009.

[252] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Taylan Cemgil, et al. A fine-grained analysis on distribution shift. *arXiv preprint arXiv:2110.11328*, 2021.

[253] Tousif Ahmed, Roberto Hoyle, Patrick Shaffer, Kay Connelly, David Crandall, and Apu Kapadia. Understanding the physical safety, security, and privacy concerns of people with visual impairments. *IEEE Internet Computing*, 21(3):56–63, 2017.

[254] Taslima Akter, Bryan Dosono, Tousif Ahmed, Apu Kapadia, and Bryan Semaan. " i am uncomfortable sharing what i can't see": Privacy concerns of the visually impaired with camera based assistive applications. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1929–1948, 2020.

[255] Kyungjun Lee, Daisuke Sato, Saki Asakawa, Hernisa Kacorri, and Chieko Asakawa. Pedestrian detection with wearable cameras for the blind: A two-way perspective. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[256] Tousif Ahmed, Apu Kapadia, Venkatesh Potluri, and Manohar Swaminathan. Up to a limit? privacy concerns of bystanders and their willingness to share additional information with visually impaired users of assistive technologies. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–27, 2018.

[257] Taslima Akter, Tousif Ahmed, Apu Kapadia, and Manohar Swaminathan. Shared privacy concerns of the visually impaired and sighted bystanders with camera based assistive technologies. *ACM Trans. Access. Comput.*, 2021.

Part II

Included Papers

Paper A

A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels

Marcus Klasson[*], **Cheng Zhang**[†], **Hedvig Kjellström**[*]

[*]KTH Royal Institute of Technology, Stockholm, Sweden
[†]Microsoft Research, Cambridge, United Kingdom

Abstract

Image classification models built into visual support systems and other assistive devices need to provide accurate predictions about their environment. We focus on an application of assistive technology for people with visual impairments, for daily activities such as shopping or cooking. In this paper, we provide a new benchmark dataset for a challenging task in this application – classification of fruits, vegetables, and refrigerated products, e.g. milk packages and juice cartons, in grocery stores. To enable the learning process to utilize multiple sources of structured information, this dataset not only contains a large volume of natural images but also includes the corresponding information of the product from an online shopping website. Such information encompasses the hierarchical structure of the object classes, as well as an iconic image of each type of object. This dataset can be used to train and evaluate image classification models for helping visually impaired people in natural environments. Additionally, we provide benchmark results evaluated on pretrained convolutional neural networks often used for image understanding purposes, and also a multi-view variational autoencoder, which is capable of utilizing the rich product information in the dataset.

1 Introduction

In this paper, we focus on the application of image recognition models implemented into assistive technologies for people with visual impairments. Such tech-

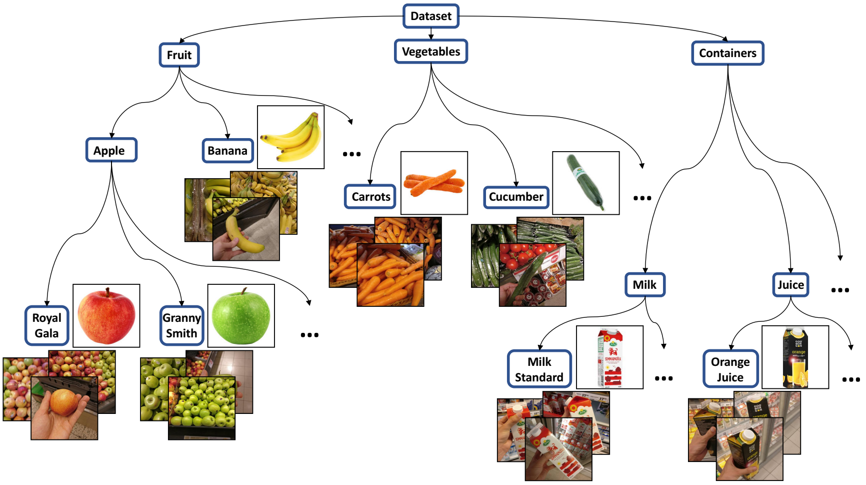


Figure 1: The primary contribution of this paper is a dataset of grocery items, for the purpose of training a visual recognition system to aid visually impaired people. The dataset is organized according to a hierarchical class structure, as illustrated above. A novel aspect of the dataset is that each class, apart from the semantic label, also has a visual label in the form of an iconic image.

nologies already exist in the form of mobile applications, e.g. Microsoft's Seeing AI [1] and Aipoly Vision [2], and as wearable artificial vision devices, e.g. Orcam MyEye [3] and the Sound of Vision system introduced in [4]. These products have the ability to support people with visual impairments in many different situations, such as reading text documents, describing the user's environment and recognizing people the user may know.

We here address a complementary scenario not handled by current systems on the market: visual support when shopping for grocery items considering a large range of eatable objects, including fruits, vegetables, milk, and juices. In the case of fruits and vegetables, these are usually stacked in large bins in grocery stores as shown in Figure 2(a-f). A common problem in grocery stores is that similar items are often stacked next to each other; therefore, items are often misplaced into neighboring bins. Figure 2a shows a mix of red and green apples, where it might be difficult for the system to determine which kind of apple is the actual target. Humans can distinguish between groceries without vision to some degree, e.g. by touching and smelling them, but it requires prior knowledge about texture and fragrance of food items.

Moreover, in addition to raw grocery items, there are also items that can only be differentiated with the help of visual information, e.g. milk, juice, and yogurt cartons, see Figure 2(g-i). Such items usually have barcodes, that are readable using the existing assistive devices described above. However, the barcodes are not easily located by visually impaired persons. Thus, an assistive vision device that fully relies on natural image understanding would be of significant added value for a visually impaired person shopping in a grocery store.

Image recognition models used for this task typically require training images collected in similar environments. However, current benchmark datasets, such as ImageNet [5] and CIFAR-100 [6], do contain images of fruits and vegetables, but are not suitable for this type of assistive application, since the target objects are commonly not presented in this type of natural environments, with occlusion and cluttered backgrounds. To address this issue, we present a novel dataset containing natural images of various raw grocery items and refrigerated products, e.g. milk, juice, and yogurt, taken in grocery stores. As part of our dataset, we collect images taken with single and multiple target objects, from various perspectives, and with noisy backgrounds.

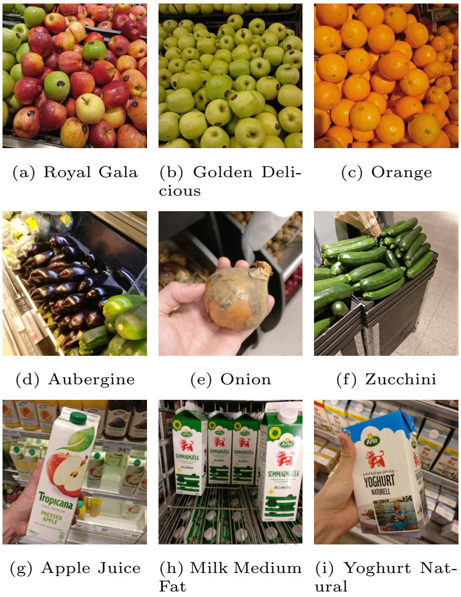
In computer vision, previous studies have shown that model performance can be improved by extending the model to utilize other data sources, e.g. text, audio, in various machine learning tasks [7–10]. Descriptions of images are rather common to computer vision datasets, e.g. Flickr30k [11], whereas the datasets in [8, 12] includes both descriptions and a reference image with clean background to some objects. Therefore, in addition to the natural images, we have collected iconic images with a single object centered in the image (see Figure 3) and a corresponding product description to each grocery item. In this work, we also demonstrate how we can benefit from using additional information about the natural images by applying the multi-view generative model.

To summarize, the contribution of this paper is a dataset of natural images of raw and refrigerated grocery items, which could be used for evaluating and training image recognition systems to assist visually impaired people in a grocery store. The dataset labels have a hierarchical structure with both coarse- and fine-grained classes (see Figure 1). Moreover, each class also has an iconic image and a product description, which makes the dataset applicable to multimodal learning models. The dataset is described in Section 3.

We provide multiple benchmark results using various deep neural networks, such as Alexnet [13], VGG [14], DenseNet [15], as well as deep generative models, such as VAE [16]. Furthermore, we adapt a multi-view VAE model to make use of the iconic images for each class (Section 4), and show that it improves the classification accuracy given the same model setting (Section 5). Last, we discuss possible future directions for fully using the additional information provided with the dataset and adopt more advanced machine learning methods, such as visual-semantic embeddings, to learn efficient representations of the images.

2 Related Work

Many popular image datasets have been collected by downloading images from the web [5, 6, 8, 12, 17–21]. If the dataset contains a large amount of images, it is convenient to make use of crowdsourcing to get annotations for recognition tasks [5, 6, 22]. For some datasets, the crowdsourcers are also asked to put bounding

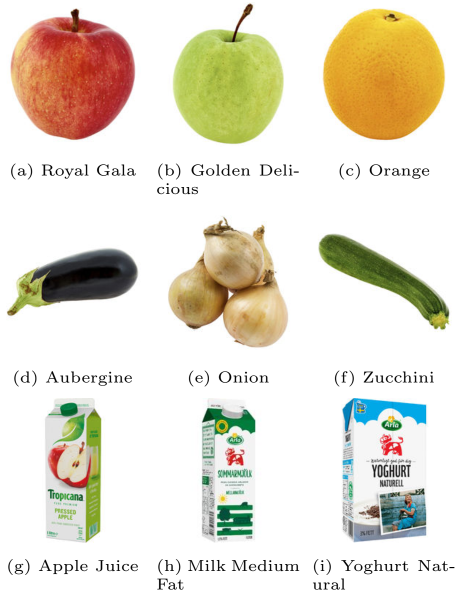


(a) Royal Gala (b) Golden Delicious (c) Orange

(d) Aubergine (e) Onion (f) Zucchini

(g) Apple Juice (h) Milk Medium Fat (i) Yoghurt Natural

Figure 2: Examples of natural images in our dataset, where each image have been taken inside a grocery store. Image examples of fruits, vegetables and refrigerated products are presented in each row respectively.



(a) Royal Gala (b) Golden Delicious (c) Orange

(d) Aubergine (e) Onion (f) Zucchini

(g) Apple Juice (h) Milk Medium Fat (i) Yoghurt Natural

Figure 3: Examples of iconic images downloaded from a grocery shopping website, which corresponds to the target items in the images in Figure 2.

boxes around the object to be labeled for object detection tasks [8, 17, 20]. In [18] and [6], the target objects are usually centered and takes up most content of the image itself. Another significant characteristic is that web images usually are biased in the sense that they have been taken with the object focus in mind; they have good lighting settings and are typically clean from occlusions, since the collectors have used general search words for the object classes, e.g. *car*, *horse*, or *apple*.

Some datasets include additional information about the images beyond the single class label, e.g. text descriptions of what is present in the image and bounding boxes around objects. These datasets can be used in several different computer vision tasks, such as image classification, object detection, and image segmentation. Structured labeling is another important property of a dataset, which provides flexibility when classifying images. In [8, 12], all of these features exist and moreover they include reference images to each object class, which in [12] is used for labeling multiple categories present in images, while in [8] these images are used for fine-grained recognition. Our dataset includes a reference image, i.e. the iconic image, and a product description for every class, and we have also labeled the grocery items in a structured manner.

Other image datasets of fruits and vegetables for classification purposes are the FIDS30 database [23] and the dataset in [24]. The images in FIDS30 were downloaded from the web and contain background noise as well as single or multiple instances of the object. In [24], all pixels belonging to the object are extracted from the original image, such that all images have white backgrounds with the same brightness condition. There also exist datasets for detecting fruits in orchards for robotic harvesting purposes, which are very challenging since the images contain plenty of background and various lighting conditions, and the targeted fruits are often occluded or of the same color as the background [25, 26].

Another dataset that is highly relevant to our application need is presented in [27]. They collected a dataset for training and evaluating the image classifier by extracting images from video recordings of 23 main classes, which are subdivided into 98 classes, of raw grocery items (fruits and vegetables) in different grocery stores. Using this dataset, a mobile application was developed to recognize food products in grocery store environments, which provides the user with details and health recommendations about the item along with other proposals of similar food items. For each class, there exists a product description with nutrition values to assist the user in shopping scenarios. The main difference between this work and our dataset is firstly the clean iconic images (visual labels) for each class in our dataset, and secondly that we have also collected images of refrigerated items, such as dairy and juice containers, where visual information is required to distinguish between the products.

3 Our Dataset

We have collected images from fruit and vegetable sections and refrigerated sections with dairy and juice products in 18 different grocery stores. The dataset consists of 5125 images from 81 fine-grained classes, where the number of images in each class range from 30 to 138. Figure 4 displays a histogram over the number of images per class. As illustrated in Figure 1, the class structure is hierarchical, and there are 46 coarse-grained classes. Figure 2 shows examples of the collected natural images. For each fine-grained class, we have downloaded an iconic image of the item and also a product description including origin country, an appreciated weight and nutrient values of the item from a grocery store website. Some examples of downloaded iconic images can be seen in Figure 3.

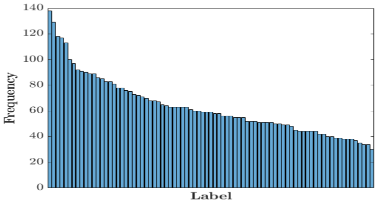


Figure 4: Histogram over the number of images in each class in the dataset.

Our aim has been to collect the natural images under the same condition as they would be as part of an assistive application on a mobile phone. All images have been taken with a 16-megapixel Android smartphone camera from different distances and angles. Occasionally, the images include other items in the background or even items that have been misplaced in the wrong shelf along with the targeted item. It is important that image classifiers that are used for assisting devices are capable of performing well with such noise since these are typical settings in a grocery store environments. The lighting conditions in the images can also vary depending on where the items are located in the store. Sometimes the images are taken while the photographer is holding the item in the hand. This is often the case for refrigerated products since these containers are usually stacked compactly in the refrigerators. For these images, we have consciously varied the position of the object, such that the item is not always centered in the image or present in its entirety.

We also split the data into a training set and test set based on the application need. Since the images have been taken in several different stores at specific days and time stamps, parts of the data will have similar lighting conditions and backgrounds for each photo occasion. To remove any such biasing correlations, all images of a certain class taken at a certain store are assigned to either the test set or training set. Moreover, we balance the class sizes to as large extent as possible in both the training and test set. After the partitioning, the training and test set contains 2640 and 2485 images respectively. Predefining a training and test set also makes it easier for other users to compare their results to the evaluations in this paper.

The task is to classify natural images using mobile devices to aid visually impaired people. The additional information such as the hierarchical structure of the class labels, iconic images, and product descriptions can be used to improve the performance of the computer vision system. Every class label is associated with a product description. Thus, the product description itself can be part of the output for visually impaired persons as they may not be able to read what is printed on a carton box or a label tag on a fruit bin in the store.

The dataset is intended for research purposes and we are open to contributions with more images and new suitable classes. Our dataset is available at https://github.com/marcusklasson/GroceryStoreDataset. Detailed instructions on how to contribute to the dataset can be found on our dataset webpage.

4 Classification Methods

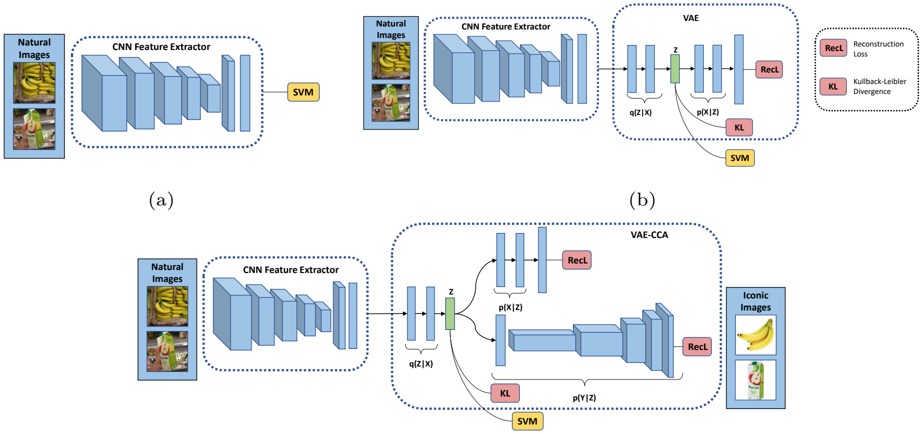
We here describe the classification methods and approaches that we have used to provide benchmark results to the dataset. We apply both deterministic deep neural networks as well as a deep generative model used for representation learning to the natural images that we have collected. Furthermore, we utilize the additional information – iconic images – from our dataset with a multi-view deep generative model. This model can utilize different data sources and obtain superior representation quality as well as high interpretability. For a fair evaluation, we use a linear classifier with the learned representation from the different methods.

Deep Neural Networks. CNNs have been the state-of-the-art models in image classification ever since AlexNet [13] achieved the best classification accuracy in ILSVRC in 2012. However, in general, computer vision models require lots of labeled data to achieve satisfactory performance, which has resulted in interest for adapting CNNs that have already been trained on a large amount of training data to other image datasets. When adapting pretrained CNNs to new datasets, we can either use it directly as a feature extractor, a.k.a use the off-the-shelf features, [28, 29], or fine-tune it [30–34]. Using off-the-shelf features, we need to specify which feature representation we should extract from the network and use these for training a new classifier. Fine-tuning a CNN involves adjusting the pretrained model parameters, such that the network can e.g. classify images from a dataset different from what the CNN was trained on before. We can either choose to fine-tune the whole network or select some layer parameters to adjust while keeping the others fixed. One important factor on deciding which approach to choose is the size of the new dataset and how similar the new dataset is to the dataset which the CNN was previously trained on. A rule of thumb here is that the closer the features are to the classification layer, the features become more specific to the training data and task [33].

Using off-the-shelf CNN features and fine-tuned CNNs have been successfully applied in [28, 29] and [30, 31, 34] respectively. In [28, 29], it is shown that the pretrained features have sufficient representational power to generalize well to other visual recognition tasks with simple linear classifiers, such as Support Vector Machines (SVMs), without fine-tuning the parameters of the CNN to the new task. In [30, 34], all CNN parameters are fine-tuned, whereas in [31] the pretrained CNN layer parameters are kept fixed and only an adaptation layer of two fully connected layers are trained on the new task. The results from these works motivate why we should evaluate our dataset on fine-tuned CNNs or linear classifiers trained on off-the-shelf feature representations instead of training an image recognition model from scratch.

Variational Autoencoders with only natural images. Deep generative models, e.g. the variational autoencoder (VAE) [16, 35, 36], have become widely used in the machine learning community thanks to their generative nature. We thus use VAEs for representation learning as the second benchmarking method. For efficiency, we use low-level pretrained features from a CNN as inputs to the VAE.

The latent representations from VAEs are encodings of the underlying factors for how the data are generated. VAEs belongs to the family of latent variable models, which commonly has the form $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, where $p(\mathbf{z})$ is a prior distribution over the latent variables $\mathbf{z}$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is the likelihood over the data $\mathbf{x}$ given $\mathbf{z}$. The prior distribution is often assumed to be Gaussian, $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \,|\, \mathbf{0}, \mathbf{I})$, whereas the likelihood distribution depends on the values of $\mathbf{x}$. The likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is referred to as a decoder represented as a neural network parameterized by $\boldsymbol{\theta}$. An encoder network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ parameterized by $\boldsymbol{\phi}$



(a)

(b)

(c)

Figure 5: The architectures for the classification methods described in Section 4. In this paper, we use either a pretrained AlexNet, VGG16 or DenseNet-169 as the CNN feature extractor, but it may be replaced with any CNN architecture. Note that the pretrained CNN can be fine-tuned. The encoder and decoder of the VAE in 5b consist of two fully-connected layers. VAE-CCA in 5c uses the DCGAN architecture as an iconic image decoder and the same encoder and feature vector decoder as the VAE.

is introduced as an approximation of the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, which is intractable since it requires computing the integral $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$. When the prior distribution is a Gaussian, the approximate posterior is also modeled as a Gaussian, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z} \,|\, \boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x}) \odot \mathbf{I})$, with some mean $\boldsymbol{\mu}(\mathbf{x})$ and variance $\boldsymbol{\sigma}^2(\mathbf{x})$ computed by the encoder network. The goal is to maximize the marginal log-likelihood by defining a lower bound using $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = &\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \right] \\ &- D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z})). \end{aligned} \tag{1}$$

The last term is the Kullback-Leibler (KL) divergence of the approximate posterior from the true posterior. The lower bound $\mathcal{L}$ is called the evidence lower bound (ELBO) and can be optimized with stochastic gradient descent via backpropagation [16, 37]. VAE is a probabilistic framework. Many extensions such as utilizing structured priors [38] or using continual learning [39] have been explored. In the following method, we describe how to make use of the iconic images while retaining the unsupervised learning setting in VAEs.

Utilizing iconic images with multi-view VAEs. Utilizing extra information has shown to be useful in many applications with various model designs [38, 40–43]. For computer vision tasks, natural language is the most commonly used modality to aid the visual representation learning. However, the consistency of the language and visual embeddings has no guarantee. As an example with our dataset, the product description of a Royal Gala apple explains the appearance of a red apple. But if the description is represented with word embeddings, e.g. word2vec [44], the word 'royal' will probably be more similar to the words 'king' and 'queen' than 'apple'. Therefore, if available, additional visual information about objects might be more beneficial for learning meaningful representations instead of text. In this work, with our collected dataset, we propose to utilize the iconic images for the representation learning of natural images using a multi-view VAE. Since the natural images can include background noise and grocery items different from the targeted one, the role of the iconic image will be to guide the model to which features that are of interest in the natural image.

The VAE can be extended to modeling multiple views of data, where a latent variable $\mathbf{z}$ is assumed to have generated the views [40,42]. Considering two views $\mathbf{x}$ and $\mathbf{y}$, the joint distribution over the paired random variables $(\mathbf{x}, \mathbf{y})$ and latent variable $\mathbf{z}$ can be written as $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y},\mathbf{z}) = p(\mathbf{z})p_{\boldsymbol{\theta}^{(1)}}(\mathbf{x}\,|\,\mathbf{z})p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y}\,|\,\mathbf{z})$, where both $p_{\boldsymbol{\theta}^{(1)}}(\mathbf{x}\,|\,\mathbf{z})$ and $p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y}\,|\,\mathbf{z})$ are represented as neural networks with parameters $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$. Assuming that the latent variable $\mathbf{z}$ can reconstruct both $\mathbf{x}$ and $\mathbf{y}$ when only $\mathbf{x}$ is encoded into $\mathbf{z}$ by the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, then the ELBO is written as

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) \geq & \,\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x},\mathbf{y}) \\ = & \,\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}^{(1)}}(\mathbf{x}|\mathbf{z}) + \log p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y}|\mathbf{z})\right] \\ & - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\,||\,p(\mathbf{z})).\end{aligned} \tag{2}$$

This model is referred to as variational autoencoder canonical correlation analysis (VAE-CCA) and was introduced in [42]. The main motivation for using VAE-CCA is that the latent representations need to contain information about reconstructing both natural and iconic images. The main motivation for using VAE-CCA is that the latent representation needs to preserve information about how both the natural and iconic images are reconstructed. This also allows us to produce iconic images from new natural images to enhance the interpretability of the latent representation of VAE-CCA (see Section 5) [40].

5 Experimental Results

We apply the three different types of models described in Section 4 to our dataset and evaluate their performance. The natural images are propagated through a CNN pretrained on ImageNet to extract feature vectors. We experiment with both the off-the-shelf features as well as fine-tuning the CNN. When using off-the-shelf features, we simply extract feature vectors and train an SVM on those. For the fine-tuned CNN, we report both results from the softmax classifier used in the actual fine-tuning procedure and training an SVM with extracted fine-tuned feature vectors.

Table 1: Fine-grained classification (81 classes) accuracies with the methods described in Section 5.1. Each row displays from which network architecture and layer that we extracted the feature vectors of the natural images. The columns show the result from the classifiers that we used (see Section 5.1).

	SVM	SVM-ft	VAE+SVM	VAE+SVM-ft	VAE-CCA+SVM	VAE-CCA+SVM-ft
AlexNet_6	69.2	72.6	65.6	70.7	67.8	71.5
AlexNet_7	65.0	70.7	63.0	68.7	65.0	70.9
VGG16_6	62.1	73.3	57.5	71.9	60.7	73.0
VGG16_7	57.3	71.7	56.8	67.8	56.8	71.3
DenseNet-169	72.5	85.0	65.4	79.1	72.6	80.4

These extracted feature vectors are also used for VAE and VAE-CCA which makes further compression. We perform classification for those VAE based models by training a classifier, e.g. an SVM, on the data encoded into the latent representation. We use this classification approach for both VAE and VAE-CCA. In all classification experiments, except when we fine-tune the CNN, we use a linear SVM trained with the one-vs-one approach as in [29].

We experiment with three different pretrained CNN architectures, namely AlexNet [13], VGG16 [14] and DenseNet-169 [15]. For AlexNet and VGG16, we extract feature vectors of size 4096 from the two last fully connected (FC) layers before the classification layer. The features from the n^{th} hidden layer are denoted as AlexNet_n and VGG16_n. As an example, the last hidden FC layer in AlexNet is denoted as AlexNet_7, the input of which is output from AlexNet_6. For DenseNet-169, we extract the features of size 1664 from the average pooling layer before its classification layer.

5.1 Experimental Setups

The following setups were used in the experiments:

Setup 1. Train an SVM on extracted off-the-shelf features from a pretrained CNN, which is denoted as SVM in the results. We also fine-tune the CNN by replacing the final layer with a new softmax layer and denote these results as Fine-tune. We denote training an SVM on extracted finetuned feature vectors as SVM-ft.

Setup 2. Extract feature vectors with a pretrained CNN of the natural images and learn a latent representation $\mathbf{z}$ with a VAE. Then the data is encoded into the latent space and we train an SVM with these latent representations, which used for classification. We denote the results as VAE+SVM when using off-the-shelf feature vectors, whereas using the fine-tuned feature vectors are denoted as VAE+SVM-ft. In all experiments with the VAE, we used the architecture from [45], i.e. the latent layer having 200 hidden units and both encoder and decoder consisting of two FC layers with 1,000 hidden units each.

Table 2: Coarse-grained classification (46 classes) accuracies with an SVM for the methods described in Section 5.1 that uses off-the-shelf feature representations. Each row displays from network architecture and layer that we extracted the feature vectors of the natural images and the columns show the result for the classification methods.

	SVM	VAE+SVM	VAE-CCA+SVM
AlexNet$_6$	78.0	74.2	76.4
AlexNet$_7$	75.4	73.2	74.4
VGG16$_6$	76.6	74.2	74.9
VGG16$_7$	72.8	71.7	72.3
DenseNet-169	85.2	79.5	82.0

Table 3: Fine-grained classification accuracies from fine-tuned CNNs pretrained on ImageNet, where the column shows which architecture that has been fine-tuned. A standard softmax layer is used as the last layer.

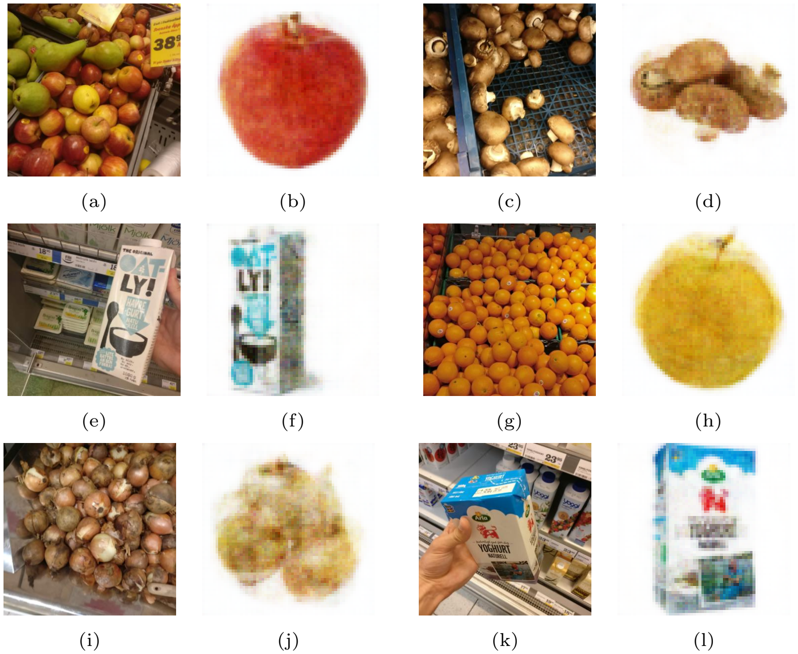
	Fine-tune
AlexNet	69.3
VGG16	73.8
DenseNet-169	84.0

Setup 3. Each natural image is paired with its corresponding iconic image. We train VAE-CCA similarly as the VAE, but instead, we learn a joint latent representation that is used to reconstruct the extracted feature vectors $\mathbf{x}$ and the iconic images $\mathbf{y}$. The classification is performed with the same steps as in Setup 2 and denotes the results similarly with VAE-CCA+SVM and VAE-CCA+SVM-ft. Our VAE-CCA model takes the feature vectors $\mathbf{x}$ as input and encodes them into a latent layer with 200 hidden units. The encoder and the feature vector decoder uses the same architecture, i.e. two FC layers with 512 hidden units, whereas the iconic image decoder uses the DCGAN [46] architecture.

Figure 5 displays the three experimental setups described above. We report both fine-grained and coarse-grained classification results with an SVM in Table 1 and 2 respectively. In Table 3, we report the fine-grained classification results from fine-tuned CNNs.

When fine-tuning the CNNs, we replace the final layer with a softmax layer applicable to our dataset with randomly initialized weights drawn from a Gaussian with zero mean and standard deviation 0.01 [34]. For AlexNet and VGG16, we fine-tune the networks for 30 epochs with two different learning rates, 0.01 for the new classification layer and 0.001 for the pretrained layers. Both learning rates are reduced by half after every fifth epoch. The DenseNet-169 is fine-tuned for 30 epochs with momentum of 0.9 and an initial learning rate of 0.001, which decays with 10^{-6} after each epoch. We report the classification results from the softmax activation after the fine-tuned classification layer. We also report classification results from an SVM trained with feature representations from a fine-tuned CNN, which are extracted from FC6 and FC7 of the AlexNet and VGG16 and from the last average pooling layer in DenseNet-169.

The VAE and VAE-CCA models are trained for 50 epochs with Adam [47] for optimizing the ELBOs in Equation 1 and 2 respectively. We use a constant learning rate of 0.0001 and set the minibatch size to 64. The extracted feature vectors are rescaled with standardization before training the VAE and VAE-CCA models to stabilize the learning.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 6: Examples of six natural images in the test set that have been decoded into product iconic images by the iconic image decoder $p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y}\,|\,\mathbf{z})$ using VAE-CCA model as in Figure 5c. This result is obtained with the fine-tuned DenseNet-169 features, which corresponds to VAE-CCA+SVM-ft in Table 1. We show the natural image and corresponding decoded iconic image next to each other. The classes for all images are (a, b) Royal Gala Apple, (c, d) Brown Cap Mushroom, (e, f) Oatly Oatgurt, (g, h) Orange, (i, j) Onion, and (k, l) Arla Natural Yogurt.

5.2 Results

The fine-grained classification results for all methods using an SVM as classifier are shown in Table 1. We also provide coarse-grained classification results for some of the methods in Table 2 to demonstrate the possibility of hierarchical evaluation that our labeling of the data provides (see Figure 1). The accuracies in the coarse-grained classification are naturally higher than the accuracies in the corresponding columns in Table 1. Table 3 shows fine-grained classification accuracies from a softmax classifier in the fine-tuned CNNs. We note that fine-tuning the networks gives consistently better results than training an SVM on off-the-shelf features (see Table 1).

Fine-tuning the entire network results improves the classification performance consistently for each method in Table 1. The performance is clearly enhanced for features extracted from fine-tuned VGG16 and DenseNet-169, which improves the classification accuracy by 10% in most cases for SVM-ft, VAE+SVM-ft, and VAE-CCA+SVM-ft. For AlexNet and VGG16, we see that the performance drops when extracting the features from layer FC7 instead of FC6. The reason might be that the off-the-shelf features in FC7 are more difficult to transfer to other datasets since the weights are biased towards classifying objects in the ImageNet database. The performance drops also when we use fine-tuned features, which could be due to the small learning rate we use for the pretrained layers, such that the later layers are still ImageNet-specific. We might circumvent this drop by increasing the learning rate for the later pretrained layers and keeping the learning rate for earlier layers small.

The VAE-CCA model achieves mostly higher classification accuracies than the VAE model in both Table 1 and 2. This indicates that the latent representation separates the classes more distinctly than the VAE by jointly learning to reconstruct the extracted feature vectors and iconic images. However, further compressing the feature vectors with VAE and VAE-CCA will lower the classification accuracy compared to applying the feature vectors to a classifier directly. Since both VAE and VAE-CCA compresses the feature vectors into the latent representation, there is a risk of losing information about the natural images. We might receive better performance by increasing the dimension of the latent representation at the expense of speed in both training and classification.

In Figure 6, we show results from the iconic image decoder $p_{\boldsymbol{\theta}^{(2)}}(\mathbf{y} \mid \mathbf{z})$ when translating natural images from the test set into iconic images with VAE-CCA and a fine-tuned DenseNet-169 as feature extractor. Such visualization can demonstrate the quality of the representation using the model, as well as enhancing the interpretability of the method. Using VAE-CCA in the proposed manner, we see that with challenging natural images, the model is still able to learn an effective representation which can be decoded to the correct iconic image. For example, some pears have been misplaced in the bin for Royal Gala apples in Figure 6a, but still the image decoder manages to decode a blurry red apple seen in Figure 6b. In Figure 6h, a mix of an orange and an apple are decoded from a bin of oranges in Figure 6g, which indicates these fruits are encoded close to each other in the learned latent space. Even if Figure 6e includes much of the background, the iconic image decoder is still able to reconstruct the iconic images accurately in Figure 6f, which illustrates that the latent representation is able to explain away irrelevant information in the natural image and preserved the features of the oatgurt package. Thus, using VAE-CCA with iconic images as the second view not only advances the classification accuracy but also provides us with the means to understand the model.

6 Conclusions

This paper presents a dataset of images of various raw and packaged grocery items, such as fruits, vegetables, and dairy and juice products. We have used a structured labeling of the items, such that grocery items can be grouped into more general (coarse-grained) classes and also divided into fine-grained classes. For each class, we have a clean iconic image and a text description of the item, which can be used for adding visual and semantic information about the items in the modeling. The intended use of this dataset is to train and benchmark assistive systems for visually impaired people when they shop in a grocery store. Such a system would complement existing visual assistive technology, which is confined to grocery items with barcodes. We also present preliminary benchmark results for the dataset on the task of image classification.

We make the dataset publicly available for research purposes at https://github.com/marcusklasson/GroceryStoreDataset. Additionally, we will both continue collecting natural images, as well as ask for public contributions of natural images in shopping scenarios to enlarge our dataset.

For future research, we will advance our model design to utilize the structured nature of our labels. Additionally, we will design a model that use the product description of the objects in addition to the iconic images. One immediate next step is to extend the current VAE-CCA model to three views, where the third view is the description of the product.

References

[1] Microsoft Seeing AI app. https://www.microsoft.com/en-us/seeing-ai/. Accessed on 2018-02-22.

[2] Aipoly Vision app. https://www.aipoly.com/. Accessed on 2018-02-28.

[3] Orcam. https://www.orcam.com/en/. Accessed on 2018-02-28.

[4] S. Caraiman, A. Morar, M. Owczarek, A. Burlacu, D. Rzeszotarski, N. Botezatu, P. Herghelegiu, F. Moldoveanu, P. Strumillo, and A. Moldoveanu. Computer vision for the visually impaired: the sound of vision system. In *IEEE International Conference on Computer Vision Workshops*, 2017.

[5] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[7] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, 2013.

[8] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *AAAI Conference on Artificial Intelligence*, 2017.

[9] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[10] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning*, 2011.

[11] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *IEEE International Conference on Computer Vision*, 2015.

[12] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[16] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[17] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[18] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[19] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[20] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[21] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, 2015.

[23] Škrjanec Marko. Automatic fruit recognition using computer vision. (Mentor: Matej Kristan), Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013. FIDS30 dataset was accessed 2018-02-24 at http://www.vicos.si/Downloads/FIDS30.

[24] Horea Muresan and Mihai Oltean. Fruit recognition from images using deep learning. Technical report, Babes-Bolyai University, 2017.

[25] Suchet Bargoti and James Patrick Underwood. Deep fruit detection in orchards. In *IEEE International Conference on Robotics and Automation*, 2017.

[26] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222, 2016.

[27] Georg Waltner, Michael Schwarz, Stefan Ladstätter, Anna Weber, Patrick Luley, Horst Bischof, Meinrad Lindschinger, Irene Schmid, and Lucas Paletta. Mango - mobile augmented reality with functional eating guidance and food awareness. In *International Workshop on Multimedia Assisted Dietary Management*, 2015.

[28] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[29] Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2014.

[31] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[32] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[33] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

[34] Ning Zhang, Jeff Donahue, Ross B. Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, 2014.

[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

[36] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. *arXiv preprint arXiv:1711.05597*, 2017.

[37] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016.

[38] Judith Butepage, Jiawei He, Cheng Zhang, Leonid Sigal, and Stephan Mandt. Informed priors for deep representation learning. In *Symposium on Advances in Approximate Bayesian Inference*, 2018.

[39] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[40] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. In *International Conference on Learning Representations*, 2018.

[41] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[42] Weiran Wang, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis. *CoRR*, abs/1610.03454, 2016.

[43] Cheng Zhang, Hedvig Kjellström, and Carl Henrik Ek. Inter-battery topic representation learning. In *European Conference on Computer Vision*, pages 210–226. Springer, 2016.

[44] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. 2015.

[46] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Paper B

Using Variational Multi-View Learning for Classification of Grocery Items

Marcus Klasson*, **Cheng Zhang**†, **Hedvig Kjellström***

*KTH Royal Institute of Technology, Stockholm, Sweden
†Microsoft Research, Cambridge, United Kingdom

Abstract

An essential task for computer vision-based assistive technologies is to help visually impaired people to recognize objects in constrained environments, for instance, recognizing food items in grocery stores. In this paper, we introduce a novel dataset with natural images of groceries – fruits, vegetables, and packaged products – where all images have been taken inside grocery stores to resemble a shopping scenario. Additionally, we download iconic images and text descriptions for each item that can be utilized for better representation learning of groceries. We select a multi-view generative model, which can combine the different item information into lower-dimensional representations. The experiments show that utilizing the additional information yields higher accuracies on classifying grocery items over only using the natural images. We observe that iconic images help to construct representations separated by visual differences of the items, while text descriptions enable the model to distinguish between visually similar items by different ingredients.

1 Introduction

hej hej här är en artikel

what do you think this is?

Paper C

Learn the Time to Learn: Replay Scheduling in Continual Learning

Marcus Klasson[*], Hedvig Kjellström[*], Cheng Zhang[†]

[*]KTH Royal Institute of Technology, Stockholm, Sweden
[†]Microsoft Research, Cambridge, United Kingdom

Abstract

Replay-based continual learning has shown to be successful in mitigating catastrophic forgetting, where most works focus on improving the sample quality in the commonly small replay memory. However, in many real-world applications, replay memories would be limited by constraints on processing time rather than storage capacity as most organizations store all historical data in the cloud. Inspired by human learning, we demonstrate that scheduling over the time to replay is critical to the final performance with finite memory resources. To this end, we propose to learn the time to learn for a continual learning system, in which we learn schedules over which tasks to replay at different times. We use Monte Carlo tree search to illustrate this idea and show that our method can be combined with any replay-based method and memory selection technique. We perform extensive evaluation showing that learning replay schedules can significantly improve the performance compared to baselines without learned scheduling. Our results indicate that the learned schedules are also consistent with human learning insights.

1 Introduction

Many organizations deploying machine learning systems receive large volumes of data daily where these new data are often associated with new tasks. Although all historical data are stored in the cloud in practice, retraining machine learning systems on a daily basis is prohibitive both in time and cost. In this setting, the systems must continuously adapt to new tasks without forgetting the previously

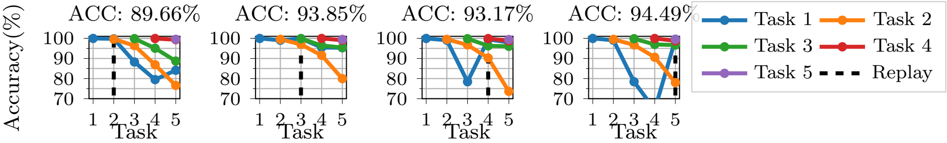


Figure 1: Task accuracies on Split MNIST [16] when replaying only 10 samples of classes 0/1 at a single time step. The black vertical line indicates when replay is used. ACC denotes the average accuracy over all tasks after learning Task 5. Results are averaged over 5 seeds. These results show that the time to replay the previous task is critical for the final performance.

learned abilities. Continual learning (CL) methods [1, 2] address this challenge where, in particular, replay-based methods [3, 4] have shown to be very effective in achieving great prediction performance and retaining knowledge of old tasks. Replay-based methods mitigate catastrophic forgetting by revisiting a small set of samples, which is feasible to process compared to the size of the historical data. In the traditional CL literature, the replay memory is limited due to the assumption that historical data are not available. In the real-world setting where historical data are in fact always available, the requirement of small memory remains due to processing time and cost issues.

Most research on replay-based CL has been focused on the sample quality in the memory [3, 5–10] or data compression to increase the memory capacity [4, 11, 12]. Common for these methods is that the memory allocates an equal amount of space for storing samples from old tasks. When learning new tasks, the whole memory is replayed to mitigate catastrophic forgetting. However, in life-long learning settings, this simple strategy would be inefficient as the memory must store a large number of tasks. Furthermore, these methods ignore the time to learn old tasks again which is important in human learning. Humans are CL systems, and different methods have been developed to enhance memory retention, such as spaced repetition [13–15] which is often used in education. These education methods focus on the scheduling of learning and rehearsal of previous learned knowledge.

We argue that finding the proper schedule of which tasks to replay in the fixed memory setting is critical for CL. To demonstrate our claim, we perform a simple experiment on the Split MNIST [16] dataset where each task consists of learning the digits 0/1, 2/3, etc. arriving in sequence. The replay memory contains data from task 1 and can only be replayed at one point in time. Figure 1 shows how the task performances progress over time when the memory is replayed at different time steps. In this example, the best final performance is achieved when the memory is used when learning task 5. Note that choosing different time points to replay the same memory leads to noticeably different results in the final performance. These results indicate that scheduling the time when to apply replay can influence the final performance significantly of a CL system.

To this end, we propose learning the time to learn, in which we learn replay schedules of which tasks to replay at different times inspired from human learning [13]. We illustrate the advantages with replay scheduling by using Monte Carlo tree search (MCTS) [17] to learn policies for replay. More specifically, we train a neural network on the current task dataset mixed with the scheduled replay samples and measure the final performance of the network to evaluate the replay schedules selected by MCTS. In summary, our contributions are:

- We propose a new CL setting where historical data is available while the processing time is limited, in order to adjust current CL research closer to real-world needs (Section 3.1). In this new setting, we introduce replay scheduling in CL where we learn the time to learn which tasks to replay (Section 3.2).
- We use MCTS as an example method to illustrate how to learn replay scheduling policies by creating finite sets of memory compositions that can be replayed at every task (Section 3.3).
- We demonstrate with six benchmark datasets that learned scheduling can improve the CL performance significantly in the fixed size memory setting (Section **??** and **??**). Moreover, we show that our method can be combined with any memory selection technique and replay-based method (Section **??** and **??**), as well as being efficient in situations where the memory size is even smaller than the number of classes (Section **??**).

2 Related Work

In this section, we give a brief overview of CL methods, essentially replay-based methods, as well as spaced repetition techniques for human CL.

Continual Learning. Traditional CL can be divided into three main areas, namely regularization-based, architecture-based, and replay-based approaches. Regularization-based methods aim to mitigate catastrophic forgetting by protecting parameters influencing the predictive performance on known tasks from wide changes and use the rest of the parameters for learning new tasks [8, 18–23, 165]. Architecture-based methods isolate task-specific parameters by either increasing network capacity [24–26] or freezing parts of the network [27, 28] to maintain good performance on previous tasks. Replay-based methods mix samples from old tasks with the current dataset to mitigate catastrophic forgetting, where the replay samples are either stored in an external memory [3, 4, 29, 30] or generated using a generative model [31, 32]. Regularization-based approaches and dynamic architectures have been combined with replay-based approaches to methods to overcome their limitations [8,12,19,33–42]. Our work relates most to replay-based methods with external memory which we spend more time on describing in the next paragraph.

Replay-based Continual Learning. A commonly used memory selection strategy of replay samples is random selection. Much research effort has focused on selecting higher quality samples to store in memory [3, 5–10, 29, 30, 43].

Chaudhry et al. [3] reviews several selection strategies in scenarios with tiny memory capacity, e.g., reservoir sampling [44], first-in first-out buffer [30], k-Means, and Mean-of-Features [9]. However, more elaborate selection strategies have been shown to give little benefit over random selection for image classification problems [4, 19]. More recently, there has been work on compressing raw images to feature representations to increase the number of memory examples for replay [4, 11, 12]. Our approach differs from the above mentioned works since we focus on learning to select which tasks to replay at the current task rather than improving memory selection or compression quality of the memory samples. Replay scheduling can however be combined with any selection strategy and feature compression method.

Human Continual Learning. Humans are CL systems in the sense of learning tasks and concepts sequentially. Furthermore, humans have the ability to memorize experiences but forgets learned knowledge gradually rather than catastrophically [45]. Different learning techniques have been suggested for humans to memorize better [46,47]. An example is spaced repetition where time intervals between rehearsal are gradually increased to improve long-term memory retention [13], where the earliest documented works are from Ebbinghaus [14]. Further studies have shown that memory training schedules with adjusted spaced repetition are better at preserving memory than using uniformly spaced rehearsal times [15,48]. Several works in CL with neural networks are inspired by human learning techniques, including spaced repetition [49–51], sleep mechanisms [23, 27, 52], and memory reactivation [4, 53]. Replay scheduling is also inspired by spaced repetition, where we learn schedules of which tasks to replay at different times.

3 Method

In this section, we describe our new problem setting in CL where historical data are available while the processing time is limited when learning new tasks. To this end, we present our method for learning replay schedules in CL where the idea is to learn schedules of which tasks the network should replay at different times. We use Monte Carlo tree search (MCTS) [17] to learn a scheduling policy by encouraging searches for promising replay schedules based on the classification accuracy.

3.1 Problem Setting

We focus on a slightly new setting, considering the needs for CL in the real-world where all historical data can be available since data storage is cheap. However, as this data volume is typically huge, we are constrained to use a small amount of historical data for replay when learning new tasks due to limitations on the amount of compute rather than data storage capability. Thus, the goal is to learn how to select subsets of historical data to efficiently mitigate catastrophic forgetting. We refer to these subsets of historical data as the *replay memory* throughout the paper, where the size of the replay memory affects the processing time when learning a new task. Moreover, we focus on composing the replay memory based on the seen tasks in the historical data rather than single stored instances.

Here, we introduce the notation of our problem setting which resembles the traditional CL setting for image classification. We let a neural network $f_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta}$, learn T tasks sequentially from the datasets $\mathcal{D}_1, \dots, \mathcal{D}_T$ arriving one at a time. The t-th dataset $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{N_t}$ consists of N_t samples where $\boldsymbol{x}_t^{(i)}$ and $y_t^{(i)}$ are the i-th data point and class label respectively. The training objective at task t is given by

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N_t} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(i)}), y_t^{(i)}), \tag{1}$$

where $\ell(\cdot)$ is the loss function, e.g., cross-entropy loss in our case. When learning task t, the network $f_{\boldsymbol{\theta}}$ is at risk of catastrophically forgetting the previous $t-1$ tasks. Replay-based methods mitigate forgetting historical tasks by storing old examples in an external replay memory, that is mixed with the current task dataset during training. Next, we describe our method for constructing this replay memory.

We assume that historical data from old tasks are accessible at any time step t. However, since the historical data is prohibitively large, we can only fill a small replay memory $\mathcal{M}$ with M historical samples for replay due to processing time constraints. The challenge is how to fill the replay memory with M samples that efficiently retain the knowledge of old tasks when learning new tasks. We focus on selecting the samples on task-level by deciding on the task proportion $(a_1, \dots, a_{t-1})$ of samples to fetch from each task, where $a_i \geq 0$ is the proportion of M examples from task i to place in $\mathcal{M}$ and $\sum_{i=1}^{t-1} a_i = 1$. Consequently, we need a method for choosing these task proportions of which old tasks to replay. To simplify this selection, we construct a discrete set of choices for possible task proportions that can be used for constructing the replay memory $\mathcal{M}$.

3.2 Replay Scheduling in Continual Learning

In this section, we describe our replay scheduling method for selecting the replay memory at different time steps. We define a replay schedule as a sequence $S = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_{T-1})$, where $\boldsymbol{a}_i = (a_1, \dots, a_{T-1})$ for $1 \leq i \leq T-1$ is the sequence of task proportions used for determining how many samples per task to fill the replay memory with at task i. To make the selection of task proportions tractable, we construct an action space with a discrete number of choices for task proportions from old tasks. We use the following method to construct this action space: At the time to learn task t, we have $t-1$ historical tasks that we can choose from. We create $t-1$ bins $\boldsymbol{b}_t = [b_1, \dots, b_{t-1}]$ and choose a task index to sample for each bin $b_i \in \{1, \dots, t-1\}$. We treat the bins as interchangeable and only

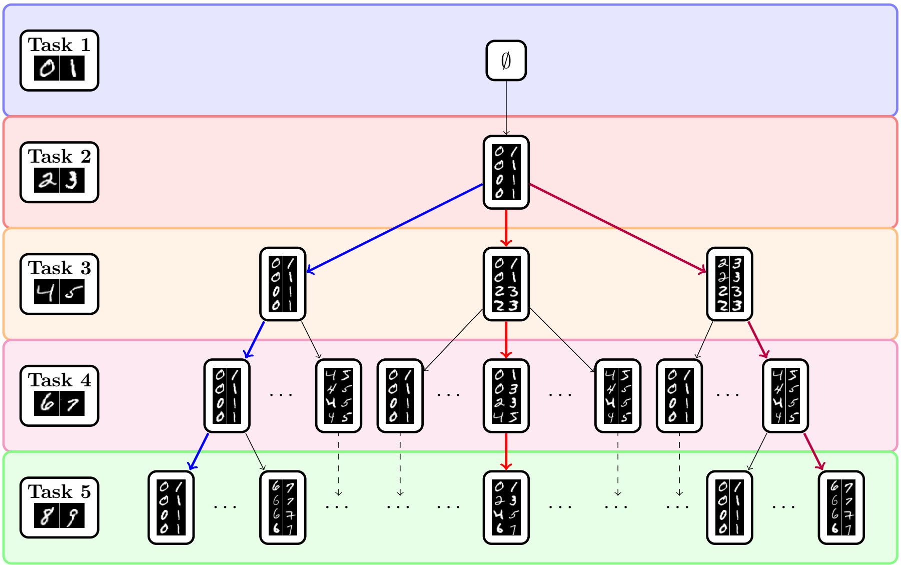


Figure 2: An exemplar tree of replay memory compositions from the proposed discretization method described in Section **??** for Split MNIST. The replay memories from one replay schedule are found by traversing from task 1-5 through the tree on the right hand side. The replay memory compositions have been structured according to the task where they can be used for replay. Note that the replay memory at task 1 is the empty set, i.e., $\mathcal{M} = \emptyset$. Example images for each task are shown on the left.

keep the unique choices. For example, at task 3, we have seen task 1 and 2, so the unique choices of vectors are $[1, 1], [1, 2], [2, 2]$, where $[1, 1]$ indicates that all memory samples are from task 1, $[1, 2]$ indicates that half memory is from task 1 and the other half are from task etc. The task proportions are then computed by counting the number of occurrences of each task index in $\boldsymbol{b}_t$ and dividing by $t-1$, such that $\boldsymbol{a}_t = \texttt{bincount}(\boldsymbol{b}_t)/(t-1)$. If the memory size M is not divisible by the task proportion value for task i, we round the number of replay samples from task i, i.e., $a_i \cdot M$, up or down accordingly while keeping the memory size fixed. From this specification, we can build a tree of different replay schedules to evaluate with the network.

Figure 2 shows an example of a replay schedule tree with Split MNIST [16] where the memory size is $M = 8$. The figure shows the current tasks with example images of the task classes on the left, and the right side shows examples of possible replay memories that can be evaluated. The memory starts as the empty set, i.e. $\mathcal{M}_1 = \emptyset$, at task 1. Before learning task 2, $\mathcal{M}_2$ is filled with M task 1 examples since this is the only task seen so far. At task 3, the memory compositions we can choose from are M examples from either task 1 or 2, as well as equally filling $\mathcal{M}_3$ with four examples each from both tasks. A replay schedule is represented as a path traversal of different replay memory compositions from task 1 to task 5. We have color-coded three examples of possible schedules in

Figure 2 to use for illustration: the blue path represents a replay schedule where only task 1 examples are replayed. The red path represents using an equally distributed amount of memory samples per task in the memory, and the purple path represents a schedule where the memory is filled M examples from the previously visited task. Note that all other possible paths in the tree are also valid replay schedules.

3.3 Monte Carlo Tree Search for Replay Schedules

Our proposed setup to learn replay schedules is to discretize the number of possible task proportions per task. With our discretization method, the action space of task proportions becomes shaped as a tree where each node represents a task proportion that corresponds to a specific composition of the replay memory $\mathcal{M}$, as can be seen in Figure 2. To find the most efficient replay schedule S in mitigating catastrophic forgetting, we can perform an exhaustive search (e.g., breadth-first search) where all possible replay schedules are evaluated. However, since the tree grows fast with the number of tasks, we need a scalable method that enables tree searches in large action spaces. To this end, we propose to use Monte Carlo tree search (MCTS) [17] since MCTS has been successful in applications with similar conditions [54–57]. In our setting, MCTS can help us to concentrate the search in directions towards promising replay schedules in terms of classification performance and thus learn the time to learn.

Before outlining the steps for performing MCTS, we describe the notation for the tree search. The tree has T levels where level t corresponds to task t. Each tree level t contains a set of nodes $V_t = \{v_t^i\}_{i=1}^{K_t}$ where K_t is the number of nodes at level t. Every node v_t^i is related to a sequence of task proportions $\boldsymbol{a}_t^i$, which can be used to retrieve the replay memory from the historical data. Referring back to Figure 2, the replay memory composition on the root node v_1^1 is the empty set $\mathcal{M} = \emptyset$, since the historical data is empty. At task 2, the only task proportion is $\boldsymbol{a}_2^1 = (a_1, a_2, ..., a_{T-1}) = (1.0, 0.0, ..., 0.0)$, so the replay memory at node v_2^1 is only filled with samples from task 1. In the rest of the paper, we denote a node at task t as v_t by ignoring the superscript i to avoid clutter in the notation.

At every visited node v_t, we store the corresponding task proportions $\boldsymbol{a}_t$ in the replay schedule S. The final replay schedule is then used for constructing the replay memory at each task during the CL training. Next, we briefly outline the MCTS steps for performing the search of replay schedules:

Selection. An MCTS iteration begins by selecting the next node to visit from the root node. If the current node v_t has visited all its children during the previous iterations, the next node is selected by evaluating the Upper Confidence Tree (UCT) [58] function for all children. We use the following UCT function proposed by [55] to evaluate the score for moving from node v_t to its child v_{t+1}:

$$UCT(v_t, v_{t+1}) = \max(Q(v_{t+1})) + C\sqrt{\frac{2\log(N(v_t))}{N(v_{t+1})}}. \tag{2}$$

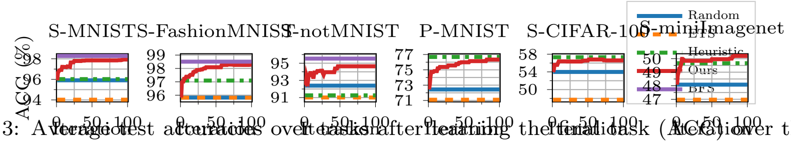


Figure 3: Average test accuracies over tasks after learning the final task (ACC) over the MCTS simulations for all datasets, where 'S' and 'P' are used as short for 'Split' and 'Permuted'. We compare performance for RS-MCTS (Ours) against random replay schedules (Random), Equal Task Schedule (ETS), and Heuristic Scheduling (Heuristic) baselines. For the first three datasets, we show the best ACC found from a breadth-first search (BFS) as an upper bound. All results have been averaged over 5 seeds. These results show that replay scheduling can improve over ETS and outperform or perform on par with Heuristic across different datasets and network architectures.

The reward function $Q(\cdot)$ contains the rewards for previous searches that has passed through the child v_{t+1}. The exploration constant $C \geq 0$ determines the degree of exploration to consider based on the number of visits $N(v_t)$ and $N(v_{t+1})$ to the corresponding nodes in the tree. The child node to visit next is the one with the highest UCT score.

Expansion. Whenever the current node v_t has unvisited child nodes, the search tree is expanded with one of the unvisited child nodes v_{t+1} selected with uniform sampling.

Simulation and Reward. After the expansion step, the search proceeds by selecting the succeeding nodes uniformly at random until reaching a terminal node v_T. The replay schedule S is then collected by appending the task proportions from the visited nodes during the iteration. We train the network with the replay memories retrieved from S and evaluate the average accuracy over all tasks after learning task T as the reward r, i.e., $r = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}^{(val)}$, where $A_{T,i}^{(val)}$ is the validation accuracy of task i at task T.

Backpropagation. Reward r is backpropagated from the expanded node v_t to the root v_1, where the reward function $Q(\cdot)$ and number of visits $N(\cdot)$ are updated at each node.

We provide pseudocode in Algorithm 3 in Appendix 2. Finally, we emphasize that we use MCTS as an example method for illustrating the benefits of learning replay schedules in our new proposed CL setting.

4 Experiments

We evaluated our replay scheduling method empirically on six common benchmark datasets for CL. We denote our method as Replay Scheduling MCTS (RS-MCTS) and select the result on the held-out test sets from the replay schedule that yielded the best reward on the validation set during the search. Full details on experimental settings are in Appendix ?? and additional results are in Appendix ??. Our code is available in the supplementary material.

Datasets. We conduct experiments on six datasets commonly used as benchmarks in the CL literature: Split MNIST [16, 59], Fashion-MNIST [60], Split

Table 1: Performance comparison with ACC between RS-MCTS (Ours), Random scheduling (Random), Equal Task Schedule (ETS), and Heuristic Scheduling (Heuristic) with various memory selection methods evaluated across all datasets. Replay memory sizes are $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation averaged over 5 seeds. Ours performs better or on par with the baselines on most datasets and selection methods, where MoF yields the best results in general.

		5-task Datasets			**10- and 20-task Datasets**		
Selection	**Method**	S-MNIST	S-FashionMNIST	S-notMNIST	P-MNIST	S-CIFAR-100	S-miniIma
Uniform	Random	95.91 (± 1.56)	95.82 (± 1.45)	92.39 (± 1.29)	72.44 (± 1.15)	53.99 (± 0.51)	48.08 (±
	ETS	94.02 (± 4.25)	95.81 (± 3.53)	91.01 (± 1.39)	71.09 (± 2.31)	47.70 (± 2.16)	46.97 (±
	Heuristic	96.02 (± 2.32)	97.09 (± 0.62)	91.26 (± 3.99)	**76.68** (± 2.13)	**57.31** (± 1.21)	49.66 (±
	Ours	**97.93** (± 0.56)	**98.27** (± 0.17)	**94.64** (± 0.39)	76.34 (± 0.98)	56.60 (± 1.13)	**50.20** (±
k-means	Random	94.24 (± 3.20)	96.30 (± 1.62)	91.64 (± 1.39)	74.30 (± 1.43)	53.18 (± 1.66)	49.47 (±
	ETS	92.89 (± 3.53)	96.47 (± 0.85)	**93.80** (± 0.82)	69.40 (± 1.32)	47.51 (± 1.14)	45.82 (±
	Heuristic	96.28 (± 1.68)	95.78 (± 1.50)	91.75 (± 0.94)	75.57 (± 1.18)	54.31 (± 3.94)	49.25 (±
	Ours	**98.20** (± 0.16)	**98.48** (± 0.26)	93.61 (± 0.71)	**77.74** (± 0.80)	**56.95** (± 0.92)	**50.47** (±
k-center	Random	96.40 (± 0.68)	95.57 (± 3.16)	92.61 (± 1.70)	71.41 (± 2.75)	48.46 (± 0.31)	44.76 (±
	ETS	94.84 (± 1.40)	97.28 (± 0.50)	91.08 (± 2.48)	69.11 (± 1.69)	44.13 (± 1.06)	41.35 (±
	Heuristic	94.55 (± 2.79)	94.08 (± 3.72)	92.06 (± 1.20)	74.33 (± 2.00)	50.32 (± 1.97)	44.13 (±
	Ours	**98.24** (± 0.36)	**98.06** (± 0.35)	**94.26** (± 0.37)	**76.55** (± 1.16)	**51.37** (± 1.63)	**46.76** (±
MoF	Random	95.18 (± 3.18)	95.76 (± 1.41)	91.33 (± 1.75)	77.96 (± 1.84)	61.93 (± 1.05)	54.50 (±
	ETS	97.04 (± 1.23)	96.48 (± 1.33)	92.64 (± 0.87)	77.62 (± 1.12)	60.43 (± 1.17)	56.12 (±
	Heuristic	96.46 (± 2.41)	95.84 (± 0.89)	93.24 (± 0.77)	77.27 (± 1.45)	55.60 (± 2.70)	52.30 (±
	Ours	**98.37** (± 0.24)	**97.84** (± 0.32)	**94.62** (± 0.42)	**81.58** (± 0.75)	**64.22** (± 0.65)	**57.70** (±

notMNIST [?], Permuted MNIST [61], Split CIFAR-100 [62], and Split miniImagenet [63]. We randomly sample 15% of the training data from each task to use for validation when computing the reward for the MCTS simulations.

Baselines. We compare RS-MCTS to using 1) random replay schedules (Random), 2) equal task schedules (ETS), and 3) a heuristic scheduling method (Heuristic). The ETS baseline uses equal task proportions, such that $M/(t-1)$ samples per task are replayed during learning of task t,and use both training and validations sets for training such that ETS use the same amount of data as RS-MCTS. The Heuristic baseline replays the tasks which accuracy on the validation set is below a certain threshold proportional to the best achieved validation accuracy on the task. Here, the replay memory is filled with M/k samples per task where k is the number of selected tasks. If $k = 0$, then we skip applying replay at the current task. See Appendix **??** for more details on the Heuristic baseline.

Architectures. We use a multi-head output layer for all datasets except for Permuted MNIST where the network uses single-head output. We use a 2-layer MLP with 256 hidden units for Split MNIST, Split FashionMNIST, Split notMNIST, and Permuted MNIST. For Split CIFAR-100, we use the CNN architecture used in Vinyals et al. [63] and Schwartz et al. [23]. For Split miniImagenet, we apply the reduced ResNet-18 from [30].

Evaluation Metric. We use the average test accuracy over all tasks after learning the final task, i.e., ACC $= \frac{1}{T}\sum_{i=1}^{T} A_{T,i}$ where $A_{T,i}$ is the test accuracy of task i after learning task T. We report means and standard deviations of ACC using 5 different seeds on all datasets.

4.1 Performance Progress of RS-MCTS

In the first experiments, we show that the replay schedules from RS-MCTS yield better performance than replaying an equal amount of samples per task. The replay memory size is fixed to $M = 10$ for Split MNIST, FashionMNIST, and notMNIST, and $M = 100$ for Permuted MNIST, Split CIFAR-100, and Split miniImagenet. Uniform sampling is used as the memory selection method for all methods in this experiment. For the 5-task datasets, we provide the optimal replay schedule found from a breadth-first search (BFS) over all 1050 possible replay schedules in our action space (which corresponds to a tree with depth of 4) as an upper bound for RS-MCTS. As the search space grows fast with the number of tasks, BFS becomes computationally infeasible when we have 10 or more tasks.

Figure 3 shows the progress of ACC over iterations by RS-MCTS for all datasets. We also show the best ACC metrics for Random, ETS, Heuristic, and BFS (where appropriate) as straight lines. We observe that RS-MCTS outperforms Random and ETS successively with more iterations. Furthermore, RS-MCTS approaches the upper limit of BFS on the 5-task datasets. For Permuted MNIST and Split CIFAR-100, the Heuristic baseline and RS-MCTS perform on par after 50 iterations. This shows that Heuristic with careful tuning of the validation accuracy threshold can be a strong baseline when comparing replay scheduling methods. The top row of Table 1 shows the ACC for each method for this experiment. We note that RS-MCTS outperforms ETS significantly on most datasets and performs on par with Heuristic.

4.2 Replay Schedule Visualizations

We visualize a learned replay schedule from Split CIFAR-100 with memory size $M = 100$ to gain insights into the behavior of the scheduling policy from RS-MCTS. Figure 4 shows a bubble plot of the task proportions that are used for filling the replay memory at every task. Each circle color corresponds to a historical task and the circle size represents its proportion of replay samples at the current task. The sum of all points from all circles at each column is fixed at the different time steps since the memory size M is fixed. The task proportions vary dynamically over time in a sophisticated nonlinear way which would be hard to replace by a heuristic method. Moreover, we can observe space repetition style scheduling on many tasks, e.g., task 1-3 are replayed with similar proportion at the initial tasks but eventually starts varying the time interval between replay. Also, task 4 and 6 need less replay in their early stages, which could potentially be that they are simpler or correlated with other tasks. We provide a similar

Table 2: Performance comparison with ACC between scheduling methods RS-MCTS (Ours), Random, ETS, and Heuristic combined with replay-based methods HAL, MER, DER, and DER++. Replay memory sizes are $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation averaged over 5 seeds. Results on Heuristic where some seed did not converge is denoted by *. Applying RS-MCTS to each method can enhance the performance compared to using the baseline schedules.

		5-task Datasets			**10- and 20-task Datasets**		
Method	**Schedule**	S-MNIST	S-FashionMNIST	S-notMNIST	P-MNIST	S-CIFAR-100	S-minil
HAL	Random	97.24 (± 0.70)	86.74 (± 6.05)	93.61 (± 1.31)	88.49 (± 0.99)	36.09 (± 1.77)	38.51
	ETS	94.02 (± 4.25)	95.81 (± 3.53)	91.01 (± 1.39)	88.46 (± 0.86)	34.90 (± 2.02)	38.13
	Heuristic	97.69 (± 0.19)	*74.16 (± 11.19)	93.64 (± 0.93)	*66.63 (± 28.50)	35.07 (± 1.29)	39.51
	Ours	**97.93** (± 0.56)	**98.27** (± 0.17)	**94.64** (± 0.39)	**89.14** (± 0.74)	**40.22** (± 1.57)	**41.39**
MER	Random	93.07 (± 0.81)	85.53 (± 3.30)	91.13 (± 0.86)	75.90 (± 1.34)	42.96 (± 1.70)	31.48
	ETS	92.89 (± 3.53)	96.47 (± 0.85)	**93.80** (± 0.82)	73.01 (± 0.96)	43.38 (± 1.81)	33.58
	Heuristic	94.30 (± 2.79)	96.91 (± 0.62)	90.90 (± 1.30)	**83.86** (± 3.19)	40.90 (± 1.70)	**34.22**
	Ours	**98.20** (± 0.16)	**98.48** (± 0.26)	93.61 (± 0.71)	79.72 (± 0.71)	**44.29** (± 0.69)	32.74
DER	Random	98.23 (± 0.53)	96.56 (± 1.79)	92.89 (± 0.86)	87.51 (± 1.10)	56.83 (± 0.76)	42.19
	ETS	98.17 (± 0.35)	97.69 (± 0.58)	94.74 (± 1.05)	85.71 (± 0.75)	52.58 (± 1.49)	35.50
	Heuristic	94.57 (± 1.71)	*72.49 (± 19.32)	*77.88 (± 12.58)	81.56 (± 2.28)	55.75 (± 1.08)	**43.62**
	Ours	**99.02** (± 0.10)	**98.33** (± 0.51)	**95.02** (± 0.33)	**90.11** (± 0.18)	**58.99** (± 0.98)	43.46
DER++	Random	97.90 (± 0.52)	97.10 (± 1.03)	93.29 (± 1.43)	87.89 (± 1.10)	58.49 (± 1.44)	48.40
	ETS	97.98 (± 0.52)	98.12 (± 0.40)	94.53 (± 1.02)	85.25 (± 0.88)	52.54 (± 1.06)	41.36
	Heuristic	92.35 (± 2.42)	*67.31 (± 21.20)	93.88 (± 1.33)	79.17 (± 2.44)	56.70 (± 1.27)	45.73
	Ours	**98.84** (± 0.21)	**98.38** (± 0.43)	**94.73** (± 0.20)	**89.84** (± 0.22)	**59.23** (± 0.83)	**49.45**

visualization for Split MNIST in Figure 7 in Appendix 4.1 to bring more insights to the benefits of replay scheduling.

4.3 Alternative Memory Selection Methods

We show that our method can be combined with any memory selection method for storing replay samples. In addition to uniform sampling, we apply various memory selection methods commonly used in the CL literature, namely k-means clustering, k-center clustering [8], and Mean-of-Features (MoF) [9]. We compare our method and ETS combined with these different selection methods. The replay memory sizes are $M = 10$ for the 5-task datasets and $M = 100$ for the 10- and 20-task datasets. Table 1 shows the results across all datasets. We note that using the replay schedule from RS-MCTS outperforms the baselines when using the alternative selection methods, where MoF performs the best on most datasets.

4.4 Applying Scheduling to Recent Replay Methods

In this experiment, we show that replay scheduling can be combined with any replay method to enhance the CL performance. We combine RS-MCTS with Hindsight Anchor Learning (HAL) [35], Meta-Experience Replay (MER) [64], Dark Experience Replay (DER) [33], and DER++. We provide the hyperparameter settings in Appendix **??**. Table 2 shows the performance comparison between our proposed replay scheduling against using Random, ETS, and Heuristic sched-

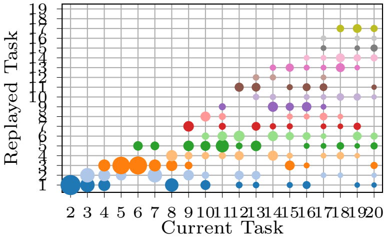


Figure 4: Replay schedule learned from Split CIFAR-100 visualized as a bubble plot. The task proportions vary dynamically over time which would be hard to replace by a heuristic method.

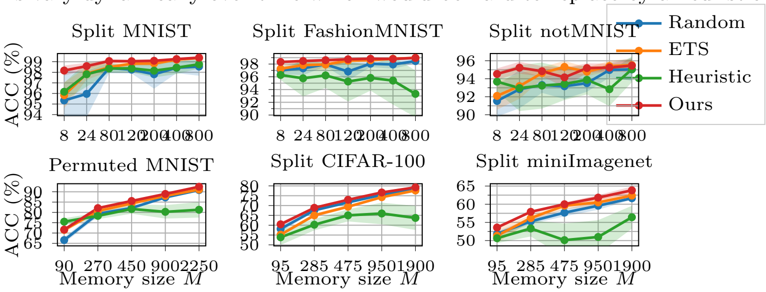


Figure 5: Average test accuracies over tasks after learning the final task (ACC) over different replay memory sizes M for the RS-MCTS (Ours) and the Random, ETS, and Heuristic baselines on all datasets. All results have been averaged over 5 seeds. The results show that replay scheduling can outperform replaying with random and ETS schedules, especially for small M, on both small and large datasets across different backbone choices. Furthermore, our method requires less careful tuning than the Heuristic baseline as M increases.

ules for each method. The results confirm that replay scheduling is important for the final performance given the same memory constraints and it can benefit any existing CL framework.

4.5 Results with Varying Replay Memory Size

We show that our method can improve the performance across different choices of memory size. In this experiment, we set the replay memory size to ensure the ETS baseline replays an equal number of samples per class at the final task. The memory size is set to $M = n_{cpt} \cdot n_{spc} \cdot (T-1)$, where n_{cpt} is the number of classes per task in the dataset and n_{spc} are the number of samples per class we wish to replay at task T for the ETS baseline. In Figure 5, we observe that RS-MCTS obtains better task accuracies than ETS, especially for small memory sizes. Both RS-MCTS and ETS perform better than Heuristic as M increases showing that Heuristic requires careful tuning of the validation accuracy threshold.

4.6 Efficiency of Replay Scheduling

We illustrate the efficiency of replay scheduling with comparisons to several common replay-based CL baselines in an even more extreme memory setting. Our

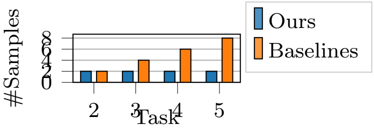


Figure 6: Number of replayed samples per task for the 5-task datasets in the tiny memory setting. Ours use $M = 2$ samples for replay, while the baselines increment their memory per task.

Table 3: Performance comparison with ACC averaged over 5 seeds in the 1 ex/class memory setting evaluated across all datasets. RS-MCTS (Ours) has replay memory size $M = 2$ and $M = 50$ for the 5-task and 10/20-task datasets respectively. The baselines replay all available memory samples. RS-MCTS performs on par with the best baselines on all datasets except S-CIFAR-100.

	5-task Datasets		
Method	S-MNIST	S-FashionMNIST	S-notMNIST
Random	92.56 (± 2.90)	92.70 (± 3.78)	89.53 (± 3.96)
A-GEM	94.97 (± 1.50)	94.81 (± 0.86)	92.27 (± 1.16)
ER-Ring	94.94 (± 1.56)	95.83 (± 2.15)	91.10 (± 1.89)
Uniform	95.77 (± 1.12)	97.12 (± 1.57)	92.14 (± 1.45)
Ours	96.07 (± 1.60)	97.17 (± 0.78)	93.41 (± 1.11)

	10- and 20-task Datasets		
Method	P-MNIST	S-CIFAR-100	S-miniImagenet
Random	70.02 (± 1.76)	48.62 (± 1.02)	48.85 (± 1.38)
A-GEM	64.71 (± 1.78)	42.22 (± 2.13)	32.06 (± 1.83)
ER-Ring	69.73 (± 1.13)	53.93 (± 1.13)	49.82 (± 1.69)
Uniform	69.85 (± 1.01)	52.63 (± 1.62)	50.56 (± 1.07)
Ours	72.52 (± 0.54)	51.50 (± 1.19)	50.70 (± 0.54)

goal is to investigate if scheduling over which tasks to replay can be more efficient in situations where the memory size is even smaller than the number of classes. To this end, we set the replay memory size for our method to $M = 2$ for the 5-task datasets, such that only 2 samples can be selected for replay at all times. For the 10- and 20-task datasets which have 100 classes, we set $M = 50$. We then compare against the most memory efficient CL baselines, namely A-GEM [34], ER-Ring [3] which show promising results with 1 sample per class for replay, and with uniform memory selection as reference. Additionally, we compare to using random replay schedules (Random) with the same memory setting as for RS-MCTS. We visualize the memory usage for our method and the baselines when training on a 5-task dataset in Figure 6. Figure 8 in Appendix 5 shows the memory usage for the other datasets.

Table 3 shows the ACC for each method across all datasets. Despite using significantly fewer samples for replay, RS-MCTS performs better or on par with the best baselines on all datasets except Split CIFAR-100. These results show that replay scheduling can be even more efficient than the state-of-the-art memory efficient replay-based methods, which indicate that learning the time to learn is an important research direction in CL.

5 Conclusions

We propose learning the time to learn, i.e., in a real-world CL context, learning schedules of which tasks to replay at different times. To the best of our knowledge, we are the first to consider the time to learn in machine learning inspired by human learning techniques. We illustrate with MCTS how replay schedules can be learned and show that they can produce significantly improved results when comparing against methods without learned scheduling under the same memory budgets. Furthermore, the dynamic behavior of the learned schedules showed similarities to human learning techniques, such as spaced repetition, by replaying previous tasks with varying time intervals. Finally, we showed that replay scheduling can be combined with any replay-based method as well as utilize the memory efficiently even when the memory size is smaller than the number of classes.

In future work, we would like to explore choosing memory samples on an instance level as the current work selects samples on task level. This would need a policy learning method that scales to large action spaces which is a research challenge by itself. Also, using MCTS for learning replay scheduling policies can be inefficient, especially in CL settings, in terms of training time and since it needs to learn for each dataset and application separately. In future work, we will incorporate reinforcement learning methods that generalize and extend to learning general policies that can be directly applied to any new application and domain.

Appendix

This supplementary material is structured as follows:

- Appendix 1: Full details of the experimental settings.
- Appendix 2: Pseudocode of RS-MCTS in Algorithm 3 to provide more details about our method.
- Appendix 3 describes the Heuristic scheduling baseline.
- Appendix 4 shows additional experimental results.
- Appendix 5 includes additional figures.
- Our code is provided in the zip-file `code_replay_scheduling_mcts_icml22.zip` as part of the supplementary material and will be made publicly available upon acceptance.

1 Experimental Settings

In this section, we describe the full details of the experimental settings used in this paper.

Datasets. We conduct experiments on six datasets commonly used in the continual learning literature. Split MNIST [16] is a variant of the MNIST [59] dataset where the classes have been divided into 5 tasks incoming in the order 0/1, 2/3, 4/5, 6/7, and 8/9. Split Fashion-MNIST [60] is of similar size to MNIST and consists of grayscale images of different clothes, where the classes have been divided into the 5 tasks T-shirt/Trouser, Pullover/Dress, Coat/Sandals, Shirt/Sneaker, and Bag/Ankle boots. Similar to MNIST, Split notMNIST [**?**] consists of 10 classes of the letters A-J with various fonts, where the classes are divided into the 5 tasks A/B, C/D, E/F, G/H, and I/J. We use training/test split provided by C:ebrahimi2020adversarial for Split notMNIST. Permuted MNIST [61] dataset consists of applying a unique random permutation of the pixels of the images in original MNIST to create each task, except for the first task that is to learn the original MNIST dataset. We reduce the original MNIST dataset to 10k samples and create 9 unique random permutations to get a 10-task version of Permuted MNIST. In Split CIFAR-100 [62], the 100 classes are divided into 20 tasks with 5 classes for each task [30, 208]. Similarly, Split miniImagenet [63] consists of 100 classes randomly chosen from the original Imagenet dataset where the 100 classes are divided into 20 tasks with 5 classes per task.

Network Architectures. We use a 2-layer MLP with 256 hidden units and ReLU activation for Split MNIST, Split FashionMNIST, Split notMNIST, and Permuted MNIST. We use a multi-head output layer for each dataset except Permuted MNIST where the network uses single-head output layer. For Split CIFAR-100, we use a multi-head CNN architecture built according to the CNN in [18, 23, 63], which consists of four 3x3 convolutional blocks, i.e. convolutional layer followed by batch normalization [65], with 64 filters, ReLU activations, and 2x2 Max-pooling. For Split mniImagenet, we use the reduced ResNet-18 from [30] with multi-head output layer.

Hyperparameters. We train all networks with the Adam optimizer [66] with learning rate $\eta = 0.001$ and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Note that the learning rate for Adam is not reset before training on a new task. Next, we give details on number of training epochs and batch sizes specific for each dataset:

- Split MNIST: 10 epochs/task, batch size 128.
- Split FashionMNIST: 30 epochs/task, batch size 128.
- Split notMNIST: 50 epochs/task, batch size 128.
- Permuted MNIST: 20 epochs/task, batch size 128.

- Split CIFAR-100: 25 epochs/task, batch size 256.
- Split miniImagenet: 1 epoch/task (task 1 trained for 5 epochs as warm up), batch size 32.

Monte Carlo Tree Search. We run RS-MCTS for 100 iterations in all experiments. The replay schedules used in the reported results on the held-out test sets are from the replay schedule that gave the highest reward on the validation sets. The exploration constant for UCT in Equation 2 is set to $C = 0.1$ in all experiments [55].

Computational Cost. All experiments were performed on one NVIDIA GeForce RTW 2080Ti. The wall clock time for ETS on Split MNIST was around 1.5 minutes, and RS-MCTS and BFS takes 40 seconds on average to run one iteration, where BFS runs 1050 iterations in total for Split MNIST.

Implementations. We adapted the implementation released by Borsos et al. [6] for the memory selection strategies Uniform sampling, k-means clustering, k-center clustering [8], and Mean-of-Features [9]. For HAL [35], MER [64], DER [33], and DER++, we follow the implementations released by [33] for each method to apply them to our replay scheduling methods. Furthermore, we follow the implementations released by C:chaudhry2019tiny and C:mirzadeh2021cl-gym for A-GEM [34] and ER-Ring [3]. For MCTS, we adapted the implementation from `https://github.com/int8/monte-carlo-tree-search` to search for replay schedules.

Experimental Settings for Single Task Replay Memory Experiment. We motivated the need for replay scheduling in continual learning with Figure 1 in Section 1. This simple experiment was performed on Split MNIST where the replay memory only contains samples from the first task, i.e., learning the classes 0/1. Furthermore, the memory can only be replayed at one point in time and we show the performance on each task when the memory is replayed at different time steps. We set the memory size to $M = 10$ samples such that the memory holds 5 samples from both classes. We use the same network architecture and hyperparameters as described above for Split MNIST. The ACC metric above each subfigure corresponds to the ACC for training a network with the single task memory replay at different tasks. We observe that choosing different time points to replay the same memory leads to noticeably different results in the final performance, and in this example, the best final performance is achieved when the memory is used when learning task 5. Therefore, we argue that finding the proper schedule of what tasks to replay at what time in the fixed memory situation can be critical for continual learning.

2 Replay Scheduling Monte Carlo Tree Search Algorithm

We provide pseudo-code in Algorithm 3 outlining the steps for our method Replay Scheduling Monte Carlo tree search (RS-MCTS) described in the main paper (Section 3.3). The MCTS procedure selects actions over which task proportions to fill the replay memory with at every task, where the selected task proportions are stored in the replay schedule S. The schedule is then passed to EvaluateReplaySchedule$(\cdot)$ where the continual learning part executes the training with replay memories filled according to the schedule. The reward for the schedule S is the average validation accuracy over all tasks after learning task T, i.e., ACC, which is backpropagated through the tree to update the statistics of the selected nodes. The schedule S_{best} yielding the best ACC score is returned to be used for evaluation on the held-out test sets.

The function GetReplayMemory$(\cdot)$ is the policy for retrieving the replay memory $\mathcal{M}$ from the historical data given the task proportion $\boldsymbol{a}$. The number of samples per task determined by the task proportions are rounded up or down accordingly to fill $\mathcal{M}$ with M replay samples in total. The function GetTaskProportion$(\cdot)$ simply returns the task proportion that is related to given node.

Algorithm 3 Replay Scheduling Monte Carlo tree search

```
Require: Tree nodes $v_{1:T}$, Datasets $\mathcal{D}_{1:T}$, Learning rate $\eta$
Require: Replay memory size $M$
ACC$_{best} \leftarrow 0$, $S_{best} \leftarrow ()$
while within computational budget do
    $S \leftarrow ()$
    $v_t, S \leftarrow$ TreePolicy($v_1, S$)
    $v_T, S \leftarrow$ DefaultPolicy($v_t, S$)
    ACC $\leftarrow$ EvaluateReplaySchedule($\mathcal{D}_{1:T}, S, M$)
    Backpropagate($v_t$, ACC)
    if ACC > ACC$_{best}$ then
        ACC$_{best} \leftarrow$ ACC
        $S_{best} \leftarrow S$
    end if
end while
return ACC$_{best}, S_{best}$

function TreePolicy($v_t, S$)
    while $v_t$ is non-terminal do
        if $v_t$ not fully expanded then
            return Expansion($v_t, S$)
        else
            $v_t \leftarrow$ BestChild($v_t$)
            $S$.append($\boldsymbol{a}_t$), where $\boldsymbol{a}_t \leftarrow$ GetTaskProportion($v_t$)
        end if
    end while
    return $v_t, S$
end function

function Expansion($v_t, S$)
    Sample $v_{t+1}$ uniformly among unvisited children of $v_t$
    $S$.append($\boldsymbol{a}_{t+1}$), where $\boldsymbol{a}_{t+1} \leftarrow$ GetTaskProportion($v_{t+1}$)
    Add new child $v_{t+1}$ to node $v_t$
    return $v_{t+1}, S$
end function

function BestChild($v_t$)
    $v_{t+1} = \underset{v_{t+1} \in \text{ children of } v}{\arg\max} \max(Q(v_{t+1})) + C\sqrt{\frac{2\log(N(v_t))}{N(v_{t+1})}}$
    return $v_{t+1}$
end function

function DefaultPolicy($v_t, S$)
    while $v_t$ is non-terminal do
        Sample $v_{t+1}$ uniformly among children of $v_t$
        $S$.append($\boldsymbol{a}_{t+1}$), where $\boldsymbol{a}_{t+1} \leftarrow$ GetTaskProportion($v_{t+1}$)
        Update $v_t \leftarrow v_{t+1}$
    end while
    return $v_t, S$
end function

function EvaluateReplaySchedule($\mathcal{D}_{1:T}, S, M$)
    Initialize neural network $f_{\boldsymbol{\theta}}$
    for $t = 1, \ldots, T$ do
        $\boldsymbol{a}_t \leftarrow S[t-1]$
```

3 Heuristic Scheduling Baseline

We implemented a heuristic scheduling baseline to compare against RS-MCTS. The baseline keeps a validation set for the old tasks and replays the tasks which validation accuracy is below a certain threshold. We set the threshold in the following way: Let $A_{t,i}$ be the validation accuracy for task t evaluated at time step i. The best evaluated validation accuracy for task t at time i is given by $A_{t,i}^{(best)} = \max(\{A_{t,1}, ..., A_{t,i}\})$. The condition for replaying task t on the next time step is then $A_{t,i} < \tau A_{t,i}^{(best)}$, where $\tau \in [0, 1]$ is a ratio controlling how much the current accuracy on task t is allowed to decrease w.r.t. the best accuracy. The replay memory is filled with M/k, where k is the number of tasks that need to be replayed according to their decrease in validation accuracy. This heuristic scheduling corresponds to the intuition of re-learning when a task has been forgotten. Training on the current task is performed without replay if the accuracy on all old tasks is above their corresponding threshold.

Grid search for τ. We performed a coarse-to-fine grid search for the ratio τ on each dataset. The best value for τ is selected according to the highest mean accuracy on the validation set averaged over 5 seeds. The validation set consists of 15% of the training data and is the same for RS-MCTS. We use the same experimental settings as described in Appendix 1. The memory sizes are set to $M = 10$ and $M = 100$ for the 5-task datasets and the 10/20-task datasets respectively, and we apply uniform sampling as the memory selection method. We provide the ranges for τ that was used on each dataset and put the best value in **bold**:

- Split MNIST: $\tau =$ [0.9, 0.93, 0.95, **0.96**, 0.97, 0.98, 0.99]
- Split FashionMNIST: $\tau =$ [0.9, 0.93, 0.95, 0.96, **0.97**, 0.98, 0.99]
- Split notMNIST: $\tau =$ [0.9, 0.93, 0.95, 0.96, 0.97, **0.98**, 0.99]
- Permuted MNIST: $\tau =$ [0.5, 0.55, 0.6, 0.65, 0.7, **0.75**, 0.8, 0.9, 0.95, 0.97, 0.99]
- Split CIFAR-100: $\tau =$ [0.3, 0.4, 0.45, **0.5**, 0.55, 0.6, 0.65, 0.7, 0.8, 0.9, 0.95, 0.97, 0.99]
- Split miniImagenet: $\tau =$ [0.5, 0.6, 0.65, 0.7, **0.75**, 0.8, 0.85, 0.9, 0.95, 0.97, 0.99]

Note that we use these values for τ on all experiments with Heuristic for the corresponding datasets. The performance for this heuristic highly depends on careful tuning for the ratio τ when the memory size or memory selection method changes, as can be seen in in Figure 5 and Table 1.

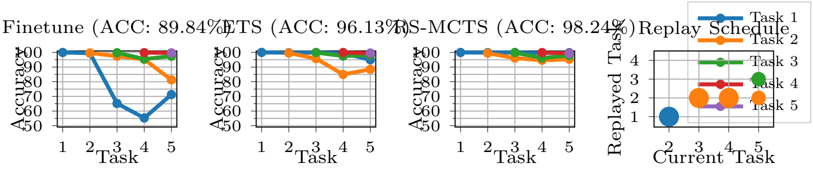


Figure 7: Comparison of test classification accuracies for Task 1-5 on Split MNIST from a network trained without replay (Fine-tuning), ETS, and RS-MCTS. The ACC metric for each method is shown on top of each figure. We also visualize the replay schedule found by RS-MCTS as a bubble plot to the right. The memory size is set to $M = 10$ with uniform memory selection for ETS and RS-MCTS. Results are shown for 1 seed.

4 Additional Experimental Results

In this section, we bring more insights to the benefits of replay scheduling in Section 4.1 as well as provide metrics for catastrophic forgetting in Section 4.2.

4.1 Replay Schedule Visualization for Split MNIST

In Figure 7, we show the progress in test classification performance for each task when using ETS and RS-MCTS with memory size $M = 10$ on Split MNIST. For comparison, we also show the performance from a network that is fine-tuning on the current task without using replay. Both ETS and RS-MCTS overcome catastrophic forgetting to a large degree compared to the fine-tuning network. Our method RS-MCTS further improves the performance compared to ETS with the same memory, which indicates that learning the time to learn can be more efficient against catastrophic forgetting. Especially, Task 1 and 2 seems to be the most difficult task to remember since it has the lowest final performance using the fine-tuning network. Both ETS and RS-MCTS manage to retain their performance on Task 1 using replay, however, RS-MCTS remembers Task 2 better than ETS by around 5%.

To bring more insights to this behavior, we have visualized the task proportions of the replay examples using a bubble plot showing the corresponding replay schedule from RS-MCTS in Figure 7(right). At Task 3 and 4, we see that the schedule fills the memory with data from Task 2 and discards replaying Task 1. This helps the network to retain knowledge about Task 2 better than ETS at the cost of forgetting Task 3 slightly when learning Task 4. This shows that the learned policy has considered the difficulty level of different tasks. At the next task, the RS-MCTS schedule has decided to rehearse Task 3 and reduces replaying Task 2 when learning Task 5. This behavior is similar to spaced repetition, where increasing the time interval between rehearsals helps memory retention. We emphasize that even on datasets with few tasks, using learned replay schedules can overcome catastrophic forgetting better than standard ETS approaches.

4.2 Analysis of Catastrophic Forgetting

We have compared the degree of catastrophic forgetting for our method against the baselines by measuring the backward transfer (BWT) metric from lopez2017gradient, which is given by

$$\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} A_{T,i} - A_{i,i}, \tag{3}$$

where $A_{t,i}$ is the test accuracy for task t after learning task i. Table 4 shows the ACC and BWT metrics for the experiments in Section **??**. In general, the BWT metric is consistently better when the corresponding ACC is better. We find an exception in Table 4 on Split CIFAR-100 and Split miniImagenet between Ours and Heuristic with uniform selection method, where Heuristic has better BWT while its mean of ACC is slightly lower than ACC for Ours. Table 5 shows the ACC and BWT metrics for the experiments in Section **??**, where we see a similar pattern that better ACC yields better BWT. The BWT of RS-MCTS is on par with the other baselines except on Split CIFAR-100 where the ACC on our method was a bit lower than the best baselines.

4.3 Applying Scheduling to Recent Replay Methods

In Section **??**, we showed that RS-MCTS can be applied to any replay method. We combined RS-MCTS together with four recent replay methods, namely Hindsight Anchor Learning (HAL) [35], Meta Experience Replay (MER) [64], and Dark Experience Replay (DER) [33]. Table 6 shows the ACC and BWT for all methods combined with the scheduling from Random, ETS, Heuristic, and RS-MCTS. We observe that RS-MCTS can further improve the performance for each of the replay methods across the different datasets.

Hyperparameters. We present the hyperparameters used for each method below. We used the same architectures and hyperparameters as described in Appendix 1 for all datasets if not mentioned otherwise. We used the Adam optimizer with learning rate $\eta = 0.001$ for MER, DER, and DER++. For HAL, we used the SGD optimizer since using Adam made the model diverge in our experiments.

- Hindsight Anchor Learning (HAL):
  - Split MNIST: learning rate $\eta = 0.1$, regularization $\lambda = 0.1$, mean embedding strength $\gamma = 0.5$, decay rate $\beta = 0.7$, gradient steps on anchors $k = 100$
  - Split FashionMNIST: learning rate $\eta = 0.1$, regularization $\lambda = 0.1$, mean embedding strength $\gamma = 0.1$, decay rate $\beta = 0.5$, gradient steps on anchors $k = 100$

  - Split notMNIST: learning rate $\eta = 0.1$, regularization $\lambda = 0.1$, mean embedding strength $\gamma = 0.1$, decay rate $\beta = 0.5$, gradient steps on anchors $k = 100$
  - Permuted MNIST: learning rate $\eta = 0.1$, regularization $\lambda = 0.1$, mean embedding strength $\gamma = 0.1$, decay rate $\beta = 0.5$, gradient steps on anchors $k = 100$
  - Split CIFAR-100: learning rate $\eta = 0.03$, regularization $\lambda = 1.0$, mean embedding strength $\gamma = 0.1$, decay rate $\beta = 0.5$, gradient steps on anchors $k = 100$
  - Split miniImagenet: learning rate $\eta = 0.03$, regularization $\lambda = 0.3$, mean embedding strength $\gamma = 0.1$, decay rate $\beta = 0.5$, gradient steps on anchors $k = 100$

- Meta Experience Replay (MER):
  - Split MNIST: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 1.0$
  - Split FashionMNIST: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 0.01$
  - Split notMNIST: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 1.0$
  - Permuted MNIST: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 1.0$
  - Split CIFAR-100: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 0.1$
  - Split miniImagenet: across batch meta-learning rate $\gamma = 1.0$, within batch meta-learning rate $\beta = 0.1$

- Dark Experience Replay (DER):
  - Split MNIST: loss coefficient memory logits $\alpha = 0.2$
  - Split FashionMNIST: loss coefficient memory logits $\alpha = 0.2$
  - Split notMNIST: loss coefficient memory logits $\alpha = 0.1$
  - Permuted MNIST: loss coefficient memory logits $\alpha = 1.0$
  - Split CIFAR-100: loss coefficient memory logits $\alpha = 1.0$
  - Split miniImagenet: loss coefficient memory logits $\alpha = 0.1$

- Dark Experience Replay++ (DER++):
  - Split MNIST: loss coefficient memory logits $\alpha = 0.2$, loss coefficient memory labels $\beta = 1.0$

- Split FashionMNIST: loss coefficient memory logits $\alpha = 0.2$, loss coefficient memory labels $\beta = 1.0$
- Split notMNIST: loss coefficient memory logits $\alpha = 0.1$, loss coefficient memory labels $\beta = 1.0$
- Permuted MNIST: loss coefficient memory logits $\alpha = 1.0$, loss coefficient memory labels $\beta = 1.0$
- Split CIFAR-100: loss coefficient memory logits $\alpha = 1.0$, loss coefficient memory labels $\beta = 1.0$
- Split miniImagenet: loss coefficient memory logits $\alpha = 0.1$, loss coefficient memory labels $\beta = 1.0$

Table 4: Performance comparison with ACC and BWT metrics for all datasets between RS-MCTS and the baselines with various memory selection methods. The memory size is set to $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation of ACC and BWT, where all results have been averaged over 5 seeds. RS-MCTS performs better or on par than the baselines on most datasets and selection methods, where MoF yields the best performance in general.

		Split MNIST		**Split FashionMNIST**		**Split notMNIS**	
Selection	**Method**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BW
Uniform	Random	95.91 (± 1.56)	-4.79 (± 1.95)	95.82 (± 1.45)	-4.35 (± 1.79)	92.39 (± 1.29)	-4.56
	ETS	94.02 (± 4.25)	-7.22 (± 5.33)	95.81 (± 3.53)	-4.45 (± 4.34)	91.01 (± 1.39)	-6.16
	Heuristic	96.02 (± 2.32)	-4.64 (± 2.90)	97.09 (± 0.62)	-2.82 (± 0.84)	91.26 (± 3.99)	-6.06
	Ours	97.93 (± 0.56)	-2.27 (± 0.71)	98.27 (± 0.17)	-1.29 (± 0.20)	94.64 (± 0.39)	-1.47
k-means	Random	94.24 (± 3.20)	-6.88 (± 4.00)	96.30 (± 1.62)	-3.77 (± 2.05)	91.64 (± 1.39)	-5.64
	ETS	92.89 (± 3.53)	-8.66 (± 4.42)	96.47 (± 0.85)	-3.55 (± 1.07)	93.80 (± 0.82)	-2.84
	Heuristic	96.28 (± 1.68)	-4.32 (± 2.11)	95.78 (± 1.50)	-4.46 (± 1.87)	91.75 (± 0.94)	-5.60
	Ours	98.20 (± 0.16)	-1.94 (± 0.22)	98.48 (± 0.26)	-1.04 (± 0.31)	93.61 (± 0.71)	-3.11
k-center	Random	96.40 (± 0.68)	-4.21 (± 0.84)	95.57 (± 3.16)	-7.20 (± 3.93)	92.61 (± 1.70)	-4.14
	ETS	94.84 (± 1.40)	-6.20 (± 1.77)	97.28 (± 0.50)	-2.58 (± 0.66)	91.08 (± 2.48)	-6.39
	Heuristic	94.55 (± 2.79)	-6.47 (± 3.50)	94.08 (± 3.72)	-6.59 (± 4.57)	92.06 (± 1.20)	-4.70
	Ours	98.24 (± 0.36)	-1.93 (± 0.44)	98.06 (± 0.35)	-1.59 (± 0.45)	94.26 (± 0.37)	-1.97
MoF	Random	95.18 (± 3.18)	-5.73 (± 3.95)	95.76 (± 1.41)	-4.41 (± 1.75)	91.33 (± 1.75)	-5.94
	ETS	97.04 (± 1.23)	-3.46 (± 1.50)	96.48 (± 1.33)	-3.55 (± 1.73)	92.64 (± 0.87)	-4.57
	Heuristic	96.46 (± 2.41)	-4.09 (± 3.01)	95.84 (± 0.89)	-4.39 (± 1.15)	93.24 (± 0.77)	-3.48
	Ours	98.37 (± 0.24)	-1.70 (± 0.28)	97.84 (± 0.32)	-1.81 (± 0.39)	94.62 (± 0.42)	-1.80

		Permuted MNIST		**Split CIFAR-100**		**Split miniImage**	
Selection	**Method**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BW
Uniform	Random	72.44 (± 1.15)	-25.56 (± 1.39)	53.99 (± 0.51)	-34.19 (± 0.66)	48.08 (± 1.36)	-15.98
	ETS	71.09 (± 2.31)	-27.39 (± 2.59)	47.70 (± 2.16)	-41.68 (± 2.37)	46.97 (± 1.24)	-18.32
	Heuristic	76.68 (± 2.13)	-20.82 (± 2.41)	57.31 (± 1.21)	-30.76 (± 1.45)	49.66 (± 1.10)	-12.04
	Ours	76.34 (± 0.98)	-21.21 (± 1.16)	56.60 (± 1.13)	-31.39 (± 1.11)	50.20 (± 0.72)	-13.46
k-means	Random	74.30 (± 1.43)	-23.50 (± 1.64)	53.18 (± 1.66)	-35.15 (± 1.61)	49.47 (± 2.70)	-14.10
	ETS	69.40 (± 1.32)	-29.23 (± 1.47)	47.51 (± 1.14)	-41.77 (± 1.30)	45.82 (± 0.92)	-19.53
	Heuristic	75.57 (± 1.18)	-22.11 (± 1.22)	54.31 (± 3.94)	-33.80 (± 4.24)	49.25 (± 1.00)	-12.92
	Ours	77.74 (± 0.80)	-19.66 (± 0.95)	56.95 (± 0.92)	-30.92 (± 0.83)	50.47 (± 0.85)	-13.31
k-center	Random	71.41 (± 2.75)	-26.73 (± 3.11)	48.46 (± 0.31)	-39.89 (± 0.27)	44.76 (± 0.96)	-18.72
	ETS	69.11 (± 1.69)	-29.58 (± 1.81)	44.13 (± 1.06)	-45.28 (± 1.04)	41.35 (± 0.96)	-23.71
	Heuristic	74.33 (± 2.00)	-23.45 (± 2.27)	50.32 (± 1.97)	-37.99 (± 2.14)	44.13 (± 0.95)	-18.26
	Ours	76.55 (± 1.16)	-21.06 (± 1.32)	51.37 (± 1.63)	-37.01 (± 1.62)	46.76 (± 0.96)	-16.56
MoF	Random	77.96 (± 1.84)	-19.44 (± 2.13)	61.93 (± 1.05)	-25.89 (± 1.07)	54.50 (± 1.33)	-8.64
	ETS	77.62 (± 1.12)	-20.10 (± 1.26)	60.43 (± 1.17)	-28.22 (± 1.26)	56.12 (± 1.12)	-8.93
	Heuristic	77.27 (± 1.45)	-20.15 (± 1.63)	55.60 (± 2.70)	-32.57 (± 2.77)	52.30 (± 0.59)	-9.61
	Ours	81.58 (± 0.75)	-15.41 (± 0.86)	64.22 (± 0.65)	-23.48 (± 1.02)	57.70 (± 0.51)	-5.31

Table 5: Performance comparison with ACC and BWT metrics for all datasets between RS-MCTS and the baselines in the setting where only 1 sample per class can be replayed. The memory sizes are set to $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation of ACC and BWT, where all results have been averaged over 5 seeds. RS-MCTS performs on par with the best baselines for both metrics on all datasets except S-CIFAR-100.

	Split MNIST		**Split FashionMNIST**		**Split notMNIST**	
Method	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%
Random	92.56 (± 2.90)	-8.97 (± 3.62)	92.70 (± 3.78)	-8.24 (± 4.75)	89.53 (± 3.96)	-8.13 (± 5.
A-GEM	94.97 (± 1.50)	-6.03 (± 1.87)	94.81 (± 0.86)	-5.65 (± 1.06)	92.27 (± 1.16)	-4.17 (± 1.
ER-Ring	94.94 (± 1.56)	-6.07 (± 1.92)	95.83 (± 2.15)	-4.38 (± 2.59)	91.10 (± 1.89)	-6.27 (± 2.
Uniform	95.77 (± 1.12)	-5.02 (± 1.39)	97.12 (± 1.57)	-2.79 (± 1.98)	92.14 (± 1.45)	-4.90 (± 1.
RS-MCTS (Ours)	96.07 (± 1.60)	-4.59 (± 2.01)	97.17 (± 0.78)	-2.64 (± 0.99)	93.41 (± 1.11)	-3.36 (± 1.

	Permuted MNIST		**Split CIFAR-100**		**Split miniImagenet**	
Method	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%
Random	70.02 (± 1.76)	-28.22 (± 1.92)	48.62 (± 1.02)	-39.95 (± 1.10)	48.85 (± 1.38)	-14.55 (± 1
A-GEM	64.71 (± 1.78)	-34.41 (± 2.05)	42.22 (± 2.13)	-46.90 (± 2.21)	32.06 (± 1.83)	-30.81 (± 1
ER-Ring	69.73 (± 1.13)	-28.87 (± 1.29)	53.93 (± 1.13)	-34.91 (± 1.18)	49.82 (± 1.69)	-14.38 (± 1
Uniform	69.85 (± 1.01)	-28.74 (± 1.17)	52.63 (± 1.62)	-36.43 (± 1.81)	50.56 (± 1.07)	-13.52 (± 1
RS-MCTS (Ours)	72.52 (± 0.54)	-25.43 (± 0.65)	51.50 (± 1.19)	-37.01 (± 1.08)	50.70 (± 0.54)	-12.60 (± 1

Table 6: Performance comparison with ACC and BWT metrics between scheduling methods RS-MCTS (Ours), Random, ETS, and Heuristic when combining them with replay-based methods Hindsight Anchor Learning (HAL), Meta Experience Replay (MER), Dark Experience Replay (DER), and DER++. Replay memory sizes are $M = 10$ and $M = 100$ for the 5-task and 10/20-task datasets respectively. We report the mean and standard deviation averaged over 5 seeds. Results on Heuristic where some seed did not converge is denoted by *. Applying RS-MCTS to each method can enhance the performance compared to using the baseline schedules.

		Split MNIST		**Split FashionMNIST**		**Split no**
Method	**Schedule**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑
HAL	Random	97.24 (± 0.70)	-2.77 (± 0.90)	86.74 (± 6.05)	-15.54 (± 7.58)	93.61 (± 1.31)
	ETS	97.21 (± 1.25)	-2.80 (± 1.59)	96.75 (± 0.50)	-2.84 (± 0.75)	92.16 (± 1.82)
	Heuristic	97.69 (± 0.19)	-2.22 (± 0.24)	*74.16 (± 11.19)	-31.26 (± 14.00)	93.64 (± 0.93)
	Ours	97.96 (± 0.15)	-1.85 (± 0.18)	97.56 (± 0.51)	-2.02 (± 0.63)	94.47 (± 0.82)
MER	Random	93.07 (± 0.81)	-8.36 (± 0.99)	85.53 (± 3.30)	-0.56 (± 3.36)	91.13 (± 0.86)
	ETS	92.97 (± 1.73)	-8.52 (± 2.15)	84.88 (± 3.85)	-3.34 (± 5.59)	90.56 (± 0.83)
	Heuristic	94.30 (± 2.79)	-6.46 (± 3.50)	96.91 (± 0.62)	-1.34 (± 0.76)	90.90 (± 1.30)
	Ours	96.44 (± 0.72)	-4.14 (± 0.94)	86.67 (± 4.09)	0.85 (± 3.85)	92.44 (± 0.77)
DER	Random	98.23 (± 0.53)	-1.89 (± 0.65)	96.56 (± 1.79)	-3.48 (± 2.22)	92.89 (± 0.86)
	ETS	98.17 (± 0.35)	-2.00 (± 0.42)	97.69 (± 0.58)	-2.05 (± 0.71)	94.74 (± 1.05)
	Heuristic	94.57 (± 1.71)	-6.08 (± 2.09)	*72.49 (± 19.32)	-20.88 (± 11.46)	*77.88 (± 12.58)
	Ours	99.02 (± 0.10)	-0.91 (± 0.13)	98.33 (± 0.51)	-1.26 (± 0.63)	95.02 (± 0.33)
DER++	Random	97.90 (± 0.52)	-2.32 (± 0.67)	97.10 (± 1.03)	-2.77 (± 1.29)	93.29 (± 1.43)
	ETS	97.98 (± 0.52)	-2.24 (± 0.66)	98.12 (± 0.40)	-1.59 (± 0.52)	94.53 (± 1.02)
	Heuristic	92.35 (± 2.42)	-8.83 (± 2.99)	*67.31 (± 21.20)	-24.86 (± 16.34)	93.88 (± 1.33)
	Ours	98.84 (± 0.21)	-1.14 (± 0.26)	98.38 (± 0.43)	-1.17 (± 0.51)	94.73 (± 0.20)

		Permuted MNIST		**Split CIFAR-100**		**Split min**
Method	**Schedule**	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑
HAL	Random	88.49 (± 0.99)	-7.03 (± 1.05)	36.09 (± 1.77)	-17.49 (± 1.78)	38.51 (± 2.22)
	ETS	88.46 (± 0.86)	-7.26 (± 0.90)	34.90 (± 2.02)	-18.92 (± 0.91)	38.13 (± 1.18)
	Heuristic	*66.63 (± 28.50)	-29.68 (± 27.90)	35.07 (± 1.29)	-24.76 (± 2.41)	39.51 (± 1.49)
	Ours	89.14 (± 0.74)	-6.29 (± 0.74)	40.22 (± 1.57)	-12.77 (± 1.30)	41.39 (± 1.15)
MER	Random	75.90 (± 1.34)	-21.69 (± 1.47)	42.96 (± 1.70)	-34.01 (± 2.07)	31.48 (± 1.65)
	ETS	73.01 (± 0.96)	-25.19 (± 1.10)	43.38 (± 1.81)	-34.84 (± 1.98)	33.58 (± 1.53)
	Heuristic	83.86 (± 3.19)	-12.48 (± 3.60)	40.90 (± 1.70)	-44.10 (± 2.03)	34.22 (± 1.93)
	Ours	79.72 (± 0.71)	-17.42 (± 0.78)	44.29 (± 0.69)	-32.73 (± 0.88)	32.74 (± 1.29)
DER	Random	87.51 (± 1.10)	-8.81 (± 1.28)	56.83 (± 0.76)	-27.34 (± 0.63)	42.19 (± 0.67)
	ETS	85.71 (± 0.75)	-11.15 (± 0.87)	52.58 (± 1.49)	-32.93 (± 2.04)	35.50 (± 2.84)
	Heuristic	81.56 (± 2.28)	-15.06 (± 2.51)	55.75 (± 1.08)	-31.27 (± 1.02)	43.62 (± 0.88)
	Ours	90.11 (± 0.18)	-5.89 (± 0.23)	58.99 (± 0.98)	-24.95 (± 0.64)	43.46 (± 0.95)
DER++	Random	87.89 (± 1.10	-8.35 (± 1.33)	58.49 (± 1.44)	-25.48 (± 1.55)	48.40 (± 0.69)
	ETS	85.25 (± 0.88)	-11.60 (± 1.03)	52.54 (± 1.06)	-33.22 (± 1.51)	41.36 (± 2.90)
	Heuristic	79.17 (± 2.44)	-17.68 (± 2.68)	56.70 (± 1.27)	-30.33 (± 1.41)	45.73 (± 0.84)
	Ours	89.84 (± 0.22)	-6.13 (± 0.29)	59.23 (± 0.83)	-24.61 (± 0.91)	49.45 (± 0.68)

5 Additional Figures

Memory usage. We visualize the memory usage for Permuted MNIST and the 20-task datasets Split CIFAR-100 and Split miniImagenet in Figure 8 for our method and the baselines used in the experiment in Section **??**. Our method uses a fixed memory size of $M = 50$ samples for replay on all three datasets. The memory size capacity for our method is reached after learning task 6 and task 11 on the Permuted MNIST and the 20-task datasets respectively, while the baselines continue incrementing their replay memory size.

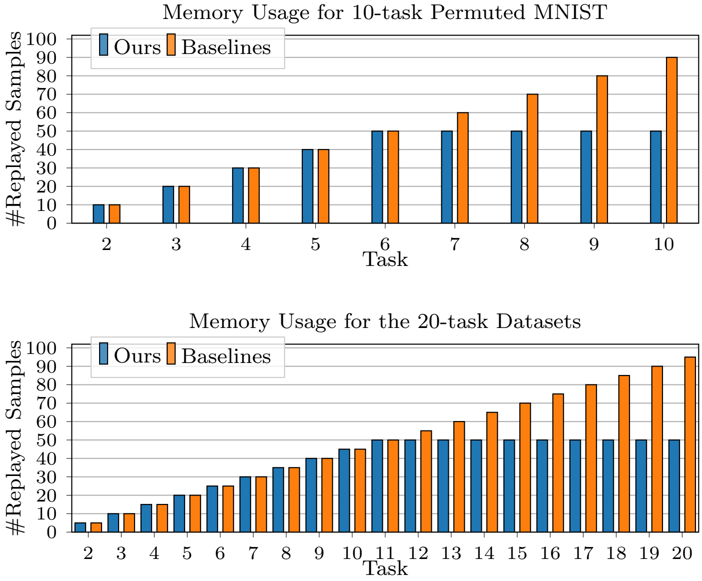


Figure 8: Number of replayed samples per task for 10-task Permuted MNIST (top) and the 20-task datasets in the experiment in Section **??**. The fixed memory size $M = 50$ for our method is reached after learning task 6 and task 11 on the Permuted MNIST and the 20-task datasets respectively, while the baselines continue incrementing their number of replay samples per task.

References

[1] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2(6), 2019.

[2] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[4] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020.

[5] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019.

[6] Zalán Borsos, Mojmír Mutnỳ, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.

[7] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *ICML*, 2020.

[8] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[10] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

[11] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *European Conference on Computer Vision*, pages 699–715. Springer, 2020.

[12] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. *arXiv preprint arXiv:1912.01100*, 2019.

[13] Frank N Dempster. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330, 1989.

[14] Hermann Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

[15] T. Landauer and Robert Bjork. Optimum rehearsal patterns and name learning. *Practical aspects of memory*, 1, 11 1977.

[16] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[17] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[18] Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). *arXiv preprint arXiv:1911.09514*, 2019.

[19] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[21] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[22] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[23] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[24] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[25] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019.

[26] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[27] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[28] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[29] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[30] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

[31] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[32] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[33] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.

[34] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[35] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6993–7001, 2021.

[36] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision – ECCV 2020*, pages 86–102. Springer International Publishing, 2020.

[37] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *arXiv preprint arXiv:2003.09553*, 2020.

[38] KJ Joseph and Vineeth N Balasubramanian. Meta-consolidation for continual learning. *arXiv preprint arXiv:2010.00352*, 2020.

[39] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020.

[40] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. *arXiv preprint arXiv:2004.14070*, 2020.

[41] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

[42] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[43] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, 2019.

[44] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[45] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[46] John Dunlosky, Katherine A. Rawson, Elizabeth J. Marsh, Mitchell J. Nathan, and Daniel T. Willingham. Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest*, 14(1):4–58, 2013.

[47] Judy Willis. Review of research: Brain-based teaching strategies for improving students' memory, learning, and test-taking success. *Childhood Education*, 83(5):310–315, 2007.

[48] Karri S Hawley, Katie E Cherry, Emily O Boudreaux, and Erin M Jackson. A comparison of adjusted spaced retrieval versus a uniform expanded retrieval schedule for learning a name–face association in older adults with probable alzheimer's disease. *Journal of Clinical and Experimental Neuropsychology*, 30(6):639–649, 2008.

[49] Hadi Amiri, Timothy Miller, and Guergana Savova. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2410, 2017.

[50] Kanyin Feng, Xiao Zhao, Jing Liu, Ying Cai, Zhifang Ye, Chuansheng Chen, and Gui Xue. Spaced learning enhances episodic memory by increasing neural pattern similarity across repetitions. *Journal of Neuroscience*, 39(27):5351–5360, 2019.

[51] Paul Smolen, Yili Zhang, and John H Byrne. The right time to learn: mechanisms and optimization of spaced learning. *Nature Reviews Neuroscience*, 17(2):77, 2016.

[52] Philip J Ball, Yingzhen Li, Angus Lamb, and Cheng Zhang. A study on efficiency in continual learning inspired by human learning. *arXiv preprint arXiv:2010.15187*, 2020.

[53] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020.

[54] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[55] Muhammad Umar Chaudhry and Jee-Hyong Lee. Feature selection for high dimensional data using monte carlo tree search. *IEEE Access*, 6:76036–76048, 2018.

[56] Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of uct with patterns in monte-carlo go. Technical Report RR-6062, INRIA, 2006. inria-00117266v3f.

[57] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[58] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[61] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[62] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[63] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[64] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[65] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[66] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Appendix A

Meta Policy Learning for Replay Scheduling in Continual Learning

Marcus Klasson[*], Hedvig Kjellström[*], Cheng Zhang[†]

[*]KTH Royal Institute of Technology, Stockholm, Sweden
[†]Microsoft Research, Cambridge, United Kingdom

Abstract

Learning the time to replay different tasks have been demonstrated to be important in continual learning. However, a replay scheduling policy that can be applied in any continual learning scenario is currently absent, which makes replay scheduling infeasible in real-world scenarios. To this end, we propose using reinforcement learning to enable learning general policies that can generalize across different data domains. Thus, an efficient replay schedule can be used in any new real-world continual learning scenario to improve the continual learning performance without any additional computational cost. We compare the learned policies to several replay scheduling baselines and show that the learned policies can improve the continual learning performance given unseen tasks and datasets.

1 Introduction

hej hej här är en artikel

what do you think this is?